DISS. ETH NO. 30024

On Abstract Models in Cryptography and Their Applications

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES

(Dr. sc. ETH Zurich)

presented by

Julia Kastner

M. Sc., Karlsruhe Institute of Technology

born on 18.01.1995

accepted on the recommendation of

Prof. Dr. Dennis Hofheinz
Prof. Dr. Ueli Maurer
Prof. Dr. Stefano Tessaro

2024

Acknowledgments

First of all, I want to thank Dennis Hofheinz for being a great doctoral advisor. In your case, Dennis, I think the word advisor can really be taken literally as you not only supervised my thesis, but also took the time to give me good advice on whatever question I had related to being a researcher, be it on concrete research questions, student supervision and teaching, work-life balance, future career or any other topic. I also want to thank you for fostering such a pleasant social atmosphere in the group with many fun lunch conversations. Thank you for making my time as a doctoral student so enjoyable!

I would also like to thank Ueli Maurer and Stefano Tessaro for taking the time to be on my Committee for the Doctoral Exam and all the work that entails. I am grateful to all my co-authors Thomas Agrikola, Nicholas Brandt, Dennis Hofheinz, Kristina Hostáková, Karen Klein, Julian Loss, Ky Nguyen, Michael Reichle, Omar Renawi, Akin Ünal, Bogdan Ursu, and Jiayu Xu for the productive collaboration on works during my doctoral studies. Thank you Claudia for the administrative support.

I would also like to thank my current and former colleagues both at ETH as well as KIT. In particular, thank you Akin for Lossy the Trapdoor Squirrel, Alex for my first taste of scientific writing, Bogdan for seeking drama and rarely finding it, Cecilia for snake tales, Diana for tarot readings, Fabio for your firm stance on veganism, Gianluca for fun language facts, Geoffroy for the most interesting conversation topics, Giovanni for many bouldering adventures, Guilherme for organizing social events at conferences, Jiaxin for supervising my master's thesis, Karen for travelling advice and stories, Kien for the coffee cult initiation ritual, Kristina for organising 'the betrayal', Laura for finger-physio advice, Lisa for translating 'squirrel' into all the languages, Marta for many fun hikes, Matilda for VMI events, Matteo for graph-theoretic observations and matchmaking, Michael K. for your undying love for expected polynomial time, Michael R. for Christmas market visits, Nico for many political discussions, Roman for your enthusiasm for cycling, Suvradip for sweets, Sven for vegan pancakes, Thomas A. for teaching me how to play table tennis, Thomas W. for the fastest coffee refills of Karlsruhe, and Varun for asking the important questions.

Finally, I thank my friends, family for their support. Thank you Daniele for all the cuteness shared. Thank you Markus for putting up with squirrels and yarn for so long. Thank you Jakob for the many inliner excursions, thank you Dominik for the countless hours spent together throughout our studies, thank you Killian for opening my mind to new ideas. Thank you to the bouldering gang for all the adventures, the pilgrimage to Font, and an unforgettable Mallorca trip. Thank you to the knitting group for many group video calls during lockdown. Thank you Mama and Papa for your continuous support, thank you Adam for being my favourite little brother, thank you Oma Hedi for always stressing the importance of education especially for us girls and women, thank you Oma Anne for the best cooking, thank you Opa for your creativity.

Contents

Abstract

When talking about the security of a cryptographic scheme, researchers often model it as a *game* between a *challenger* and an *adversary*. Such a game gives a clear description of the ways we consider the adversary to interact with the scheme whose security we want to prove. A security reduction then only needs to implement these ways of interactions which we usually call 'oracles'. However, the scheme may have some underlying building blocks, such as cryptographic groups or hash functions. For example, if the scheme uses a hash function, a security proof can be in the Random Oracle Model (ROM) where the hash function is replaced by an idealized function which returns uniformly random values from the output space of the hash function and can be accessed only through an oracle, thus giving the adversary no access to the code of the function. This gives a reduction some additional leverage over the adversary, for example it can *observe* what queries the adversary makes to the random oracle, or it can also *program* the random oracle, i.e. provide the adversary with a random oracle implementation that returns values that are useful for the reduction. Another case where the ROM is useful is when the reduction wants to rewind the adversary and replay it using a different hash function evaluation, for example in the context of Fiat-Shamir based signatures. When rewinding, we consider the adversary to be a computer program that the reduction can run multiple times and where it can determine the random coins given to the program. Other than that, the reduction gets only black-box access.

In this thesis, we consider rewinding as a proof strategy in several contexts. First, we look into a novel strategy of so-called *undirected* rewinding. In this rewinding strategy, we rewind the adversary to every step of its execution, and then in some cases change some of its inputs at the step it has been rewound to (e.g. change responses to oracle queries it made). We apply this strategy to circumvent existing lower bounds (Kamath, Klein, Pietrzak and Walter, TCC 2021) for proving the adaptive security of the Goldreich-Goldwasser-Micali Pseudo-Random Function (GGM-PRF) as a Prefix-Constrained PRF (PC-PRF). A PC-PRF allows the holder of the secret PRF key to hand out prefix keys that allow others to evaluate the PRF on values that start with a specific prefix, while the function should still be pseudo-random on other inputs. In the adaptive setting, the adversary may query such prefix keys even depending on the keys it saw before, which makes it difficult for a reduction to predict its behaviour. We circumvent this issue by rewinding the adversary to learn some of its future choices and provide an analysis that shows that the reduction loss is *polynomial in the input length* rather than super-polynomial which would be the case for a straight-line reduction.

We furthermore apply the undirected rewinding technique to prove the adaptive security of the Logical Key Hierarchy (LKH) protocol for server-assisted group key exchange. Also here it is tricky for straight-line reductions to figure out when to embed challenges, and we can help the reduction with a rewinding strategy.

We then turn to the setting of blind and partially blind signatures. A blind signature scheme is a two-party protocol between a *Signer* and a *User* where the signer holds the secret signing key and the user has a message it wants to have signed. We require *blindness*, i.e. even a malicious Signer should not learn the User's message, and *one-more unforgeability*, i.e. a malicious User should not be able to produce more message-signature pairs than it requested signatures from the Signer. In a partially blind scheme, signatures are issued with respect to a shared tag info and the two security properties need to hold with respect to each tag. We revisit the influential partially blind scheme by Abe and Okamoto (CRYPTO 2000) whose construction and proof have served as an inspiration for several other blind and partially blind signature schemes. We point out a subtle gap in their rewinding-based proof of one-more unforgeability. We then show how to mend the gap with a new analysis of the success probability of the forking strategy of the reduction.

After this, we turn to the popular blind signature scheme by Abe (EUROCRYPT 2001) which is known to have a flaw in the rewinding based security proof (pointed out by Okuhbo and Abe, SCIS 2003). While the proof strategy for the Abe-Okamoto scheme can be applied, it incurs a rather large loss. We therefore turn to another abstract model in cryptography for proving one-more unforgeability: the Algebraic Group Model (AGM) (introduced by Fuchsbauer, Kiltz, Loss, CRYPTO 2018). In the AGM, adversaries are assumed to be *algebraic*. Intuitively speaking, this means the adversary computes new group elements only using the group operation. In the AGM, this intuition is modelled by requiring the adversary to always output an explanation how it computed the group elements in its output. We show how this algebraic explanation can be exploited by a reduction to show the *concurrent* security of Abe's blind signature scheme and even of a new partially blind variant that we introduce. We furthermore investigate the *sequential* security of Blind Schnorr Signatures, and show that sequential One-More Unforgeability of Blind Schnorr Signatures can be shown assuming the hardness of the One-More Discrete Logarithm Problem in the AGM + ROM. We furthermore show that it is indeed necessary for an algebraic reduction (even against an algebraic adversary) to query the Discrete Logarithm Oracle as many times as the adversary closes a signing session.

Whenever confronted with a new abstract model, one may ask how realistic it is and how it compares to other models and assumptions. We therefore provide some evidence towards the algebraic group model being realistic. Namely, we construct a group scheme that we call the *algebraic wrapper* from strong, but falsifiable assumptions. The algebraic wrapper allows the person who set up the group parameters to 'extract' an algebraic explanation in a limited manner. We show that several existing proofs from the AGM can be translated into the algebraic wrapper setting, lending some credibility to proofs in the AGM.

Zusammenfassung

Wenn sie über die Sicherheit eines kryptografischen Schemas sprecen, modellieren Forscher diese oft als ein *Spiel* zwischen einem *Herausforderer* und einem *Angreifer*. So ein Spiel gibt eine klare Beschreibung der Arten, die wir betrachten, wie der Angreifer mit dem Schema interagieren kann, dessen Sicherheit wir beweisen wollen. Eine Sicherheitsreduktion muss dann nur diese Arten der Interaktion implementieren, die wir üblicherweise als *Orakel* bezeichnen. Allerdings kann es sein, dass das Schema zugrundeliegende Bausteine wie Hashfunktionen oder kryptographische Gruppen verwendet. Wenn das Schma zum Beispiel eine Hashfunktion verwendet, kann ein Sicherheitsbeweis im Zufalsorakelmodell (ROM) sein, wo die Hashfunktion durich eine idealisierte Funktion ersetzt wird, die zufällige Werte aus dem Ausgabebereich der Funktion zurückgibt, und auf die nur durch ein Orakel zugegriffen werden kann, also der Angreifer keinen Zugriff auf den Programmbeschrieb der Hashfunktion hat. Das gibt der Reduktion einen zusätzlichen Hebel gegenüber dem Angreifer, beispielsweise kann sie die Anfragen des Angreifers an das Orakel *beobachten* oder sie kann das Orakel *programmieren*, also das Orakel für den Angreifer so implementieren, dass es Wrte zurückgibt, die für die Reduktion nützlich sind. Ein anderer Fall, in dem das ROM nützlich ist, ist wenn die Reduktion den Angreifer zurückspulen möchte und ihn dann erneut laufen lassen will, aber mit einer anderen Auswertung der Hashfunktion, zum Beispiel im Kontext von Fiat-Shamir basierten Signaturen. Wenn wir den Angreifer zurückspulen, betrachten wir ihn als Computerprogramm, das die Reduktion mehrfach laufen lassen kann und dessen Zufallsentscheidungen sie bestimmen darf. Abgesehen davon bekommt die Reduktion nur Black-Box-Zugriff.

In dieser Doktorarbeit betrachten wir Zurückspulen als Beweisstrategie in mehreren Kontexten. Zuerst schauen wir uns eine neue Strategie des sogenannten *ungerichteten* Zurückspulens an. In dieser Zurückspulungsstrategie spulen wir den Angreifer zu jedem Schritt seiner Ausführung zurück und verändern dann in manchen Fällen seine Eingaben an der Stelle, zu der wir ihn zurückgespult haben (z.B: ändern die Antworten auf Orakel-Anfragen, die er gemacht hat). Wir wenden diese Strategie an, um existierende untere Schranken (Kamath, Klein Pietzrak und Walter, TCC 2021) für die adaptive Sicherheit der Goldreich-Goldwasser-Micali Pseudozufallsfunktion (GGM-PRF) als präfix-beschränkte PRF (PC-PRF) zu umgehen. Eine PC-PRF erlaubt es, dem Halter des geheimen PRF-Schlüssels, Präfixschlüssel auszuhändigen, die es anderen erlaubem, die PRF auf Werten auszuwerten, die mit einem bestimmten Präfix anfangen, während die Funktion weiterhin auf anderen Eingaben pseudozufällig sein sollte. Im adaptiven Rahmen darf der Angreifer solche Präfixschlüssel sogar in Abhängigkeit von Präfixschlüsseln, die er bereits gesehen hat anfragen, was es für die Reduktion schwierig macht, sein Verhalten vorherzusehen. Wir umgehen dieses Problem, indem wir den Angreifer zurückspulen um einige seiter zukünftigen Entscheidungen zu lernen und liefern eine Analyse, die zeigt, dass der Reduktionsverlust *polynomiell in der Eingabelänge* anstatt superpolynomiell ist, wie es der Fall für eine geradlinige Reduktion wäre.

Wir wenden die ungerichtete Zurückspulungsstrategie ausserdem an, um die adaptive Sicherheit des Logischen Schlüsselhierarchie-Protokolls (LKH) für Server-unterstützten Grupenschlüsselaustausch zu beweisen. Auch hier ist es schwierig für geradlinige Reduktionen, herauszufinden, wann Herausforderungen eingebettet werden können, und wir können der Reduktion mit einer Zurückspulungsstrategie helfen.

Wir wenden uns dann blinden und teilweise blinden Signatuen zu. Ein blindes Signaturschema ist ein Zweiparteienprotokoll zwischen einem *Signierer* und einem *Benutzer*, bei dem der Signierer einen geheimen Signierungsschlüssel hat, und der Benutzer eine Nachricht, die er signiert haben möchte. Wir erfordern *Blindheit*, d.h. sogar ein bösartiger Signierer sollte nicht in der Lage sein, die Nachricht des Benutzers herauszufinden, und *Einmal-Mehr-Unfälschbarkeit*, d.h. ein bösartiger Benutzer sollte nicht in der Lage sein, mehr Nachrichten-Signatur-Paare zu erzeugen, als er Signaturen vom Signierer angefragt hat. In einem teilweise blinden Schema werden Signaturen bezüglich einer gemeinsamen Beschriftung info ausgestellt und die beiden Sicherheitseigenschaften müssen bezüglich jeder Beschriftung gelten. Wir greifen das einflussreiche teilweise blinde Signaturverfahren von Abe und Okamoto (CRYPTO 2000) wieder auf, dessen Konstruktion und Beweis als Inspiration für viele andere blinde und teilweise blinde Schemata gedient haben. Wir zeigen eine subtile Lücke in deren zurückspulungsbasiertem Beweis der Einmal-Mehr-Unfälschbarkeit auf. Wir zeigen dann, wie man die Lücke mit einer neuen Analyse der Erfolgswahrscheinlichkeit der Gabelungssstrategie der Reduktion schliessen kann.

Danach wenden wir uns dem beliebten blinden Signaturschema von Abe (EUROCRYPT 2001) zu, von dem es bekannt ist, dass es einen Makel im zurückspulungsbasierten Sicherheitsbeweis enthält (zuerst entdeckt von Okuhbo und Abe, SCIS 2003). Während die Beweisstrategie vom Abe-Okamoto-Schema angewandt werden kann, bringt diese einen grossen Verlust mit sich. Daher verwenden wir ein anderes abstraktes Modell der Kryptographie um die Einmal-Mehr-Unfälschbarkeit zu beweisen: das Algebraische Gruppenmodell (AGM) (eingeführt von Fuchsbauer, Kiltz und Loss, CRYPTO 2018). Im AGM wird davon ausgegangen, dass Angreifer sich *algebraisch* verhalten. Intuitiv heißt dass, dass der Angreifer neue Gruppenelemente ausschliesslich durch die Gruppenmultiplikation berechnet. Im AGM wird diese Intuition dadurch modelliert, dass vom Angreifer verlangt wird, dass er immer eine Erklärung ausgibt, wie er die Gruppenelemente in seiner Ausgabe berechnet hat. Wir zeigen wie diese algebraische Erklärung von einer Reduktion ausgenutzt werden kann, um die *nebenläufige* Sicherheit von Abe's blindem Signaturschema und sogar von einer teilweise blinden Variante, die wir einführen, zu beweisen. Wir untersuchen weiterhin die *sequentielle* Sicherheit von blinden Schnorr-Signaturen und zeigen, dass die sequentielle Einmal-Mehr-Unfälschbarkeit von blinden Schnorr-Signaturen im AGM+ROM bewiesen werden kann, unter der Annahme, dass das Einmal-Mehr-Diskreten-Logarithmus-Problem (OMDL) schwierig ist. Wir zeigen ausserdem, dass es tatsächlich notwendig ist, dass eine algebraische Reduktion (sogar gegen einen algebraischen Angreifer) das diskreter-Logarithmus-Orakel so oft anfragt wie der Angreifer Signatursitzungen schliesst.

Wenn man mit einem neuen abstrakten Modell konfrontiert ist, kann man sich fragen, wie realistisch es ist und wie es sich im Vergleich zu anderen Modellen und Annahmen verhält. Wir bieten daher etwas Beweismaterial in die Richtung, dass das AGM realistisch ist. Nämlich konstruieren wir ein Gruppenschema, das wir die *algebraische Verpackung* nennen, von starken aber falsifizierbaren Annahmen. Die algebraische Verpackung erlaubt es der Person, die die Gruppenparameter aufgesetzt hat, eine algebraische Erklärung in einer beschränkten Art und Weise zu ‘extrahieren’. Wir zeigen, dass mehrere existierende Beweise aus dem AGM in den Rahmen der algebraischen Verpackung übersetzt werden können und verleihen Beweisen im AGM damit etwas Glaubwürdigkeit.

List of Publications

Articles Included in Thesis

The following are articles where one or multiple main results are included in the thesis.

- D. Hofheinz, J. Kastner and K. Klein. 'The Power of Undirected Rewindings for Adaptive Security'. In: *CRYPTO 2023, Part II*. ed. by H. Handschuh and A. Lysyanskaya. Vol. 14082. LNCS. Springer, Heidelberg, Aug. 2023, pp. 725–758. DOI: 10.1007/978-3-031-38545-2_24

- J. Kastner, J. Loss and J. Xu. 'The Abe-Okamoto Partially Blind Signature Scheme Revisited'. In: *ASIACRYPT 2022, Part IV*. ed. by S. Agrawal and D. Lin. Vol. 13794. LNCS. Springer, Heidelberg, Dec. 2022, pp. 279–309. DOI: 10.1007/978-3-031-22972-5_10

- J. Kastner, J. Loss and J. Xu. 'On Pairing-Free Blind Signature Schemes in the Algebraic Group Model'. In: *PKC 2022, Part II*. ed. by G. Hanaoka, J. Shikata and Y. Watanabe. Vol. 13178. LNCS. Springer, Heidelberg, Mar. 2022, pp. 468–497. DOI: 10.1007/978-3-030-97131-1_16

- T. Agrikola, D. Hofheinz and J. Kastner. 'On Instantiating the Algebraic Group Model from Falsifiable Assumptions'. In: *EUROCRYPT 2020, Part II*. ed. by A. Canteaut and Y. Ishai. Vol. 12106. LNCS. Springer, Heidelberg, May 2020, pp. 96–126. DOI: 10.1007/978-3-030-45724-2_4

  For this article we focus on the applications of the algebraic wrapper to proofs in the AGM. We omit the detailed proofs for the properties of the wrapper but rather only sketch them. We refer the reader to the full version of [AHK20] or to [Agr21] for the full proofs.

Articles Partially Included in Thesis

We use a minor result from the following article to improve the presentation of one result from [KLX22a].

- J. Kastner, J. Loss and O. Renawi. 'Concurrent Security of Anonymous Credentials Light, Revisited'. In: *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. CCS '23. Copenhagen, Denmark: Association for Computing Machinery, 2023, pp. 45–59. ISBN: 9798400700507. DOI: 10.1145/3576915.3623184. URL: https://doi.org/10.1145/3576915.3623184

Articles and Unpublished Manuscripts not Included in Thesis

The following articles and manuscripts were written during the doctoral studies, but are not included in the thesis.

- N. Brandt, D. Hofheinz, J. Kastner and A. Ünal. 'The Price of Verifiability: Lower Bounds for Verifiable Random Functions'. In: *TCC 2022, Part II*. ed. by E. Kiltz and V. Vaikuntanathan. Vol. 13748. LNCS. Springer, Heidelberg, Nov. 2022, pp. 747–776. DOI: `10.1007/978-3-031-22365-5_26`
- D. Hofheinz, K. Hostáková, J. Kastner, K. Klein and A. Ünal. *Compact Lossy Trapdoor Functions and Selective Opening Security From LWE*. PKC 2024. To appear. 2024. URL: `https://eprint.iacr.org/2023/864`
- D. Hofheinz, J. Kastner, A. Ünal and B. Ursu. *Decoding LTFs in the Generic Group Model*. Cryptology ePrint Archive, Paper 2023/866. `https://eprint.iacr.org/2023/866`. 2023. URL: `https://eprint.iacr.org/2023/866`
- J. Kastner, K. Nguyen and M. Reichle. *Pairing-Free Blind Signatures from Standard Assumptions in the ROM*. In Submission. 2023

Chapter 1

Introduction

1.1 Motivation

The desire of humans to send secret messages hidden from potential curious eyes dates back thousands of years with one well-known example stemming from Caesar being said to have sent 'encrypted' messages using a substitution cipher for his private correspondence [Sue21]. Other 'encryption' methods of the time even relied on the method itself remaining secret, so if an adversary was to find out about the method one would have had to come up with a new method.

It was only much later that Kerckhoffs [Ker83] came up with guidelines for developing secure cryptographic systems. The main principle that is still relevant today is that the system itself should not require secrecy for it to remain secure. At the time, many existing cryptosystems did not fulfill this criterion and security was assumed if there was no existing attack.

The first modern proof of security of a cryptographic scheme was given by Shannon [Sha49] for the one-time pad which had at the time been in use for several decades[Mil82; Ver26]. In a seminal paper, Diffie and Hellman [DH76] introduced many important concepts for modern cryptography. For example, they described public key encryption, i.e. schemes where there are two types of keys, one for encryption which can be posted in a public place, and a secret decryption key, digital signatures, a primitive that allows to sign using a secret signing key and verify signatures using a public verification key, as well as a method for exchanging secret keys for encryption. As a building block to construct these schemes, they introduced so-called trapdoor one-way functions. Such functions have the property that the function f is easy to compute, however the inverse f^{-1} is hard to compute - unless one has access to some additional information called the 'trapdoor'. While they did not have a concrete example of such a trapdoor one-way function, just two years after, Rivest, Shamir and Adleman [RSA78] had found one.

However, their signature scheme based on the RSA assumption remained without a formal proof of security for many years, until Bellare and Rogaway [BR93] introduced the *Random Oracle Model* (ROM), an abstract model in cryptography. The Trapdoor One-Way Function Full Domain Hash (TDOWF-FDH)signature, of which RSA Full Domain Hash is an instantiation works as follows. The underlying building blocks are a hash function H and a trapdoor one-way function f with inverse f^{-1}. The trapdoor is the secret key, whereas the one-way function f is the public key. To sign a message m, one computes $\sigma = f^{-1}(H(m))$ using the trapdoor to compute f^{-1}. Verification is easy as one can simply check $f(\sigma) = H(m)$.

1.1.1 The Random Oracle Model - an Abstract Model in Cryptography

In the ROM, the hash function is replaced by an oracle that implements a random function. The proof of security operates via a reduction that breaks the one-way property of the trapdoor OWF. The reduction receives a challenge, i.e. a value on which to invert the function along with a description of the function (but not the trapdoor) as an input. It then has to use the adversary somehow to invert the function. The adversary may also request signatures on messages of his choice from the reduction, so the reduction has to find a way to generate such signatures. In the random oracle model, it is possible to give a proof for TDOWF-FDH, namely the reduction can *program* (i.e. implement) the random oracle in such a a way that a) it can sign using only the function f by implementing $H(m) = f(\sigma)$ for values σ of its choice and b) it can have the random oracle H output its inversion challenge for a hash query of the adversary, thus using the adversary as an inverter. In [BR93], the authors give many more examples of applications of the ROM such as public key encryption and non-interactive zero knowledge proofs.

One application that we will also use in this thesis is that of rewinding as a proof strategy. For some cryptographic primitives, for example those that build on the Fiat-Shamir Heuristic [FS87], a single run of a successful adversary may not be enough to extract information that solves a cryptographic hardness assumption. The Fiat-Shamir Heurisitic is a way to transform a Σ-protocol [Cra97] into a non-interactive zero-knowledge proof of knowledge or a signature. It works by replacing the random choices of the verifier with a hash of the messages sent so far between prover and verifier. As in a Σ-protocol, the witness can be computed from two transcripts of the protocol with the same first prover messages but different verifier challenges, a witness can be extracted from an adversary producing non-interactive Fiat-Shamir based proofs in the ROM as follows. First, run the adversary with the random oracle, answering queries via lazy sampling. Then, rewind the adversary to the point just after it has made the query to the random oracle that resulted in the proof it output in the first run. As the adversary has already made its query, it is the same 'first prover message' as in the first run, but now we re-sample which response we give to the adversary, i.e. replace the hash value by a uniformly random different hash value. If the adversary uses the same hash query again for its forgery, the reduction can obtain a witness from the two transcripts as they contain the same first message but different verifier challenges. The probability of success for this strategy can be computed using the *forking lemma* by [PS00]. In [PS00], this technique was used to prove security of Schnorr Signatures [Sch90] as well as the Okamoto-Schnorr blind signature scheme [Oka93]. Later in this thesis (namely in Chapters 3 and 4), we will see more applications of rewinding as a proof strategy.

1.1.2 Abstract Models for Cryptographic Groups

Just like hash functions can be idealized in the ROM, around the same time, Nechaev [Nec94] and Shoup [Sho97] introduced the Generic Group Model (GGM) as a means to justify the hardness of the discrete logarithm problem in groups of prime order. In their variant of the GGM, anyone interacting with the group only gets to see random 'labels'. To perform the group operation, one needs to query an oracle that will invert the labelling, add exponents together, and output the label of the sum of the exponents. This allows to prove the lower bounds for various assumptions in groups (including the discrete logarithm assumption) using the following strategy: Internally, the group oracle is replaced by an oracle that instead of operating on $\mathbb{Z}_q$, operates on a polynomial ring over $\mathbb{Z}_q$, for example in the discrete logarithm case, the group oracle is replaced by an oracle that operates on $\mathbb{Z}_q[X]$. The discrete logarithm challenge is internally replaced by the formal variable X whereas the group generator is given as the label of 1. Then, the adversary is allowed to interact with this alternative oracle, but as X is not an actual group exponent, there is no way the adversary can learn anything about the exponent. However, as this oracle internally uses a different group, this implementation is not perfect. The probability of making a 'mistake' that is noticeable for the adversary can however be bounded using the Schwartz-Zippel-Lemma [Sch80; Zip79; DL78]. Later on, Maurer [Mau05] introduced a variant of the GGM that gets rid of the labelling function where algorithms only interact with the group in a black-box manner. While many works refer to 'the GGM' only, a recent work by Zhandry [Zha22] showed that the models are indeed different in the sense that there are games that one can It is Maurer's variant of the GGM that Fuchsbauer, Kiltz and Loss [FKL18] used to justify the Algebraic Group Model (AGM) [Los23] and Zhandry [Zha22] indeed shows that algorithms that work in Maurer's model are also algebraic. Other than in the GGM(s), an adversary in the AGM is allowed full access to a cryptographic group as long as it outputs an algebraic explanation of what it did, i.e. how it computed the group elements in its output from the group elements in its input using the group operation. The notion of such algebraic algorithms has been around for longer though. Boneh and Venkatesan [BV98] introduced algebraic algorithms in the context of a meta-reduction to show that RSA may not be equivalent to factoring. Later on, Paillier and Vergnaud [PV05] refined this definition stating that for algebraic algorithms there must be an extractor to extract the algebraic explanation from the adversary. Both works consider only algebraicity of reductions though, whereas the work by Fuchsbauer, Kiltz and Loss [FKL18] is the first to consider algebraic adversaries as well. While being a very new model, it has since found plenty of applications for proving security of Digital Signatures [FKL18; FPS20; TZ23], Blind Signatures [FPS20; TZ22; KLX22a; Cri+23], Threshold Signatures [CKM23], Zero-Knowledge Protocols [FKL18] as well as more general families of group based hardness assumptions [BFL20]. Researchers also came up with variants of the model extending it to having the adversary explain its decision bit output algebraically [RS20] as well as to a more fine-grained explanation version where the adversary needs to additionally provide the order in which it computed the group operations [KLX20].

All of this research interest sparks a natural question that we will discuss in the following.

1.1.3 How Realistic are Abstract Models?

To answer such a question, one first has to think about what it means for an abstract model to be realistic. Take for example the ROM: On the one hand, as discussed above, there are many schemes whose proofs use the ROM where no attacks are known to contradict these proofs. On the other hand, in Canetti, Goldreich and Halevi [CGH98] constructed examples of schemes that can be proven secure in the ROM, but are insecure when the Random Oracle is replaced by any real hash function. The schemes constructed contain a specifically inserted breaking point. We give a high-level intuition of how this breaking point is inserted. This builds on correlation intractable functions, which are, informally speaking, function ensembles for which it is hard to find input-output pairs that fulfill a certain evasive relation. Random oracles fulfill this criterion as their outputs are random, however for any real world hash function one can define the relation of input-output pairs of the function and thus find a relation that will always be fulfilled. The idea of 'breaking' any scheme that uses the ROM, say, for sake of this exposition a signature scheme, is to modify the signing algorithm such that it first checks whether the message to be signed and its hash fulfill a certain relation, and if yes output the secret key, if no run the normal signing algorithm. Signature verification works analogously by outputting 1 if message and hash fulfill the relation and using the old verification algorithm otherwise. The scheme remains both correct and secure in the ROM as it is hard for an adversary to find a message that makes the signing oracle output the secret key, but it is easy to break in the real world. This means, there can be no hash function that is 'as good' as a Random Oracle. However, the schemes constructed in that work are not schemes intended for use in the real world, but rather constructed specifically so that they break as soon as the hash function is implemented in the real world. This result has also been extended by Dent [Den02] for Shoup's version of the GGM.

While on the other hand, Dent's impossibility result did not apply to Maurer's version of the GGM, Zhandry [Zha22] showed that also Maurer's model cannot be instantiated by providing a public key encryption scheme that is secure in Maurer's model but cannot be secure for any real world group. This impossibility result also extends to Shoup's model without making usage of the interpretation of the group exponentiation function as a Random Oracle. The same paper also showed that the AGM is uninstantiable, i.e. there exists a cryptographic game that is hard to win in the AGM (assuming the discrete logarithm assumption is hard), but easy to win outside of the AGM. The trick of how to construct such a game is to 'feed' the adversary a group element label of the solution of the game in a non-algebraic way. This allows any non-algebraic adversary to output the solution, whereas an algebraic adversary that also has to come up with an algebraic explanation would have to break the discrete logarithm assumption. At around the same time, another work [ZZK22], showed that the AGM is incomparable to Shoup's GGM, i.e. there are games that are secure in one model, but not the other and vice versa.

While none of the above models are fully instantiable, there are works providing partial instantiations that can be used for some applications. For the ROM, there are Universal Computational Extractors (UCE) [BHK13], a type of hash function that cannot be distinguished from a Random Oracle in a particular 2-stage game. They show that this type of hash function can be used in various types of encryption schemes that rely on the ROM and still yield a proof of security. However, later on Brzuska, Farshim and Mittelbach [BFM14] showed that a certain subclass of UCEs is itself again uninstantiable, assuming the existence of indistinguishability obfuscation, while on the other hand Jost and Maurer [JM18] showed that the Merkle-Damgård construction satisfied a UCE security notion if the underlying round function had certain properties, and Brzuska and Mittelbach [BM14] provided a construction of a hash function instantiating a certain class of UCE.

In the case of the GGM, there have been multiple approaches for instantiation. On the one hand, there are several concrete constructions of groups in which certain hardness assumptions hold, like for example [AH18]. In their construction of a cryptographic group, the interactive Uber assumption holds. The Uber Assumption Family [Boy08; BBG05] is a family of assumptions generalizing the Diffie-Hellman assumption [DH76].

On the other hand, more recently Pseudo-Generic Groups [Bau+22] were introduced as the Shoup GGM-analogue of UCEs. While this notion captures one characterization of what it means for a group to behave 'like a generic group', no provable instantiations beyond the GGM itself have been presented so far.

For the AGM, there are two main instantiation attempts. One of them shows equivalence with a strong algebraic knowledge assumption [KP19], the other (which we will discuss in a bit more detail in Chapter 7) uses strong falsifiable assumptions to obtain some form of algebraic explanation with respect to a fixed basis of group elements [AHK20].

1.1.4 Our Contributions

In this work, we interact with abstract models and proof strategies derived from them in several ways. In particular, we show the following things:

- *adaptive security* of the Goldreich-Goldwasser-Micali PRF as a prefix-constrained PRF using *rewinding*
- *adaptive security* of the Logical Key Hierarchy protocol using *rewinding*

- *one-more unforgeability* of the Abe-Okamoto partially blind signatue scheme using *rewinding in the ROM*
- *one-more unforgeability* of a partially blind variant of Abe's blind signature scheme in the *AGM + ROM*
- *sequential one-more unforgeability* of blind Schnorr signatures in the *AGM + ROM*, as well as the optimality of that reduction w.r.t. amount of discrete logarithm queries made by the reduction
- how to use a *partial instantiation of the AGM*, called the algebraic wrapper, to transfer AGM proofs to the *standard model*

We give a more detailed overview over these topics in the following.

1.2 Adaptive Security in Game-Based Notions

When proving the security of a cryptographic scheme, it is a common strategy to formulate what it even means to be 'secure' as a so called *game* where the adversary plays against a challenger [BR04; Sho04]. The game has an initial setup, and then the adversary is started with an initial input and access to some oracles defined by the game. In the end, the adversary outputs its solution to the challenge posed. For some types of cryptographic primitives, it makes sense to consider a setting where one of the oracles is one that allows the adversary to *corrupt* certain parts of the structure of the primitive, thus learning some secret information. In such a setting, we distinguish between the *selective* setting where the adversary has to specify at the beginning of the game all the parts it wants to corrupt, and the *adaptive* setting where it can corrupt at any point in time, and in particular after it has already learned secret information from previous corruptions. For example, in Chapter 3, we will consider a prefix-constrained pseudorandom function (PC-PRF). A pseudorandom function F takes as input a secret key k as well as a value x to evaluate the function on. The pseudorandomness game works as follows. During setup, the game samples a secret key. Then, the adversary is allowed access to an oracle that either implements $F_k(\cdot)$ or a truly random function $R(\cdot)$. At the end, the adversary outputs a bit as a guess whether it was given access to F_k or to R and it wins if this guess was correct. As a security property, we want that the winning probability of the adversary is close to $\frac{1}{2}$. For a PC-PRF, the adversary additionally gets to obtain so-called prefix-constrained keys, that is keys $k_{\bar{x}}$ that allow the adversary to evaluate F_k on all inputs that have $\bar{x}$ as a prefix. In this case, we say the adversary has *corrupted* this key. As described above, we can consider the selective case, where the adversary specifies which prefixes it would like to know the keys for at the beginning, or the adaptive case, where the adversary gets access to an oracle that allows it to corrupt arbitrary keys. In addition to this, the adversary gets to also query a challenge oracle on a value that it does know a prefix key for, and this challenge oracle is like in the standard pseudorandomness game either a real implementation of F_k, or a truly random function R.

When proving the security of a cryptographic primitive, one often employs a reduction that shows that an adversary against the primitive can be used to break the security of some underlying building block or some mathematical hardness assumption.

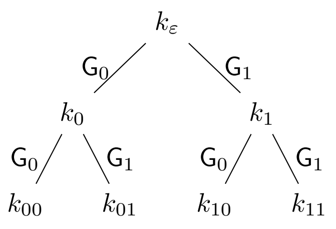
1.2.1 The Goldreich-Goldwasser-Micali PRF

In Chapter 3, we consider the case of the Goldreich-Goldwasser-Micali (GGM) PRF [GGM84a]. It is known that this PRF can be viewed as a PC-PRF [Kia+13; BW13; BGI14]. The GGM PRF is built on a so-called pseudorandom generator (PRG). A PRG is a function that maps an input x to an output y where y is longer than x (we consider the case where y is twice as long as x, i.e. $x \in \{0,1\}^n$ and $y \in \{0,1\}^{2n}$). In the pseudorandomness game of a PRG the adversary is given access to an oracle that either samples random values from the input space of the PRG and outputs the corresponding PRG values, i.e. it samples $x \xleftarrow{\$} \{0,1\}^n$ and outputs $\mathsf{G}(x)$, or it samples random values from $\{0,1\}^{2n}$ directly and outputs those. The adversary then has to guess which oracle it interacted with. The GGM PRF now works as follows. The key space is $\mathcal{K} = \{0,1\}^n$. On input of a value x, the PRF proceeds as follows. For each bit of x, it evaluates the PRG, first on the secret key k. If the first bit of x is 0 it takes the first half of the output of the PRG to continue, otherwise the second. It iterates this process with the output of the previous round as the input always using the next bit of x to select which half of the output to use. The last output value of the PRG, i.e. when there are no more bits of x to process, is also the output value of the PRF. This evaluation can also be expressed as a tree, namely one where the key k sits at the root, and the two child keys are the two halves of the PRG evaluated on k, the children of those are the halves of the PRG outputs of the PRG evaluated on the first level etc. A picture of this tree can be seen in Figure 1.1a.

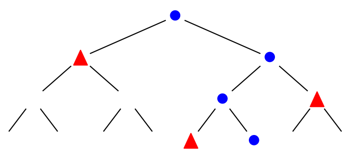
The proof for the plain PRF security proceeds by replacing the keys in the tree layer by layer from top to bottom using the PRG security in each hybrid game. This layering is necessary as PRG security only holds for inputs that are chosen uniformly at random from the input space.

For the prefix-constrained setting, the proof of selective security follows a similar strategy, but this time only replacing the keys on the path to the challenge value and on its co-path[1]. This is possible as by the definition of the game and the validity of the adversary, there are no corrupted keys on the path to the challenge, and in the selective setting the adversary has to disclose at the beginning which leaf will be its challenge, so the reductions between hybrids know which vertices lie on the path to the challenge. A depiction of the GGM PRF tree with some replaced keys can be seen in Figure 1.1b.

For adaptive security, it is known that any straight-line reduction[2] to the pseudorandomness property of the underlying PRG will have a super-polynomial loss in the input length [Kam+21]. Therefore, it makes sense to consider rewinding as a proof strategy.



(a) The GGM PRF evaluation where G_0 denotes the PRG G restricted to the first half of the output and G_1 denotes G restricted to the second half.



(b) GGM PRF tree with randomized keys along the path (blue ●) and co-path (red ▲) to k_{101}, as desirable when $x^* = 101$ is selected as challenge.

Figure 1.1: The GGM PRF evaluation (Figure 1.1a) and tree (Figure 1.1b).

[1]the co-path of a path in a tree consists of the vertices adjacent to the path
[2]i.e. one that does not rewind the adversary

1.2.2 Rewinding Techniques

If we imagine the adversary as a piece of code, like a computer program, the reduction may be able to run this computer program, and also reset it to previous points and re-run it with different inputs. We call this technique of resetting and rerunning the adversary *rewinding*.

Rewinding is a common strategy for to extract witnesses from adversaries in settings with Σ-protocols [Cra97] as well as signatures that result from the Fiat-Shamir [FS87] transformation of a Σ-protocol. In particular, Schnorr signatures [Sch90] were proven secure using the *forking lemma* by [PS96; PS00]. Intuitively, the forking lemma says that if one runs the adversary a first time and it is successful (i.e. the adversary wins its game, for example by forging a signature), rewinding it has a reasonable probability to result in a successful run again. Usually, applications of the forking lemma are settings in which the adversary is rewound until the reduction has two successful transcripts which it can then use to extract a witness or secret key from the adversary. This is particularly useful for reductions attempting to solve a computational problem like the discrete logarithm problem in a finite group.

In this thesis, we apply the (a variant of) the forking lemma to very carefully analyse a partially blind signature scheme by Abe and Okamoto [AO00]. We provide some more details on this later on, but first we want to discuss a different way we apply rewinding in this thesis. Namely, in Chapter 3, we consider the use of rewinding to obtain tighter security guarantees for adaptive security. Let's return to the case of the GGM PRF as a PC-PRF. The difficulty in the adaptive setting is that the reduction does not know in advance which keys the adversary will want to corrupt and which will be on the path to the challenge. We note that in the case of the GGM PRF, if one has two keys $k_{\bar{x}}$ and $k_{\bar{x}'}$ where $\bar{x}$ is a prefix of $\bar{x}'$ one can easily check consistency by evaluating the PRG along the path from $\bar{x}$ to $\bar{x}'$ in the tree. Thus, if a reduction was to replace two such keys (or in fact anything below $\bar{x}$ in the tree) by uniformly random values using the PRG property, the adversary would be able to notice and the reduction would fail. We depict such a scenario in Figure 1.2. However, if the adversary is valid, i.e. never asks for a key that

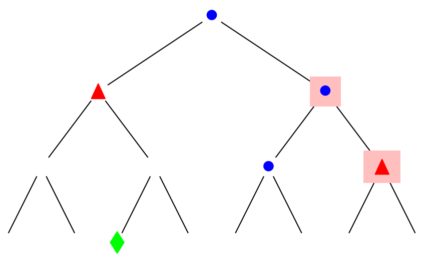


Figure 1.2: A problematic case when having to guess the challenge or parts thereof: The GGM tree with some nodes on a path (indicated by ●) and its co-path (indicated by ▲) replaced by random values. This hybrid would be desirable if the adversary chose 100 or 101 as the challenge. It leads to a failure however, if the adversary chooses a different challenge (here vertex 010, denoted by ◆) and corrupts a vertex along the replaced path as well as one of its offspring, here the vertices 1 and 11, marked by the pink boxes.

would allow him to evaluate the PRF on its challenge, we can be sure that any vertex on the path and co-path to the challenge is never *below* a corrupted key and thus those vertices are safe to replace. We therefore want to employ rewinding to give the reduction some partial information about the location of the challenge input.

There are two main aspects to consider in the rewinding strategy, namely

1. which point to 'rewind to' an what to alter for subsequent runs (the most common form of rewinding strategy is to rewind to a suitable 'relevant' point in the execution of the adversary, e.g. the time a hash query is made that is later on used in a forged signature, and for subsequent runs some or all random-coins that the challenger/reduction uses for oracle responses after the rewound point are resampled uniformly at random)
2. when to stop rewinding an resampling (this could be after a fixed number of rewinding rounds or when the current run of the adversary fulfills a certain property, e.g. it outputs a valid forgery using the same hash query again)

Let us first consider case 1. In our undirected rewinding strategy, we actually rewind to every step in the adversary's transcript with the challenger. However, for the PRF case, at some points in the transcript, there is no possibility for re-randomization and thus our strategy is equivalent to not rewinding at these points.

Informally speaking, for the purpose of this exposition the transcript consists of the following types of events:

corruption events these happen whenever the adversary requests to corrupt a key

challenge event this event is triggered by the adversary requesting a PRF challenge

key events these are the keys (corrupted or challenge) that the adversary receives

PRG evaluation events whenever the adversary requests a corruption or a challenge, the game internally has to derive the corresponding keys. This adds PRG evaluation events for any not yet derived key on the path from the last derived key to the currently requested key. For example, if the key k_{00} is requested to be corrupted as the first corruption in the tree, this adds PRG evaluation events for the input of the root key k_ε as well as for the input of k_0. If later on, the key k_{010} is corrupted, this only adds a PRG evaluation event from k_{01} because k_ε and k_0 have already been queried to the PRG to derive k_{01} along with k_{00} earlier.

It is easy to see that the only type of event where the reduction can apply any kind of re-randomization is the PRG evaluation event, and only if the key above has already been re-randomized in a previous hybrid. As explained above, it is furthermore important that only PRG evaluations along the path to the challenge are replaced by random function evaluations. Therefore, the point in the transcript that we want to alter in the hop from the ith to the $i+1$st hybrid is the $i+1$st PRG evaluation event along the path to the challenge. As this happens only when rewinding the adversary, the challenge it used in the previous round is known and thus this path is well-defined. We further note that we use a notion of PRG security where we can query the PRG oracle multiple times, and thus it is possible that the reduction, if it is unhappy with the second run, rewinds the adversary again to the same spot and re-samples with another potentially random value until the adversary behaves in the desired way.

We will now discuss what this 'desired way' is, namely when to stop rewinding at a certain index.

One possible strategy could be that the reduction keeps replacing the PRG evaluation event in question (and re-evaluates the PRG for all events that depend on it) until the adversary re-uses the exact same input value as its challenge value. This is however problematic as the number of possible challenge values is exponential in the input length of the PRF. Therefore, there is no guarantee that the same input value might re-occur soon and the reduction might have an exponential running time.

We therefore employ a more sophisticated strategy. We note that the transcript itself is actually of a length polynomial in the input length of the PRF. Thus, preserving some function that is an index inside the transcript is a more feasible goal. The choice of index that we will use is the position of this 'replacable' PRG evaluation event described before, i.e. the $i+1$st PRG evaluation event along the path to the challenge. As this PRG evaluation event has to happen *somewhere* in the transcript, there are only as many choices for this value as there are positions in the transcript.[3] By a probabilistic argument we can therefore bound the number of repetitions of this rewinding process to a polynomial.

1.2.3 Another Application: The Logical Key Hierarchy

As a second application of the rewinding strategy, we prove the security of the Logical Key Hierarchy (LKH) [WHA98; WGL00; Can+99] in the variant of [Pan07], a mechanism for server assisted key exchange in group messaging applications. The LKH works as follows: Keys are arranged in a tree, where the shared key is the root key. User keys are located at the leaves and each inner vertex is assigned a key as well. In addition to a collection of vertices with keys, each edge is labelled with a ciphertext that is an encryption of the parent key under the child key. This allows someone in possession of a leaf key to decrypt the keys along the path up to the root and finally the root key. We show the structure of the tree in Figure 1.3 The LKH protocol also supports updating keys (in the scenario of group messaging, imagine the setting where a user leaves or joins the group, or a user's key is leaked to an adversary). Instead of updating all keys in the system, a key update requires only updating the keys on the path from the leaf to the root, as well as the ciphertexts on the ingoing edges.

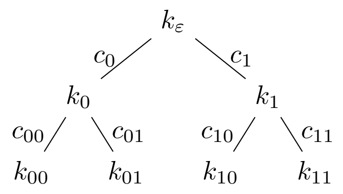


Figure 1.3: Depiction of the LKH structure as a tree. The vertices correspond to keys whereas the labels on the edges correspond to ciphertexts of the parent vertex key under the child vertex key.

We formulate a security game for this protocol. The game initializes the tree, and the adversary gets to see the ciphertexts labelling the edges. The adversary is then given the opportunity to corrupt leaf nodes. This triggers an update along the path to the root as described above. The adversary can also request to be challenged on the root key. In this case, it either receives the real root key that is encrypted in the ciphertexts on the two incoming edges of the root, or it receives a fresh random root key. It has to output a guess of whether the key it got was a real or random key and wins if its guess was correct.

The rewinding proof strategy is similar to the PC-PRF case, however the conditions are more complicated to formulate. In particular, as the tree edges are in a sense 'directed' from the leaves to the root, applying the IND-CPA property to gradually 'forget' the root key and all information that might be leaked about it, needs a more intricate hybrid argument.

This argument uses a strategy called *pebbling*. Edge-pebbling proceeds over multiple rounds where pebbles can be placed or removed from edges in a graph according to the following rule. In each round, an edge is allowed to be pebbled or unpebbled if all of the incoming edges of its source vertex are pebbled.

[3]Another way to view this condition is that we rewind until the challenge is again in the same subtree underneath the PRG evaluation event's position in the tree, but it is not as obvious to argue why the probability of hitting this subtree again is high enough.

In a tree this means that an edge can be pebbled whenever the two edges coming from the children are pebbled.

This has the following correspondence to the LKH game: An edge is pebbled when the ciphertext on it is an encryption of a random key and not the key of the vertex at the end of the edge. Thus, in order to pebble an edge (using IND-CPA security), the ciphertexts of the two ingoing edges of the source vertex cannot contain information about the encryption key. The same holds for switching back edges (ciphertexts) to the unpebbled (real encryption) setting.

To pebble the graph up to the root vertex, we use the edge pebbling algorithm of [Jaf+17] which recursively pebbles resp. unpebbles the child edges.

The hybrids of our rewinding based proof correspond to pebbling configurations in this algorithm, and each step uses an IND-CPA reduction to either place or remove a pebble. Again, we need to talk about the two rewinding criteria. For this, we first consider what kind of events can occur in the transcript with the adversary.

corruption queries these indicate that the adversary requested to corrupt a leaf key

challenge queries these indicate that the adversary requested to be challenged

new key events these are triggered by the adversary corrupting a key and the subsequent update of the path to the root. The keys are not leaked to the adversary, but the event indicates that the game internally sampled a new key.

ciphertext events these are the ciphertexts output to the adversary either as part of the initial tree or as part of an update

corrupted keys corrupted keys output to the adversary in response to a corruption query

challenge key challenge key (either real or random root key) output to the adversary in response to a challenge query

The tricky thing for the reduction between two hybrids to decide is whether a ciphertext event on an edge that needs to be pebbled or unpebbled in this step is a point in time when the adversary will no longer obtain a key to decrypt this ciphertext. Like before, this corresponds to an index within the transcript and thus has a polynomial range. Therefore, by the same probablistic argument as before, we can preserve this index when rewinding and resampling. However, for technical reasons, the argument becomes a bit more complicated as in between hybrids, we also need to switch some stopping and rewinding conditions. We give a full proof of the LKH security in Section 3.3.

1.3 (Partially) Blind Signatures

We turn to another type of scheme in which rewinding as a proof technique has a long tradition [PS00]. Blind Signatures [Cha82] are a primitive by which a *user* can obtain a signature from a *signer* on a message of its choice without revealing said message. There are two desired security properties for blind signatures. The *blindness* property protects honest users from malicious signers and guarantees that a signer cannot link message-signature pairs to interactions it had with users, whereas the *one-more unforgeability* property protects honest signers from malicious users by guaranteeing that a user cannot generate more message-signature pairs than it obtained through interaction with the signer. We note that this is a different unforgeability notion than that of (non-blind) digital signatures as the messages used in signing queries are unknown to the challenger due to the blindness property and thus we can only count interactions and signatures.

Blind signatures find applications in many different contexts - originally introduced for electronic payments [Cha82; CFN90; OO92] they found early applications electronic voting [Cha88; FOO93], and anonymous credentials [Bra94; CL01]. In recent years, the field has found renewed interest due to applications in blockchain contexts [YL19; Bus+23] and for private authentication, e.g. for VPN services [Goo].

One popular way to construct blind signatures is from the discrete-logarithm based identification scheme by Schnorr [Sch90], with potential adaptions. In blind Schnorr signatures [Sch01], the signers secret key is a scalar x, its public key is $\mathbf{y} = \mathbf{g}^x$ where $\mathbf{g}$ is a generator of group with prime order q. The signer's side of this protocol is a proof of knowledge of the secret key, whereas the user-side of the protocol generates a Fiat-Shamir style signature on the message with some assistance from the signer. We describe the interaction in more detail in the following. For signing, the signer sends the first message $\mathbf{R} = \mathbf{g}^r$ for a uniformly random scalar $r \xleftarrow{\$} \mathbb{Z}_q$. The user then blinds the group element using blinding factors $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_q$ to obtain $\mathbf{R}' := \mathbf{R} \cdot \mathbf{g}^\alpha \cdot \mathbf{y}^\beta$ and computes the hash $c' = H(\mathbf{R}', m)$ where m is the message of its choice. It sends the challenge $c = c' - \beta$ to the signer. The signer computes $s = r - cx$ and sends c, s to the user. The user de-blinds the value $s' = s + \alpha$ and obtains the signature (c', s'). Verification[4] works by checking that $H(\mathbf{g}^{s'}\mathbf{y}^{c'}, m) = c'$. While blind Schnorr signatures only have proofs in idealized group models [Sch01; FPS20; KLX22a], cannot be proven secure using the standard forking technique [BL13b], and are vulnerable to the ROS-attack in the concurrent setting [Sch01; Wag02; Ben+21], the blinding technique serves as a basis for many similar discrete-log based schemes, e.g. [Abe01; AO00; TZ22].

With a lot of renewed interest in Blind Signature, much research has been conducted in recent years toward constructing blind signatures from various underlying constructs such as pairing-free groups [HKL19; TZ22; Cri+23], pairing groups [HLW23], lattices [Rüc10; Beu+23; AHJ21; Agr+22; dK22], as well as isogenies [Kat+23].

1.3.1 The Scheme by Abe and Okamoto

In particular Abe and Okamoto [AO00] used the OR-proof technique of [CDS94] to combine two blind Schnorr signatures into one partially blind signature. Partially blind signatures [AF96] are a generalization of blind signatures where the signer and user can agree on a shared information called the *tag* and create signatures that are bound to said tag. The blindness property is then only required to hold for message-signature pairs that belong to the same tag, and one-more unforgeability also needs to hold within each tag (i.e. the adversary should not be able to produce more message-signature pairs than requested for each tag).

The Abe-Okamoto protocol has two keys, one public key that works like the Schnorr public key, i.e. $\mathbf{y} = \mathbf{g}^x$ where x is the secret key, and a tag key $\mathbf{z} = H^*(\text{info})$ where H^* is a hash function mapping into the group. The signer's side of the protocol proves knowledge of the secret key or the discrete log of the tag key. That is, it distributes the challenge e into two values $c + d$ and uses them as separate challenges for the two commitments it sent before. In an honest run of the protocol with the signer, the tag key comes out of a hash function and therefore the signer does not actually know the discrete logarithm. It therefore simulates that half of the OR-proof by choosing d, s in advance and computing $\mathbf{b} := \mathbf{g}^s\mathbf{z}^d$, whereas the commitment for the $\mathbf{y}$-proof is chosen honestly as $\mathbf{a} := \mathbf{g}^u$. The user blinds both values like

[4]This is a non-standard version of Schnorr signatures where the sign of s' is inverted. We provide this version as it is more in line with the two (partially) blind signature schemes described in the following.

$$
\begin{array}{lcr}
\textbf{Signer} & & \textbf{User} \\
\mathsf{sk} = x & & \mathsf{pk} = \mathbf{y} \\
\mathsf{pk} = (\mathbf{y} = \mathbf{g}^x), \mathbf{z} = H^*(\mathsf{info}) & & m, \mathsf{info}, \mathbf{z} = H^*(\mathsf{info}) \\
\hline
u, s, d \xleftarrow{\$} \mathbb{Z}_q & & \\
\mathbf{a} := \mathbf{g}^u & & \\
\mathbf{b} := \mathbf{g}^s \cdot \mathbf{z}^d & & \\
 & \xrightarrow{\mathbf{a},\mathbf{b}} & \\
 & & t_1, t_2, t_3, t_4 \xleftarrow{\$} \mathbb{Z}_q \\
 & & \boldsymbol{\alpha} := \mathbf{g}^{t_1} \cdot \mathbf{y}^{t_2} \cdot \mathbf{a} \\
 & & \boldsymbol{\beta} := \mathbf{g}^{t_3} \cdot \mathbf{y}^{t_4} \cdot \mathbf{b} \\
 & & h := H(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{z}, m) \\
 & & e := h - t_2 - t_4 \\
 & \xleftarrow{e} & \\
c := e - d & & \\
r := u - cx & & \\
 & \xrightarrow{c,d,r,s} & \\
 & & \rho := r + t_1 \\
 & & \omega := c + t_2 \\
 & & \sigma := s + t_3 \\
 & & \delta := d + t_4 \\
 & & \omega + \delta \stackrel{?}{=} H(\mathbf{g}^{\rho} \cdot \mathbf{y}^{\omega}, \mathbf{g}^{\sigma} \cdot \mathbf{z}^{\delta}, \mathbf{z}, m) \\
 & & \Downarrow \\
 & & \mathsf{sig} := (\rho, \omega, \sigma, \delta)
\end{array}
$$

Figure 1.4: The Abe-Okamoto scheme with parts corresponding to the $\mathbf{y}$-side of the protocol marked in purple and parts corresponding to the $\mathbf{z}$-side of the protocol marked in green.

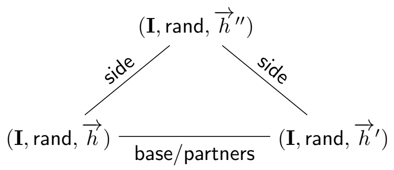
in the Schnorr signature and hashes them together with the message. It then sends the blinded challenge e to the signer to obtain c, d, r, s and de-blinds them to obtain a signature.

In the security proof, the hash function H^* is modelled as a random oracle which allows the reduction to swap the roles of $\mathbf{z}$ and $\mathbf{y}$ and answer signing queries using the discrete logarithm of $\mathbf{z}$ as a secret key.

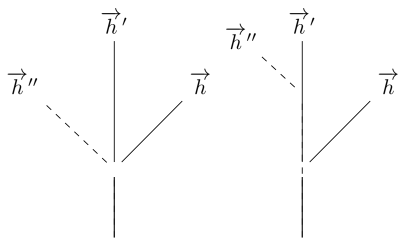
This witness indistinguishability property was used in the original work to provide a proof of security using the rewinding-based forking technique. The proof idea is essentially that the reduction can choose whether the it uses the real secret key to generate signatures and embed a discrete logarithm challenge in the tag key, or embed its challenge in the public key and use the discrete logarithm of the tag key as a secret key. Intuitively, as signatures are non-interactive proofs of knowledge of one of the two possible secret keys, it should be possible to extract one of the secret keys from the adversary through rewinding. Furthermore, as the two modes of operation of the reduction are indistinguishable, the reduction should have a decent chance at extract the secret key that it does not already possess itself.

However, when the reduction rewinds the adversary, the adversary also changes its queries to the reduction in most cases, and thus the reduction has to respond differently than in the first round. The combination of the first and second transcript therefore reveals which witness the reduction used internally and it is not obvious how this affects the signatures that the adversary outputs in the two runs.

The original work deals with this using the following observation: as the adversary needs to make as many hash queries to the random oracle H as it outputs signatures, i.e. more hash queries than it sends


$(\mathbf{I}, \text{rand}, \overrightarrow{h}'')$
side
side
$(\mathbf{I}, \text{rand}, \overrightarrow{h})$
base/partners
$(\mathbf{I}, \text{rand}, \overrightarrow{h}')$

(a) A triangle consists of a pair of partners (the base) and one additional tuple (the top). A pair consisting of the top and one of the base corners is called a side.



(b) Left: forking as in a triangle (solid lines are the base, dashed lines are the top); right: not a triangle (forking at wrong point).

Figure 1.5: Triangles

challenges e to the reduction to close signing sessions. This means that there must be pairs vectors of random oracle responses for each 'instance'[5] that lead to the same challenges by the adversary. For these pairs of random oracle responses, even forking runs do not reveal the witness used by the reduction, and thus the witness extracted from the adversary's signatures is independent of the witness used by the reduction. Following [AO00], we call such pairs of instances, hash responses, along with the random coins of the adversary *partners*.

However, these pairs of partners are hard to find for the reduction, so Abe and Okamoto [AO00] found a way to extend the desirable properties of partnering pairs to more general forking vectors. In particular, they show that there are vectors that fork off from both partners in the same place and at least one of the forks between a partner and the third vector must yield the same secret key as the two partners. We call the two partners the *base* of the triangle, and the third vector the *triangle top*. A depiction of such a forking configuration can be seen in Figure 1.5. The two forks between a partner and the top are called the *sides* of the triangle. The original work now claims that if a large majority of the triangle sides yield a certain secret key, then also a large fraction of the triangle bases would need to yield the same witness. More specifically, they claim that if $\frac{4}{5}$ of triangle sides yield a certain witness, then also $\frac{3}{5}$ of triangle bases would yield the same witness. This argument heavily relies on the observation that if both triangle sides do not yield a specific witness, then also the corresponding base cannot yield that witness.

We observe that this counting argument is flawed in the following way: it works if the triangles do not share components with each other, as then a fraction of more than one half of triangle sides not yielding the desired witness also means that there must be triangles where both sides do not yield said witness. However, if the triangles share components (which is in general the case and also needed for the extension of the desirable properties of bases to be meaningful), this argument does not work any more. Namely, if a triangle side is shared between many triangles, the corresponding opposing sides may all not yield the desired witness, so there is a lot of these, but the single shared side yields the witness, and so the transferral to the base does not happen. We show an example of such a configuration where the sharing of sides of triangles leads to problems in Figure 1.6, middle picture.

Now, one may say we can consider only triangles that do not share sides with each other, however, the sharing of bases cannot be avoided. If triangles were not allowed to share bases, the amount of forking pairs with desirable properties would at most triple from considering bases to considering also

[5]An instance consists of a public key along with the random choices the challenger makes during signing sessions.

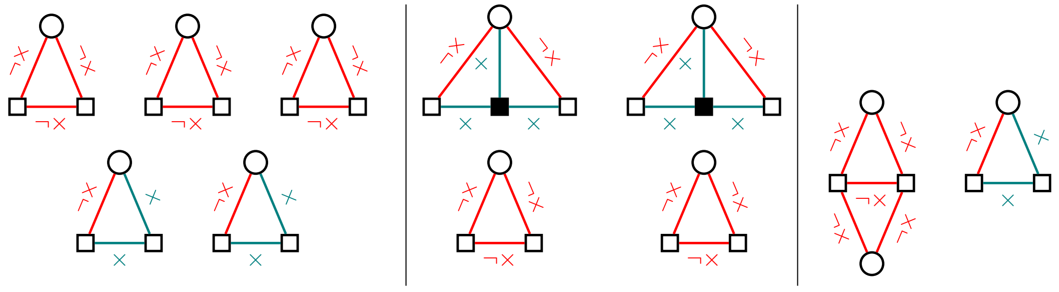


Figure 1.6: Claim in [AO00] that if at least $\frac{4}{5}$ of triangle sides are unsuccessful (i.e., yield the undesirable witness $\neg\times$), then at least $\frac{3}{5}$ of bases (incident to two square nodes) also yield this witness. This holds for non-overlapping triangles (left), but not for triangles overlapping in sides (middle, with $\frac{3}{5}$ yielding the desirable witness $\times$) or in bases (right, with $\frac{1}{2}$ of bases still yielding the desirable witness $\times$).

triangles - as bases are hard to find for the reduction, these nice forking pairs would then also be hard to find.

So, let's consider the case where triangles may share bases. In this case, the counting argument from [AO00] does not work either, as there could be 'heavier' triangle bases that have more corresponding triangle sides. Thus, if a lot of the 'bad' triangle sides accumulate on very few of these 'heavy' triangle bases, only these very few triangle bases would be forced to be bad, leaving plenty of 'lighter' triangle bases that still yield the desired witness but do not have as many triangle sides attached to them. An example of such a scenario is depicted in Figure 1.6 in the right picture.

Resolving the Gap

We rewrite most of the proof using modern techniques to model the forking proof. One key feature is that we use what we call a 'deterministic wrapper' which allows us to talk about all random choices that a reduction would make during the interaction with the adversary at once (in particular these are random coins used for responding to signing queries). We call this collection of responses by the reduction (along with the public key) an 'instance', which is given to the deterministic wrapper along with some random coins for the adversary and a vector of values to be used when responding to hash queries. We can then argue about the interaction of the adversary with the wrapper when the wrapper is running on these inputs. This allows us to define forking pairs, partners, and triangles in the spirit of the proof of [AO00]. In order to show witness indistinguishability for tuples that are partners to each other, we introduce the transcript mapping function Φ. Informally speaking, this function maps tuples of an instance, random coins, and a hash response vector where the instance uses the $\mathbf{y}$ -side (resp. $\mathbf{z}$-side) witness to such tuples where the instance uses the $\mathbf{z}$-side (resp. $\mathbf{y}$-side) witness to such that the two produce the same transcript between the wrapper and adversary. The random coins and hash response vectors are preserved by the mapping. A key property of this mapping is that it preserves partnering tuples, however it does not preserve the triangle top to base corner relationship or other non-partnering forking runs.

This mapping allows us to argue about tuples that have certain properties both before and after applying Φ. In particular, we can show that there exists a large enough set of partner tuples that are base corners of triangles both before and after mapping, and also that there is a large enough set that has *many* triangle tops both before and after mapping. This resolves the second issue we described, i.e. the possibility that there could be bases yielding the 'bad' witness that have many triangle tops and bases

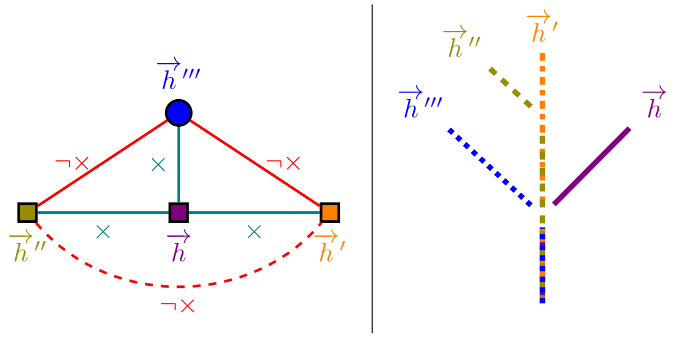


Figure 1.7: Left: In the problematic case where too many triangles share sides, we show that this would result in more triangle bases that yield the bad witness (here depicted as a dashed line). Right: Forking situation that would lead to such a triangle configuration. We bound the number of vectors that behave like $\vec{h}''$.

yielding the 'good' witness that only have few corresponding triangle tops.

We however still need to show that the second potentially bad scenario does not occur. Recall that this scenario revolved around triangles sharing sides, i.e. in particular sharing base corners, and that those shared sides would be the ones yielding the desired witness, whereas the larger set of not shared sides would yield only the undesired witness. We show that the set of non-shared sides cannot be too much larger than the set of shared sides, namely we bound the difference by a factor of roughly ℓ. The strategy to do is involves the following observation: triangle sides can only be shared without producing additional triangles between the non-shared base corners if the non-shared base corners fork from each other at the wrong point. Thus, by considering the forking point where most of the partners fork, we can see that at least a fraction of roughly $\frac{1}{\ell}$ must yield triangles. We depict this in Figure 1.7.

1.3.2 Abe's Blind Signature Scheme

A drawback of the scheme by Abe and Okamoto is that, like blind Schnorr signatures, it is susceptible to the ROS attack whenever the adversary can open $\log q$ many signing sessions in parallel where q is the order of the group. As the group order is roughly 2^λ for a security parameter λ, this means that even a linear or polynomial number of signing sessions would be problematic. This heavily limits the use of the scheme in the real world, as a signer that wants to prevent this attack would have to heavily restrict the amount of open sessions at a time, making themselves susceptible to Denial-of-Service attacks instead. It is also not clear how one would avoid concurrency in a situation where the signer runs multiple servers and signs with the same key on all of them.

Abe [Abe01] introduced a scheme with a similar idea as [AO00], however with a more complex structure in the $\mathbf{z}$-branch of the OR-proof. This scheme evades the ROS attack as each signing session uses a fresh 'session key' in the $\mathbf{z}$-branch, while still linking all session keys to the main key. As the ROS attack heavily relies on linearly combining values from different signing sessions into signatures, it cannot be applied any more. We show a graphic in Figure 1.8 where the $\mathbf{y}$ and $\mathbf{z}$ branches are marked in colour as well as the two sub-branches for the session keys $\mathbf{z}_1$ and $\mathbf{z}_2$. However, the original proof not only carries over the flaws from [AO00], but has a further issue also related to analysing the success probability related to witness indistinguishability. This flaw was first pointed out by [OA03] who gave a proof in the generic group model.

In Chapter 5, we revisit this scheme. We point out that while the scheme was originally designed as a fully blind scheme (using just one fixed 'tag key' $\mathbf{z}$ that is a hash of the rest of the public key), it can easily be extended to a partially blind scheme by using the key part $\mathbf{z}$ as a tag key instead. We provide a full concurrent security proof in the algebraic group model that avoids the problematic rewinding step from the original work.

We further note that the proof technique from Chapter 4 can be applied to Abe's scheme as well (we briefly sketch how to do so in Section 5.4.4), however, it incurs a loss that is superpolynomial in the number of signing sessions. This is inherent in the case of Abe-Okamoto due to the ROS-vulnerability, however in the case of Abe's scheme one can hope for a better proof.

We review the proof strategy for Abe's blind signature scheme in the following.

Technical Overview over Chapter 5

The security proof in Chapter 5 follows the two-step approach of Abe's original work. We first prove that it is infeasible for the adversary to forge a signature using a fresh session key that was not issued by the signer. This implies that the adversary must output two signatures that use the same (blinded) session key. However, in this case a witness can be extracted with a high probability, as the adversary would have to solve an information-theoretically hard variant of the ROS problem to output two such signatures without revealing a witness.

The proof uses the assumption that the adversary is *algebraic*. This means that whenever the adversary outputs a group element, be it as part of a signing query, in a signature, or to a random oracle, it must also submit an explanation of how this group element can be computed from the input group elements it has seen so far (which may come from the public key or various oracle responses).

This explanation can then be used by the reduction to avoid the rewinding step and compute one of the secret keys needed for signing messages only from the explanation along with the signature.

However, as the reduction will again employ the witness indistinguishability to simulate with different possible secret keys, we still need to argue in both halves of the proof that it is not possible for the adversary to reliably output a representation that will yield the same witness that the reduction already has. This argument is different than the one used in Chapter 4, as here there is no rewinding step.

While in the rewinding based setting, the two forking runs together reveal the witness internally used by the reduction, the single run by the reduction in the AGM does not reveal anything about the witness the reduction has internally.

However, the adversary may submit representations involving group elements whose internal representation is not fully known at the time of the hash query. This could allow the adversary to use the signing oracle to retrospectively set the representation in such a way that it matches that obtained from the signature. We prove a claim that states that this is not possible using an exhaustive case distinction over different types of representations the adversary could submit.

We note that this case distinction is greatly simplified in comparison to that of [KLX22a] as we employ a lemma from [KLR23a] to catch several (sub-)cases at once.

1.3.3 Sequential Security of Blind Schnorr Signatures in the AGM

In Chapter 6, we consider the security of Blind Schnorr Signatures in the AGM + ROM. Our results are two-fold. On the one hand, we prove that the scheme can be proven sequentially secure assuming the hardness of the one-more discrete logarithm problem (OMDL). The proof strategy is to embed discrete

$$
\begin{array}{lcr}
\textbf{Signer} & & \textbf{User} \\
\mathsf{sk} = x & & \\
\mathsf{pk} = (\mathbf{g}, \mathbf{h}, \mathbf{y} = \mathbf{g}^x) & & \mathsf{pk} = (\mathbf{g}, \mathbf{h}, \mathbf{y}) \\
\mathsf{info} & & m, \mathsf{info} \\
\mathbf{z} \leftarrow H_1(\mathsf{pk}, \mathsf{info}) & & \mathbf{z} \leftarrow H_1(\mathsf{pk}, \mathsf{info}) \\
\hline
u, d, s_1, s_2 \overset{\$}{\leftarrow} \mathbb{Z}_q & & \\
\mathrm{rnd} \overset{\$}{\leftarrow} \{0,1\}^\lambda & & \\
\mathbf{z}_1 \leftarrow H_2(\mathrm{rnd}),\ \mathbf{z}_2 \leftarrow \mathbf{z}/\mathbf{z}_1 & & \\
\mathbf{a} \leftarrow \mathbf{g}^u & & \\
\mathbf{b_1} \leftarrow \mathbf{g}^{s_1} \cdot \mathbf{z}_1^d & & \\
\mathbf{b_2} \leftarrow \mathbf{h}^{s_2} \cdot \mathbf{z}_2^d & \xrightarrow{\mathbf{a}, \mathbf{b_1}, \mathbf{b_2}, \mathrm{rnd}} & \tau, \gamma, t_1, t_2, t_3, t_4, t_5 \overset{\$}{\leftarrow} \mathbb{Z}_q \\
& & \mathbf{z}_1 \leftarrow H_2(\mathrm{rnd}) \\
& & \alpha \leftarrow \mathbf{a} \cdot \mathbf{g}^{t_1} \cdot \mathbf{y}^{t_2} \\
& & \zeta \leftarrow \mathbf{z}^\gamma,\ \zeta_1 \leftarrow \mathbf{z}_1^\gamma,\ \zeta_2 \leftarrow \zeta/\zeta_1 \\
& & \beta_1 \leftarrow \mathbf{b}_1^\gamma \cdot \mathbf{g}^{t_3} \cdot \zeta_1^{t_4} \\
& & \beta_2 \leftarrow \mathbf{b}_2^\gamma \cdot \mathbf{h}^{t_5} \cdot \zeta_2^{t_4} \\
& & \eta \leftarrow \mathbf{z}^\tau \\
& & h \leftarrow H_3(\zeta, \zeta_1, \alpha, \beta_1, \beta_2, \eta, m, \mathsf{info}) \\
& \xleftarrow{e} & e \leftarrow h - t_2 - t_4 \\
c \leftarrow e - d & & \\
r \leftarrow u - c \cdot x & & \\
& \xrightarrow{c, r, d, s_1, s_2} & \rho \leftarrow r + t_1,\ \omega \leftarrow c + t_2 \\
& & \sigma_1 \leftarrow \gamma \cdot s_1 + t_3 \\
& & \sigma_2 \leftarrow \gamma \cdot s_2 + t_5 \\
& & \delta \leftarrow d + t_4 \\
& & \mu \leftarrow \tau - \delta \cdot \gamma \\
& & \delta + \omega \overset{?}{=} H_3(\zeta, \zeta_1, \mathbf{g}^\rho \mathbf{y}^\omega, \mathbf{g}^{\sigma_1} \zeta_1^\delta, \mathbf{h}^{\sigma_2} \zeta_2^\delta, \mathbf{z}^\mu \zeta^\delta, m, \mathsf{info}) \\
& & \Downarrow \\
& & (m, (\zeta, \zeta_1, \rho, \omega, \sigma_1, \sigma_2, \delta, \mu))
\end{array}
$$

Figure 1.8: Interaction between Signer and User for the partially blind version of BSA. Components related to proving knowledge of the discrete logarithm of $\mathbf{z}_1$ resp. ζ_1 to $\mathbf{g}$ are marked in orange, components related to proving knowledge of the discrete logarithm of $\mathbf{z}_2$ resp. ζ_2 to $\mathbf{h}$ are marked in teal, components connecting the two or related to proving knowledge of the discrete logarithm of ζ to $\mathbf{z}$ are marked in green, and components related to proving knowledge of $\mathbf{y}$ to $\mathbf{g}$ are marked in purple.

logarithm challenges in the public key as well as in the messages sent as response to Sign_1 queries. The reduction then uses the discrete logarithm oracle to respond to Sign_2 queries without having to know the secret key. It finally solves the discrete logarithm challenges by solving for the secret key using the representations submitted in hash queries along with the signatures. Once it has the secret key, it can combine it with the values for c and s it learned during its own Sign_2 responses to solve all the other challenges. This strategy uses as many discrete logarithm queries as the user closes signing sessions, but due to the sequentiality, it avoids the ROS-problem that appears in [FPS20] when they prove security of Blind Schnorr Signatures in the concurrent setting.

We then turn to the second result, namely we prove that our reduction is optimal with respect to the amount of discrete logarithm queries made. In particular, we employ a meta-reduction against an algebraic reduction that simulates an algebraic adversary to the reduction and solves the OMDL problem for $\ell - 1$ discrete logarithm queries and ℓ challenges whenever the reduction solves the problem.

The meta-reduction's strategy is to provoke the reduction to output a linearly independent system of linear equations through the algebraic explanations provided for the public key and first signer messages. It then uses this system of equations to either solve for the secret key directly or to solve the discrete logarithm challenges contained in the public key to obtain the secret key. The meta-reduction can then sign arbitrary messages to submit to the reduction, which in turn breaks the OMDL assumption.

1.4 The Algebraic Group Model and the Algebraic Wrapper

As already sketched in a previous section of this introduction, in the Algebraic Group Model [FKL18], one considers algebraic adversaries. More formally, let $\mathbf{x}_1, \ldots, \mathbf{x}_n$, be the group elements that the adversary has seen so far, and let $\mathbf{y}$ be a group element in the output of the adversary. Then, the adversary is required to additionally output a vector $\overrightarrow{z} \in \mathbb{Z}_q^n$ such that

$$\mathbf{y} = \prod_{i=1}^{n} \mathbf{x}_i^{z_i}.$$

Since its introduction in [FKL18], the AGM has found applications proving security in many areas of cryptography, such as signatures [FKL18; TZ23; FPS20], blind signatures [TZ22; FPS20; KLX22a; Cri+23], zero-knowledge proofs [FKL18], as well as more general cryptographic assumptions [BFL20] from the Uber-Framework [BBG05; Boy08].

Furthermore, researchers have proposed extensions to the model, such as a decisional variant where the adversary has to 'explain' its decision bit [RS20], a variant that allows for counting the number of group operations made by the adversary to arrive at the group element [KLX20], as well as a variant where the adversary is allowed to sample group elements without knowledge of an algebraic representation [LPS23].

In recent years however, the AGM has received some criticism. On the one hand, [ZZK22] show that the AGM is in fact incomparable to the variant of the Generic Group Model introduced by Shoup [Sho97], i.e. there are examples of protocols that are secure in one model but not the other. Their proof technique relies on 'feeding' an adversary the label of a group element that is a solution to the problem. In the GGM, the adversary can thus easily win the game by outputting said group element. In the AGM however, the adversary would be required to also explain how it computed the group element from its inputs which would reveal a discrete logarithm relation.

In another recent work, Zhandry [Zha22] compared the two variants of the GGM by Shoup [Sho97] and by Maurer [Mau05] and showed that in the case of Maurer's GGM variant, it indeed holds that any

$$\left(\underbrace{\mathbf{x}}_{\substack{\text{Group Element}\\ \text{from } \mathbb{G}}}, \underbrace{c}_{\substack{\text{Encrypted}\\ \text{Algebraic Explanation}}}, \underbrace{\pi}_{\substack{\text{Proof of Consistency}\\ \text{between } \mathbf{x} \text{ and } c}}\right)$$

Figure 1.9: Structure of a group element in the algebraic wrapper $\mathbb{H}$.

generic algorithm is also an algebraic algorithm, while at the same time he proves that the full AGM is uninstantiable.

1.4.1 The Algebraic Wrapper as an Approximation of the AGM

In [AHK20] (we review this in Chapter 7), we showed that, despite the uninstantiability of a full AGM, it is possible to do a partial instantiation based on falsifiable assumptions. In this partial instantiation, called the *algebraic wrapper*, a reduction would attack a problem in a base group, but the adversary would attack a different problem in a constructed group that allows the reduction to extract a restricted algebraic representation. The algebraic wrapper works as follows: During setup, the reduction can choose a basis of group elements from the base group that will be used as a basis for the explanations that the adversary submits. Group elements are then represented as an element from the base group with some auxiliary information attached. The auxiliary information is essentially an encryption of the algebraic explanation with respect to the basis contained in the group parameters along with a NIZK proof that the explanation matches the group element from the base group. That is, a group element of the algebraic wrapper looks as depicted in Figure 1.9.

In order to compute the group operation, one needs to do the following three things:

1. multiply the two base group elements - this can easily be done using the base group operation
2. provide an encrypted explanation. This can be done for example through homomorphism of the encryption scheme which allows to add the two previous explanation vectors together
3. provide a NIZK proof of correctness. This is the trickiest part as one would need the witnesses of the encryption scheme.

We resolve the difficulty of the last steps by relying on indistinguishability obfuscation (iO). This is a primitive that allows to *obfuscate circuits* in such a way that if one has two circuits that compute the same function, one can input one of these circuits into the obfuscator to obtain an obfuscated circuit that computes the same function. The security guarantee states that an adversary won't be able to tell which of the original circuits was the input to the obfuscation.

In the algebraic wrapper, the group parameters contain an obfuscated circuit that allows users of the group to compute the group operation, including recomputing the proof of consistency for new group elements.

For technical reasons, we also need to include a 'trapdoor' that allows a reduction to output an obfuscated circuit that computes the group operation even when the secret key of the encryption scheme is unknown. This trapdoor can be 'activated' in some hybrids to allow for efficient implementations of the addition function even when it does not have access to all the information required to do so.

This leads to the following group parameters

- the group parameters $\mathsf{pp}_{\mathbb{G}}$ of the underlying base group
- a base for the algebraic explanations (as group elements from $\mathbb{G}$)
- a public key pk of a homomorphic public key encryption scheme
- a CRS for the NIZK
- the NIZK generation trapdoor (in standard mode deactivated)
- an obfuscated circuit for the group operation
- an obfuscated circuit for re-randomizing group element encodings

We note that the proof of consistency can be created using different types of witnesses. On the one hand, one can generate a proof using the encryption randomness and the plaintext (which is a valid explanation), whereas on the other hand, one can use the secret key as a witness to show that the ciphertext in question decrypts to a valid explanation. Alternatively, the proof of consistency can also be generated using the trapdoor witness. In the security proof, we can use the indistinguishability property to switch between a circuit that uses the public key and randomness and one that uses the secret key. This allows us to 'forget' about the secret key of the encryption scheme and use the IND-CPA security of the encryption to switch the algebraic representations of the group elements output by the challenger. We call this property of the algebraic wrapper *k-switching*.

In the standard application of the wrapper, the base will be some random group elements, and only the first entry is used by the challenger for its output group elements. A reduction using the algebraic wrapper, however, will put its own challenge elements (and possibly some related group elements) into the basis so that it can output related group elements in the algebraic wrapper without actually knowing their discrete logarithm.

This allows us to transfer some proofs from the AGM into the setting of the algebraic wrapper. We give an overview over these proof strategies in the following.

1.4.2 Transferring Proofs from the AGM to the Algebraic Wrapper

We show how to transfer the proofs of various Diffie-Hellman-type assumptions based on DL like in [FKL18]. To this end, we come up with some base elements for the Diffie-Hellan assumption in question, namely there is a base element for each of the random variables of the assumption, i.e. for CDH, there is a base element that corresponds to the generator $\mathbf{g}$, one for $\mathbf{g}^x$, and one for $\mathbf{g}^y$. This allows the reduction to embed its own DL challenge in either of those base elements (after application of the appropriate switching lemmata). These proofs are mostly a proof-of-concept to get the reader familiar with the techniques. In particular, for the Diffie-Hellman assumptions we already employ what we call the *symmetrization technique* where we add group elements to the basis of the wrapper to hide from the adversary where the discrete-logarithm challenge is embedded. If we only added one group element fro $\mathbf{g}^x$ (in addition to the one for $\mathbf{g}$), an adversary could always present the reduction with a solution that only allowed to solve for the discrete logarithm of $\mathbf{g}^y$, but never for the discrete logarithm of $\mathbf{g}^x$ which is what our reduction is interested in. However, when there are two equivalent base elements, the embedding choice of the reduction remains information-theoretically hidden from the adversary.

We then turn to proofs for Schnorr Signatures as well as Signed ElGamal - the underlying AGM proof strategy is based on [FPS20]. As the EUF-CMA game for Schnorr signatures is interactive, we have to take care of how to simulate Random Oracle Queries as well as Signing queries. However, we consider a special type of Random Oracle that ignores parts of its input, namely auxiliary information provided in the group elements. As each group element has a unique identifier, we base the hash function outputs solely on this unique identifier while observing the auxiliary information submitted. Another thing we need to concern ourselves with is how the reduction responds to signing queries. As usual for Schnorr signatures, the reduction will sample values c, s, and construct the group element as $\widehat{R} = \mathbf{g}^s \cdot \mathsf{pk}^{-c}$. However, in the case of the algebraic wrapper, this alternate method of generating group elements may be noticeable to an adversary if we are not careful. We want to apply the k-switching property here, however applying it to all signatures would yield a non-tight reduction whereas the reduction in the AGM is tight. So, we add some group elements from the algebraic wrapper that we call *origin elements*. These elements are used by the games and reduction internally to derive all group elements in the public key as well as in the oracle responses. As there is only a constant number of origin elements, doing a k-switching argument over those elements only leads to a constant loss.

1.4.3 Limitations of the Algebraic Wrapper

It is a natural question to ask whether the algebraic wrapper can also be used to transfer other proofs, such as those presented in Chapters 5 and 6 or other proofs from [FKL18; BFL20]. There are two key limiting factors to the transferrall of proof techniques. The first is whether the symmetrization technique is applicable. This is for example not the case for our proof of one-more unforgeability of Abe's blind signature scheme, as the reduction relies heavily on the witness indistinguishability of the scheme. In the AGM, this witness indistinguishability is perfect as the group elements do not reveal any information regarding how they were computed. However, in the algebraic wrapper, the reduction just like the adversary, has to output wrapper group elements which contain information about how they were computed. Due to the interactivity of the one-more unforgeability game, it is not possible for the reduction to send first-round elements that already contain the correct representation or match the internal representation to the 'external' representation provided in the second signer response. Thus, while the internal representation is hidden computationally, an information-theoretical argument is no longer possible and it is unclear how to replace some of the information-theoretical arguments in the proof of Abe's scheme.

On the other hand, some proofs in the AGM rely on non-constant size assumptions, such as q-type assumptions (where the adversary gets q group elements where q is polynomial) or interactive assumptions such as the one-more discrete logarithm assumption. Examples of such proofs are proofs of sequential (see Chapter 6, [KLX22a]) or concurrent (see [FPS20]) security of blind Schnorr signatures where security is based on the one-more discrete logarithm assumption.

The difficulty with using such non-constant sized assumptions in the base group of the algebraic wrapper is that as the reduction w.l.o.g. does not know the discrete logarithms of any of the challenge group elements of the assumption, it would have to embed any challenge element it ever intends to send to the adversary in the basis of the algebraic wrapper. Such an embedding would lead to a polynomial blow-up (instead of a mere constant blow-up of the size of the wrapper group elements in comparison to the base group elements). Even worse (as our construction is of a theoretical and proof-of-concept nature), such an embedding of a large number of group elements would create a cyclic dependency between the adversary's input and potential upper bounds on the adversary's query complexity (say for a non-interactive assumption the adversary is bounded to make at most q queries with q a polynomial in its input size, and so the reduction attacks an assumption of size q - now if the input size gets larger by a factor of q, the adversary might make more than q queries).

1.5 Outline of the Thesis

In Chapter 2 we recall important definitions and lemmata. Chapter 3 is based on [HKK23] and we discuss the proofs of adaptive security for the Goldreich-Goldwasser-Micali PRF as a prefix-constrained PRF based on the Pseudorandomness of the Underlying PRG as well as the adaptive security of the Logical Key Hierarchy based on the IND-CPA security of the underlying secret key encryption scheme. In Chapter 4, we present the full proof of One-More Unforgeability of the Abe-Okamoto partially blind signature scheme based on the discrete logarithm problem in the ROM. This chapter is based on [KLX22c]. In Chapter 5, we present the full proof of One-More Unforgeability of Abe's blind signature scheme, also discussing how to extend it to the partially blind setting. This proof is slightly different from the one originally presented in [KLX22a] as we found a way to apply a lemma from a later work [KLR23a] to simplify some steps. The proof is in the AGM+ROM and shows tight security under the discrete logarithm assumption. In Chapter 6, we discuss the sequential security of blind Schnorr signatures in the AGM+ROM under the one-more discrete logarithm assumption. This chapter is also based on work presented originally in [KLX22a]. In Chapter 7 we present the algebraic wrapper, a way to partially instantiate the AGM, and discuss how to apply it to some existing proofs that use the AGM. This chapter is based on [AHK20]. We conclude in Chapter 8.

Chapter 2

Preliminaries

2.1 Notation

We denote by $[\ell] := \{1, \ldots, \ell\}$ and by $[\ell]_0 := \{0, 1, \ldots, \ell\}$ for a natural number $\ell \in \mathbb{N}$. For a vector $\overrightarrow{h}$, its i-th entry is denoted by h_i, and the vector of its first i entries is denoted by $\overrightarrow{h}_{[i]}$. Given a finite set S, the notation $x \xleftarrow{\$} S$ means a uniformly random assignment of an element of S to the variable x. For a vector $\overrightarrow{x} \in X^n$, we denote by $\overrightarrow{x}' \xleftarrow{\$} X^n_{|\overrightarrow{x}_{[i]}}$ that $\overrightarrow{x}'$ is sampled uniformly at random from $\{\overrightarrow{x}' \in X^n | \overrightarrow{x}'_{[i]} = \overrightarrow{x}_{[i]}\}$. For an algorithm A, we use t_{A} to denote its running time.

Throughout this document λ denotes the security parameter.

A function negl: $\mathbb{N} \to \mathbb{R}$ is *negligible* in λ if for every constant $c \in \mathbb{N}$, there exists a bound $n_c \in \mathbb{R}$, such that for all $n \geq n_c$, $|\mathsf{negl}(n)| \leq n^{-c}$.

Given an algorithm A, the notation $y \xleftarrow{\$} A(x)$ means evaluation of A on input of x with fresh random coins and assignment to the variable y. The notation $\mathsf{A}^{\mathcal{O}}$ indicates that the algorithm A is given oracle access to $\mathcal{O}$. Given a random variable B, supp(B) denotes the support of B.

Let $\mathbb{G}$ be a finite cyclic group with generator $\mathbf{g}$ and order q. For $x \in \mathbb{Z}_q$, the notation $[x]_{\mathbb{G}}$ denotes the group element $\mathbf{g}^x$. Note that using this notation does not imply knowledge of x. Let $\mathbb{K}$ be a field and V be a vector space over $\mathbb{K}$ of finite dimension n. For $i \in [n]$, $\overrightarrow{e_i}$ denotes the vector which carries 1 in its i-th entry and 0 in all other entries.

In game based proofs, out_i denotes the output of game G_i. Further, we will use this notation to highlight differences to previous hybrids.

2.1.1 Sets and Bitstrings.

For two sets $\mathcal{X}$, $\mathcal{Y}$ we denote the symmetric difference between $\mathcal{X}$ and $\mathcal{Y}$ as $\mathcal{X} \Delta \mathcal{Y} := (\mathcal{X} \setminus \mathcal{Y}) \cup (\mathcal{Y} \setminus \mathcal{X})$. With $\{0,1\}^n$, $\{0,1\}^{\leq n}$, and $\{0,1\}^{<n}$, we mean all bitstrings of length exactly n, at most n, and less than n, respectively. The lexicographic ordering upon bitstrings x is denoted with $\leq_{\mathsf{lex}}$. If x is a prefix of x', we write $x \leq_{\mathsf{pfx}} x'$, for a proper pefix we write $x <_{\mathsf{pfx}} x'$. For a finite vector $x = (x_1, \ldots, x_n) \in \Sigma^n$ over an alphabet Σ, we denote by $\mathsf{pfx}_j(x)$ the prefix $(x_1, \ldots, x_j)$ of x. The symbol $\|$ denotes string or sequence concatenation.

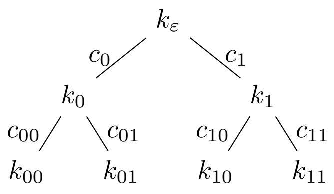


Figure 2.1: A depth-2 binary tree with node and edge names.

2.1.2 Tree Notation.

For our applications, we will consider complete binary trees whose depth we generally denote by d. We derive generic names for nodes and edges from our applications: concretely, we denote the root node as k_ε (where ε is the empty bitstring), and the two child nodes of each node k_x as $k_{x\|0}$ and $k_{x\|1}$. For each $x \in \{0,1\}^{<d}$ and $b \in \{0,1\}$, there is an edge $c_{x\|b}$ between k_x and $k_{x\|b}$. (See Figure 2.1 for an example with $d = 2$.) For a binary tree of depth d and a path $P = (k_x, \ldots, k_\varepsilon)$ from a leaf $x \in \{0,1\}^d$ to the root, the *co-path* of P consists of the sibling vertices of the vertices on P. More formally, writing $x = (x_1, \ldots, x_d)$, the co-path consists of the vertices $(k_{\mathsf{pfx}_{d-1}(x)\|(1-x_d)}, k_{\mathsf{pfx}_{d-2}(x)\|(1-x_{d-1})}, \ldots, k_{1-x_1})$.

2.1.3 Probabilities, Distributions, and Predicates.

If $\mathcal{D}$ is a distribution over some set $\mathcal{X}$, then

$$\rho_{\mathcal{D}}(x) := \Pr_{X \leftarrow \mathcal{D}}[X = x].$$

Furthermore, if $f : \mathcal{X} \to \mathcal{Y}$ is a function, then $f(\mathcal{D})$ denotes the distribution over $\mathcal{Y}$ that arises by applying f to values sampled from $\mathcal{D}$. If $\mathrm{P} : \mathcal{X} \to \{\mathtt{true}, \mathtt{false}\}$ is a predicate, then $\mathcal{D} \mid \mathrm{P}$ denotes the conditional distribution of $\mathcal{D}$ conditioned on $\mathrm{P}(\cdot) = \mathtt{true}$. As a special case, we consider equalities as predicates P in the above sense, and may write, e.g., $\mathcal{D} \mid [f(\cdot) = y]$.

For two random variables X, Y (which may depend on the security parameter λ), we write $X \equiv Y$ if they are identically distributed, $X \overset{\mathrm{s}}{\approx}_\delta Y$ if their statistical distance is at most δ, and $X \overset{\mathrm{c}}{\approx} Y$ if they are computationally indistinguishable.

2.2 Security Games

We use the standard notion of (prose-based) *security games* [BR04; Sho04] to present our proofs. We denote the binary output of a game $\mathbf{G}$ with an adversary A as $\mathbf{G}^{\mathsf{A}}$ and say that A *wins* $\mathbf{G}$ if $\mathbf{G}^{\mathsf{A}} = 1$.

2.3 Computational Problems

2.3.1 Subset Membership Problem

Let $\mathcal{L} = (\mathcal{L}_\lambda)_{\lambda \in \mathbb{N}}$ be a family of families of languages $L \subseteq X_\lambda$ in a universe $X_\lambda = X$. Further, let R be an efficiently computable witness relation, such that $x \in L$ if and only if there exists a witness

$w \in \{0,1\}^{\mathsf{poly}(|x|)}$ with $R(x,w) = 1$ (for a fixed polynomial poly). We assume that we are able to efficiently and uniformly sample elements from L together with a corresponding witness, and that we are able to efficiently and uniformly sample elements from $X \setminus L$.

Definition 2.3.1 (Subset membership problem, [CS02]). *A* subset membership problem $L \subseteq X$ *is hard, if for any PPT adversary* A*, the advantage*

$$\mathrm{Adv}^{\mathtt{smp}}_{L,\mathsf{A}}\lambda := \Pr[x \leftarrow L\colon \mathsf{A}(1^\lambda, x) = 1] - \Pr[x \leftarrow X \setminus L\colon \mathsf{A}(1^\lambda, x) = 1]$$

is negligible in λ.

We additionally require that for every L and every $x \in L$, there exists exactly one witness $r \in \{0,1\}^*$ with $R(x,w) = 1$. Note that given a cyclic group $\mathbb{G}$ of prime order p in which DDH is assumed to hold, the Diffie-Hellman language $L_{[(1,x)]_\mathbb{G}} := \{[(y, xy)]_\mathbb{G} \mid y \in \mathbb{Z}_p\}$ (for randomly chosen generators $[1]_\mathbb{G}, [x]_\mathbb{G}$) satisfies this definition. Another instantiation of Definition 2.3.1 is the language containing all commitments to a fixed value using a perfectly binding commitment scheme with unique opening.

2.3.2 Problems in Groups

This definition is derived from the definition of a non-interactive computational problem in [FJS19].

Definition 2.3.2 (Non-interactive computational problems in groups). *Let* $\mathbb{G}$ *be a cyclic group with group generation algorithm* Setup*. We say that* $\mathcal{P} = (\mathcal{G}, \mathcal{C})$ *is a* non-interactive computational problem *in* $\mathbb{G}$ *if the two procedures* $\mathcal{G}$ *and* $\mathcal{C}$ *are of the following syntax:*

Setup. *Takes the group parameters as input and outputs a problem instance (the challenge)* $\mathcal{I} = (C_1, \ldots C_u, C') \in \mathbb{G}^u \times \{0,1\}^*$ *and a state* st

Output Determination *Takes as input the group parameters, the state from above, the problem instance* $\mathcal{I}$ *and an attempt at a solution* $\mathcal{S} = (S_1, \ldots S_w, S') \in \mathbb{G}^w \times \{0,1\}^*$*. It is possible that* $w = 0$*.* $\mathcal{C}$ *outputs* 1 *if the attempted solution* $\mathcal{S}$ *is considered a* correct *solution to* $\mathcal{I}$ *and* 0 *if it is not a correct solution.*

An example of a computational problem is the discrete logarithm problem $\mathbf{DLOG}$ (see Definition 2.3.4). We will later also consider some *interactive* computational problems that come with an online-phase that allows the adversary access to some oracles. We will explain the winning conditions specifically to those problems.

Definition 2.3.3 (Hardness of a non-interactive computational problem). *We say that a non-interactive computational problem* $\mathcal{P} = (\mathcal{G}, \mathcal{C})$ *is* hard with respect to a group generator algorithm Setup *if for all PPT adversaries* A, $\mathrm{Adv}^{\mathcal{P}}_{\mathsf{Setup},\mathsf{A}}\lambda := \Pr[\mathrm{Exp}^{\mathcal{P}}_{\mathsf{Setup},\mathsf{A}}(\lambda) = 1]$ *is negligible.*

```
$\underline{\mathrm{Exp}^{\mathcal{P}}_{\mathsf{Setup},\mathsf{A}}(\lambda)}$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$(\mathsf{st}, \mathcal{I}) \leftarrow \mathcal{G}(\mathsf{pp})$
$\mathcal{S} \leftarrow \mathsf{A}(1^\lambda, \mathcal{I})$
$\mathbf{return}\ \mathcal{C}(\mathsf{pp}, \mathsf{st}, \mathcal{S}, \mathcal{I})$
```

Definition 2.3.4 (Discrete Logarithm Problem). *For public parameters* $\mathsf{pp} = (\mathbb{G}, q, \mathbf{g})$ *for a group* $\mathbb{G}$ *with order* q *and generator* $\mathbf{g}$*, we describe the discrete logarithm game* $\mathbf{DLOG}_\mathbb{G}$ *with adversary* A *as follows:*

Setup. *Sample $x \xleftarrow{\$} \mathbb{Z}_q$ and set $\mathbf{y} := \mathbf{g}^x$. Output $(\mathsf{pp}, \mathbf{y})$ to* A*.*

Output Determination. *When* A *outputs $x' \in \mathbb{Z}_q$, return 1 if $g^{x'} = \mathbf{y}$ and 0 otherwise.*

We define the advantage of A *as*

$$\mathrm{Adv}_{\mathsf{A}}^{\mathbf{DLOG}_{\mathbb{G}}} := \Pr[\mathbf{DLOG}_{\mathbb{G}}^{\mathsf{A}} = 1]$$

where the probability goes over the randomness of the game as well as the randomness of the adversary A*. We say that the discrete logarithm problem is (t, ϵ)-*hard *in $\mathbb{G}$ if for any adversary* A *that runs in time at most t, it holds that*

$$\mathrm{Adv}_{\mathsf{A}}^{\mathbf{DLOG}_{\mathbb{G}}} \leq \epsilon.$$

(When it is clear from context, we may omit $\mathbb{G}$ and only write $\mathbf{DLOG}$ for the game.)

Definition 2.3.5 (One-More-Discrete Logarithm Problem (OMDL))**.** *For a stateful algorithm* A *and a positive integer ℓ, we define the game ℓ-$\mathbf{OMDL}$ as follows:*

Setup. *Initialize $C = \emptyset$. Run* A *on input $\mathbf{g}$.*

Online Phase. A *is given access to the following oracles:*

> **Oracle** chal *takes no input and samples a group element $\mathbf{y} \xleftarrow{\$} \mathbb{G}$. It sets $C \leftarrow C \cup \{\mathbf{y}\}$ and returns $\mathbf{y}$.*
>
> **Oracle** dlog *takes as input a group element $\mathbf{y}$. It returns $\mathrm{dlog}_{\mathbf{g}}\,\mathbf{y}$. We assume that* dlog *can be queried at most ℓ many times.*

Output Determination. *When* A *outputs $(\mathbf{y}_i, x_i)_{i=1}^{\ell+1}$, return 1 if for all $i \in [\ell+1]$: $\mathbf{y}_i \in C$, $\mathbf{g}^{x_i} = \mathbf{y}_i$, and $y_i \neq y_j$ for all $j \neq i$. Otherwise, return 0.*

We define the advantage of A *in ℓ-$\mathbf{OMDL}$ as*

$$\mathrm{Adv}_{\mathsf{A},\ell}^{\mathbf{OMDL}} := \Pr\left[\ell\text{-}\mathbf{OMDL}^{\mathsf{A}} = 1\right].$$

Definition 2.3.6 (Decsional Diffie-Hellman Problem (DDH))**.** *For an algorithm* A *we define the game $\mathbf{DDH}$ as follows:*

Setup. *Sample $x, y, z \xleftarrow{\$} \mathbb{Z}_q$ and $b \xleftarrow{\$} \{0,1\}$. Run* A *on input $(\mathbf{g}, \mathbf{g}^x, \mathbf{g}^y, \mathbf{g}^{xy+bz})$*

Output Determination. *When* A *outputs b', return 1 if $b = b'$ and 0 otherwise.*

We define the advantage of A *in $\mathbf{DDH}$ as*

$$\mathrm{Adv}_{\mathsf{A}}^{\mathbf{DDH}} := \left|\Pr[\mathbf{DDH}^{\mathsf{A}} = 1] - \frac{1}{2}\right|.$$

2.4 (Partially) Blind Signatures

The definitions in this section are taken from [KLX22c; KLX22a] and mostly follow [AO00].

Definition 2.4.1 (Partially Blind Signature scheme). *A three-move* partially blind signature scheme $\mathsf{PBS} = (\mathsf{KeyGen}, \mathsf{Sign} = (\mathsf{Sign}_1, \mathsf{Sign}_2), \mathsf{User} = (\mathsf{User}_1, \mathsf{User}_2), \mathsf{Verify})$ *consists of the following* PPT *algorithms:*

Key Generation. *On input public parameters* pp*, the probabilistic algorithm* KeyGen *outputs a public key* pk *and a secret key* sk*. Henceforth we assume that* pp *is provided to all parties (including the adversary) as an input, and do not explicitly write it.*

Signer: *The interactive signer* $\mathsf{Sign} = (\mathsf{Sign}_1, \mathsf{Sign}_2)$ *has two phases:*

Sign_1**:** *On input a tag* info *and a secret key* sk*, the probabilistic algorithm* Sign_1 *outputs an internal signer state* $\mathsf{st}_{\mathsf{Sign}}$*, and a response* R*.*

Sign_2**:** *On input the secret key* sk*, a challenge value* e*, and the corresponding internal state* $\mathsf{st}_{\mathsf{Sign}}$*, the deterministic algorithm* Sign_2 *outputs a response* S*.*

User. *The interactive user* $\mathsf{User} = (\mathsf{User}_1, \mathsf{User}_2)$ *has two phases:*

User_1**:** *On input a public key* pk*, a tag* info*, a message* m*, and a* Sign_1 *response* R*, the probabilistic algorithm* User_1 *outputs a challenge value* e *and an internal user state* $\mathsf{st}_{\mathsf{User}}$*.*

User_2**:** *On input a public key* pk*, a* Sign_2 *response* S*, and the corresponding internal user state* $\mathsf{st}_{\mathsf{User}}$*, the deterministic algorithm* User_2 *outputs a signature* sig *on message* m *along with the tag* info*.*

Verification. *On input a public key* pk*, a message* m*, a signature* sig*, and a tag* info*, the deterministic algorithm* Verify *outputs either* 1 *or* 0*, where* 1 *indicates that the signature is valid, and* 0 *that it is not.*

We say a partially blind signature scheme PBS *is* (perfectly) correct *if for all* $\mathsf{pk}, m, \mathsf{sig}, \mathsf{info}$ *that result from an honest interaction between signer and user,* $\mathsf{Verify}(\mathsf{pk}, m, \mathsf{sig}, \mathsf{info}) = 1$*.*

We now define the one-more-unforgeability of a partially blind signature scheme. We do not focus on partial blindness in this paper; we include the definition for completeness, and for a proof that the Abe-Okamoto scheme is partially blind, see the original paper [AO00].

Definition 2.4.2 (ℓ-(Sequential-)One-More-Unforgeability (ℓ-(SEQ-)OMUF)). *For a stateful algorithm* A*, a three-move partially blind signature scheme* BS*, and a positive integer* ℓ*, we define the game* ℓ-$\mathbf{OMUF}_{\mathsf{BS}}$ *(*ℓ-$\mathbf{SEQ}$-$\mathbf{OMUF}_{\mathsf{BS}}$*) as follows:*

Setup. *Sample* $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{BS.KeyGen}(\mathsf{pp})$ *and run* A *on input* $(\mathsf{pk}, \mathsf{pp})$*.*

Online Phase. A *is given access to the oracles* sign_1 *and* sign_2 *that behave as follows.*

Oracle sign_1**:** *On input* info*, it samples a fresh session identifier* $\mathtt{id}$ *(If sequential, it checks if* $\mathtt{session}_{\mathtt{id}-1} = \mathtt{open}$ *and returns* $\perp$ *if yes). If* info *has not been requested before, it initializes a counter* $\ell_{\mathsf{closed},\mathsf{info}} := 0$*. It sets* $\mathtt{session}_{\mathtt{id}} \leftarrow \mathtt{open}$ *and generates* $(C_{\mathtt{id}}, \mathsf{st}_{\mathtt{id}}) \xleftarrow{\$} \mathsf{BS.Sign}_1(\mathsf{sk}, \mathsf{info})$*. Then it returns* $C_{\mathtt{id}}$ *and* $\mathtt{id}$*.*

Oracle sign_2**:** *If* $\sum_{\mathsf{info}} \ell_{\mathtt{closed},\mathsf{info}} < \ell$, sign_2 *takes as input a challenge* e *and a session identifier* $\mathtt{id}$. *If* $\mathtt{session}_{\mathtt{id}} \neq \mathtt{open}$, *it returns* $\perp$. *Otherwise, it sets* $\ell_{\mathtt{closed},\mathsf{info}} \leftarrow \ell_{\mathtt{closed},\mathsf{info}} + 1$ *and* $\mathtt{session}_{\mathtt{id}} \leftarrow \mathtt{closed}$. *Then it generates the response* R *via* $R \xleftarrow{\$} \mathsf{BS.Sign}_2(\mathsf{sk}, \mathsf{st}_{\mathtt{id}}, e)$ *and returns* R.

Output Determination. *When* A *outputs tuples* $(m_1, \sigma_1, \mathsf{info}_1), \ldots, (m_k, \sigma_k, \mathsf{info}_k)$, *return* 1 *if there exists a tag* $\overline{\mathsf{info}}$ *such that* $\left|\{(m_i, \sigma_i, \mathsf{info}_i) \middle| \mathsf{info}_i = \overline{\mathsf{info}}\}\right| \geq \ell_{\mathtt{closed},\overline{\mathsf{info}}} + 1$ *(where by convention* $\ell_{\mathtt{closed},\mathsf{info}} := 0$ *for any* info *that has not been requested to the signing oracles) and for all* $i \in [k]$: $\mathsf{BS.Verify}(\mathsf{pk}, \sigma_i, m_i, \mathsf{info}_i) = 1$ *and* $(m_i, \sigma_i, \mathsf{info}_i) \neq (m_j, \sigma_j, \mathsf{info}_j)$ *for all* $j \neq i$. *Otherwise, return* 0.

We define the advantage of A *in* $\mathbf{OMUF}_{\mathsf{BS}}$ *as*

$$\mathrm{Adv}^{\mathbf{OMUF}}_{\mathsf{A},\mathsf{BS},\ell}(\lambda) := \Pr\left[\ell\text{-}\mathbf{OMUF}^{\mathsf{A}}_{\mathsf{BS}} = 1\right].$$

We say that a blind signature scheme BS *is* (t, ϵ, ℓ)*-one-more unforgeable if for any adversary that runs in time at most* $t(\lambda)$ *the advantage* $\mathrm{Adv}^{\mathbf{OMUF}}_{\mathsf{A},\mathsf{BS},\ell}(\lambda) \leq \epsilon(\lambda)$. *And, respectively for* $\mathbf{SEQ\text{-}OMUF}_{\mathsf{BS}}$

$$\mathrm{Adv}^{\mathbf{SEQ\text{-}OMUF}}_{\mathsf{A},\mathsf{BS},\ell}(\lambda) := \Pr\left[\ell\text{-}\mathbf{SEQ\text{-}OMUF}^{\mathsf{A}}_{\mathsf{BS}} = 1\right].$$

We say that a blind signature scheme BS *is* (t, ϵ, ℓ)*-sequentially-one-more unforgeable if for any adversary that runs in time at most* $t(\lambda)$ *the advantage* $\mathrm{Adv}^{\mathbf{SEQ\text{-}OMUF}}_{\mathsf{A},\mathsf{BS},\ell}(\lambda) \leq \epsilon(\lambda)$.

Definition 2.4.3 (Partial Blindness)**.** *For a three-move partially blind signature scheme* PBS, *we define the partial blindness game* $\mathbf{PBLIND}_{\mathsf{PBS}}$ *with an adversary* S *(in the role of the signer) as follows:*

Setup. *The game samples* $b \xleftarrow{\$} \{0,1\}$. *It then runs* S *on input* pp.

Online Phase. *When* S *outputs messages* $\widetilde{m}_0$ *and* $\widetilde{m}_1$, *a tag* info, *and a public key* pk, *the game checks if* pk *is a valid public key if so, it assigns* $m_0 := \widetilde{m}_b$, $m_1 := \widetilde{m}_{1-b}$. *If* pk *is not a valid public key, the game aborts and outputs* 0. S *is given access to oracles* user_1 *and* user_2, *which behave as follows.*

Oracle user_1**:** *On input a bit* b' *and a* Sign_1 *response* R, *if the session* b' *is not yet open, the oracle marks session* b' *as open and generates a state and a challenge as* $(\mathsf{st}_{b'}, e) \xleftarrow{\$} \mathsf{PBS.User}_1(\mathsf{pk}, m_{b'}, R, \mathsf{info})$. *It returns* e *to* S. *Otherwise, it returns* $\perp$.

Oracle user_2**:** *On input of a* Sign_2 *response* S *and a bit* b', *if the session* b' *is open, the oracle computes the signature* $\mathsf{sig}_{b'} := \mathsf{PBS.User}_2(\mathsf{pk}, \mathsf{st}_{b'}, R)$. *It marks session* b' *as closed and saves* $\mathsf{sig}_{b'}$. *If both sessions are closed and produced signatures, the oracle outputs the two signatures* $\mathsf{sig}_0, \mathsf{sig}_1$ *to* S.

Output Determination. *If both sessions are closed and produced signatures, the game outputs* 1 *iff* S *outputs a bit* b^* *s.t.* $b^* = b$. *Otherwise, it outputs* 0.

We define the advantage of S *as*

$$\mathrm{Adv}^{\mathbf{PBLIND}}_{\mathsf{S},\mathsf{PBS}}(\lambda) = \left|\Pr\left[\mathbf{PBLIND}^{\mathsf{S}}_{\mathsf{PBS}} = 1\right] - \frac{1}{2}\right|$$

where the probability goes over the randomness of the game as well as the randomness of the adversary S. *We say the scheme* PBS *is* (t, ϵ)-partially blind *if for any adversary* S *running in time at most* $t(\lambda)$,

$$\mathrm{Adv}^{\mathbf{PBLIND}}_{\mathsf{S},\mathsf{PBS}}(\lambda) \leq \epsilon(\lambda).$$

2.5 The Algebraic Group Model

In the following, let pp be public parameters that describe a group $\mathbb{G}$ of prime order q with generator $\mathbf{g}$. (We assume for simplicity that pp also includes the security parameter λ.) We denote the neutral element by ϵ and write all other group elements in bold face. We further write $\mathbb{Z}_q$ for $\mathbb{Z}/q\mathbb{Z}$.

Definition 2.5.1 (Algebraic Algorithm)**.** *We say that an algorithm* A *is* algebraic *if, for any group element* $\mathbf{y} \in \mathbb{G}$ *that it outputs, it also outputs a list of* algebraic coefficients $\overrightarrow{z} \in \mathbb{Z}_q^t$, *i.e.,*

$$(\mathbf{y}, \overrightarrow{z}) \overset{\$}{\leftarrow} \mathsf{A}(\overrightarrow{\mathbf{x}})$$

such that

$$\mathbf{y} = \prod \mathbf{x}_i^{z_i}$$

We denote this representation as $[\mathbf{y}]_{\overrightarrow{\mathbf{x}}}$. *For an adversary* A *that has access to oracles during its runtime, we impose the above restriction to all group elements that it outputs to an oracle. Similarly, all group elements that* A *receives through oracle interactions are treated as inputs to* A*; hence, such group elements become part of* $\overrightarrow{\mathbf{x}}$ *when* A *outputs group elements (and hence algebraic coefficients) at a later point.*

In the algebraic group model (AGM), all algorithms are treated as algebraic algorithms.

2.6 Probability Theory

We will need a special Chernoff bound. We state without proof:

Lemma 2.6.1. *Let* $E_1, \ldots, E_\ell$ *be independent events that each occur with probability* p. *Then*

$$\Pr\big[\bigvee_{t=1}^{\ell} E_t\big] \;\geq\; 1 - 1/e^{\ell p/2}.$$

The following lemma is straightforward:

Lemma 2.6.2. *Let* $\mathcal{D}$ *be a distribution over* $\mathcal{X}$, *and* $f : \mathcal{X} \to \mathcal{Y}$ *be a function. Consider random variables* X, X_0 *with*

$$X_0 \leftarrow \mathcal{D} \qquad\qquad X \leftarrow \mathcal{D} \mid [f(\cdot) = f(X_0)].$$

Then X *is distributed according to* $\mathcal{D}$, *i.e., we have* $\forall x \in \mathcal{X} : \Pr[X = x] = \Pr[X_0 = x] = \rho_{\mathcal{D}}(x)$.

Intuitively, Lemma 2.6.2 states that resampling conditioned on a "current value" $f(X_0)$ does not change the distribution.

Proof.

$$\begin{aligned}
\Pr[X = x] &= \sum_{y \in \mathcal{Y}} \Pr[X = x \wedge f(X) = y] \\
&= \sum_{y \in \mathcal{Y}} \Pr[X = x \mid f(X) = y] \cdot \Pr[f(X) = y]
\end{aligned}$$

$$= \sum_{y \in \mathcal{Y}} \Pr[X_0 = x \mid f(X_0) = y] \cdot \Pr[f(X_0) = y]$$
$$= \sum_{y \in \mathcal{Y}} \Pr[X_0 = x \wedge f(X_0) = y] \;=\; \Pr[X_0 = x].$$

□

We review the classical splitting lemma and what we call the bucket lemma (first introduced in [KLX22c]), which will help facilitate our proofs later on.

Lemma 2.6.3 (Splitting Lemma [PS00]). *Let $A \subset X \times Y$ such that*

$$\Pr_{(x,y) \xleftarrow{\$} X \times Y}[(x,y) \in A] \geq \epsilon.$$

For any $\alpha \in [0, \epsilon)$ define

$$B = \left\{ (x,y) \in X \times Y \,\middle|\, \Pr_{y' \xleftarrow{\$} Y}[(x,y') \in A] \geq \epsilon - \alpha \right\}.$$

(B is sometimes called the heavy row *of A.) Then the following statements hold:*

1. $\Pr_{(x,y) \xleftarrow{\$} X \times Y}[(x,y) \in B] \geq \alpha$
2. $\forall (x,y) \in B$: $\Pr_{y' \xleftarrow{\$} Y}[(x,y') \in A] \geq \epsilon - \alpha$
3. $\Pr_{(x,y) \xleftarrow{\$} X \times Y}[(x,y) \in B \mid (x,y) \in A] \geq \frac{\alpha}{\epsilon}$

Lemma 2.6.4 (Bucket Lemma). *Let X be a finite set, $b \in \mathbb{Z}^+$, and let $B_1, \ldots, B_b \subset X$ s.t. $\bigcup_{i=1}^{b} B_i = X$. Then for all $\alpha \in (0,1)$ there exists a set $G_\alpha \subset X$ such that*

1. $|G_\alpha| > (1-\alpha) \cdot |X|$.
2. *For all $x \in G_\alpha$, there exists $i \in [b]$ s.t. $x \in B_i$ and $|B_i| \geq \alpha \cdot \frac{|X|}{b}$.*

Proof. Fix α. Let $F_\alpha \subset X$ be the set of elements that do not belong to *any* B_i ($i \in [b]$) with $|B_i| \geq \alpha \cdot \frac{|X|}{b}$. It therefore holds that $F_\alpha \subset \bigcup_{B_i : |B_i| < \alpha \cdot \frac{|X|}{b}} B_i$. We now compute an upper bound for the size of F_α as

$$|F_\alpha| \leq \left| \bigcup_{i : |B_i| < \alpha \cdot \frac{|X|}{b}} B_i \right| \leq \sum_{i : |B_i| < \alpha \cdot \frac{|X|}{b}} |B_i|$$
$$< \sum_{i : |B_i| < \alpha \cdot \frac{|X|}{b}} \alpha \cdot \frac{|X|}{b} \leq b \cdot \left(\alpha \cdot \frac{|X|}{b} \right) = \alpha \cdot |X|$$

Setting $G_\alpha = X \setminus F_\alpha$ yields the statement. □

The next lemma (originally introduced in [HKK23]) is a probabilistic version of the "bucket lemma" of [KLX22c], which in turn generalizes the "splitting lemma" of [PS96].

Lemma 2.6.5. *Let $\mathcal{D}$ be a distribution over $\mathcal{X}$, and $f : \mathcal{X} \to \mathcal{Y}$ be a function with finite range $\mathcal{Y}$. For any $\alpha \in [0,1]$,*

$$\Pr_{X \leftarrow \mathcal{D}} \left[\rho_{f(\mathcal{D})}(f(X)) \geq \alpha \right] \geq 1 - \alpha \cdot |\mathcal{Y}|.$$

Intuitively, Lemma 2.6.5 states that it is likely that an $X \leftarrow \mathcal{D}$ has a "somewhat common" value of $f(X)$.

Proof.

$$\begin{aligned}\Pr_{X \leftarrow \mathcal{D}} \left[\rho_{f(\mathcal{D})}(f(X)) \geq \alpha \right] &= \sum_{\substack{y \in \mathcal{Y} \\ \rho_{f(\mathcal{D})}(y) \geq \alpha}} \rho_{f(\mathcal{D})}(y) \\ &= \sum_{y \in \mathcal{Y}} \rho_{f(\mathcal{D})}(y) - \sum_{\substack{y \in \mathcal{Y} \\ \rho_{f(\mathcal{D})}(y) < \alpha}} \rho_{f(\mathcal{D})}(y) \geq 1 - |\mathcal{Y}| \cdot \alpha.\end{aligned}$$

□

We recall the following lemma in its variant over $\mathbb{Z}_q$.

Lemma 2.6.6. *Schwartz-Zippel Lemma ([Sch80; Zip79; DL78]) Let $P \in \mathbb{Z}_q[X_1, \ldots, X_n]$ be a non-zero polynomial of total degree $d \geq 0$ over $\mathbb{Z}_q$. Let S be a finite subset of $\mathbb{Z}_q$. Then it holds that*

$$\Pr_{x_1,\ldots,x_n \xleftarrow{\$} S} [P(x_1, \ldots, x_n) = 0] \leq \frac{d}{|S|}.$$

2.7 Pseudorandom Generators and Functions

For convenience, we define pseudorandom number generators (PRGs) with a multi-instance security notion (that is however easily seen to be polynomially equivalent to the ordinary one-instance notion using a hybrid argument):

Definition 2.7.1 ($(\mathcal{Q}, t, \delta)$-hard pseudorandom generator (PRG))**.** *An efficiently computable function* $\mathsf{G} : \{0,1\}^n \mapsto \{0,1\}^m$ *with $m > n$ is a $(\mathcal{Q}, t, \delta)$-hard* pseudo-random generator (PRG) *if every probabilistic adversary* A *that makes at most $\mathcal{Q}$ oracle queries and runs in time at most t satisfies $|\mathrm{Adv}^{\mathsf{PR}}_{\mathsf{G},\mathsf{A}}(\lambda)| \leq \delta$, where*

$$\mathrm{Adv}^{\mathsf{PR}}_{\mathsf{G},\mathsf{A}}(\lambda) := \Pr[\mathsf{MI\text{-}PRG}^{\mathsf{A}}_{\mathsf{G}}(\lambda) = 1] - 1/2$$

for the experiment $\mathsf{MI\text{-}PRG}^{\mathsf{A}}_{\mathsf{G}}$ *defined in Figure 2.2.*

Asymptotically, we say that G *is a secure PRG if for all polynomials $\mathcal{Q}, t$ in λ, there is a negligible $\delta = \delta(\lambda)$, so that* G *is a $(\mathcal{Q}, t, \delta)$-hard PRG.*

In our setting, we will only be interested in PRGs with $n = \lambda$ and $m = 2\lambda$.

Definition 2.7.2 (Prefix-constrained pseudorandom functions)**.** *Consider an efficiently computable function* $\mathsf{F} : \{0,1\}^\lambda \times \{0,1\}^n \to \{0,1\}^m$ *that takes as input a key $k \in \{0,1\}^\lambda$ and an input $x \in \{0,1\}^n$, and outputs an image $y \in \{0,1\}^m$.*

We say that F *is a* prefix-constrained pseudorandom function (PC-PRF) *if there are polynomial-time algorithms* constrain *and* ceval *with the following properties:* constrain *may be probabilistic, takes as input*

```
Algorithm 1: MI-PRG$_{\mathsf{F}}^{\mathsf{A}}(\lambda)$
1 $b \leftarrow \{0,1\}$
2 $b' \leftarrow \mathsf{A}^{\mathrm{challenge}}(1^\lambda)$
3 return $[b = b']$
```

```
Algorithm 2: challenge()
1 $s \leftarrow \{0,1\}^n$
2 $y_0 := \mathsf{G}(s)$; $y_1 \leftarrow \{0,1\}^m$
3 return $y_b$
```

Figure 2.2: Multi-instance PRG indistinguishability game

a key $k \in \{0,1\}^\lambda$ and a prefix $x' \in \{0,1\}^{\leq n}$, and outputs a constrained key $k_{x'}$. ceval *is deterministic, takes as input such a constrained key $k_{x'}$ and an input $x \in \{0,1\}^n$, and outputs an image $y \in \{0,1\}^m$. We require that for all λ, $k \in \{0,1\}^\lambda$, $k_{x'} \leftarrow$ constrain(k, x'), and $x \in \{0,1\}^n$ with $x' \leq_{\mathsf{pfx}} x$, we have*

$$\mathsf{ceval}(k_{x'}, x) \;=\; \mathsf{F}(k, x).$$

The main security property of PC-PRFs is indistinguishability:

Definition 2.7.3 ($(\mathcal{Q}, t, \delta)$-indistinguishability for PC-PRFs)**.** *Let* F *be a PC-PRF as in Definition 2.7.2. We say that* F *is* $(\mathcal{Q}, t, \delta)$-indistinguishable *if for every probabilistic adversary* A *that runs in time at most t, makes at most $\mathcal{Q}$ queries to the* constrain *oracle and at most one query to the* challenge *oracle in the* PC-PRF$_{\mathsf{F},\mathsf{A}}$ *experiment, we have $|\mathrm{Adv}_{\mathsf{F},\mathsf{A}}^{\mathsf{PC\text{-}PRF}}(\lambda)| \leq \delta$, where*

$$\mathrm{Adv}_{\mathsf{F},\mathsf{A}}^{\mathsf{PC\text{-}PRF}}(\lambda) := \Pr[\mathsf{PC\text{-}PRF}_{\mathsf{F}}^{\mathsf{A}}(\lambda) = 1] - 1/2$$

for the experiment PC-PRF$_{\mathsf{F}}^{\mathsf{A}}$ *defined in Figure 2.3.*

Asymptotically, we say that F *is an indistinguishable PC-PRF if for all polynomials $\mathcal{Q}, t$ in λ, there is a negligible $\delta = \delta(\lambda)$, so that* F *is $(\mathcal{Q}, t, \delta)$-indistinguishable.*

```
Algorithm 3: PC-PRF$_{\mathsf{F}}^{\mathsf{A}}(\lambda)$
1 $b \leftarrow \{0,1\}$
2 $k \leftarrow \{0,1\}^\lambda$
3 $X := \emptyset$
4 $x^* := \varepsilon$
5 $b' \leftarrow \mathsf{A}^{\mathrm{constrain,challenge}}(1^\lambda)$
6 return $[b = b']$
```

```
Algorithm 4: constrain($x'$)
1 if $x' \leq_{\mathsf{pfx}} x^*$ then return $\perp$
2 $X := X \cup \{x'\}$
3 $k_{x'} \leftarrow \mathsf{constrain}(k, x')$
4 return $k_{x'}$
```

```
Algorithm 5: challenge($x$)
1 if $\exists x' \in X : x' \leq_{\mathsf{pfx}} x$ then return $\perp$
2 $x^* := x$
3 $y_0^* := \mathsf{F}(k, x)$; $y_1^* \leftarrow \{0,1\}^m$
4 return $y_b^*$
```

Figure 2.3: CP-PRF indistinguishability game

2.8 Secret-Key Encryption

Definition 2.8.1 (Secret-key encryption)**.** *A* secret-key encryption scheme *consists of the following algorithms* SKE $= (\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$*:*

```
Algorithm 6: IND-CPA$^{\mathsf{A}}_{\mathsf{SKE}}(\lambda)$
1 $b \leftarrow \{0,1\}$
2 $U := [\,]$          // empty array
3 $b' \leftarrow \mathsf{A}^{LoR,NU}(1^\lambda)$
4 return $[b = b']$
```

```
Algorithm 7: $NU$
1 $k \leftarrow \mathbf{Gen}(1^\lambda)$
2 $U[\mathsf{len}(U)+1] := k$          // append to array
```

```
Algorithm 8: $LoR(i, m_0, m_1)$
1 $c \leftarrow \mathbf{Enc}(U[i], m_b)$          // $\bot$ if $U[i]$ undef'd
2 return $c$
```

Figure 2.4: Many-user, many-challenge IND-CPA game

$\mathbf{Gen}(1^\lambda)$ *takes as input the security parameter encoded in unary, and outputs a key k.*

$\mathbf{Enc}(k,m)$ *takes as input a key k and a message $m \in \mathcal{M}$, and outputs a ciphertext c.*

$\mathbf{Dec}(k,c)$ *takes as input a key k and a ciphertext c and outputs either a message $m \in \mathcal{M}$ or an error symbol $\bot$.*

We require correctness, *i.e., $\forall\lambda$, and $m \in \mathcal{M}$, we have*

$$\Pr[\mathbf{Dec}(k,c) = m \mid k \leftarrow \mathbf{Gen}(1^\lambda), c \leftarrow \mathbf{Enc}(k,m)] = 1.$$

Definition 2.8.2 (Many-user, many-ciphertext SKE indistinguishability)**.** *A secret-key encryption scheme* SKE *is $(\mathcal{Q}_{\mathsf{LoR}}, \mathcal{Q}_{\mathsf{NU}}, t, \delta)$-indistinguishable under chosen-plaintext attacks (short: $(\mathcal{Q}_{\mathsf{ctxt}}, \mathcal{Q}_{\mathsf{NU}}, t, \delta)$-*IND-CPA *secure) if every probabilistic adversary* A *that runs in time at most t, and makes at most $\mathcal{Q}_{\mathsf{LoR}}$ and $\mathcal{Q}_{\mathsf{NU}}$ queries to the LoR and NU oracles below, respectively, satisfies $|\mathrm{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{SKE,A}}(\lambda)| \le \delta$, where*

$$\mathrm{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{SKE,A}}(\lambda) := \Pr[\mathsf{IND\text{-}CPA}^{\mathsf{A}}_{\mathsf{SKE}}(\lambda) = 1] - 1/2$$

for the IND-CPA$^{\mathsf{A}}_{\mathsf{SKE}}$ *experiment defined in Figure 2.4.*

Asymptotically, we say that SKE *is* IND-CPA *secure if for all polynomials $\mathcal{Q}_{\mathsf{LoR}}, \mathcal{Q}_{\mathsf{NU}}, t$ in λ, there is a negligible $\delta = \delta(\lambda)$, so that* SKE *is $(\mathcal{Q}_{\mathsf{LoR}}, \mathcal{Q}_{\mathsf{NU}}, t, \delta)$-*IND-CPA *secure.*

We remark that this many-user, many-ciphertext formulation of IND-CPA security is polynomially equivalent (using a standard hybrid argument) to the traditional one-user, one-ciphertext formulation (as in, e.g., [Bel+97]).

2.9 Dual-Mode NIWI

A dual-mode NIWI proof system is a variant of NIWI proofs [FS90] offering two computationally indistinguishable modes to setup the common reference string (CRS). A binding mode CRS provides perfect soundness guarantees whereas a hiding mode CRS provides perfect witness indistinguishability guarantees.

Definition 2.9.1 (Dual-mode NIWI proof system (syntax), [GS08; Alb+16])**.** *A* dual mode non-interactive witness-indistinguishable (NIWI) proof system *for a relation $\mathcal{R}$ is a tuple of PPT algorithms $\Pi = (\mathsf{Setup}, \mathsf{HSetup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Ext})$.*

Setup(1^λ). *On input of* 1^λ, Setup *outputs a* perfectly binding *common reference string* crs *and a corresponding extraction trapdoor* td_{ext}.

HSetup(1^λ). *On input of* 1^λ, HSetup *outputs a* perfectly hiding *common reference string* crs.

Prove(crs, x, w). *On input of the CRS* crs, *a statement* x *and a corresponding witness* w, Prove *produces a proof* π.

Verify(crs, x, π). *On of the CRS* crs, *a statement* x *and a proof* π, Verify *outputs* 1 *if the proof is valid and* 0 *otherwise.*

Ext$(td_{\mathsf{ext}}, x, \pi)$. *On input the extraction trapdoor* td_{ext}, *a statement* x *and a proof* π, Ext *outputs a witness* w.

We require Π *to satisfy the CRS indistinguishability, perfect completeness, perfect soundness, perfect extractability and perfect witness-indistinguishability.*

There are several instantiations of dual-mode NIWI proof systems satisfying the above definition (or statistical variants), [GS08; PS19; HU19].

2.10 Probabilistic Indistinguishability Obfuscation

Let $\mathcal{C} = (\mathcal{C}_\lambda)_{\lambda\in\mathbb{N}}$ be a family of sets $\mathcal{C}_\lambda$ of probabilistic circuits. A *circuit sampler* for $\mathcal{C}$ is defined as a family of (efficiently samplable) distributions $S = (S_\lambda)_{\lambda\in\mathbb{N}}$, where S_λ is a distribution over triplets (C_0, C_1, z) with $C_0, C_1 \in \mathcal{C}_\lambda$ such that C_0 and C_1 take inputs of the same length and $z \in \{0,1\}^{poly(\lambda)}$.

Definition 2.10.1 (X-ind sampler, [Can+15]). *Let* $X(\lambda)$ *be a function upper bounded by* 2^λ. *The class* $\mathcal{S}^{X\text{-ind}}$ *of* X-ind samplers *for a circuit family* $\mathcal{C}$ *contains all circuit samplers* $S = (S_\lambda)_{\lambda\in\mathbb{N}}$ *for* $\mathcal{C}$ *such that for all* $\lambda \in \mathbb{N}$, *there exists a set* $\mathcal{X}_\lambda \subseteq \{0,1\}^*$ *with* $|\mathcal{X}| \leq X(\lambda)$, *such that*

X-**differing inputs.** *With overwhelming probability over the choice of* $(C_0, C_1, z) \leftarrow S_\lambda$, *for every* $x \notin \mathcal{X}_\lambda$, *for all* $r \in \{0,1\}^{m(\lambda)}$, $C_0(x; r) = C_1(x; r)$.

X-**indistinguishability.** *For all (non-uniform) adversaries* A, *the advantage*

$$X(\lambda) \cdot \left(\Pr[\mathrm{Exp}^{\texttt{sel-ind}}_{S,\mathsf{A}}(\lambda) = 1] - \frac{1}{2}\right)$$

is negligible, where $\mathrm{Exp}^{\texttt{sel-ind}}_{S,\mathsf{A}}(\lambda)$ *requires* A *to statically choose an input, samples circuits* C_0, C_1 *(and auxiliary information* z*) afterwards, evaluates the circuit* C_b *(for randomly chosen* b*) on the adversarially chosen input (let the output be* y*) and outputs* 1 *if* A *on input of* (C_0, C_1, z, y) *guesses* b *correctly.*

Definition 2.10.2 (Probabilistic indistinguishability obfuscation for a class of samplers $\mathcal{S}$ (syntax), [Can+15]). *A* probabilistic indistinguishability obfuscator *(pIO) for a class of samplers* $\mathcal{S}$ *is a uniform PPT algorithm* piO, *such that correctness and security with respect to* $\mathcal{S}$ *hold.*

[Can+15] present the to date only known construction of pIO for X-ind samplers over the family of all polynomial sized probabilistic circuits.

2.11 Statistically Correct Input Expanding pIO

Looking ahead to Chapter 7, instead of computationally correct pIO, we require a notion of statistically correct pIO, i.e. statistical closeness between evaluations of the original (probabilistic) circuit and the obfuscated (deterministic) circuit. Clearly, in general, this is impossible since the obfuscated circuit is deterministic and hence has no source of entropy other than its input. However, as long as a portion of the circuit's input is guaranteed to be outside the view of the adversary (and has sufficiently high min-entropy), the output of the obfuscated circuit and the actual probabilistic circuit can be statistically close. Therefore, we compile probabilistic circuits such that they receive an auxiliary input aux but simply ignore this input in their computation. Even though the obfuscated circuit is deterministic, the auxiliary input can be used as a source of actual entropy.

Definition 2.11.1 (ℓ-expanding pIO for the class of samplers $\mathcal{S}$)**.** *An* ℓ-expanding probabilistic indistinguishability obfuscator *for the class of samplers $\mathcal{S}$ over $\mathcal{C} = (\mathcal{C}_\lambda)_{\lambda\in\mathbb{N}}$ is a uniform PPT algorithm $\mathsf{piO}^\star_\ell$, satisfying the following properties.*

Input expanding correctness. *For all PPT adversaries* A*, all circuits* $C \in \mathcal{C}$*,*

$$\left|\Pr[\mathsf{A}^{\mathcal{O}_C(\cdot,\cdot)}(1^\lambda, C) = 1] - \Pr[\Lambda \leftarrow \mathsf{piO}^\star_\ell(1^{p(\lambda)}, C) \colon \mathsf{A}^{\mathcal{O}_\Lambda(\cdot,\cdot)}(1^\lambda, C) = 1]\right|$$

is negligible, where the oracles must not be called twice on the same input (x, aux)*.*

```
$\mathcal{O}_C(x, aux)$                    | $\mathcal{O}_\Lambda(x, aux)$
$r \leftarrow \{0,1\}^m$                   | return $\Lambda(x, aux)$
return $C(x; r)$                           |
```

Security with respect to $\mathcal{S}$**.** *For all circuit samplers $S \in \mathcal{S}$, for all PPT adversaries* A*, the advantage*

$$\begin{aligned}
&\mathrm{Adv}^{\texttt{pio-ind}(\star)}_{\mathsf{piO}^\star_\ell,S,\mathsf{A}}\lambda := \\
&\Big|\Pr\left[(C_0, C_1, z) \leftarrow S(1^\lambda) \colon \mathsf{A}(1^\lambda, C_0, C_1, z, \mathsf{piO}^\star_\ell(1^{p(\lambda)}, C_0)) = 1\right] \\
&- \Pr\left[(C_0, C_1, z) \leftarrow S(1^\lambda) \colon \mathsf{A}(1^\lambda, C_0, C_1, z, \mathsf{piO}^\star_\ell(1^{p(\lambda)}, C_1)) = 1\right]\Big|
\end{aligned}$$

is negligible in λ*.*

Support respecting. *For all circuits $C \in \mathcal{C}_\lambda$, all inputs $x \in \{0,1\}^{n'(\lambda)}$, all $aux \in \{0,1\}^{\ell(\lambda)}$, all $\Lambda \in \mathsf{supp}(\mathsf{piO}^\star_\ell(1^{p(\lambda)}, C))$, $\Lambda(x, aux) \in \mathsf{supp}(C(x))$.*

Statistical correctness with error $2^{-e(\lambda)}$**.** *For all $C \in \mathcal{C}_\lambda$ and all joint distributions (X_1, X_2) over $\{0,1\}^{n'(\lambda)} \times \{0,1\}^{\ell(\lambda)}$ with average min-entropy $\ell(\lambda) \geq \widetilde{\mathrm{H}}_\infty(X_2 \mid X_1) > m(\lambda) + 2e(\lambda) + 2$, the statistical distance between*

$$\left\{\Lambda \leftarrow \mathsf{piO}^\star_\ell(1^{p(\lambda)}, C) \colon (\Lambda, \Lambda(X_1, X_2))\right\}$$
$$\text{and } \left\{\Lambda \leftarrow \mathsf{piO}^\star_\ell(1^{p(\lambda)}, C) \colon (\Lambda, C(X_1; U_{m(\lambda)}))\right\}$$

is at most $2^{-e(\lambda)}$*.*

We note that setting $\ell := 0$ recovers the original definition of pIO for X-ind samplers due to [Can+15]. Looking ahead, our application does not require input expanding correctness.

Let S be a circuit sampler and let $\widehat{S}$ denote the circuit sampler which calls S and outputs ℓ-expanded circuits. Unfortunately, if S is an X-ind sampler does not imply that $\widehat{S}$ also satisfies the requirements to be an X-ind sampler. On a high level this is because $\widehat{X}(\lambda) := X(\lambda) \cdot 2^{\ell(\lambda)}$ is necessary for $\widehat{S}$ to satisfy the X-differing inputs property. Then, however, X-indistinguishability of S does not suffice to prove $\widehat{X}$-indistinguishability of $\widehat{S}$. Thus, we introduce the notion of ℓ-expanding X-ind samplers.

Definition 2.11.2 (ℓ-expanding X-ind sampler)**.** *Let S be a circuit sampler. With $\widehat{S}$ we denote the circuit sampler which on input of $1^{p(\lambda)+\ell(\lambda)}$ samples $(C_0, C_1, z) \leftarrow S(1^{p(\lambda)})$ and outputs the circuits $\widehat{C}_0 := \mathcal{E}_\ell(C_0), \widehat{C}_1 := \mathcal{E}_\ell(C_1)$ and auxiliary information $\widehat{z} := (C_0, C_1, z)$. The class $\mathcal{S}_\ell^{X\text{-}(\star)\text{-ind}}$ of ℓ-*expanding X-ind samplers *for a circuit family $\mathcal{C}$ contains all circuit samplers $S = (S_\lambda)_{\lambda\in\mathbb{N}}$ for $\mathcal{C}$ such that the circuit sampler $\widehat{S}$ is an X-ind sampler according to Definition 2.10.1, i.e. $\widehat{S} \in \mathcal{S}^{X\text{-ind}}$.*

On a high level, we instantiate the construction of pIO for X-ind samplers due to [Can+15] with a suitably extracting puncturable pseudorandom function (pPRF). By suitably extracting we mean that the PRF output is guaranteed to be statistically close to uniform randomness as long as the average min-entropy of the input of the PRF is sufficiently high. Such a pPRF can be constructed by composing a pPRF with a universal hash function.

Theorem 2.11.3. *Let e be an efficiently computable function. Let F be a sub-exponentially secure special extracting PRF family with distinguishing advantage $2^{-\lambda^\epsilon}$ (for some constant ϵ) and error $2^{-e(\lambda)}$ mapping $n(\lambda) = n'(\lambda) + \ell(\lambda)$ bits to $m(\lambda)$ bits which is extracting if the input average min-entropy is greater than $m(\lambda) + 2e(\lambda) + 2$. Then, there exists a statistically correct input expanding pIO* piO$_\ell^\star$ *for the class of samplers $\mathcal{S}_\ell^{X\text{-}(\star)\text{-ind}}$.*

2.12 Re-Randomizable and Fully Homomorphic Encryption

We define an IND-CPA secure PKE scheme as a tuple of PPT algorithms PKE = (KGen, Enc, Dec) in the usual sense. Furthermore, without loss of generality, we assume that sk is the random tape used for key generation. Therefore, making the random tape of KGen explicit, we write (pk, sk) = KGen(1^λ; sk).

A re-randomizable PKE scheme additionally provides an algorithm Rerand which re-randomizes a given ciphertext perfectly.

Finally, a fully homomorphic PKE scheme additionally provides an algorithm Eval which given the public key pk, an circuit C (expecting a inputs from the message space) and a ciphertexts $C_1, \ldots, C_a$, produces a ciphertext encrypting $C(\mathsf{Dec}(\mathsf{sk}, C_1), \ldots, \mathsf{Dec}(\mathsf{sk}, C_a))$.

Due to [Can+15], probabilistic indistinguishability obfuscation in conjunction with (slightly super-polynomially secure) perfectly correct and perfectly re-randomizable public-key encryption yields a perfectly correct and perfectly re-randomizable fully homomorphic encryption scheme.

Chapter 3

The Power of Undirected Rewindings for Adaptive Security

This chapter deals with a novel way of using rewinding as a proof strategy. It is based on [HKK23].

We first show some helper results in Section 3.1, namely we consider different ways to repeatedly resample from a distribution. The generic way to of resampling we consider (with respect to a set of functions $f_1, \ldots, f_{\mathcal{T}}$) is to sample an initial value X_0 from the distribution, and then for increasing t starting from 1 re-sample until the function f_t has the same value on X_{t-1} and the current sample X_t. It is easy to see that all the samples, including the last one, are distributed according to the original distribution. We show that there is an equivalent way of sampling that splits these f_t into two functions g_t and h_t, where h_t is a condition applied to the distribution itself, whereas g_t is still achieved via resampling until $g_t(X_{t-1}) = g_t(X_t)$. This way of resampling is closer to our rewinding strategy, as later on, the rewinding games and reductions will preserve a prefix of the transcript between challenger and adversary (this will be h_t) and resample until the transcript has certain properties again (this corresponds to g_t). We further prove a lemma bounding the number of re-samplings needed until g_t is fulfilled again which will be useful to bound the running time of our reductions later on.

We then turn to our proof of adaptive security of the GGM PRF as a PC-PRF in Section 3.2. The security is based on the pseudorandomness of the underlying PRG. The proof strategy is to gradually replace the prefix keys on the path to the challenge by uniformly random keys. In order for this to work, the reduction needs to know which keys will be prefix keys of the challenge. To learn this, it employs the rewinding strategy described above.

Finally, in Section 3.3, we look at the security of the logical key hierarchy protocol, which can be used for server-assisted key exchange in group messaging applications. We again want to apply our rewinding technique to figure out when it is 'safe' for a reduction to embed its IND-CPA challenge in a vertex in the tree. The goal of the reduction is to be able to embed an IND-CPA challenge at the root of the LKH tree. To do this, it needs to 'forget' the two keys just below the root. This means, a previous game needs to embed an IND-CPA challenge there to change these keys to something random. Overall, this line of thought results in a *pebbling* strategy which we explain in Section 3.3.1. We then further prove a technical lemma that we can use to bound the distance between certain hybrid games in the proof of security of LKH in Section 3.3.2. We then finally turn to the main proof of security of LKH in Section 3.3.3. In the proof, hybrids correspond to pebbling configurations and switching between pebbling configurations is done via IND-CPA reductions. The reductions figure out the right

Algorithm 9: Repeated resampling, generic

```
   Input: $\mathcal{D}, f_1, \ldots, f_\mathcal{T}$
1  $X_0 \leftarrow \mathcal{D}$
2  for $t := 1$ to $\mathcal{T}$ do
3      $X_t \leftarrow \mathcal{D} \mid [f_t(\cdot) = f_t(X_{t-1})]$
4  end
5  return $X_\mathcal{T}$
```

time during the hybrid to embed an IND-CPA challenge using the rewinding technique. As the pebbling strategy leads to different rewinding end conditions between some hybrids, we need to do some additional technical work to show that these end conditions are actually equivalent. Therefore, this proof is a bit more involved than that of the GGM PRF.

3.1 Analysis of a Repeated Resampling Algorithm

Overview. In this section, we will provide a few helper results for our upcoming applications. Specifically, we will investigate what happens when we first sample some X_0 from a distribution (which can be a run with an adversary A), and then resample conditioned on parts of X_0. (This latter operation corresponds to rewinding and rerunning A until a certain property of the full run is preserved.)

As explained in the introduction, the main difference to previous rewinding treatments is that we consider "undirected" rewindings, which translates to resampling conditioned on a-priori fixed properties of X_0. This will enable us to deduce that this resampling does not change the output distribution, and that resampling is likely to preserve any "sufficiently common" property of the initial X_0 in the process.

Generic Framework. In the following, let $\mathcal{D}$ be a distribution over some set $\mathcal{X}$, and assume functions $f_1, \ldots, f_\mathcal{T} : \mathcal{X} \to \mathcal{Y}$ for a finite set $\mathcal{Y}$. Now consider Algorithm 9. Algorithm 9 starts with a fresh $\mathcal{D}$-sample, and then repeatedly resamples while preserving the value of the functions f_t on those samples. We have:

Lemma 3.1.1. *All X_t defined through Algorithm 9 are distributed according to $\mathcal{D}$, i.e., $\forall t, x : \Pr[X_t = x] = \rho_\mathcal{D}(x)$.*

Proof. For X_0, this is clear. For $X_{t-1} \leftarrow \mathcal{D}$, we obtain $\forall x : \Pr[X_t = x] = \rho_\mathcal{D}(x)$ by applying Lemma 2.6.2. □

This in particular holds for Algorithm 9's output $X_\mathcal{T}$. Hence, Algorithm 9 would seem like an unnecessarily complicated way to sample from $\mathcal{D}$. However, in the following, we will refine Algorithm 9 to better capture our upcoming rewinding process.

Split Resampling. Now Algorithm 10 performs the generation of the X_t through a different, yet conceptually equivalent form of resampling. More concretely, Algorithm 10 conditions not only on one function value $f_t(X_{t-1})$, but on two function values $g_t(X_{t-1})$ and $h_t(X_{t-1})$. Here, we assume functions $g_t : \mathcal{X} \to \mathcal{Y}$ and $h_t : \mathcal{X} \to \mathcal{Z}$ for a finite set $\mathcal{Y}$ and a set $\mathcal{Z}$. This "double resampling" is done in a somewhat peculiar way: the distribution already conditioned on $h_t(X_{t-1})$ is sampled until a value X_t with $g_t(X_t) = g_t(X_{t-1})$ appears. Still, we obtain as before:

Algorithm 10: Repeated resampling, split

```
   Input: $\mathcal{D}, g_1, \dots, g_{\mathcal{T}}, h_1, \dots, h_{\mathcal{T}}$
1  $X_0 \leftarrow \mathcal{D}$
2  for $t := 1$ to $\mathcal{T}$ do
3      repeat
4          $X_t \leftarrow \mathcal{D} \mid [h_t(\cdot) = h_t(X_{t-1})]$
5      until $g_t(X_t) = g_t(X_{t-1})$
6  end
7  return $X_{\mathcal{T}}$
```

Lemma 3.1.2. *All X_t defined through Algorithm 10 are distributed according to $\mathcal{D}$, i.e., $\forall t, x : \Pr[X_t = x] = \rho_{\mathcal{D}}(x)$.*

Proof. For any $t \in [\mathcal{T}]$, the $\mathbf{repeat}$ loop samples X_t from

$$\big(\mathcal{D} \mid [h_t(\cdot) = h_t(X_{t-1})]\big) \;\mid\; [g_t(\cdot) = g_t(X_{t-1})] \;=\; \mathcal{D} \mid [f_t(\cdot) = f_t(X_{t-1})]$$

for the function $f_t(X) = (g_t(X), h_t(X))$. Hence, Algorithm 10 is equivalent to Algorithm 9 (for these f_t), and Lemma 3.1.1 yields the statement. □

Additionally, we can bound the runtime of Algorithm 10:

Lemma 3.1.3. *Let T_t^{rep} be the number of all iterations of the $\mathbf{repeat}$ loop for this value of t in Algorithm 10. For any $\gamma \in (0, 1]$, we have*

$$\Pr[\,\forall t \in [\mathcal{T}] : T_t^{\mathsf{rep}} \le \underbrace{2\mathcal{T} \cdot \ln(2\mathcal{T}/\gamma) \cdot |\mathcal{Y}|/\gamma}_{=:T_{\mathbf{max}}(\mathcal{T}, |\mathcal{Y}|, \gamma)}\,] \;\ge\; 1 - \gamma, \tag{3.1}$$

where $\mathcal{Y}$ is the (finite) domain of the g_t.

Proof. First fix a $t \in [\mathcal{T}]$. By Lemma 3.1.2, X_{t-1} is distributed according to $\mathcal{D}$. Hence, using Lemma 2.6.2, X_{t-1} is also distributed according to

$$\mathcal{D}' \;:=\; \mathcal{D} \mid [h_t(\cdot) = h_t(X^*)]$$

for some independently chosen $X^* \leftarrow \mathcal{D}$. Since $h_t(X_t) = h_t(X^*)$ by definition, in each iteration of Line 4, X_t is also distributed according to $\mathcal{D}'$. Now invoke Lemma 2.6.5 with $\alpha := \gamma/(2\mathcal{T} \cdot |\mathcal{Y}|)$, distribution $\mathcal{D}'$, and function g_t. This yields

$$\Pr_{X_t \leftarrow \mathcal{D}'}[g_t(X_t) = g_t(X_{t-1})] \;\ge\; \alpha, \tag{3.2}$$

except with probability $\gamma/(2\mathcal{T})$ (over X_{t-1}).

Recall that T_t^{rep} is the number of iterations of the $\mathbf{repeat}$ loop for this t. Conditioned on Eq. (3.2), Lemma 2.6.1 (instantiated with E_t as the event that the t-th iteration succeeds, $p := \alpha$, and $\ell := 2\ln(2\mathcal{T}/\gamma)/\alpha$) shows

$$\Pr\left[T_t^{\mathsf{rep}} \;\le\; \frac{2\ln(2\mathcal{T}/\gamma)}{\alpha}\right] \;\ge\; 1 - \frac{\gamma}{2\mathcal{T}}, \tag{3.3}$$

where the probability is taken (only) over the resamplings in the loop. Now a union bound shows that Eq. (3.2) and the bound from Eq. (3.3) hold for all t, except with probability γ. This finally yields Eq. (3.1). □

The split approach of Algorithm 10 reflects our upcoming rewinding scenario. In particular, conditioning on a "common partial history" $h_t(X_t) = h_t(X_{t-1})$ will correspond to rewinding a simulation up to the t-th "branching point", while $g_t(X_t) = g_t(X_{t-1})$ is a condition we hope the rewound simulation to fulfill. We will be able to sample from $\mathcal{D} \mid [h_t(\cdot) = h_t(X_{t-1})]$ directly through rewinding, but will then have to condition on $g_t(\cdot) = g_t(X_{t-1})$ by a brute-force $\mathbf{repeat}$ loop.

3.2 Adaptive Security for the GGM PC-PRF

In this section we use the results on repeated resampling from Section 3.1 to prove that the PRF construction by Goldreich, Goldwasser and Micali [GGM84b] is adaptively secure as a prefix-constrained pseudorandom function (PC-PRF), based on the security of the underlying PRG.

Definition 3.2.1 (GGM PRF)**.** *Given a length-doubling PRG* $\mathsf{G} : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ *and an input length* $d = d(\lambda)$*, the GGM PRF* $\mathsf{F}_d : \{0,1\}^\lambda \times \{0,1\}^d \to \{0,1\}^\lambda$ *with key space* $\{0,1\}^\lambda$ *is defined as*

$$\mathsf{F}_d(k, x) = k_x \text{ where } k_\varepsilon = k \text{ and } \forall x' \in \{0,1\}^{<d} : k_{x'\|0} \| k_{x'\|1} = \mathsf{G}(k_{x'}).$$

It was noted independently in [Kia+13], [BW13], and [BGI14] that the above PRF construction allows for the use as a prefix-constrained PRF (PC-PRF), with

$$\mathsf{constrain}(k, x') := (x', \mathsf{F}_{|x'|}(k, x')) = (x', k_{x'}) \text{ for } x' \in \{0,1\}^{<d}.$$

For ease of presentation, in the following we will often refer to $k_{x'}$ as the constrained key for x'. The algorithm ceval, on input a constrained key $(x', k_{x'})$ for a prefix x' of x and the string $x = x'\|x'' \in \{0,1\}^d$, then computes k_x as

$$\mathsf{ceval}((x', k_{x'}), x) := \mathsf{constrain}(k_{x'}, x'').$$

3.2.1 Proving Security from PR

We now define the security experiment $\mathrm{Exp}^{\mathsf{GGMPRF}}_{\mathsf{G},\mathsf{A},d}(\lambda)$. Security in the sense of the following definition immediately implies adaptive security of the GGM PC-PRF (see also Remark 3.2.3).

Definition 3.2.2 (GGMPRF Security Experiment)**.** *Let* $\mathsf{G} : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ *be a length-doubling PRG, let* $d = d(\lambda)$ *an input length, and let* A *be a probabilistic adversary. We denote the first half of the output of* G *on input* k *by* $\mathsf{G}_0(k)$*, the second half by* $\mathsf{G}_1(k)$*.*

Setup. *The experiment* $\mathrm{Exp}^{\mathsf{GGMPRF}}_{\mathsf{G},\mathsf{A},d}(\lambda)$ *initially samples uniformly at random a challenge bit* $b_{\mathsf{ggmprf}} \leftarrow \{0,1\}$ *and a key* $k_\varepsilon \leftarrow \{0,1\}^\lambda$*.*

Online Phase. *The adversary is allowed to make the following queries.*

Corruption Queries. A *can adaptively make corruption queries for strings* $x \in \{0,1\}^{\leq d}$*. This initiates the computation of all so far undefined keys* $k_{x'b}$ *with* $x' <_{\mathsf{pfx}} x$ *and* $b \in \{0,1\}$ *as* $k_{x'b} := \mathsf{G}_b(k_{x'})$*, and exposes* k_x *to* A*.*

Challenge. *At any point,* A *may stop the game and ask to be challenged on* $x^* \in \{0,1\}^d$*, and then has to distinguish the real key* k_{x^*} *(case* $b_{\mathsf{ggmprf}} = 0$*) from a random key (case* $b_{\mathsf{ggmprf}} = 1$*). To make the game non-trivial, for the challenge* x^* *it must hold that no corruption of any prefix of* x^* *was made throughout the game.*

Output Determination *The output of the experiment is 1 if* A *correctly guesses the bit* b_{ggmprf}*, and 0 otherwise.*

We define the advantage of A *in this game as*

$$\mathrm{Adv}^{\mathsf{GGMPRF}}_{\mathsf{G},\mathsf{A},d}(\lambda) := \Pr[\mathrm{Exp}^{\mathsf{GGMPRF}}_{\mathsf{G},\mathsf{A},d}(\lambda)(\lambda) = 1] - 1/2.$$

We say that GGMPRF *security holds (for* G *and* d*) if* $\mathrm{Adv}^{\mathsf{GGMPRF}}_{\mathsf{G},\mathsf{A},d}(\lambda)$ *is negligible for every probabilistic polynomial-time* A*.*

One can view this security experiment as a game on a binary tree of depth d as defined in Section 2.1, where the adversary can adaptively compromise labels k_x. For security, we require that keys that cannot be computed trivially from compromised keys should remain pseudorandom.

Remark 3.2.3. *We note that we consider adversaries that make their challenge query as the last query. This is not a restriction as any adaptive adversary can be transformed into such an adversary with only* d *additional constrained key queries, using the following reduction: All queries and responses until the challenge query are forwarded. Once the adversary submits the challenge query, the reduction queries all constrained keys on the co-path before forwarding the challenge query and its response. To answer any future constrained key queries, the reduction uses the previously queried constrained keys on the co-path.*

To see that security of the GGM PRF as a PC-PRF follows from our results on the GGMPRF *security experiment, note that a reduction can answer adversarial constrained key queries and PRF evaluation queries in the* PC-PRF *security experiment for the GGM PRF by making corresponding corruption queries in the* GGMPRF *security experiment. In particular, this means that for any adversary* A *that has advantage* $\mathrm{Adv}^{\mathsf{GGMPRF}}_{\mathsf{G},\mathsf{A},d}(\lambda)$*, runs in time* t_{A}*, and makes* $\mathcal{Q}_{\mathsf{corrupt}}$ *constrained key queries, there exists an adversary* B *that runs in time* t_{B} *roughly equal*[1] *to* t_{A} *with*

$$\mathrm{Adv}^{\mathsf{PC\text{-}PRF}}_{\mathsf{F}_d,\mathsf{A}}(\lambda) = \mathrm{Adv}^{\mathsf{GGMPRF}}_{\mathsf{G},\mathsf{A},d}(\lambda)$$

and makes $\mathcal{Q}'_{\mathsf{corrupt}} \leq \mathcal{Q}_{\mathsf{corrupt}} + d$ *constrained key queries.*

Our strategy. Let us fix a length-doubling PRG G and a depth/input length $d = d(\lambda)$. Let us first consider a *selective* setting where an adversary A has to commit to the challenge x^* in the beginning of the game. For this setting, we can bound the success probability of any PPT adversary A by a sequence of $d+1$ hybrid games where in the ith hybrid game, the first i PRG evaluations on the path from the root to x^* are replaced by random sampling, i.e. the keys $k_{x\|0}, k_{x\|1}$ for all $x \leq_{\mathsf{pfx}} x^*$ with $|x| < i$ are sampled uniformly at random instead of computing $\mathsf{G}(k_x)$. For each $i \in [d]$, one can then prove that games $i-1$ and i are indistinguishable based on the security of the PRG G. Furthermore, since in game d, the key k_{x^*} is sampled independently and uniformly at random, the cases $b_{\mathsf{ggmprf}} = 0$ and $b_{\mathsf{ggmprf}} = 1$ are information-theoretically indistinguishable, hence the advantage of A is 0 in this game. We thus obtain an upper bound on A's advantage in the selective GGMPRF experiment in terms of PR security of the PRG G, with a security loss linear in d.

Also in the adaptive setting, where A can make its choices on the fly, we will bound A's advantage to win the GGMPRF game through a similar hybrid argument. Again, we will start with the original GGMPRF game above and apply a number of successive changes until finally A's view is independent of the challenge bit b_{ggmprf}. Since we make a liberal use of rewindings, it will be helpful to formalize A's view:

[1]By "roughly equal", we mean that B runs A only once, but as discussed with up to d added oracle queries and some additional constrain operations.

Definition 3.2.4 (Adversarial view)**.** *In a run of the experiment* $\mathrm{Exp}^{\mathsf{GGMPRF}}_{\mathsf{G},\mathsf{A},d}(\lambda)$ *from Definition 3.2.2, we define* A*'s view* view_A *in this run as a sequence* $(\mathsf{ev}_1, \dots, \mathsf{ev}_\ell)$ *of events, where each* ev_i *can be one of the following:*

Query. *One of* A*'s queries (without reply), either of the form* $(\mathsf{corrupt}, x)$ *for a corruption query, or* $(\mathsf{challenge}, x^*)$.

New Keys. *Every time new keys* $k_{x\|0}, k_{x\|1}$ *are defined, right before that, a corresponding* (PRG, x) *event is appended to* view*. Concretely, a query* $(\mathsf{corrupt}, x)$ *or* $(\mathsf{challenge}, x^*)$ *in* view *automatically causes also entries* (PRG, x') *for all proper prefixes* x' *of* x *for which no* PRG *query has been issued yet, to be appended immediately after that* $(\mathsf{corrupt}, x)$ *entry. Entries* (PRG, x) *defined at the same time are ordered in* view *with keys closer to the root (i.e., with shorter* x*) first.*

Corrupted Key. *A key* (key, x, k_x) *as a response to a corruption query.*

Challenge Key. *The response to the final challenge query, in the form* $(\mathsf{challenge}, x^*, k)$ *(i.e., depending on* b_{ggmprf} *with either* k *being the real key* k_{x^*} *or a random value). This event comes after the corresponding* (PRG, x) *events which are triggered by the challenge query.*

Decision Bit. *The final output bit* b_A *of* A*, in the form* $(\mathsf{guess}, b_\mathsf{A})$*. This event is the last in* view*, and we may write* $\mathrm{out}_\mathsf{A}(\mathsf{view})$ *to denote that bit* b_A.

We are now ready to formulate and prove our main result:

Theorem 3.2.5. *Let* $\mathsf{G} : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ *be a PRG. Then*

- *for every* GGMPRF *adversary* A *that runs in time* t_A *and makes at most* $\mathcal{Q}_{\mathsf{corrupt}}$ *corrupt queries,*
- *for every* GGMPRF *depth* d *and every* $\gamma \in (0, 1]$,

there is a PR *adversary* B *that runs in time* t_B*, makes at most* $\mathcal{Q}_\mathsf{B}$ *oracle queries, and for which*

$$\mathrm{Adv}^{\mathsf{PR}}_{\mathsf{G},\mathsf{B}}(\lambda) \;\geq\; \frac{1}{2d} \cdot \left(\mathrm{Adv}^{\mathsf{GGMPRF}}_{\mathsf{G},\mathsf{A},d}(\lambda) - \gamma\right), \tag{3.4}$$

where

$$t_\mathsf{B} \;\lessapprox\; \left(2 \cdot \ln(2 \cdot \mathcal{T}/\gamma) \cdot \mathcal{T}^4/\gamma\right) \cdot t_\mathsf{A} \quad \textit{and} \quad \mathcal{Q}_\mathsf{B} \;\leq\; 2 \cdot \ln(2 \cdot \mathcal{T}/\gamma) \cdot \mathcal{T}^3/\gamma \tag{3.5}$$

with $\mathcal{T} \leq ((d+2) \cdot \mathcal{Q}_{\mathsf{corrupt}} + 2)$.

Before we proceed to a proof, we notice that Theorem 3.2.5 implies asymptotic security when setting γ accordingly:

Corollary 3.2.6 (G secure $\Rightarrow$ GGM PRF secure PC-PRF)**.** *If* G *is a secure PRG (as in Definition 2.7.1) and* $d = d(\lambda)$ *is a polynomial, then the GGM PRF* F_d *is an indistinguishable PC-PRF (as in Definition 2.7.3).*

Proof of Corollary 3.2.6. Assume for contradiction that there is a polynomial-time adversary A′ against the PC-PRF indistinguishability with non-negligible advantage. By Remark 3.2.3, this immediately yields a polynomial-time GGMPRF adversary A with (the same) non-negligible advantage $\varepsilon_\mathsf{A} := \mathrm{Adv}^{\mathsf{GGMPRF}}_{\mathsf{G},\mathsf{A},d}(\lambda)$. Since ε_A is non-negligible, there exists a polynomial p such that for infinitely many values of λ, we have $\varepsilon_\mathsf{A} \geq 1/p(\lambda)$.

Now set $\gamma = 1/(2p(\lambda))$ and invoke Theorem 3.2.5. We obtain a PR adversary B with (by Eq. (3.5)) polynomial runtime and non-negligible advantage

$$\mathrm{Adv}^{\mathsf{PR}}_{\mathsf{G},\mathsf{B}}(\lambda) \overset{Eq.\ (3.4)}{\geq} \frac{1}{2d} \cdot \left(\varepsilon_{\mathsf{A}} - \gamma\right) \overset{(*)}{\geq} \frac{1}{2d} \cdot \left(\frac{1}{p(\lambda)} - \gamma\right) = \frac{1}{4d \cdot p(\lambda)},$$

where $(*)$ holds (only) for infinitely many λ. □

Proof of Theorem 3.2.5. Fix A and d. In the following, we will consider a number of hybrid games, with Game ggmprf being the original GGMPRF experiment. Denoting with out_i the output of $\mathrm{Game}\ i$, we trivially get

$$\Pr[\mathrm{out}_{\mathsf{ggmprf}} = 1] = \mathrm{Adv}^{\mathsf{GGMPRF}}_{\mathsf{G},\mathsf{A},d}(\lambda) + 1/2. \tag{3.6}$$

Moving on, we will formulate $\mathrm{Game}\ i$ (for $0 \leq i \leq d$) in a (for us) convenient way, see Algorithm 11. This formulation outsources the bulk of the game into the sampling of A's view view from a suitable distribution $\mathcal{D}^{\mathsf{view}}_{i,b_{\mathsf{ggmprf}}}$. In our upcoming refinements, we will only change $\mathcal{D}^{\mathsf{view}}_{i,b_{\mathsf{ggmprf}}}$ and investigate the

Algorithm 11: $\mathrm{Game}\ i$, with the bulk of the work outsourced into the sampling from $\mathcal{D}^{\mathsf{view}}_{i,b_{\mathsf{ggmprf}}}$.

```
1 $b_{\mathsf{ggmprf}} \leftarrow \{0,1\}$
2 view $\leftarrow \mathcal{D}^{\mathsf{view}}_{i,b_{\mathsf{ggmprf}}}$          // view has the format from Definition 3.2.4
3 return $[b_{\mathsf{ggmprf}} = \mathrm{out}_{\mathsf{A}}(\mathsf{view})]$          // returns 1 iff $b_{\mathsf{ggmprf}} = \mathrm{out}_{\mathsf{A}}(\mathsf{view})$
```

effects on out_i.

The distributions $\mathcal{D}^{\mathsf{view}}_{i,b_{\mathsf{ggmprf}}}$. To define the distribution $\mathcal{D}^{\mathsf{view}}_{i,b_{\mathsf{ggmprf}}}$ for $\mathrm{Game}\ i$ with $i \in [d]_0$, we use the following notation:

- $\mathcal{D}^{\mathsf{view}}_{\mathsf{ggmprf},b_{\mathsf{ggmprf}}}$ is the distribution of A-views (as in Definition 3.2.4) that is induced by running the GGMPRF experiment with challenge bit b_{ggmprf} (that decides whether A is challenged with k_{x^*} or a random key).
- len(view) is the length of a given A-view view (measured in events).
- pfx$_t$(view) outputs the prefix of view up to (and including) the t-th event (as defined in Section 2.1).
- lastpre$_t$(view) on input view $= (\mathsf{ev}_1, \ldots, \mathsf{ev}_{\mathcal{T}})$ outputs the largest index $t' \leq t$ such that event $\mathsf{ev}_{t'}$ defines a key on the path from the root to x^*, i.e.,

$$\mathsf{lastpre}_t(\mathsf{view}) := \max\left(\left\{ t' \;\middle|\; \begin{array}{c} \mathsf{ev}_{t'} = (\mathsf{PRG}, x) \\ \wedge\, t' \leq t \,\wedge\, x <_{\mathsf{pfx}} x^* \end{array} \right\} \cup \{0\}\right).$$

- $B \in \mathbb{N}$ is a bound on the number of repetitions of Lines 6 to 13 in Algorithm 12 for each t. In case of B unsuccessful repetitions for one t, the whole algorithm outputs $\perp$. We will fix a suitable value for B later.

Now consider Algorithm 12. Our distribution $\mathcal{D}^{\mathsf{view}}_{i,b_{\mathsf{ggmprf}}}$ will be defined almost like $\mathcal{D}^{\mathsf{view}}_{\mathsf{ggmprf},b_{\mathsf{ggmprf}}}$ (i.e., like A's view in a GGMPRF run), but will additionally replace PRG evaluations by random sampling as indicated by index i and use rewindings at every step. Concretely, fix an i and consider Algorithm 12, which programmatically defines $\mathcal{D}^{\mathsf{view}}_{i,b_{\mathsf{ggmprf}}}$ as its output.

For an adversary A with view $\mathsf{view} = (\mathsf{ev}_1, \dots, \mathsf{ev}_{\mathcal{T}})$ we denote by *rewinding* the adversary to time t the cutting-off of the view at point $t-1$, i.e. $(\mathsf{ev}_1, \dots, \mathsf{ev}_{t-1})$ and resetting the adversary to the state it had directly before ev_t. (By keeping track of A's state throughout our rewindings, this will always be possible.)

We say we *resample* from point t (after rewinding A to t) if we rerun A from t onwards, using fresh challenger random coins from that point onwards. In some cases, we will also rerun A with a specific replacement (e.g., an embedded computational challenge) in ev_t (if ev_t contains an answer to one of A's previous queries). The view resulting from rewinding to t and then resampling from point t is $\mathsf{view}' = (\mathsf{ev}_1, \dots, \mathsf{ev}_{t-1}, \mathsf{ev}'_t, \dots, \mathsf{ev}'_{\mathcal{T}'})$.

Algorithm 12: Sampler for $\mathcal{D}^{\mathsf{view}}_{i,b_{\mathsf{ggmprf}}}$

```
Input: $i \in \{0, \dots, d\}, b_{\mathsf{ggmprf}} \in \{0,1\}, B \in \mathbb{N}$        // $\mathsf{len}, \mathsf{lastpre}_{i,t}, B$ described in proof
1  $\mathsf{view}_0 \leftarrow \mathcal{D}^{\mathsf{view}}_{\mathsf{ggmprf},b_{\mathsf{ggmprf}}}$
2  $\mathcal{T} := \mathsf{len}(\mathsf{view}_{\mathsf{GGMPRF}})$                    // Length of $\mathsf{view}_{\mathsf{GGMPRF}}$ (in entries)
3  for $t := 1$ to $\mathcal{T}$ do
4      Write $\mathsf{view}_{t-1} = (\mathsf{ev}_{t-1,1}, \dots, \mathsf{ev}_{t-1,\mathcal{T}})$
5      repeat                               // Output $\perp$ if $B$ repetitions fail for this $t$
6          Rewind adversary to point $t$
7          if $\mathsf{ev}_{t-1,t} = (\mathsf{PRG}, x)$ with $|x| \leq i$ and $\mathsf{lastpre}_t(\mathsf{view}_{t-1}) = t$ then      // Checks if
             $\mathsf{ev}_{t-1,t}$ defines a PRG evaluation to be replaced by random
8              Sample fresh $k_{x\|0}, k_{x\|1} \leftarrow \{0,1\}^\lambda$                 // Fresh independent keys
9              Resample from point $t+1$ to obtain
                 $\mathsf{view}_t = (\mathsf{ev}_{t-1,0}, \dots, \mathsf{ev}_{t-1,t-1}, \mathsf{ev}_{t-1,t}, \mathsf{ev}_{t,t+1}, \dots, \mathsf{ev}_{t,\tau})$
10         else
11             Resample from point $t$ to obtain $\mathsf{view}_t = (\mathsf{ev}_{t-1,0}, \dots, \mathsf{ev}_{t-1,t-1}, \mathsf{ev}_{t,t}, \dots, \mathsf{ev}_{t,\tau})$
12         end
13     until $\mathsf{lastpre}_t(\mathsf{view}_t) = \mathsf{lastpre}_t(\mathsf{view}_{t-1})$ and $\mathsf{len}(\mathsf{view}_t) = \mathsf{len}(\mathsf{view}_{t-1})$
14 end
15 return $\mathsf{view}_{\mathcal{T}}$
```

Having defined our hybrid distributions $\mathcal{D}^{\mathsf{view}}_{i,b_{\mathsf{ggmprf}}}$, we will additionally consider the distribution $\widetilde{\mathcal{D}^{\mathsf{view}}_{0,b_{\mathsf{ggmprf}}}}$ which is defined as the output of a variant of Algorithm 12 for $i = 0$ without a bound B on the runtime. (Hence, $\widetilde{\mathcal{D}^{\mathsf{view}}_{0,b_{\mathsf{ggmprf}}}}$ will not be efficiently sampleable in general.) We will first show that the distribution $\widetilde{\mathcal{D}^{\mathsf{view}}_{0,b_{\mathsf{ggmprf}}}}$ coincides with the distribution of views in Game ggmprf. Here we will use our results from Section 3.1, namely Lemma 3.1.2, for the distribution $\mathcal{D} = \mathcal{D}^{\mathsf{view}}_{\mathsf{ggmprf},b_{\mathsf{ggmprf}}}$ where functions h_t on input view will output the $(t-1)$-sized prefix of view, and resampling conditioned on $h_t(\mathsf{view})$ simply means rewinding and rerunning from point t. The stopping conditions $g_t(\mathsf{view})$ will preserve (1) the value of $\mathsf{lastpre}_t(\mathsf{view})$, and (2) the length of view. Intuitively, preserving $\mathsf{lastpre}_t(\mathsf{view})$ implies that "PRG embedding slots" along the path to the challenge x^* defined prior to point t remain the same. This

$$\begin{tikzcd}
\mathrm{Game}\ \mathsf{ggmprf} \arrow[r, dash, "\equiv"] & \widetilde{\mathcal{D}^{\mathsf{view}}_{0,b_{\mathsf{ggmprf}}}} \arrow[d, dash, "\overset{s}{\approx}_{\gamma}"] & & & & \\
& \mathcal{D}^{\mathsf{view}}_{0,b_{\mathsf{ggmprf}}} \arrow[r, dash, "\overset{c}{\approx}"] & \mathcal{D}^{\mathsf{view}}_{1,b_{\mathsf{ggmprf}}} \arrow[r, dash, "\overset{c}{\approx}"] & \mathcal{D}^{\mathsf{view}}_{2,b_{\mathsf{ggmprf}}} \arrow[r, dash, "\overset{c}{\approx}"] & \cdots \arrow[r, dash, "\overset{c}{\approx}"] & \mathcal{D}^{\mathsf{view}}_{d,b_{\mathsf{ggmprf}}}
\end{tikzcd}$$

Figure 3.1: Sequence of hybrids. Perfect indistinguishability ($\equiv$) is shown in Proposition 1, statistical distance ($\overset{s}{\approx}_{\gamma}$) is shown in Proposition 2, and computational indistinguishability ($\overset{c}{\approx}$) is shown in Proposition 3.

implies that no preimages of previously embedded PRG images have to be revealed, and the rewinding did not "undo" any of the progress made so far.

Again using our results from Section 3.1, namely Lemma 3.1.3 with similar interpretation as above, we will then choose the bound B such that the probability of an abort in $\mathcal{D}^{\mathsf{view}}_{0,b_{\mathsf{ggmprf}}}$ can be bounded by γ. Then we will show that A has no advantage in $\mathrm{Game}\ d$ since the view sampled according to $\mathcal{D}^{\mathsf{view}}_{d,b_{\mathsf{ggmprf}}}$ is independent of b_{ggmprf}. Finally, we will argue that $\mathcal{D}^{\mathsf{view}}_{0,b_{\mathsf{ggmprf}}}$ is computationally indistinguishable from $\mathcal{D}^{\mathsf{view}}_{d,b_{\mathsf{ggmprf}}}$ by the pseudorandomness of G. Combining these results will allow us to conclude the proof. Our path along this sequence of hybrids can be seen in Figure 3.1.

Proposition 1. $\mathcal{D}^{\mathsf{view}}_{\mathsf{ggmprf},b_{\mathsf{ggmprf}}} \equiv \widetilde{\mathcal{D}^{\mathsf{view}}_{0,b_{\mathsf{ggmprf}}}}$.

Proof. This follows from Lemma 3.1.2, where $\mathcal{D} = \mathcal{D}^{\mathsf{view}}_{\mathsf{ggmprf},b_{\mathsf{ggmprf}}}$, $h_t(\mathsf{view}_s) = (\mathsf{ev}_{s,1}, \dots, \mathsf{ev}_{s,t-1})$, and $g_t(\mathsf{view}_s) = (\mathsf{lastpre}_t(\mathsf{view}_s), \mathsf{len}(\mathsf{view}_s))$. We note that for $i = 0$, the **if** on Line 7 never returns true, and thus the sampling procedure always enters the **else** branch which behaves just as in Lemma 3.1.2. □

Proposition 2 (Abort probability)**.** *For*

$$B := 2 \cdot \ln\left(2 \cdot ((d+2) \cdot \mathcal{Q}_{\mathsf{corrupt}} + 2)/\gamma\right) \cdot \left((d+2) \cdot \mathcal{Q}_{\mathsf{corrupt}} + 2\right)^3 / \gamma,$$

we have $\Pr[\bot \leftarrow \mathcal{D}^{\mathsf{view}}_{0,b_{\mathsf{lkh}}}] \leq \gamma$.

Proof. To prove this claim, we consider the process of sampling from $\mathcal{D}^{\mathsf{view}}_{0,b_{\mathsf{lkh}}}$ according to Algorithm 12 and bound the probability that any of the iterations in the "**for**" loop runs the "**repeat**" loop more than B times.

By Lemma 3.1.3, with $g_t(\mathsf{view}) = (\mathsf{lastpre}_t(\mathsf{view}), \mathsf{len}(\mathsf{view}))$ and $h_t(\mathsf{view}) = (\mathsf{ev}_1, \dots, \mathsf{ev}_t)$ for $\mathsf{view} = (\mathsf{ev}_1, \dots, \mathsf{ev}_{\mathsf{len}(\mathsf{view})})$, it holds that for any $\gamma \in (0, 1]$ (thus in particular the γ from the theorem statement)

$$\Pr[\ \forall t \in [\mathcal{T}] : T^{\mathsf{rep}}_t \leq 2\mathcal{T} \cdot \ln(2\mathcal{T}/\gamma) \cdot |\mathcal{Y}|/\gamma\ ] \ \geq\ 1 - \gamma \tag{3.7}$$

where T^{rep}_t denotes the number of runs of the "**repeat**" loop in the t-th iteration of the "**for**" loop. We note that

$$\mathsf{len}(\mathsf{view}) \leq \underbrace{\mathcal{Q}_{\mathsf{corrupt}} + 1}_{\text{Query Events}} + \underbrace{d \cdot \mathcal{Q}_{\mathsf{corrupt}}}_{\text{PRG Events}} + \underbrace{\mathcal{Q}_{\mathsf{corrupt}} + 1}_{\text{Corr./Chal. Key Events}}$$

for any view resulting from a run of an adversary that makes at most $\mathcal{Q}_{\mathsf{corrupt}}$ constrained key queries. This means that $\mathsf{len}(\mathsf{view})$ can take values up to $\mathcal{T}_{\max} = (d+2) \cdot \mathcal{Q}_{\mathsf{corrupt}} + 2$. Furthermore, $\mathsf{lastpre}_t(\mathsf{view})$

takes values from 0 to $\mathrm{len}(\mathrm{view})$. Thus, we can bound the size of the range $\mathcal{Y}$ of the g_t with $|\mathcal{Y}| \leq ((d+2)\cdot \mathcal{Q}_{\mathsf{corrupt}} + 2)^2$.

Plugging this into Eq. (3.7) yields

$$\Pr\left[\, \forall t \in [\mathcal{T}] : T_t^{\mathsf{rep}} \leq 2\mathcal{T} \cdot \ln(2\mathcal{T}/\gamma) \cdot ((d+2)\cdot \mathcal{Q}_{\mathsf{corrupt}} + 2)^2 / \gamma \,\right] \;\geq\; 1-\gamma. \tag{3.8}$$

Thus, using the bound for $\mathrm{len}(\mathrm{view})$ for $\mathcal{T}$ again, i.e. $\mathcal{T} \leq (d+2)\cdot \mathcal{Q}_{\mathsf{corrupt}} + 2$, gives

$$\Pr\left[\, \forall t \in [\mathcal{T}] : T_t^{\mathsf{rep}} \leq B \,\right] \;\geq\; 1-\gamma. \tag{3.9}$$

which yields the claim. □

Proposition 3 (PR $\Rightarrow \mathcal{D}^{\mathsf{view}}_{0,b_{\mathsf{ggmprf}}} \overset{c}{\approx} \mathcal{D}^{\mathsf{view}}_{d,b_{\mathsf{ggmprf}}}$). *Let B be as in Claim 2. If* G *is* PR *secure, then the distributions $\mathcal{D}^{\mathsf{view}}_{0,b_{\mathsf{ggmprf}}}$ and $\mathcal{D}^{\mathsf{view}}_{d,b_{\mathsf{ggmprf}}}$ are computationally indistinguishable. More precisely, there exists a* PR *adversary* C *that runs in time t_{C} and makes $\mathcal{Q}_{\mathsf{C}}$ oracle queries, such that*

$$\mathrm{Adv}^{\mathsf{PR}}_{\mathsf{G,C}}(\lambda) \;=\; \frac{1}{2d}\cdot\left(\mathrm{Adv}^{\mathsf{GGMPRF},\,0}_{\mathsf{G,A},d}(\lambda) - \mathrm{Adv}^{\mathsf{GGMPRF},\,d}_{\mathsf{G,A},d}(\lambda)\right) \tag{3.10}$$

$$t_{\mathsf{C}} \;\lessapprox\; \left(2\cdot\ln(2\cdot\mathcal{T}_{\max}/\gamma)\cdot\mathcal{T}^4_{\max}/\gamma\right)\cdot t_{\mathsf{A}} \quad \textit{and} \quad \mathcal{Q}_{\mathsf{C}} \;\leq\; 2\cdot\ln(2\cdot\mathcal{T}/\gamma)\cdot\mathcal{T}^3/\gamma \tag{3.11}$$

with $\mathcal{T} \leq \mathcal{T}_{\max} = (d+2)\cdot\mathcal{Q}_{\mathsf{corrupt}} + 2$.

Proof. To generate a sample $\mathsf{view}_{\mathcal{T}}$, our PR adversary C modifies the procedure of Algorithm 12 by first sampling an index $i^* \leftarrow [d]$ and a bit $b_{\mathsf{pr}} \leftarrow \{0,1\}$ uniformly at random, and then embedding a PR challenge in the "**if**" clause in the "**repeat**" loop, see Algorithm 13. C outputs 0 if A succeeds and 1 else.

Note that the key for node $\mathsf{pfx}_{i^*-1}(x^*)$ is sampled freshly and uniformly at random. Thus, we have that for $b_{\mathsf{pr}} = 0$ and $i^* = i$ the modified algorithm samples from exactly the same distribution as Algorithm 12 on input $i-1$ (and same $b_{\mathsf{ggmprf}} \in \{0,1\}, B \in \mathbb{N}$), and for $b_{\mathsf{pr}} = 1$ and $i^* = i$ from the same distribution as Algorithm 12 on input i (and same $b_{\mathsf{ggmprf}} \in \{0,1\}, B \in \mathbb{N}$). We obtain for the advantage of C:

$$\begin{aligned}
\mathrm{Adv}^{\mathsf{PR}}_{\mathsf{G,C}}(\lambda) &= \Pr[b_{\mathsf{C}} = b_{\mathsf{pr}}] - \frac{1}{2}\\
&= \frac{1}{2}\cdot\left(\Pr[b_{\mathsf{C}} = 0 \mid b_{\mathsf{pr}} = 0] - \Pr[b_{\mathsf{C}} = 0 \mid b_{\mathsf{pr}} = 1]\right)\\
&= \frac{1}{2}\cdot\frac{1}{d}\cdot\sum_{i\in[d]}\left(\Pr\left[b_{\mathsf{C}} = 0 \,\middle|\, \begin{array}{c} b_{\mathsf{pr}} = 0 \\ \wedge\, i^* = i \end{array}\right] - \Pr\left[b_{\mathsf{C}} = 0 \,\middle|\, \begin{array}{c} b_{\mathsf{pr}} = 1 \\ \wedge\, i^* = i \end{array}\right]\right)\\
&= \frac{1}{2d}\cdot\sum_{i\in[d]}\left(\Pr[\mathrm{out}_{i-1} = 1] - \Pr[\mathrm{out}_i = 1]\right)\\
&= \frac{1}{2d}\cdot\left(\mathrm{Adv}^{\mathsf{GGMPRF},\,0}_{\mathsf{G,A},d}(\lambda) - \mathrm{Adv}^{\mathsf{GGMPRF},\,d}_{\mathsf{G,A},d}(\lambda)\right).
\end{aligned}$$

C runs A at most $\mathcal{T}\cdot B$ times. Bounding $\mathcal{T} = \mathrm{len}(\mathrm{view}_0)$ by $\mathcal{T}_{\max} = (d+2)\cdot\mathcal{Q}_{\mathsf{corrupt}} + 2$ (see proof of Claim 2) and plugging in $B = \frac{2\mathcal{T}\cdot\ln(2\mathcal{T}/\gamma)\cdot((d+2)\cdot\mathcal{Q}_{\mathsf{corrupt}}+2)^2}{\gamma}$ leads to the claimed bound on t_{C}. (We assume the time complexity of random sampling, PR oracle calls and PRG evaluations to be significantly smaller than t_{A} and thus neglect the corresponding terms in our bound.)

Algorithm 13: Variant of Algorithm 12 for sampling from $\mathcal{D}^{\mathsf{view}}_{i^*-1+b_{\mathsf{pr}},b_{\mathsf{ggmprf}}}$ given oracle access to a PR challenger with challenge bit b_{pr}. The functions $\mathsf{len}, \mathsf{lastpre}_t$ are as described in the proof, $B \in \mathbb{N}$ is as in Proposition 2.

```
1  $i^* \leftarrow \{1,\dots,d\}, b_{\mathsf{ggmprf}} \leftarrow \{0,1\}$
2  $\mathsf{view}_0 \leftarrow \mathcal{D}^{\mathsf{view}}_{\mathsf{ggmprf},b_{\mathsf{ggmprf}}}$
3  $\mathcal{T} := \mathsf{len}(\mathsf{view}_0)$                                // Length of $\mathsf{view}_{\mathsf{GGMPRF}}$ (in entries)
4  for $t := 1$ to $\mathcal{T}$ do
5      Write $\mathsf{view}_{t-1} = (\mathsf{ev}_{t-1,1},\dots,\mathsf{ev}_{t-1,\mathcal{T}})$
6      repeat                                 // Output $\perp$ if $B$ repetitions fail for this $t$
7          Rewind adversary to point $t$
8          if $\mathsf{ev}_{t-1,t} = (\mathsf{PRG}, x)$ with $|x| \leq i^*$ and $\mathsf{lastpre}_t(\mathsf{view}_{t-1}) = t$ then
               // Checks if $\mathsf{ev}_{t,t}$ defines a PRG evaluation to be replaced by random
9              if $|x| = i^*$ then
10                 Request fresh PR challenge $(k_0^*, k_1^*)$ from PR challenger          // Fresh PR
                    challenge
11                 Set $(k_{x\|0}, k_{x\|1}) := (k_0^*, k_1^*)$
12             else
13                 Sample fresh $k_{x\|0}, k_{x\|1} \leftarrow \{0,1\}^\lambda$                // Fresh independent keys
14             end
15             Resample from point $t+1$ to obtain
                $\mathsf{view}_t = (\mathsf{ev}_{t-1,0},\dots,\mathsf{ev}_{t-1,t-1},\mathsf{ev}_{t-1,t},\mathsf{ev}_{t,t+1}\dots,\mathsf{ev}_{t,\tau})$
16         else
17             Resample from point $t$ to obtain $\mathsf{view}_t = (\mathsf{ev}_{t-1,0},\dots,\mathsf{ev}_{t-1,t-1},\mathsf{ev}_{t,t},\dots,\mathsf{ev}_{t,\tau})$
18         end
19     until $\mathsf{lastpre}_t(\mathsf{view}_t) = \mathsf{lastpre}_t(\mathsf{view}_{t-1})$ and $\mathsf{len}(\mathsf{view}_t) = \mathsf{len}(\mathsf{view}_{t-1})$
20 end
21 return $\mathsf{view}_{\mathcal{T}}$
```

For the upper bound on the number of oracle calls $\mathcal{Q}_{\mathsf{C}}$, note that for each possible choice of i^* there is only one t such that $\mathsf{ev}_{t-1,t} = (\mathsf{PRG}, x)$ with $|x| = i^*$ and $\mathsf{lastpre}_t(\mathsf{view}_{t-1}) = t$. Thus, the inner **if** clause will apply only in one of the **for** iterations, which implies that there are as many PR calls as there are iterations of the **repeat** loop for that t. Hence, the reduction makes at most B calls to the PR oracle. ☐

Proposition 4. *$\mathcal{D}^{\mathsf{view}}_{d,b_{\mathsf{ggmprf}}}$ and b_{ggmprf} are independent, so $\mathrm{Adv}^{\mathsf{GGMPRF},\,d}_{\mathsf{G,A},d}(\lambda) = 0$.*

Proof. Recall that b_{ggmprf} is only used when responding to the challenge query, which by assumption is the last query the adversary makes. Hence, neither the abort probability nor any of the events in $\mathsf{view}_{\mathcal{T}}$ before the very last events $(\mathsf{challenge}, x^*, k)$ and $(\mathsf{guess}, b_{\mathsf{A}})$ depend on b_{ggmprf}. The latter also implies that the values for lastpre and len are independent of b_{ggmprf} for all t. As the last two events of the view (that are the only ones carrying information about b_{ggmprf}) are cut off when rewinding and resampling, no information about b_{ggmprf} is carried from view_{t-1} to view_t for any t. Furthermore, in the final view $\mathsf{view}_{\mathcal{T}}$ the challenge key k_{x^*} is sampled independently and uniformly at random, hence k has the same distribution for both cases $b_{\mathsf{ggmprf}} = 0$ and $b_{\mathsf{ggmprf}} = 0$. Thus, A has no advantage in distinguishing k_{x^*}

from a random independent key. □

To finish the proof of the theorem, it only remains to combine the above claims. In particular, we define the adversary B exactly as C from Proposition 3. The bound on the runtime of B follows immediately and for the advantage of B we have

$$\begin{aligned}
\mathrm{Adv}^{\mathsf{PR}}_{\mathsf{G,B}}(\lambda) &= \frac{1}{2d} \cdot \left(\mathrm{Adv}^{\mathsf{GGMPRF},\,0}_{\mathsf{G,A},d}(\lambda) - \mathrm{Adv}^{\mathsf{GGMPRF},\,d}_{\mathsf{G,A},d}(\lambda)\right) \\
&\geq \frac{1}{2d} \cdot \left(\mathrm{Adv}^{\widetilde{\mathsf{GGMPRF}},\,0}_{\mathsf{G,A},d}(\lambda) - \mathrm{Adv}^{\mathsf{GGMPRF},\,d}_{\mathsf{G,A},d}(\lambda) - \gamma\right) \\
&\geq \frac{1}{2d} \cdot \left(\mathrm{Adv}^{\mathsf{GGMPRF}}_{\mathsf{G,A},d}(\lambda) - \mathrm{Adv}^{\mathsf{GGMPRF},\,d}_{\mathsf{G,A},d}(\lambda) - \gamma\right) \\
&\geq \frac{1}{2d} \cdot \left(\mathrm{Adv}^{\mathsf{GGMPRF}}_{\mathsf{G,A},d}(\lambda) - \gamma\right).
\end{aligned}$$

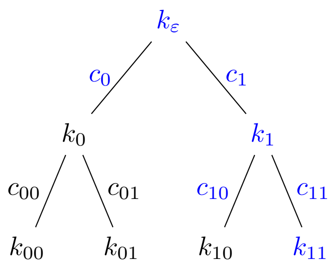
□

3.3 Adaptive Security for LKH

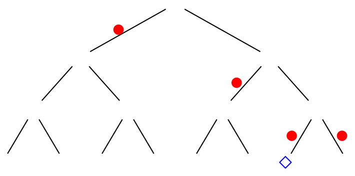
Overview. The main application we have in mind in this section is a multicast key distribution protocol called the Logical Key Hierarchy (LKH) [WHA98; WGL00; Can+99]; more precisely, we consider the rectified version by Panjwani [Pan07]. Our strategy can easily be generalized to minor modifications of LKH and therefore we do not focus on specific implementation details. Rather, we provide a very brief high-level description of the protocol as well as the security guarantees we aim to guarantee. More broadly, we believe that our results provide the core techniques to prove adaptive security also for multicast key agreement as defined in [BDT22], as well as (various versions of) the related "TreeKEM" protocol [BBR18; Kle+21] for (public-key) continuous group key agreement (CGKA).

Multicast Key Distribution (MKD). A protocol for multicast key distribution (MKD, see [Pan07]) is a server-aided secret-key protocol that enables a dynamically changing group of users to securely communicate over a broadcast channel. In an initial registration step, it is assumed that each user establishes a secret key with the server; this key infrastructure setup is however outside the protocol specification. The server then uses these shared secret keys to communicate a group key to the current set of user. Upon a join/leave request, the server refreshes the group key and sends rekey messages to the new set of users, which allow each user to derive the new group key. In the security experiment, the adversary can request join and leave operations for arbitrary users fully adaptively and learns all keys of removed users. Finally, it can request a challenge and in return obtains either the real group key or a random independent key.

Logical Key Hierarchy (LKH). A trivial MKD protocol would be to simply encrypt a freshly sampled group key to all current users after each membership change. However, for large groups this does not scale well, as it requires a linear number of encryptions. A smarter approach is taken in the Logical Key Hierarchy (LKH) protocol, as proposed in [WHA98; WGL00; Can+99]. We will consider the rectified version of LKH by Panjwani [Pan07]: LKH is based on a binary tree structure, where each node is associated with a secret key k_x and edges represent secret-key encryptions $c_{x\|b}$ of the parent key k_x under the child key $k_{x\|b}$ (see binary tree notation in Section 2.1). The keys associated to leaves in the



(a) Adding user 4 to a group of 3 users. The keys and ciphertexts that got refreshed in this process are denoted in blue.



(b) A depth-3 binary tree with red pebbles (●) at the edges c_0, c_{10}, c_{110}, and c_{111}. This configuration occurs when pebbling this graph with the pebbling algorithm (see Algorithm 14) at some step i^*. The node leaf$_{3,i^*}$ is marked with a blue diamond ($\diamond$).

Figure 3.2: (a) Adding a user in LKH. (b) A pebbling configuration occuring in the recursive pebbling strategy from Algorithm 14.

tree belong to members participating in the multicast key distribution, the key k_ε associated to the root is used as the group key. Users can be added to or removed from the group, which leads to a state update where all the keys and ciphertexts associated with nodes and edges on the path from the user's leaf to the root are refreshed (except for the edge attached to the leaf in case of a remove), and also the edges connecting these nodes to co-path nodes are refreshed (see Figure 3.2a). Note that in contrast to the trivial protocol, each remove or add operation only requires an update of a logarithmic (in the size of the group) number of ciphertexts.

3.3.1 Pebbling for LKH

Similarly to the case of GGM, the graphs that occur in the security game of LKH are trees. But now, we are interested in randomizing the key at the root of the binary tree structure. This root key can be derived from any of the leaf keys (and publicly available ciphertexts). Hence, there are now many paths to the root which we need to take into account in order to randomize the root key. We therefore will build upon the intricate "edge pebbling" strategy of [Jaf+17] to randomize k_ε, only with guesses replaced by rewindings.

Edge pebbling is a multi-round game on a graph—in our case a binary tree—, where in each step a pebble can be placed on or removed from an edge. The goal is to "pebble" the tree, which means to reach a pebbling configuration where all edges incident on the root are pebbled. The rule is the following.

Edge-pebbling rule. We can at any point add or remove a pebble on an edge c_x when all of k_x's incoming edges (i.e., edges $c_{x\|0}$ and $c_{x\|1}$, if exist) are pebbled.

In particular, we can pebble or unpebble leaf edges at any point. It is easy to see that we can pebble a binary tree in $2^{d+1} - 2$ steps (by pebbling all edges of the tree, level by level from the leaves to the root). Aiming to reduce the number of pebbles (i.e., the maximum number of pebbled edges at any given point in time), one can observe that a binary tree of depth d can be pebbled in $\Theta(2^{2d})$ steps with only $2d$ pebbles, essentially by a straightforward recursion and removing all used pebbles after pebbling upwards (see Algorithm 14, adapted from [Jaf+17, Algorithm 5]). While for our approach the number of pebbles is not that relevant, this recursive strategy will nevertheless turn out useful. In the following we will derive some useful properties of this strategy.

Algorithm 14: A recursive pebbling algorithm. The “unpebbling” steps use that all pebbling steps are reversible.

```
Input: A depth-$d$ binary graph with nodes $k_x$ ($x \in \{0,1\}^{\leq d}$) and edges $c_x$ ($1 \leq |x| \leq d$)
1  if $d = 1$ then
2      Pebble $c_0$ and $c_1$                                   // $k_0$ and $k_1$ are leaves
3  else
4      Recursively pebble the subgraph rooted at $k_0$            // Pebbles $c_{00}$ and $c_{01}$
5      Pebble $c_0$                                              // Incoming edges of $k_0$ pebbled
6      Recursively unpebble subgraph rooted at $k_0$
7      Recursively pebble the subgraph rooted at $k_1$            // Pebbles $c_{10}$ and $c_{11}$
8      Pebble $c_1$                                              // Incoming edges of $k_1$ pebbled
9      Recursively unpebble subgraph rooted at $k_1$
10 end
```

Definition 3.3.1 (Pebbling time)**.** *For $d \in \mathbb{N}$, let T_d be the pebbling time for depth-d binary trees, i.e., the runtime (measured in the number of times a basic pebbling rule is applied) of the pebbling algorithm in Algorithm 14 on a depth-d binary tree.*

Lemma 3.3.2. *We have $T_d = (2/3) \cdot (2^{2d} - 1)$.*

Proof. $T_{d+1} = 4T_d + 2$ and $T_1 = 2$ follow immediately from the structure of pebbling algorithm in Algorithm 14 . The claimed closed form of T_d can then be proven, e.g., by induction. □

Definition 3.3.3 (Edge index set)**.** *For a given run of pebbling algorithm in Algorithm 14 on a depth-d binary tree as above, let $\mathsf{edges}_{d,i}$ denote the set of indices x of edges c_x pebbled after the i-th step (i.e., application of a pebbling rule).*

Hence, $\mathsf{edges}_{d,0} = \emptyset$ and $\mathsf{edges}_{d,T_d} = \{0,1\}$. A related observation to the following was already used in [Jaf+17].

Lemma 3.3.4. *For each $i \in [T_d]_0$, there is a leaf node k_x (for $x \in \{0,1\}^d$) such that both sets $\mathsf{edges}_{d,i-1}$ and $\mathsf{edges}_{d,i}$ (where we set $\mathsf{edges}_{d,-1} := \emptyset$) consist only of edge indices on the path from k_x to k_ε, or its co-path. Formally, for each i, there is an $x \in \{0,1\}^d$, such that for each $x' \| b \in \mathsf{edges}_{d,i-1} \cup \mathsf{edges}_{d,i}$ (for $x' \in \{0,1\}^{<d}$ and $b \in \{0,1\}$), we have $x' \leq_{\mathsf{pfx}} x$.*

Proof. For $d = 1$, this is obvious, as all edges in the tree are incident to the path from the leftmost leaf to the root.

Assume the statement holds for some fixed $d \geq 1$. We will show it holds for $d + 1$. Note that the edges c_0 and c_1 lie on the path or co-path of any leaf node. As the subtrees are pebbled or unpebbled recursively as a whole, there are no pebbles in the subtree at n_1 during the pebbling or unpebbling of n_0 and vice versa. Thus, for any of the subtrees, the edge sets at any point consist of the edge set of a subtree of depth d united with potentially the edges c_0 or c_1 which lie on the path or co-path of any leaf. Therefore, the statement also holds for $d + 1$. □

Definition 3.3.5. *In the situation of Lemma 3.3.4, let $\mathsf{leaf}_{d,i}$ be the lexicographically smallest such $x \in \{0,1\}^d$.*

The following corollary is an immediate consequence of Lemma 3.3.4.

Corollary 3.3.6. *For every i, we have $|\mathsf{edges}_{d,i}| \leq 2d$.*

The following result is an easy consequence of the recursive pebbling strategy.

Lemma 3.3.7. *For each $i \in [T_d]_0$, let x_i^* be the unique index in the symmetric difference of* $\mathsf{edges}_{d,i-1}$ *and* $\mathsf{edges}_{d,i}$*, i.e. $\{x_i^*\} := \mathsf{edges}_{d,i-1} \Delta \mathsf{edges}_{d,i}$. For each $x' \in \mathsf{edges}_{d,i-1}$, it holds that $|x'| \leq |x_i^*| + 1$.*

Proof. If Algorithm 14 is currently at a recursive depth such that $d = 1$ (i.e., x_i^* is incident to a leaf node) the statement follows immediately.

Recall that by Lemma 3.3.4, all edges in $\mathsf{edges}_{d,i-1}$ and $\mathsf{edges}_{d,i}$ are incident to the path from $\mathsf{leaf}_{d,i}$ to the root. Thus, when x_i^* is being pebbled (or unpebbled), the only other pebbled edges whose label could be longer than $|x_i^*| + 1$ must be in the subtree rooted at the bottom of x_i^*, as the algorithm first pebbles this subtree before pebbling x_i^* (and thus $\mathsf{leaf}_{d,i}$ must lie in this subtree). At the point when x_i^* is pebbled or unpebbled, the only other pebbled edges in the graph are thus incident to the path from x_i^* to the root (these edges have a label length $|x'| \leq |x_i^*|$), plus the two edges incident to x_i^* directly below x_i^* (these edges have $|x'| = |x_i^*| + 1$). ☐

Lemmas 3.3.4 and 3.3.7 immediately imply the following corollary.

Corollary 3.3.8. *Let x_i^* be as in Lemma 3.3.7. For each $i \in [T_d]_0$, it holds that all edges in* $\mathsf{edges}_{d,i-1} \cup \mathsf{edges}_{d,i}$ *are incident on the unique path from x_i^* to the root.*

As an example, Figure 3.2b depicts a state that occurs when pebbling a depth-3 binary tree with Algorithm 14. The set of pebbled edges at this point is $\mathsf{edges}_{3,i} = \{0, 10, 110, 111\}$ where edge c_{111} was pebbled in step i (i.e., $x_i^* = 111$), and $\mathsf{leaf}_{d,i} = 110$.

3.3.2 A Technical Lemma

In the following we introduce a technical lemma that will help us in proving closeness of some of the hybrid games for LKH security. In particular, we will be mixing two sampling algorithms, in each of which a bad event can occur. (Later, this bad event will correspond to exceeding a certain bound for repetitions of a loop.) We want to bound the probability for this bad event in a "hybrid" sampling algorithm that starts out as one of the algorithms, and then switches to the other when a specific event occurs.

Lemma 3.3.9. *Let $I1, I2, R1, R2$ be randomized algorithms. Let G be a function with the following properties:*

1. *for any sequence $X_0, \ldots X_{\mathcal{T}}$ s.t. $X_0 \leftarrow I1$, $X_t \leftarrow R1(X_{t-1})$ for $t = 1, \ldots, \mathcal{T}$, there exists exactly one t such that $G(X_t) = 1$.*

2. *for any sequence $X_0, \ldots X_{\mathcal{T}}$ s.t. $X_0 \leftarrow I2$, $X_t \leftarrow R2(X_{t-1})$ for $t = 1, \ldots, \mathcal{T}$, there exists exactly one t such that $G(X_t) = 1$.*

3. *for any index $i = 0, \ldots, \mathcal{T}$ it holds that*

$$\Pr_{\substack{X_0 \leftarrow I1 \\ \forall t=1,\ldots\mathcal{T}:\, X_t \leftarrow R1(X_{t-1})}} [G(X_i) = 1] = \Pr_{\substack{X_0 \leftarrow I2 \\ \forall t=1,\ldots\mathcal{T}:\, X_t \leftarrow R2(X_{t-1})}} [G(X_i) = 1]$$

Algorithm 15: Algorithm P1

```
1 $X_0 \leftarrow I1()$
2 for $t := 1$ to $\mathcal{T}$ do
3     $X_t \leftarrow R1(X_{t-1})$
4 end
```

Algorithm 16: Algorithm P2

```
1 $X_0 \leftarrow I2()$
2 for $t := 1$ to $\mathcal{T}$ do
3     $X_t \leftarrow R2(X_{t-1})$
4 end
```

Algorithm 17: Algorithm P3

```
1 $X_0 \leftarrow I1()$
2 $t = 0$
3 while $\neg G(X_t)$ do
4     $t++$
5     $X_t \leftarrow R1(X_{t-1})$
6 end
7 while $t < \mathcal{T}$ do
8     $t++$
9     $X_t \leftarrow R2(X_{t-1})$
10 end
```

Figure 3.3: Algorithms for Line 10

4. *for any index $i = 0, \dots, \mathcal{T}$, the following identity of distributions holds:*

$$\left\{ X_i \middle| \begin{array}{c} X_0 \leftarrow I1 \\ \forall t \in [\mathcal{T}]\colon X_t \leftarrow R1(X_{t-1}) \\ G(X_i) = 1 \end{array} \right\} \equiv \left\{ X_i \middle| \begin{array}{c} X_0 \leftarrow I2 \\ \forall t \in [\mathcal{T}]\colon X_t \leftarrow R2(X_{t-1}) \\ G(X_i) = 1 \end{array} \right\}$$

Now consider the algorithms $P1$, $P2$, and $P3$ from Algorithms 15 to 17, and let F be an event that can occur during the sampling processes I1, I2, R1, R2 such that $\Pr[F \text{ occurs in } P1] \leq \gamma_1$ and $\Pr[F \text{ occurs in } P2] \leq \gamma_2$.

Then, we have $\Pr[F \text{ occurs in } P3] \leq \gamma_1 + \gamma_2$.

Proof. We start with a technical claim:

Proposition 5. *There exists exactly one i during any run of $P3$ such that $G(X_i) = 1$.*

Proof of Proposition 5. From case 1 of the lemma hypothesis, we know that during $P3$, G will occur at some point, as $P3$ samples the X_t exactly like $P1$ up to when G occurs.

Due to case 4, we know that when $G(X_{t^*})$ holds in either $P1$ or $P2$, the states X_{t^*} are identically distributed. As the sampling procedure for $P3$ uses $R2$ (like $P2$ does as well), and $R2$ only takes the previous state as input the 'second part' $(X_{t^*+1}, \dots, X_{\mathcal{T}})$ of a state sequence $(X_0, \dots, X_{t^*}, X_{t^*+1}, \dots, X_{\mathcal{T}})$ (where t^* is the first index where $G(X_{t^*}) = 1$) will be identically distributed to $(X'_{t^*+1}, \dots, X'_{\mathcal{T}})$ where $(X'_0, \dots X'_{t^*}, X'_{t^*+1}, \dots, X_{\mathcal{T}})$ is a state sequence generated by $P2$ with $G(X_{t^*})$.

Therefore, $G(X_t) \neq 1$ for all $t > t^*$ due to case 2. □

Let t^* be the value of t when G occurs for the first time (in any of the three algorithms). We can split up the probability for F as follows:

$$\begin{aligned} \Pr[F \text{ occurs in } P3] &= \Pr[F \text{ occurs in } P3 \text{ up to } t^*] + \Pr[F \text{ occurs in } P3 \text{ after } t^*] \\ &\leq \Pr_{\substack{X_0 \leftarrow I1 \\ \forall t=1,\dots\mathcal{T}\colon X_t \leftarrow R1(X_{t-1})}}[F] + \Pr[F \text{ occurs in } P3 \text{ after } t^*] \end{aligned}$$

where by 'F occurs in $P3$ up to t^*', we denote that F occurs before X_{t^*} is sampled or during the sampling process of X_{t^*} and by 'F occurs in $P3$ after t^*' we denote that F occurs during the sampling processes of $X_{t^*+1}, \ldots, X_{\mathcal{T}}$. We now want to bound $\Pr[F \text{ occurs in } P3 \text{ after } t^*]$.

We will write $\Pr[E \text{ occurs in } Pi]$ for $i = 1, 2, 3$ to denote

$$\Pr\left[E \,\middle|\, \begin{array}{c} X_0 \leftarrow I1 \\ \forall t \in [\mathcal{T}] \colon X_t \leftarrow R1(X_{t-1}) \end{array}\right],$$

$$\Pr\left[E \,\middle|\, \begin{array}{c} X_0 \leftarrow I2 \\ \forall t \in [\mathcal{T}] \colon X_t \leftarrow R2(X_{t-1}) \end{array}\right],$$

and

$$\Pr\left[E \,\middle|\, \begin{array}{c} X_0 \leftarrow I1 \\ \forall t = 1, \ldots t^* \colon X_t \leftarrow R1(X_{t-1}) \\ \forall t = t^* + 1, \ldots \mathcal{T} \colon X_t = R2(t, X_{t-1}) \end{array}\right],$$

respectively, i.e., the probability of E happening when the sampling of the states X_i is done according to the processes $P1, P2$, or $P3$, respectively.

$$\begin{aligned}
& \Pr[F \text{ occurs after } t^* \text{ in } P3] \\
\overset{\text{Proposition 5}}{=} & \sum_{t=0}^{\mathcal{T}} \Pr[F \text{ occurs after } t^* \text{ in } P3 \wedge t^* = t \text{ in } P3] \\
= & \sum_{t=0}^{\mathcal{T}} \Pr[F \text{ occurs after } t^* \text{ in } P3 \mid t^* = t \text{ in } P3] \cdot \Pr[t^* = t \text{ in } P3] \\
= & \sum_{t=0}^{\mathcal{T}} \Pr[F \text{ occurs after } t^* \text{ in } P3 \mid t^* = t \text{ in } P3] \cdot \Pr[t^* = t \text{ in } P1] \\
\overset{\text{case 3}}{=} & \sum_{t=0}^{\mathcal{T}} \Pr[F \text{ occurs after } t^* \text{ in } P3 \mid t^* = t \text{ in } P3] \cdot \Pr[t^* = t \text{ in } P2] \\
\overset{\text{case 4}}{=} & \sum_{t=0}^{\mathcal{T}} \Pr[F \text{ occurs after } t^* \text{ in } P2 \mid t^* = t \text{ in } P2] \cdot \Pr[t^* = t \text{ in } P2] \\
= & \sum_{t=0}^{\mathcal{T}} \Pr[F \text{ occurs after } t^* \text{ in } P2 \wedge t^* = t \text{ in } P2] \\
\leq & \Pr[F \text{ occurs in } P2] \leq \gamma_2
\end{aligned}$$

Altogether, this yields

$$\Pr[F \text{ occurs in } P3] \;\leq\; \gamma_1 + \gamma_2.$$

□

3.3.3 Proving Security from IND-CPA

We now define the LKH security experiment $\mathrm{Exp}^{\mathsf{LKH}}_{\mathsf{SKE},\mathsf{A},d}(\lambda)$. This game models the security of LKH as an MKD protocol.

Definition 3.3.10 (LKH security experiment)**.** *Let* SKE $= (\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$ *be a secret-key encryption scheme and* $d = d(\lambda)$ *some depth.*

Setup. *The* LKH *experiment* $\mathrm{Exp}^{\mathsf{LKH}}_{\mathsf{SKE},\mathsf{A},d}(\lambda)$ *initially samples uniformly at random a challenge bit* $b_{\mathsf{lkh}} \leftarrow \{0,1\}$ *and keys* $k_x \leftarrow \{0,1\}^\lambda$ *for each* $x \in \{0,1\}^{\leq d}$*. It then computes ciphertexts* $c_{x\|b}$ *for all* $x \in \{0,1\}^{<d}$ *and* $b \in \{0,1\}$*, which encrypt key* k_x *under key* $k_{x\|b}$*. The adversary* A *receives the ciphertexts* $c_{x\|b}$*.*

Online Phase. *The adversary can make the following queries.*

Corruption Queries A *can adaptively corrupt "leaf" keys* k_x *(for* $x \in \{0,1\}^d$*). This exposes* k_x *to* A*, and results in a refresh of not only* k_x*, but also all* $k_{x'}$ *for proper prefixes* x' *of* x*. Furthermore, fresh encryptions of those* $k_{x'}$ *under keys* $k_{x'\|0}$ *and* $k_{x'\|1}$ *are generated and exposed to* A*.*

Challenge. *At any point,* A *may stop the game by asking to be challenged to distinguish the then-current key* k_ε *(case* $b_{\mathsf{lkh}} = 0$*) from a random key (case* $b_{\mathsf{lkh}} = 1$*).*

Output Determination. *descriptionThe output of the experiment is 1 if* A *correctly guesses the bit* b_{lkh}*, and 0 otherwise.*

We define the advantage of A *in this game as*

$$\mathrm{Adv}^{\mathsf{LKH}}_{\mathsf{SKE},\mathsf{A},d}(\lambda) := \Pr[\mathrm{Exp}^{\mathsf{LKH}}_{\mathsf{SKE},\mathsf{A},d}(\lambda)(\lambda) = 1] - 1/2.$$

Asymptotically, we say that LKH *is secure (with* SKE*) if for every polynomial-time* A *and every constant* $c \in \mathbb{N}$*, the advantage* $\mathrm{Adv}^{\mathsf{LKH}}_{\mathsf{SKE},\mathsf{A},c\cdot\log(\lambda)}(\lambda)$ *is negligible.*[2]

Our strategy. We will prove that A has a negligible advantage to win the game $\mathrm{Exp}^{\mathsf{LKH}}_{\mathsf{SKE},\mathsf{A},d}(\lambda)$ through a large hybrid argument. We will start with the game above and apply a number of successive changes until finally A's view is independent of the real final key k_ε. Similar to Section 3.2, we make a liberal use of rewindings, thus, it will be helpful to formalize A's view:

Definition 3.3.11 (Adversarial view)**.** *In a run of the* LKH *experiment* $\mathrm{Exp}^{\mathsf{LKH}}_{\mathsf{SKE},\mathsf{A},d}(\lambda)$ *from Definition 3.3.10, we define* A*'s view* view$_\mathsf{A}$ *in this run as a sequence* $(\mathsf{ev}_1, \dots, \mathsf{ev}_\mathcal{T})$ *of events, where each* ev_i *can be one of the following:*

Query. *One of* A*'s queries (without reply), either of the form* $(\mathsf{corrupt}, x)$*, or* challenge*.*

New key. *Every time a new key* k_x *is defined, right before that, a corresponding* (newkey, x) *event is appended to* view*. Concretely,* view *starts with* (newkey, x) *events for* $x \in \{0,1\}^{\leq d}$*. Furthermore, a query* $(\mathsf{corrupt}, x)$ *automatically causes also entries* (newkey, x') *for a prefix* x' *of* x *to be appended immediately after that* corrupt *entry. Entries* (newkey, x) *defined at the same time are ordered in* view *with keys further from the root (i.e., with longer* x*) first.*

Ciphertext. *An event* $(\mathsf{ctxt}, x\|b, c_{x\|b})$ *for a ciphertext* $c_{x\|b} = \mathbf{Enc}(k_{x\|b}, k_x)$*, either as part of* A*'s initial input, or as a side effect of a corruption query.* ctxt *entries defined at the same time are ordered with ciphertexts furthest from the root (i.e., with longer* x*) first, and lexicographically (according to* $x\|b$*) for* x *of the same length.*

[2]Like previous works, we focus on a logarithmic depth and thus to polynomially many users.

Corrupted key. *A key* (key, x, k_x) *as a result of a corruption query. We assume that this key appears before the corresponding new key events and the ciphertexts that are sent to* A *in the same reply.*

Challenge key. *The result of the final challenge query, in the form* $(\mathsf{challenge}, k)$ *(i.e., depending on* b_{B} *with either* k *being a key* k_ε *or a random value).*

Decision bit. *The final output bit* b_{A} *of* A*, in the form* $(\mathsf{guess}, b_{\mathsf{A}})$*. This event is the last in* view*, and we may write* $\mathrm{out}_{\mathsf{A}}(\mathsf{view})$ *for the output bit* b_{A}*.*

Remark 3.3.12. *Note that the ordering of events above implies for an event* $(\mathsf{corrupt}, x)$ *that it is followed by an event* (key, x, k_x)*, then a sequence of events* (newkey, x') *for all prefixes* x' *of* x*, in decreasing length, and then a sequence* $(\mathsf{ctxt}, x' \| b, c_{x'\|b})$ *for all strict prefixes* x' *of* x *and bits* $b \in \{0,1\}$*, again ordered by decreasing length, and siblings ordered alphabetically. For example, for depth* $d = 3$*, a* $(\mathsf{corrupt}, 010)$ *event causes the following sequence of events:* $(\mathsf{key}, 010, k_{010})$, $(\mathsf{newkey}, 010)$, $(\mathsf{newkey}, 01)$, $(\mathsf{newkey}, 0)$, $(\mathsf{newkey}, \varepsilon)$, $(\mathsf{ctxt}, 010, c_{010})$, $(\mathsf{ctxt}, 011, c_{011})$, $(\mathsf{ctxt}, 00, c_{00})$, $(\mathsf{ctxt}, 01, c_{01})$, $(\mathsf{ctxt}, 0, c_0)$, *and* $(\mathsf{ctxt}, 1, c_1)$.

We are now ready to formulate and prove our main result:

Theorem 3.3.13. *Let* $\mathsf{SKE} = (\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$ *be an SKE scheme. Then*

- *for every* LKH *adversary* A *that runs in time* t_{A} *and places at most* $\mathcal{Q}_{\mathsf{corrupt}}$ *corruption queries,*
- *for every* LKH *depth* d *and every* $\gamma \in (0,1]$*,*

there is an IND-CPA *adversary* B *that makes at most* $\mathcal{Q}_{\mathsf{LoR}} \le 2 \cdot \ln(2\mathcal{T}/\gamma) \cdot \mathcal{T}^4 \cdot (d+1)/\gamma$ *LoR queries,* $\mathcal{Q}_{\mathsf{NU}} \le 2 \cdot \ln(2\mathcal{T}/\gamma) \cdot \mathcal{T}^3 \cdot (d+1)/\gamma$ *new user queries, and runs in time* t_{B} *and for which*

$$\mathrm{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{SKE,B}}(\lambda) \ge \frac{1}{2} \cdot \frac{1}{T_d} \cdot \mathrm{Adv}^{\mathsf{LKH}}_{\mathsf{SKE,A},d}(\lambda) - \frac{\gamma}{2}. \tag{3.12}$$

where

$$t_{\mathsf{B}} \lessapprox 2 \cdot \ln(2\mathcal{T}/\gamma) \cdot \mathcal{T}^4 \cdot (d+1)/\gamma \cdot t_{\mathsf{A}}. \tag{3.13}$$

where $\mathcal{T} \le 2^{d+2} + (3d+1) \cdot \mathcal{Q}_{\mathsf{corrupt}}$*.*

Again, before proceeding to a proof, we remark that Theorem 3.3.13 implies asymptotic security when setting γ suitably:

Corollary 3.3.14 (SKE IND-CPA $\Rightarrow$ LKH secure)**.** *If* SKE *is* IND-CPA *secure (as in Definition 2.8.2), then* LKH *is secure with* SKE *(in the sense of Definition 3.3.10).*

Proof of Corollary 3.3.14. Fix a depth $d = c \cdot \log(\lambda)$ (for some constant $c \in \mathbb{N}$), and assume for contradiction that there is a polynomial-time adversary A against the security of LKH with non-negligible advantage $\varepsilon_{\mathsf{A}} := \mathrm{Adv}^{\mathsf{LKH}}_{\mathsf{SKE,A},d}(\lambda)$. Since ε_{A} is non-negligible, there exists a polynomial p such that for infinitely many values of λ, we have $\varepsilon_{\mathsf{A}} \ge 1/p(\lambda)$.

Now set $\gamma = 1/(2T_d p(\lambda))$ (for the value T_d from Definition 3.3.1, which is polynomially bounded by our choice of d and by Lemma 3.3.2), and invoke Theorem 3.3.13. We obtain an IND-CPA adversary B with (by Eq. (3.13)) polynomial runtime and non-negligible advantage

$$\mathrm{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{SKE,B}}(\lambda) \overset{Eq.\ (3.12)}{\ge} \frac{1}{2T_d} \cdot \varepsilon_{\mathsf{A}} - \frac{\gamma}{2} \overset{(*)}{\ge} \frac{1}{2T_d} \cdot \frac{1}{p(\lambda)} - \frac{\gamma}{2} = \frac{1}{4T_d p(\lambda)},$$

where $(*)$ holds (only) for infinitely many λ. $\square$

Algorithm 18: Game i, with the bulk of the work outsourced into the sampling from $\mathcal{D}^{\mathsf{view}}_{i,b_{\mathsf{lkh}}}$.

```
1 $b_{\mathsf{lkh}} \leftarrow \{0,1\}$
2 view $\leftarrow \mathcal{D}^{\mathsf{view}}_{i,b_{\mathsf{lkh}}}$                // view has the format from Definition 3.3.11
3 return $[b_{\mathsf{lkh}} = \mathrm{out}_{\mathsf{A}}(\mathsf{view})]$          // returns 1 iff $b_{\mathsf{lkh}} = \mathrm{out}_{\mathsf{A}}(\mathsf{view})$
```

Proof Overview.

Fix SKE, A, and d. In the following, we will consider a number of hybrid games, with Game lkh being the original LKH experiment. Denoting with out_i the output of Game i, we trivially get

$$\Pr[\mathrm{out}_{\mathsf{lkh}} = 1] \; = \; \mathrm{Adv}^{\mathsf{LKH}}_{\mathsf{SKE},\mathsf{A},d}(\lambda) + 1/2. \tag{3.14}$$

To move on, we will formulate Game i (for $0 \leq i \leq T_d$) in a (for us) convenient way, see Algorithm 18. This formulation outsources the bulk of the game into the sampling of A's view view from a suitable distribution $\mathcal{D}^{\mathsf{view}}_{i,b_{\mathsf{lkh}}}$. In our upcoming refinements, we will only change $\mathcal{D}^{\mathsf{view}}_{i,b_{\mathsf{lkh}}}$ and investigate the effects on out_i.

The Distributions $\mathcal{D}^{\mathsf{view}}_{i,b_{\mathsf{lkh}}}$. To define the distribution $\mathcal{D}^{\mathsf{view}}_{i,b_{\mathsf{lkh}}}$ for Game i, we use the following notation:

- $\mathcal{D}^{\mathsf{view}}_{\mathsf{lkh},b_{\mathsf{lkh}}}$ is the distribution of A-views (as in Definition 3.3.11) that is induced by running the LKH experiment with challenge bit b_{lkh} (that decides whether A is challenged with k_ε or a random key).
- $\mathsf{edges}_{d,i}$ is the edge index set from Definition 3.3.3 that arises out of pebbling a depth-d binary tree.
- len(view) is the length of a given A-view view (measured in events).
- pfx_t(view) outputs the prefix of view up to (and including) the t-th event (see Section 2.1).
- $\mathsf{maxcor}_{i,t}$(view) on input view $= (\mathsf{ev}_1, \ldots, \mathsf{ev}_{\mathcal{T}})$ outputs

$$\max \left(\left\{ \, |x| \;\middle|\; \mathsf{ev}_{t'} = (\mathsf{ctxt}, x, c_x) \text{ for some } t' > t \text{ and } x \leq_{\mathsf{pfx}} \mathsf{leaf}_{d,i} \, \right\} \cup \{0\} \right).$$

- $\mathsf{lastkey}_i$(view) on input view $= (\mathsf{ev}_1, \ldots, \mathsf{ev}_{\mathcal{T}})$ outputs

$$\max \{ \, t' \mid \mathsf{ev}_{t'} = (\mathsf{newkey}, x_i^*) \},$$

where $x_i^* := \mathsf{edges}_{d,i-1} \Delta \mathsf{edges}_{d,i}$ for $i \geq 1$ and $x_0^* := \mathsf{edges}_{d,0} \Delta \mathsf{edges}_{d,1}$.
- $B \in \mathbb{N}$ is a bound on the number of repetitions of Lines 6 to 14 for each t. In case of B unsuccessful repetitions for one t, the whole algorithm outputs $\perp$. We will fix a suitable value for B later.

Below we will prove some useful properties of the functions maxcor and lastkey, and are now ready to define the distributions $\mathcal{D}^{\mathsf{view}}_{i,b_{\mathsf{lkh}}}$; see Algorithm 19. Our distribution $\mathcal{D}^{\mathsf{view}}_{i,b_{\mathsf{lkh}}}$ is defined like $\mathcal{D}^{\mathsf{view}}_{\mathsf{lkh},b_{\mathsf{lkh}}}$ (i.e., like an LKH run with A), but uses rewinding and resampling (as defined in Section 3.2) at every step. Additionally, we replace certain ciphertexts c_x as indicated by $x \in \mathsf{edges}_{d,i}$ during the rewindings. Concretely, fix an i and consider Algorithm 19, which programmatically describes sampling $\mathsf{view}_{\mathcal{T}}$ according to $\mathcal{D}^{\mathsf{view}}_{i,b_{\mathsf{lkh}}}$.

Algorithm 19: Sampler for $\mathcal{D}^{\mathsf{view}}_{i\,.1,b_{\mathsf{lkh}}}$

```
Input: $i \in \{0,\dots,T_d\}, b_{\mathsf{lkh}} \in \{0,1\}, B \in \mathbb{N}$   // len, $\mathsf{pfx}_t$, $\mathsf{maxcor}_{i,t}$, $\mathsf{lastkey}_i$, $B$ described in
       proof
1  $\mathsf{view}_0 \leftarrow \mathcal{D}^{\mathsf{view}}_{\mathsf{lkh},b_{\mathsf{lkh}}}$
2  $\mathcal{T} := \mathsf{len}(\mathsf{view}_{\mathsf{lkh}})$                         // Length of $\mathsf{view}_{\mathsf{lkh}}$ (in entries)
3  for $t := 1$ to $\mathcal{T}$ do
4      Write $\mathsf{view}_{t-1} = (\mathsf{ev}_{t-1,1},\dots,\mathsf{ev}_{t-1,\mathcal{T}})$
5      repeat                          // Output $\perp$ if $B$ repetitions fail for this $t$
6          Rewind adversary to point $t$      // Checks if $\mathsf{ev}_{t,t}$ defines a ciphertext to be
            pebbled
7          if $\mathsf{ev}_{t-1,t} = (\mathsf{ctxt}, x, c_x)$ with $x \in \mathsf{edges}_{d,i}$ and $\mathsf{maxcor}_{i\,+1,t+1}(\mathsf{view}_{t-1}) < |x|$ then
8              Sample fresh $c_x^{\perp} \leftarrow \mathbf{Enc}(k_x,\perp)$              // Fresh dummy ciphertext
9              Set $\mathsf{ev}_{t,t} := (\mathsf{ctxt}, x, c_x^{\perp})$ in $\mathsf{view}_t$        // Replace $c_x$ with $c_x^{\perp}$ in $\mathsf{view}_{t-1}$
10             Resample from point $t+1$ to obtain
                $\mathsf{view}_t = (\mathsf{ev}_{t-1,0},\dots,\mathsf{ev}_{t-1,t-1},\mathsf{ev}_{t,t},\mathsf{ev}_{t,t+1},\dots,\mathsf{ev}_{t,\tau})$
11         else
12             Resample from point $t$ to obtain $\mathsf{view}_t = (\mathsf{ev}_{t-1,0},\dots,\mathsf{ev}_{t-1,t-1},\mathsf{ev}_{t,t},\dots,\mathsf{ev}_{t,\tau})$
13         end
14     until $\mathsf{maxcor}_{i\,+1,t+1}(\mathsf{view}_t) = \mathsf{maxcor}_{i\,+1,t+1}(\mathsf{view}_{t-1})$ and $\mathsf{len}(\mathsf{view}_t) = \mathsf{len}(\mathsf{view}_{t-1})$ and
        $\mathsf{lastkey}_{i\,+1}(\mathsf{view}_t) = \mathsf{lastkey}_{i\,+1}(\mathsf{view}_{t-1})$
15 end
16 return $\mathsf{view}_{\mathcal{T}}$
```

Having defined our hybrid distributions $\mathcal{D}^{\mathsf{view}}_{i,b_{\mathsf{lkh}}}$, we will additionally consider potentially inefficient procedures $\widetilde{\mathcal{D}}^{\mathsf{view}}_{i,b_{\mathsf{lkh}}}$ which are defined similar to Algorithm 19 but without a bound B on the runtime. We will first show that the distribution $\widetilde{\mathcal{D}}^{\mathsf{view}}_{0,b_{\mathsf{lkh}}}$ coincides with the distribution of views in Game lkh. Next, we will consider the intermediate distributions $\mathcal{D}^{\mathsf{view}}_{i.1,b_{\mathsf{lkh}}}$ (the difference to $\mathcal{D}^{\mathsf{view}}_{i,b_{\mathsf{lkh}}}$ being marked gray in Algorithm 19) and show that for all $i \in [T_d - 1]_0$ it holds that $\widetilde{\mathcal{D}}^{\mathsf{view}}_{i,b_{\mathsf{lkh}}}$ and $\widetilde{\mathcal{D}}^{\mathsf{view}}_{i.1,b_{\mathsf{lkh}}}$ have the same distribution.

We will then bound the difference in the probability of an abort in the games $\mathcal{D}^{\mathsf{view}}_{i,b_{\mathsf{lkh}}}$ and $\mathcal{D}^{\mathsf{view}}_{i.1,b_{\mathsf{lkh}}}$. To prove this we will need an additional technical lemma about the abort probability in such mixed sampling procedures and this is the main difference to Section 3.2 in which such a mixed resampling procedure does not occur. Setting the bound B appropriately we will be able to bound the probability of an abort in $\mathcal{D}^{\mathsf{view}}_{0,b_{\mathsf{lkh}}}$ by γ. In the subsequent claim we will show that A has no advantage in $\mathrm{Game}\ T_d$ since the view sampled according to $\mathcal{D}^{\mathsf{view}}_{T_d,b_{\mathsf{lkh}}}$ is independent of b_{lkh}. Finally, we will conclude the proof by arguing that for all $i \in [T_d - 1]_0$ the distributions $\mathcal{D}^{\mathsf{view}}_{i.1,b_{\mathsf{lkh}}}$ and $\mathcal{D}^{\mathsf{view}}_{i+1,b_{\mathsf{lkh}}}$ are computationally indistinguishable by IND-CPA security of the SKE scheme SKE. See Figure 3.4 for an overview of this sequence of arguments.

$$\begin{tikzcd}
\mathcal{D}^{\mathsf{view}}_{\mathsf{lkh},b_{\mathsf{lkh}}} \arrow[r, dash, "\equiv"] & \widetilde{\mathcal{D}^{\mathsf{view}}_{0,b_{\mathsf{lkh}}}} \arrow[r, dash, "\equiv"] \arrow[d, dash, "\overset{\mathrm{s}}{\approx}_{\gamma_0}"'] & \widetilde{\mathcal{D}^{\mathsf{view}}_{0.1,b_{\mathsf{lkh}}}} \arrow[d, dash, "\overset{\mathrm{s}}{\approx}_{\gamma_{0.1}}"'] & \widetilde{\mathcal{D}^{\mathsf{view}}_{1,b_{\mathsf{lkh}}}} \arrow[r, dash, "\equiv"] \arrow[d, dash, "\overset{\mathrm{s}}{\approx}_{\gamma_1}"'] & \widetilde{\mathcal{D}^{\mathsf{view}}_{1.1,b_{\mathsf{lkh}}}} \arrow[d, dash, dashed, "\cdots"] & \\
& \mathcal{D}^{\mathsf{view}}_{0,b_{\mathsf{lkh}}} & \mathcal{D}^{\mathsf{view}}_{0.1,b_{\mathsf{lkh}}} \arrow[r, dash, "\overset{\mathrm{c}}{\approx}"] & \mathcal{D}^{\mathsf{view}}_{1,b_{\mathsf{lkh}}} & \mathcal{D}^{\mathsf{view}}_{(T_d-1).1,b_{\mathsf{lkh}}} \arrow[r, dash, "\overset{\mathrm{c}}{\approx}"] & \mathcal{D}^{\mathsf{view}}_{T_d,b_{\mathsf{lkh}}}
\end{tikzcd}$$

Figure 3.4: Sequence of hybrids from the proof of Theorem 3.3.13. Perfect indistinguishabilities ("$\equiv$") are shown in Proposition 6 and Proposition 7, statistical indistinguishabilities ("$\overset{\mathrm{s}}{\approx}_{\gamma_i}$" and "$\overset{\mathrm{s}}{\approx}_{\gamma_{i.1}}$") with bounds γ_i and $\gamma_{i.1}$ follow from our IND-CPA reduction of Theorem 3.3.13 and Proposition 8, and computational indistinguishabilities ("$\overset{\mathrm{c}}{\approx}$") follow from the IND-CPA reduction. Note that the statement $\widetilde{\mathcal{D}^{\mathsf{view}}_{0,b_{\mathsf{lkh}}}} \overset{\mathrm{s}}{\approx}_{\gamma_0} \mathcal{D}^{\mathsf{view}}_{0,b_{\mathsf{lkh}}}$ is needed to bound $\gamma_{0.1}$.

Some Useful Properties of maxcor **and** lastkey**.**

We prove some useful properties of maxcor and lastkey.

Lemma 3.3.15. *For any $i \in \{0, \dots, T_d\}$, any* view *as described above, and any $t \in \{1, \dots, \mathsf{len}(\mathsf{view})-1\}$, it holds that* $\mathsf{maxcor}_{i,t}(\mathsf{view}) \geq \mathsf{maxcor}_{i,t+1}(\mathsf{view})$.

Proof. This follows from the fact that maxcor considers the suffix of view and so increasing from t to $t+1$ only removes events that are considered for maxcor. □

We show that for edges that we want to put pebbles on, it does not matter whether we look at the maxcor value for i or for $i+1$ to determine when is a good time to put a pebble. This will be useful when game hopping later in the proof.

Lemma 3.3.16. *For any $i \in \{0, \dots, T_d - 1\}$, any* view *as described above, any $t \in \{1, \dots, \mathsf{len}(\mathsf{view})\}$, any $x \in \mathsf{edges}_{d,i}$, it holds*

$$(\mathsf{maxcor}_{i,t}(\mathsf{view}) < |x|) \Leftrightarrow (\mathsf{maxcor}_{i+1,t}(\mathsf{view}) < |x|).$$

Proof. Let x'_{i+1} be the longest common prefix of $\mathsf{leaf}_{d,i}$ and $\mathsf{leaf}_{d,i+1}$.

We note that by Lemma 3.3.4, as $x \in \mathsf{edges}_{d,i}$, x lies on the path or the co-path of $x'_{i+1}\|0$, so in particular $|x| \leq |x'_{i+1}\|0|$. This holds because $\mathsf{edges}_{d,i}$ is a subset of both sets of edges incident on the path to $\mathsf{leaf}_{d,i}$ and $\mathsf{leaf}_{d,i+1}$.

If $\mathsf{maxcor}_{i,t+1}(\mathsf{view}_{t-1}) < |x|$, this in particular means that the length-maximal x' fulfilling the criterion from the definition of $\mathsf{maxcor}_{i,t+1}(\mathsf{view}_{t-1})$ must have $|x'| < |x|$. Thus, it must be a prefix of x'_{i+1}. Thus, it is also a prefix of $\mathsf{leaf}_{d,i+1}$. As for each (ctxt, x, c_x) event, the sibling event $(\mathsf{ctxt}, \mathsf{pfx}_{|x|-1}(x)\|(1-x_{|x|}), c_{\mathsf{pfx}_{|x|-1}(x)\|(1-x_{|x|})})$ must also occur right before or right after, this yields that if x' is the longest prefix of $\mathsf{leaf}_{d,i}$ after $t+1$, it must also be the longest prefix of $\mathsf{leaf}_{d,i+1}$ after $t+1$. This yields one implication of the equivalence. The reverse direction follows by a symmetrical argument. □

The following lemma states that the last corruption of the relevant key will already have happened before a pebble is embedded (i.e. before a ciphertext is replaced by an encryption of $\perp$). This will be useful to see that the end conditions of the **repeat** loops in different versions of Algorithm 19 are equivalent.

Lemma 3.3.17. *Let $i \in [T_d]$ and $x \in \mathsf{edges}_{d,i}$. Then,* $\mathsf{maxcor}_{i,t+1}(\mathsf{view}) < |x|$ *implies that* $\mathsf{lastkey}_i(\mathsf{view}) < t+1$ *and* $\mathsf{lastkey}_{i+1}(\mathsf{view}) < t+1$.

Proof. Let x_i^* be as in the definition of $\mathsf{lastkey}_i$. By Corollary 3.3.8, any $x \in \mathsf{edges}_{d,i}$ is incident on the path from x_i^* to the root. Thus $|x| \leq |x_i^*| + 1$. It follows that if $\mathsf{maxcor}_{i,t+1}(\mathsf{view}) < |x|$, then also $\mathsf{maxcor}_{i,t+1}(\mathsf{view}) < |x_i^*| + 1$. As any event of the form (newkey, x^*) triggers events of the form $(\mathsf{ctxt}, x^*\|b, c_{x^*\|b})$ for $b \in \{0,1\}$, any (newkey, x_i^*) event at or after $t+1$ implies that $\mathsf{maxcor}_{i,t+1}(\mathsf{view}) \geq |x_i^*| + 1$. The case for $\mathsf{lastkey}_{i+1}$ follows from Lemma 3.3.16. □

We apply the above lemmas to views that share prefixes to find that identical maxcor values imply identical lastkey values as soon as a pebble is embedded. This will again be useful in a game hop.

Corollary 3.3.18. *Let t, i be arbitrary, and let* $\mathsf{view} = (\mathsf{ev}_1, \ldots, \mathsf{ev}_{\mathcal{T}})$ *be such that* $\mathsf{ev}_t = (\mathsf{ctxt}, x, c_x)$ *with* $x \in \mathsf{edges}_{d,i}$ *and* $\mathsf{maxcor}_{i,t+1}(\mathsf{view}) < |x|$. *Then, for any* view' *with* $\mathsf{pfx}_t(\mathsf{view}') = \mathsf{pfx}_t(\mathsf{view})$, *we have*

$$\mathsf{maxcor}_{i,t+1}(\mathsf{view}') = \mathsf{maxcor}_{i,t+1}(\mathsf{view}) \;\Rightarrow\; \mathsf{lastkey}_i(\mathsf{view}') = \mathsf{lastkey}_i(\mathsf{view})$$

and

$$\mathsf{maxcor}_{i+1,t+1}(\mathsf{view}') = \mathsf{maxcor}_{i+1,t+1}(\mathsf{view}) \;\Rightarrow\; \mathsf{lastkey}_{i+1}(\mathsf{view}') = \mathsf{lastkey}_{i+1}(\mathsf{view}).$$

Proof. As $\mathsf{maxcor}_{i,t+1}(\mathsf{view}) < |x|$, by Lemma 3.3.16 $\mathsf{maxcor}_{i+1,t+1}(\mathsf{view}) < |x|$. Furthermore, if $\mathsf{maxcor}_{i,t+1}(\mathsf{view}) = \mathsf{maxcor}_{i,t+1}(\mathsf{view}')$, then also $\mathsf{maxcor}_{i,t+1}(\mathsf{view}') < |x|$ and the same implication as above holds for view'. From Lemma 3.3.17, it follows that $\mathsf{lastkey}_i(\mathsf{view}) < t+1$ and $\mathsf{lastkey}_i(\mathsf{view}') < t+1$. Therefore, as the prefixes up to t of view and view' are the same, in fact $\mathsf{lastkey}_i(\mathsf{view}') = \mathsf{lastkey}_i(\mathsf{view})$. Using a similar argument, the implication for $i+1$ also follows. □

Proof of Theorem 3.3.13.

We are now ready to prove Theorem 3.3.13.

Proof. We start with a few helper propositions to structure our proof.

Proposition 6. $\widetilde{\mathcal{D}^{\mathsf{view}}_{0,b_{\mathsf{lkh}}}} \equiv \mathcal{D}^{\mathsf{view}}_{\mathsf{lkh},b_{\mathsf{lkh}}}$.

Proof of Proposition 6. Since $\mathsf{edges}_{d,0} = \emptyset$, the **if** clause can never return true and the algorithm always enters the **else** clause. The statement therefore follows from Lemma 3.1.2, where $\mathcal{D} = \mathcal{D}^{\mathsf{view}}_{\mathsf{lkh},b_{\mathsf{lkh}}}$, $h_t(\mathsf{view}_s) = (\mathsf{ev}_{s,1}, \ldots, \mathsf{ev}_{s,t-1})$, and $g_t(\mathsf{view}_s) = (\mathsf{maxcor}_{0,t}(\mathsf{view}_s), \mathsf{len}(\mathsf{view}_s), \mathsf{lastkey}_0(\mathsf{view}_s))$. □

Proposition 7 ($\widetilde{\mathcal{D}^{\mathsf{view}}_{i,b_{\mathsf{lkh}}}} \equiv \widetilde{\mathcal{D}^{\mathsf{view}}_{i.1,b_{\mathsf{lkh}}}}$)**.** *For all $i \in [T_d - 1]_0$, we have* $\widetilde{\mathcal{D}^{\mathsf{view}}_{i,b_{\mathsf{lkh}}}} \equiv \widetilde{\mathcal{D}^{\mathsf{view}}_{i.1,b_{\mathsf{lkh}}}}$.

Proof of Proposition 7. By Lemma 3.1.2, when instantiated once with

$$h_t(\mathsf{view}) = \mathsf{pfx}_{t-1}(\mathsf{view}),$$
$$g_t(\mathsf{view}) = (\mathsf{maxcor}_{i,t+1}(\mathsf{view}), \mathsf{len}(\mathsf{view}), \mathsf{lastkey}_i(\mathsf{view})),$$

and once with h_t as above and

$$g_t(\mathsf{view}) = (\mathsf{maxcor}_{i+1,t+1}(\mathsf{view}), \mathsf{len}(\mathsf{view}), \mathsf{lastkey}_{i+1}(\mathsf{view})),$$

the distributions of the view_t up until the **if** in Line 7 of Algorithm 19 returns true for the first time are identical. Further, by Lemma 3.3.16, the conditions for the **if** are equivalent, i.e. whenever the **if** condition would return true for $i+1$ it would also return true for i and vice versa. It therefore remains to show that the end conditions of the **repeat** loop are also equivalent after the first time the **if** returned true.

To see this, note that after the first time the **if** returned true, $\mathsf{maxcor}_{i,t+1}$ as well as $\mathsf{maxcor}_{i+1,t+1}$ are upper bounded by the length of any of the edge labels in $\mathsf{edges}_{d,i}$ – this follows from Lemma 3.3.16 and Lemma 3.3.15.

As by Lemma 3.3.7, all of these edges are incident on the path from the longest common prefix of $\mathsf{leaf}_{d,i}$ and $\mathsf{leaf}_{d,i+1}$ to the root or its co-path.

Thus, if $\mathsf{maxcor}_{i,t+1}$ and $\mathsf{maxcor}_{i+1,t+1}$ are bounded by the length of such an edge label, it follows in fact that $\mathsf{maxcor}_{i,t+1}(\mathsf{view}) = \mathsf{maxcor}_{i+1,t+1}(\mathsf{view})$.

By a similar argument, and by Corollary 3.3.18, it follows that the part of the stopping condition that concerns lastkey is equivalent.

Lastly, we see that $\mathsf{len}(\mathsf{view}_t)$ does not depend on i and thus the stopping conditions of the **repeat** loops in the two algorithms are equivalent.

The statement follows. □

Proposition 8. *For all $i \in [T_d - 1]_0$, we have*

$$\gamma_{i.1} = \Pr[\bot \leftarrow \mathcal{D}^{\mathsf{view}}_{i.1,b_{\mathsf{lkh}}}] \;\leq\; \Pr[\bot \leftarrow \mathcal{D}^{\mathsf{view}}_{i,b_{\mathsf{lkh}}}] + \gamma.$$

Proof of Proposition 8. We use Line 10, where event F will be exceeding the bound B on the iterations of the **repeat** loop. We further define the procedures $I1, I2, R1_i, R2_i$ as listed in Algorithms 20 to 22.

In the following, we will want to map the first placement of a pebble in Algorithm 19 to the event G that triggers switching the algorithms in Line 10, i.e. if $G(t, \mathsf{view}_t)$ is true, the next iteration of the **for** loop will replace a ciphertext. We define the event $G_1(t, \mathsf{view}_t)$ in Algorithm 21 as 't is the smallest value among all t' that satisfy $\mathsf{ev}_{t,t'+1} = (\mathsf{ctxt}, x, c_x)$ with $x \in \mathsf{edges}_{d,i}$ and $\mathsf{maxcor}_{i,t'+2}(\mathsf{view}_t) < |x|$', and the event $G_2(t, \mathsf{view}_t)$ in Algorithm 22 as 't is the smallest value among all t' that satisfy $\mathsf{ev}_{t,t'+1} = (\mathsf{ctxt}, x, c_x)$ with $x \in \mathsf{edges}_{d,i}$ and $\mathsf{maxcor}_{i+1,t'+2}(\mathsf{view}_t) < |x|$'.

Note that by Lemma 3.3.16, these definitions are in fact equivalent, i.e., $G_1(t, \mathsf{view}_t) = 1 \Leftrightarrow G_2(t, \mathsf{view}_t) = 1$. We will therefore in the following only speak of $G = G_1 = G_2$.

As mentioned above, we define F to be the event that the **repeat** loop in $R1_i$ or $R2_i$, respectively, is repeated more than B times, where B is defined in Claim 3.3.19.

As both definitions of G refer to the smallest t, it is obvious that the event G can occur at most once in either a run initiated using $I1$ and subsequent calls to $R1_i$, or a run initiated using $I2$ with subsequent calls to $R2_i$. Thus, G fulfills Cases 1 and 2 from Line 10.

Let $P1_i$ be defined through $I1$ and $R1_i$ as in Line 10. Similarly, let $P2_i$ be defined through $I2$ and $R2_i$ as in Line 10. Let $P3_i$ be defined through $I1$, then sampling using $R1_i$ until the first occurrence of G, and then sampling using $R2_i$.

We note that $P1_i$ corresponds to sampling from $\widetilde{\mathcal{D}^{\mathsf{view}}_{0,b_{\mathsf{lkh}}}}$, $P2_i$ corresponds to sampling from $\widetilde{\mathcal{D}^{\mathsf{view}}_{i.1,b_{\mathsf{lkh}}}}$, and $P3_i$ corresponds to sampling from $\widetilde{\mathcal{D}^{\mathsf{view}}_{i,b_{\mathsf{lkh}}}}$.

```
Algorithm 20: $I1$ and $I2$
1 $\mathsf{view}_0 \leftarrow \mathcal{D}^{\mathsf{view}}_{\mathsf{lkh},b_{\mathsf{lkh}}}$
2 return $X_0 = (0, \mathsf{view}_0)$
```

```
Algorithm 21: $R1_i$
  Input: $(t-1, \mathsf{view}_{t-1})$
1 repeat
2     Rewind adversary to point $t$
3     Resample from point $t$
4 until $\mathsf{maxcor}_{i,t+1}(\mathsf{view}_t) = \mathsf{maxcor}_{i,t+1}(\mathsf{view}_{t-1})$ and
      $\mathsf{len}(\mathsf{view}_t) = \mathsf{len}(\mathsf{view}_{t-1})$ and
      $\mathsf{lastkey}_i(\mathsf{view}_t) = \mathsf{lastkey}_i(\mathsf{view}_{t-1})$
5 return $X_t = (t, \mathsf{view}_t)$
```

```
Algorithm 22: $R2_i$
   Input: $(t-1, \mathsf{view}_{t-1})$
1  repeat
2      Rewind adversary to point $t$
3      if $\mathsf{ev}_{t,t} = (\mathsf{ctxt}, x, c_x)$ with $x \in \mathsf{edges}_{d,i}$ and
         $\mathsf{maxcor}_{i+1,t+1}(\mathsf{view}_{t-1}) < |x|$ then
4          Sample fresh $c_x^{\perp} \leftarrow \mathbf{Enc}(k_x, \perp)$
5          Set $\mathsf{ev}_{t,t} := (\mathsf{ctxt}, x, c_x^{\perp})$ in $\mathsf{view}_t$
6          Resample from point $t+1$
7      else
8          Resample from point $t$
9      end
10 until
     $\mathsf{maxcor}_{i+1,t+1}(\mathsf{view}_t) = \mathsf{maxcor}_{i+1,t+1}(\mathsf{view}_{t-1})$
     and $\mathsf{len}(\mathsf{view}_t) = \mathsf{len}(\mathsf{view}_{t-1})$ and
     $\mathsf{lastkey}_{i+1}(\mathsf{view}_t) = \mathsf{lastkey}_{i+1}(\mathsf{view}_{t-1})$
11 return $X_t = (t, \mathsf{view}_t)$
```

Figure 3.5: Algorithms for the proof of Proposition 8

Claim 3.3.19. *For $B := 2\mathcal{T} \cdot \ln(2\mathcal{T}/\gamma) \cdot \left(2^{d+2} + (3d+1) \cdot \mathcal{Q}_{\mathsf{corrupt}}\right)^2 \cdot (d+1)/\gamma$, the probability of F in $P1_i$, i.e. any run of the* **repeat** *loop in Algorithm 21 exceeding B, is at most γ.*

Proof. By Lemma 3.1.3, with $g_t(x) = (\mathsf{maxcor}_{i,t}(x), \mathsf{len}(x), \mathsf{lastkey}_i(\mathsf{view}))$ and $h_t(\mathsf{view}) = \mathsf{pfx}_{t-1}(\mathsf{view})$ it holds that for any $\gamma \in (0,1]$ (thus in particular the γ from the theorem statement)

$$\Pr[\,\forall t \in [\mathcal{T}] : T_t^{\mathsf{rep}} \leq 2\mathcal{T} \cdot \ln(2\mathcal{T}/\gamma) \cdot |\mathcal{Y}|/\gamma\,] \;\geq\; 1-\gamma, \tag{3.15}$$

where T_t^{rep} denotes the number of runs of the repeat loop in the t-th iteration of the for loop, and $\mathcal{Y}$ denotes a set large enough to accommodate the range of any g_t. We note

$$\mathsf{len}(\mathsf{view}) \leq \mathcal{T}_{\max} := \underbrace{2^{d+2}}_{\text{Initial Tree}} + \underbrace{\mathcal{Q}_{\mathsf{corrupt}} + 1}_{\text{Query Events}} + \underbrace{d \cdot \mathcal{Q}_{\mathsf{corrupt}}}_{\text{New Key Events}} + \underbrace{2 \cdot \mathcal{Q}_{\mathsf{corrupt}} \cdot (d-1)}_{\text{New ct Events}}$$

for any view resulting from a run of an adversary that makes at most $\mathcal{Q}_{\mathsf{corrupt}}$ corruption queries. This means that $\mathsf{len}(\mathsf{view})$ can take values up to $\mathcal{T}_{\max} = 2^{d+2} + (3d+1) \cdot \mathcal{Q}_{\mathsf{corrupt}}$. Furthermore, $\mathsf{maxcor}_{i,t}(\mathsf{view})$ takes values from 0 to d and $\mathsf{lastkey}_i(\mathsf{view})$ takes values from 0 to $\mathsf{len}(\mathsf{view})$.

Thus, $|\mathcal{Y}| \leq \left(2^{d+2} + (3d+1) \cdot \mathcal{Q}_{\mathsf{corrupt}}\right)^2 \cdot (d+1)$.

Plugging this into Eq. (3.15) yields

$$\Pr\left[\forall t \in [\mathcal{T}] : T_t^{\mathsf{rep}} \leq 2\mathcal{T} \cdot \ln(2\mathcal{T}/\gamma) \cdot \mathcal{T}_{\max}^2 \cdot (d+1)/\gamma\,\right] \geq\; 1-\gamma. \tag{3.16}$$

Thus, we can set

$$B := 2\mathcal{T} \cdot \ln(2\mathcal{T}/\gamma) \cdot \mathcal{T}_{\max}^2 \cdot (d+1)/\gamma$$

where as before $\mathcal{T}_{\max} = 2^{d+2} + (3d+1) \cdot \mathcal{Q}_{\mathsf{corrupt}}$. □

We further note that by Lemma 3.1.2 instantiated with

$$\begin{aligned} h_t(\mathsf{view}_s) &= \mathsf{pfx}_{t-1}(\mathsf{view}_s) \\ g_t(\mathsf{view}_s) &= (\mathsf{maxcor}_{i,t+1}(\mathsf{view}_s), \mathsf{len}(\mathsf{view}_s), \mathsf{lastkey}_i(\mathsf{view}_s)), \end{aligned}$$

the X_t defined by $P1_i$ are identically distributed to $\mathcal{D}^{\mathsf{view}}_{\mathsf{lkh},b_{\mathsf{lkh}}}$. Also by Lemma 3.1.2 instantiated with

$$\begin{aligned} h_t(\mathsf{view}_s) &= \mathsf{pfx}_{t-1}(\mathsf{view}_s) \\ g_t(\mathsf{view}_s) &= (\mathsf{maxcor}_{i+1,t+1}(\mathsf{view}_s), \mathsf{len}(\mathsf{view}_s), \mathsf{lastkey}_{i+1}(\mathsf{view}_s)), \end{aligned}$$

X_t defined by $P1_{i+1}$ are identically distributed to $\mathcal{D}^{\mathsf{view}}_{\mathsf{lkh},b_{\mathsf{lkh}}}$.

Since before the occurrence of G in $P2_i$, the sampling of the X_t is equivalent to that of $P1_{i+1}$, the X_t up to G are distributed identically to the X_t in $P1_{i+1}$. It follows that G also fulfills the criteria Cases 3 and 4 from Line 10.

Let B be chosen according to Claim 3.3.19. Then, the probability of F in $P1_i$ is at most γ. We now consider the probability of F in $P2_i$.

Claim 3.3.20. *The probability of F in $P2_i$ is the same as the probability of $\bot \leftarrow \mathcal{D}^{\mathsf{view}}_{i,b_{\mathsf{lkh}}}$.*

Proof of Claim 3.3.20. To see this, recall that by Lemma 3.3.16, the conditions for the **if** in Line 7 in Algorithm 19 for i are equivalent to the condition for the **if** in Line 3 in Algorithm 22. By the same argument, the end condition for the **repeat** loop is equivalent: The condition on $\mathsf{len}(\mathsf{view}_t)$ is independent of i and thus the same. By Lemma 3.3.17, if G occurred for X_t, $\mathsf{lastkey}_i(\mathsf{view}_{t'})$ and $\mathsf{lastkey}_{i+1}(\mathsf{view}_{t'})$ will be smaller than $t+1$ for any $t' > t$. Due to Lemma 3.3.15, a similar property holds for maxcor. □

Putting this together and using Line 10, the claim follows. □

Proposition 9. *$\mathcal{D}^{\mathsf{view}}_{T_d,b_{\mathsf{lkh}}}$ is independent of b_{lkh} and in particular $\mathrm{Adv}^{\mathsf{LKH},T_d}_{\mathsf{SKE},\mathsf{A},d}(\lambda) = 0$ for every* LKH *adversary* A.

Proof of Proposition 9. Recall that b_{lkh} is only used when responding to the challenge query, which by assumption is the last query the adversary makes. Hence, neither the abort probability nor any of the events in $\mathsf{view}_{\mathcal{T}}$ before the very last events $(\mathsf{challenge}, k)$ and $(\mathsf{guess}, b_{\mathsf{A}})$ depend on b_{lkh}. The latter implies that the values of maxcor and lastkey (which are computed from corruption queries and ciphertext renewal events) as well as len are independent of b_{lkh} for all t. As the resampling procedure cuts off the tail of the view (including the last two events that may contain information about b_{lkh}) when resampling the views, no information about b_{lkh} is carried from view_{t-1} to view_t. Furthermore, since $\mathsf{edges}_{T_d} = \{0,1\}$, the final view $\mathsf{view}_{\mathcal{T}}$ does not contain any encryptions of the root key, and the root key k_ε is therefore independent of $\mathsf{view}_{\mathcal{T}}$. Thus, A has no advantage in distinguishing k_ε from a random independent key sampled by $\mathbf{Gen}(1^\lambda)$. □

We are now ready to prove the theorem.

Let A be an arbitrary LKH adversary running in time t_{A}. First, C samples $i^* \leftarrow [T_d]$. It will then simulate the game $(i^*-1).1$ or i^* depending on the challenge it gets from its own challenger. To generate a sample $\mathsf{view}_{\mathcal{T}}$, our IND-CPA adversary C modifies the procedure of Algorithm 19 by embedding an IND-CPA challenge in the if clause in the repeat loop, see Algorithm 23.

We briefly describe the algorithm. Let $x^* = \mathsf{edges}_{d,i-1} \Delta \mathsf{edges}_{d,i}$ be the edge that needs to either be pebbled or unpebbled in this game hop. The variable β indicates whether the former or the latter is the case. The core idea of Algorithm 23 is that it runs Algorithm 19, except that as the edge set it considers $\mathsf{edges}_{d,i} \cup \{x^*\}$, and when the edge x^* would be re-sampled during a rewinding to a ctxt event, the ciphertext is replaced with an IND-CPA challenge, where the permutation of the challenge messages is chosen depending on whether the edge is to be pebbled or unpebbled in the game hop (i.e. depending on β), implicitly setting the corresponding encryption key at the lower end of the edge to the challenge key. This is possible as for an edge to be pebbled or unpebbled, both other edges incident to its lower vertex need to be pebbled, i.e. both of the ciphertexts sitting on those edges need to have been replaced by encryptions of $\perp$ already, thus revealing nothing about the challenge key. Furthermore, any 'honest' encryptions that need to be made with regard to this challenge key can be obtained by calling the LoR oracle provided by the IND-CPA challenger with two identical messages.

For any $x \in \mathsf{edges}_{d,i} \setminus \{x^*\}$, the algorithm C resamples a ciphertext of $\perp$ according to the same criteria as Algorithm 19. C outputs 0 if A succeeds and 1 else.

We have that for $b_{\mathsf{indcpa}} = 0$ and $i^* = i$ the modified algorithm samples from exactly the same distribution as Algorithm 19 on input $i-1$ in the gray mode (and same $b_{\mathsf{lkh}} \in \{0,1\}, B \in \mathbb{N}$), and for $b_{\mathsf{indcpa}} = 1$ and $i^* = i$ from the same distribution as Algorithm 19 in the plain mode on input i (and same $b_{\mathsf{lkh}} \in \{0,1\}, B \in \mathbb{N}$). We obtain for the advantage of C:

$$\begin{aligned}
\mathrm{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{SKE},\mathsf{C}}(\lambda) &= \Pr[b_{\mathsf{C}} = b_{\mathsf{indcpa}}] - \frac{1}{2} \\
&= \frac{1}{2} \cdot \left(\Pr[b_{\mathsf{C}} = 0 \mid b_{\mathsf{indcpa}} = 0] + \Pr[b_{\mathsf{C}} = 1 \mid b_{\mathsf{indcpa}} = 1] - 1\right) \\
&= \frac{1}{2T_d} \cdot \sum_{i \in [T_d]} \left(\Pr\left[b_{\mathsf{C}} = 0 \,\middle|\, \begin{array}{c} b_{\mathsf{indcpa}} = 0 \\ \wedge\, i^* = i \end{array} \right] - \Pr\left[b_{\mathsf{C}} = 0 \,\middle|\, \begin{array}{c} b_{\mathsf{indcpa}} = 1 \\ \wedge\, i^* = i \end{array} \right] \right) \\
&= \frac{1}{2T_d} \cdot \sum_{i \in [T_d]} \left(\Pr[\mathrm{out}_{(i-1).1} = 1] - \Pr[\mathrm{out}_i = 1] \right) \\
&= \frac{1}{2T_d} \cdot \sum_{i \in [T_d]} \left(\mathrm{Adv}^{\mathsf{LKH},\,(i-1).1}_{\mathsf{SKE},\mathsf{A},d}(\lambda) - \mathrm{Adv}^{\mathsf{LKH},\,i}_{\mathsf{SKE},\mathsf{A},d}(\lambda) \right).
\end{aligned}$$

By Proposition 8 we have for all $i \in [T_d]$

$$\mathrm{Adv}^{\mathsf{LKH},\,(i-1).1}_{\mathsf{SKE},\mathsf{A},d}(\lambda) - \mathrm{Adv}^{\mathsf{LKH},\,i-1}_{\mathsf{SKE},\mathsf{A},d}(\lambda) \geq -\gamma$$

and hence we obtain

$$\begin{aligned}
\mathrm{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{SKE},\mathsf{C}}(\lambda) &= \frac{1}{2} \cdot \frac{1}{T_d} \cdot \sum_{i \in [T_d]} \left(\mathrm{Adv}^{\mathsf{LKH},\,(i-1).1}_{\mathsf{SKE},\mathsf{A},d}(\lambda) - \mathrm{Adv}^{\mathsf{LKH},\,i}_{\mathsf{SKE},\mathsf{A},d}(\lambda) \right) \\
&\geq \frac{1}{2} \cdot \frac{1}{T_d} \cdot \sum_{i \in [T_d]} \left(\mathrm{Adv}^{\mathsf{LKH},\,i-1}_{\mathsf{SKE},\mathsf{A},d}(\lambda) - \mathrm{Adv}^{\mathsf{LKH},\,i}_{\mathsf{SKE},\mathsf{A},d}(\lambda) - \gamma \right) \\
&\geq \frac{1}{2} \cdot \frac{1}{T_d} \cdot \left(\mathrm{Adv}^{\mathsf{LKH}}_{\mathsf{SKE},\mathsf{A},d}(\lambda) - \mathrm{Adv}^{\mathsf{LKH},\,T_d}_{\mathsf{SKE},\mathsf{A},d}(\lambda) \right) - \frac{\gamma}{2}.
\end{aligned}$$

Plugging in Proposition 9 yields

$$\mathrm{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{SKE},\mathsf{C}}(\lambda) \geq \frac{1}{2} \cdot \frac{1}{T_d} \cdot \mathrm{Adv}^{\mathsf{LKH}}_{\mathsf{SKE},\mathsf{A},d}(\lambda) - \frac{\gamma}{2}.$$

Algorithm 23: Variant of Algorithm 19 for sampling from $\mathcal{D}^{\mathsf{view}}_{i^*-1+b_{\mathsf{indcpa}},b_{\mathsf{lkh}}}$ given oracle access to an IND-CPA challenger with challenge bit b_{indcpa}. The functions len, $\mathsf{maxcor}_{i^*,t}$, $\mathsf{lastkey}_{i^*}$ are described in the proof, B is as in Claim 3.3.19.

```
1  $i^* \leftarrow \{1,\dots,T_d\}$                       // guess which games need to be distinguished
2  $b_{\mathsf{lkh}} \leftarrow \{0,1\}$
3  Initialize $\iota := 0$                                  // Counter for IND-CPA users
4  $y^*\|b^* := x^* := \mathsf{edges}_{d,i^*-1}\Delta\mathsf{edges}_{d,i^*}$, where $b^* \in \{0,1\}$      // differing edge index
     $x^* = y^*\|b^*$
5  $\beta := [x^* \in \mathsf{edges}_{d,i^*-1}]$   // bit $\beta$ indicates whether a pebble is added or removed
     in $i^*$th step
6  $\mathsf{view}_0 \leftarrow \mathcal{D}^{\mathsf{view}}_{\mathsf{lkh},b_{\mathsf{lkh}}}$
7  $\mathcal{T} := \mathsf{len}(\mathsf{view}_0)$                            // Length of $\mathsf{view}_{\mathsf{lkh}}$ (in entries)
8  $t^* := \mathsf{lastkey}_{i^*}(\mathsf{view}_0)$                   // Time when last key for $x^*$ is generated
9  for $t := 1$ to $\mathcal{T}$ do
10     Write $\mathsf{view}_{t-1} = (\mathsf{ev}_{t-1,1},\dots,\mathsf{ev}_{t-1,\mathcal{T}})$
11     repeat                                  // Output $\perp$ if $B$ repetitions fail for this $t$
12         Rewind adversary to point $t$
13         if $t = t^*$ then
14             $\iota := \iota + 1$                                 // Update current user index
15             NU()                                // Embed fresh IND-CPA challenge key
16         end
17         if $\mathsf{ev}_{t-1,t} = (\mathsf{ctxt}, x, c_x)$ with $x \in \mathsf{edges}_{d,i^*} \cup \{x^*\}$ and $\mathsf{maxcor}_{i^*,t+1}(\mathsf{view}_{t-1}) < |x|$
            then
                                   // Checks if $\mathsf{ev}_{t,t}$ defines a ciphertext to be pebbled
18             if $x = x^*$ then
19                 Set $k^*_\beta := k_{y^*}$, $k^*_{1-\beta} := \perp$
20                 $c^* \leftarrow \mathsf{LoR}(\iota, k_0, k_1)$ from IND-CPA challenger  // Fresh IND-CPA challenge
                    ciphertext
21                 Set $\mathsf{ev}_{t,t} := (\mathsf{ctxt}, x^*, c^*)$            // Replace $c_x$ with $c^*$ in $\mathsf{view}_{t-1}$
22             else
23                 Sample fresh $c_x^\perp \leftarrow \mathbf{Enc}(k_x, \perp)$             // Fresh dummy ciphertext
24                 Set $\mathsf{ev}_{t,t} := (\mathsf{ctxt}, x, c_x^\perp)$           // Replace $c_x$ with $c_x^\perp$ in $\mathsf{view}_{t-1}$
25             end
26             Resample from point $t+1$ to obtain
                $\mathsf{view}_t = (\mathsf{ev}_{t-1,0},\dots,\mathsf{ev}_{t-1,t-1},\mathsf{ev}_{t,t},\mathsf{ev}_{t,t+1},\dots,\mathsf{ev}_{t,\tau})$
27         else
28             Resample from point $t$ to obtain $\mathsf{view}_t = (\mathsf{ev}_{t-1,0},\dots,\mathsf{ev}_{t-1,t-1},\mathsf{ev}_{t,t},\dots,\mathsf{ev}_{t,\tau})$ but
                with the following change: for all $t' > t^*$ with $\mathsf{ev}_{t,t'} = (\mathsf{ctxt}, x^*, c_{x^*})$ generate
                $c_{x^*} \leftarrow \mathsf{LoR}(\iota, k_{y^*}, k_{y^*})$
29         end
30     until $\mathsf{maxcor}_{i^*,t+1}(\mathsf{view}_t) = \mathsf{maxcor}_{i^*,t+1}(\mathsf{view}_{t-1})$ and $\mathsf{len}(\mathsf{view}_t) = \mathsf{len}(\mathsf{view}_{t-1})$ and
        $\mathsf{lastkey}_{i^*}(\mathsf{view}_t) = \mathsf{lastkey}_{i^*}(\mathsf{view}_{t-1})$
31 end
32 return $[\mathrm{out}_{\mathsf{A}}(\mathsf{view}_{\mathcal{T}}) = b_{\mathsf{lkh}}]$
```

For the runtime analysis we see that the **for** loop is called $\mathcal{T}$ times and the **repeat** loop is called at most B times. During each run of the **repeat** loop, the adversary A is called once. This yields the runtime given in the theorem statement. $\square$

Chapter 4

The Abe-Okamoto Partially Blind Signature Scheme Revisited

This chapter is based on [KLX22c]. In this chapter, we revisit the Abe-Okamoto Partially Blind Signature Scheme [AO00].

As the scheme is based on the OR-proof technique, the reduction's strategy is to use one of the two OR-proof witnesses to generate signatures, while it embeds a discrete logarithm challenge in the other branch. It then hopes to extract the other witness using the forking technique, i.e. rewinding the adversary and responding with different responses to certain hash queries. While the protocol is normally witness indistinguishable, Abe and Okamoto [AO00] already observed that the concatenation of two such forking runs often reveals which witness the reduction is using internally. Therefore, they apply a probability analysis showing that the desired witness can still be extracted with a good probability. As explained in Section 1.3.1, this probability analysis contains a gap. In this chapter, we rewrite the proof of security of the Abe-Okamoto scheme and mend the gap. We first give an overview of the scheme in Section 4.1. In Section 4.2, we do a lot of preliminary work to compute the probability of extracting the desired witness. We start by describing a wrapper adversary in Section 4.2.1 which allows the reduction to consider the adversary as a deterministic machine where the wrapper also simulates all oracles to the adversary. This allows us to argue about the input-output behaviour of the wrapper when run with the adversary as a subroutine. We then provide some useful definitions related to the interaction of the wrapper with the adversary in Section 4.2.2. After that, we are ready to start our counting arguments for calculating the probability. We first count partners and triangles in Section 4.2.3. Then we introduce the transcript mapping function Φ and prove its relevant properties in Section 4.2.4 before we can also count the relevant objects in its image in Section 4.2.5. This finally allows us to find the probability of extracting a witness in Section 4.2.6. We then turn to the main forking-based proof in Section 4.3. As this proof deals only with adversaries that use a single tag info, we extend the result to multiple tags using the same strategy as Abe and Okamoto [AO00] in Section 4.4.

4.1 The Abe-Okamoto Partially Blind Signature Scheme

In this section we describe the partially blind signature scheme by Abe and Okamoto [AO00]. It runs a proof of knowledge that the signer knows either the secret key x or the discrete logarithm of the so-called *tag key* $\mathbf{z}$, which is obtained through hashing the tag info. In this way we obtain a *witness indistinguishable* scheme: an honest signer does not know $\operatorname{dlog}\mathbf{z}$ and is forced to use x for issuing signatures; while the reduction may program the random oracle so that it knows the $\operatorname{dlog}\mathbf{z}$ and can then simulate the signer without knowing the secret key x. To an outsider, e.g. an adversary, the two modes of operation, i.e. whether x or $\operatorname{dlog}\mathbf{z}$ was used are indistinguishable.

Key Generation. On input public parameters $\mathsf{pp} = (\mathbb{G}, \mathbf{g}, q, \mathsf{H}^*, \mathsf{H})$ (where H^* and H are random oracles with ranges $\mathbb{G}$ and $\mathbb{Z}_q$, respectively), KeyGen samples $x \xleftarrow{\$} \mathbb{Z}_q$ and sets $\mathbf{y} := \mathbf{g}^x$. It then outputs $(\mathsf{pk}, \mathsf{sk}) := (\mathbf{y}, x)$.

Signer. $\mathsf{Sign} = (\mathsf{Sign}_1, \mathsf{Sign}_2)$ behaves as follows:

Sign_1**:** On input info and sk, Sign_1 computes the tag key $\mathbf{z} := \mathsf{H}^*(\mathsf{info})$ and samples $u, s, d \xleftarrow{\$} \mathbb{Z}_q$. It then computes the *commitments* $\mathbf{a} := \mathbf{g}^u, \mathbf{b} := \mathbf{g}^s \cdot \mathbf{z}^d$. It outputs the response $(\mathbf{a}, \mathbf{b})$ to the user and an internal state $\mathsf{st}_{\mathsf{Sign}} := (u, s, d)$.

Sign_2**:** On input $e \in \mathbb{Z}_q, \mathsf{st}_{\mathsf{Sign}} = (u, s, d), \mathsf{sk} = x$, Sign_2 computes $c := e - d$ and $r := u - cx$. It outputs the response (r, c, s, d) to the user.

User. $\mathsf{User} = (\mathsf{User}_1, \mathsf{User}_2)$ behaves as follows:

User_1**:** On input $\mathsf{pk}, m, \mathsf{info}, \mathbf{a}, \mathbf{b}$, User_1 computes the tag key $\mathbf{z} := \mathsf{H}^*(\mathsf{info})$ and samples $t_1, t_2, t_3, t_4 \xleftarrow{\$} \mathbb{Z}_q$. It then computes $\boldsymbol{\alpha} := \mathbf{g}^{t_1} \cdot \mathbf{y}^{t_2} \cdot \mathbf{a}$ and $\boldsymbol{\beta} := \mathbf{g}^{t_3} \cdot \mathbf{z}^{t_4} \cdot \mathbf{b}$, queries $h := \mathsf{H}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{z}, m)$ for the message m it wants to sign, and computes the blinded challenge $e := h - t_2 - t_4$. It outputs e to the signer and an internal state $\mathsf{st}_{\mathsf{User}} := (t_1, t_2, t_3, t_4)$.

User_2**:** On input $\mathsf{pk}, (r, c, s, d), \mathsf{st}_{\mathsf{User}} = (t_1, t_2, t_3, t_4)$, User_1 computes $\rho := r + t_1$, $\omega := c + t_2$, $\sigma := s + t_3$, and $\delta := d + t_4$. It then verifies that $\omega + \delta = \mathsf{H}(\mathbf{g}^\rho \cdot \mathbf{y}^\omega, \mathbf{g}^\sigma \cdot \mathbf{z}^\delta, \mathbf{z}, m)$; if so, it outputs the signature $(\rho, \omega, \sigma, \delta)$. (Otherwise, it outputs $\perp$.)

Verification. On input $\mathbf{y}, m, \mathsf{info}, (\rho, \omega, \sigma, \delta)$, Verify computes $\mathbf{z} := \mathsf{H}^*(\mathsf{info})$. It outputs 1 if $\omega + \delta = \mathsf{H}(\mathbf{g}^\rho \cdot \mathbf{y}^\omega, \mathbf{g}^\sigma \cdot \mathbf{z}^\delta, \mathbf{z}, m)$ and 0 otherwise.

For a graphic illustration of the scheme, see Figure 4.1

4.2 Computing the Probability for Extracting the 'Good' Witness

As mentioned in the introduction, our analysis of the Abe-Okamoto scheme is done in two steps. In this section, we deal with the case that the adversary U only uses a *single* tag, i.e., U plays the ℓ-1-info-$\mathbf{OMUF}_{\mathsf{AO}}$ game.

4.2.1 The Deterministic OMUF Wrapper

Restricting the Adversary to Making $\ell + 1$ Hash Queries.

Suppose that the adversary U makes ℓ queries to sign_2 (henceforth "signing queries") and Q_h queries to H (henceforth "hash queries"), and uses a single tag $\overline{\mathsf{info}}$. Below we assume w.l.o.g. that U never makes the same query to H twice.

We say that a message-signature pair $(m, (\rho, \omega, \sigma, \delta))$ *corresponds to* an index $i \in [Q_h]$, or corresponds to the adversary U's i-th hash query, if this query was $\mathsf{H}(\mathbf{y}^\omega \mathbf{g}^\rho, \mathbf{z}^\delta \mathbf{g}^\sigma, \mathbf{z}, m)$. (When the message m is

$$
\begin{array}{lcr}
\text{Signer} & & \text{User} \\
\mathsf{sk} = x & & \mathsf{pk} = \mathbf{y} \\
\mathsf{pk} = (\mathbf{y} = \mathbf{g}^x), \mathbf{z} = \mathsf{H}^*(\mathsf{info}) & & m, \mathsf{info}, \mathbf{z} = \mathsf{H}^*(\mathsf{info}) \\
\hline
u, s, d \xleftarrow{\$} \mathbb{Z}_q & & \\
\mathbf{a} := \mathbf{g}^u & & \\
\mathbf{b} := \mathbf{g}^s \cdot \mathbf{z}^d & & \\
& \xrightarrow{\mathbf{a},\mathbf{b}} & \\
& & t_1, t_2, t_3, t_4 \xleftarrow{\$} \mathbb{Z}_q \\
& & \boldsymbol{\alpha} := \mathbf{g}^{t_1} \cdot \mathbf{y}^{t_2} \cdot \mathbf{a} \\
& & \boldsymbol{\beta} := \mathbf{g}^{t_3} \cdot \mathbf{y}^{t_4} \cdot \mathbf{b} \\
& & h := \mathsf{H}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{z}, m) \\
& & e := h - t_2 - t_4 \\
& \xleftarrow{e} & \\
c := e - d & & \\
r := u - cx & & \\
& \xrightarrow{c,d,r,s} & \\
& & \rho := r + t_1 \\
& & \omega := c + t_2 \\
& & \sigma := s + t_3 \\
& & \delta := d + t_4 \\
& & \omega + \delta \stackrel{?}{=} \mathsf{H}(\mathbf{g}^\rho \cdot \mathbf{y}^\omega, \mathbf{g}^\sigma \cdot \mathbf{z}^\delta, \mathbf{z}, m) \\
& & \Downarrow \\
& & \mathsf{sig} := (\rho, \omega, \sigma, \delta)
\end{array}
$$

Figure 4.1: The Abe-Okamoto scheme

clear from context, we may say that the signature $(\rho, \omega, \sigma, \delta)$ corresponds to index i.) We remark that we can further assume w.l.o.g. that there exist $\ell + 1$ hash queries of U, each of which corresponds to a distinct message-signature pair in the output of U (in particular, $Q_h \geq \ell + 1$). This is because otherwise one of the following must hold (assuming that U succeeds):

- There exists a pair $(m, (\omega, \rho, \delta, \sigma))$ that does not correspond to any hash query, i.e., $\mathsf{H}(\mathbf{y}^\omega \mathbf{g}^\rho, \mathbf{z}^\delta \mathbf{g}^\sigma, \mathbf{z}, m)$ has never been queried. In this case, U can be turned into another adversary U′ that runs the code of U and additionally makes such a hash query; obviously U and U′ have the same advantage.

- There exist two distinct pairs $(m_1, (\omega_1, \rho_1, \delta_1, \sigma_1)), (m_2, (\omega_2, \rho_2, \delta_2, \sigma_2))$ that correspond to the same hash query. In this case, we have that $m_1 = m_2$, $\mathbf{y}^{\omega_1} \mathbf{g}^{\rho_1} = \mathbf{y}^{\omega_2} \mathbf{g}^{\rho_2}$, and $\mathbf{z}^{\delta_1} \mathbf{g}^{\sigma_1} = \mathbf{z}^{\delta_2} \mathbf{g}^{\sigma_2}$. Then a reduction to the discrete logarithm problem can easily compute both x and w as $x = (\omega_1 - \omega_2)^{-1} \cdot (\rho_2 - \rho_2)$ and $w = (\delta_1 - \delta_2)^{-1} \cdot (\sigma_2 - \sigma_1)$.

It is not hard to see that any adversary U can be turned into another adversary that makes *exactly* $\ell + 1$ hash queries, with a factor of $\binom{Q_h}{\ell+1}$ loss in advantage. Formally, we define an adversary $\mathsf{M} := \mathsf{M}^{\mathsf{U}}$ that works as follows. M, on input of a public key pk, chooses a random subset I of $[Q_h]$ with $|I| = \ell + 1$, and invokes U(pk). For U's i-th query to H, if $i \notin I$, M responds with a random integer in $\mathbb{Z}_q$. For any other query (including queries to signing oracles, queries to H^*, and the i-th query to H for $i \in I$), M forwards it to the corresponding oracle of M's own challenger, and forwards the response back to

U. When U outputs a set of $\ell+1$ message-signature pairs, M checks if every pair $(m, (\rho, \omega, \sigma, \delta))$ corresponds to some index $i \in I$, that is, U's i-th hash query was $\mathsf{H}(\mathbf{y}^{\omega}\mathbf{g}^{\rho}, \mathbf{z}^{\delta}\mathbf{g}^{\sigma}, \mathbf{z}, m)$. If so, M copies U's output (and outputs $\perp$ otherwise).

Lemma 4.2.1. *For* M *described above, we have that*

$$\mathrm{Adv}_{\mathsf{M}}^{\ell\text{-1-info-}\mathbf{OMUF}_{\mathsf{AO}}} \geq \frac{\mathrm{Adv}_{\mathsf{U}}^{\ell\text{-1-info-}\mathbf{OMUF}_{\mathsf{AO}}}}{\binom{Q_h}{\ell+1}}.$$

Proof. It is straightforward that M simulates the OMUF game to U perfectly. Assume that U succeeds. By our assumption on U, there is a set of indices $I^* \subset [Q_h]$ corresponding to the message-signature pairs in U's output, with $|I^*| = \ell+1$. If $I^* = I$, then M also succeeds. Since I is a random subset of size $\ell+1$ of $[Q_h]$, the probability that $I^* = I$ is $\frac{1}{\binom{Q_h}{\ell+1}}$. The lemma follows. □

The lemma above implies that it is sufficient to consider an adversary that makes exactly $\ell+1$ (distinct) hash queries, since an upper bound of the adversary's advantage in this specific case immediately translates to such an upper bound in the general case. Below we simply assume that the adversary makes $\ell+1$ hash queries.

The Deterministic Wrapper.

For any adversary M that makes exactly $\ell+1$ distinct hash queries, we define a deterministic *wrapper* A that, given the witness and random coin tosses for one side, simulates the view of M. The wrapper uses either the $\mathbf{y}$-side witness (i.e., the secret key) x or the $\mathbf{z}$-side witness $w = \operatorname{dlog} \mathbf{z}$ to respond to sign$_2$ queries, and simulates the other side of the OR-proof using fixed values. We begin with the formal definition of an instance:

Definition 4.2.2 (Instances). *For the deterministic wrapper simulating the OMUF-game to the adversary we define two types of* instances $\mathbf{I}$. *A* $\mathbf{y}$-side *(a.k.a. honest)* instance *consists of the following components:*

$b = 0$: *bit indicating that the secret key x will be used for simulation*

x: *the secret key, also referred to as the $\mathbf{y}$-side witness*

$\mathbf{z}$: *the tag key, to be returned by oracle* H^* *for requested* $\overline{\text{info}}$

d_i, s_i: *simulator choices for $\mathbf{z}$-side part corresponding to the i-th signing session*

u_i: *discrete logarithm of the $\mathbf{y}$-side commitment $\mathbf{a}_i$ in the i-th signing session*

A $\mathbf{z}$-side instance *consists of the following components:*

$b = 1$: *bit indicating that the tag witness w will be used for simulation*

$\mathbf{y}$: *the public key*

w: *the discrete logarithm of the tag key $\mathbf{z}$ as above*

c_i, r_i: *simulator choices for $\mathbf{y}$-side part corresponding to the i-th signing session*

v_i: *discrete logarithm of the $\mathbf{z}$-side commitment $\mathbf{b}_i$ in the i-th signing session*

Let $\overrightarrow{h}$ be the vector of responses returned by random oracle H (so $\left|\overrightarrow{h}\right| = \ell + 1$), rand be the randomness used by the adversary M, and $\overline{\mathsf{info}}$ be the tag used in the OMUF game. We define a deterministic wrapper A $:= \mathsf{A}^{\mathsf{M}}_{\overline{\mathsf{info}}}$ that runs on $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ as shown in Figure 4.2. The wrapper allows us to argue about which $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ tuples cause the adversary to succeed.

A has two simulation modes. For $b = 0$, it runs the honest signer's algorithm to simulate both sign_1 and sign_2 oracle queries; for H^* queries, it responds with $\mathbf{z}$ if the input is $\overline{\mathsf{info}}$ and $\perp$ for all other inputs. In mode $b = 1$, A knows w and not x. It therefore runs the so-called $\mathbf{z}$-side signer (see Figure 4.3), which is the honest signer's algorithm except that w is treated as the secret key. A responds to queries to H^* with $\mathbf{g}^w$ for $\overline{\mathsf{info}}$ and $\perp$ otherwise. In both modes, A responds to queries to H using entries in the hash vector $\overrightarrow{h}$. Finally, upon receiving M's output message-signature pairs, A checks if they are all valid, and if so, A copies M's output (and outputs $\perp$ otherwise).

It is easy to see that

$$t_{\mathsf{A}} = t_{\mathsf{M}} + \mathrm{O}(\ell) = t_{\mathsf{U}} + \mathrm{O}(\ell) + \mathrm{O}({Q_h}^2) = t_{\mathsf{U}} + \mathrm{O}(\ell) + \mathrm{O}({Q_h}^2),$$

where the term $\mathrm{O}(\ell)$ comes from verifying $\ell + 1$ signatures, and $\mathrm{O}({Q_h}^2)$ comes from identifying the hash indices that correspond to signatures.

The Set of Successful Tuples.

Let

$$\mathsf{Succ} := \{(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) | \mathsf{A}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \neq \perp\}$$

be the set of all "successful" input tuples to the wrapper A. For a pair of instance and randomness $\mathbf{I}, \mathsf{rand}$, it is also useful to define $\mathsf{Succ}_{\mathbf{I},\mathsf{rand}}$ as the set of successful input tuples with instance $\mathbf{I}$ and randomness rand, i.e.,

$$\mathsf{Succ}_{\mathbf{I},\mathsf{rand}} := \left\{ (\mathbf{I}', \mathsf{rand}', \overrightarrow{h}) \in \mathsf{Succ} \middle| \begin{array}{c} \mathbf{I}' = \mathbf{I} \\ \mathsf{rand}' = \mathsf{rand} \end{array} \right\}.$$

In the following we further denote by $\mathcal{I}$ the set of all possible instances, by $\mathcal{R}$ the set of all possible randomness of A, and by ϵ the success probability of A, i.e.,

$$\epsilon := \frac{|\mathsf{Succ}|}{\left|\mathcal{I} \times \mathcal{R} \times \mathbb{Z}_q^{\ell+1}\right|}$$

We show in Lemma 4.2.21 below (in Section 4.2.4) that the simulation using the $\mathbf{z}$-side witness is perfectly indistinguishable from the real execution where the $\mathbf{y}$-side witness is used (this is called the *witness indistinguishability* of the scheme), i.e., A simulates the OMUF game to M perfectly. Furthermore, if M succeeds, then so does A, since A copies M's output in this case (see lines 10–11 of Figure 4.2). Therefore,

$$\epsilon = \mathrm{Adv}_{\mathsf{M}}^{\ell\text{-}1\text{-}\mathsf{info}\text{-}\mathbf{OMUF}_{\mathsf{AO}}}.$$

4.2.2 Basic Definitions

We first define some concepts related to the wrapper A's input tuple $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$, that will be used throughout the security proof.

```
A($\mathbf{I}$, rand, $\overrightarrow{h}$)
parse $b$ from $\mathbf{I}$
if $b = 0$
    parse $(b, x, \mathbf{z}, \overrightarrow{d}, \overrightarrow{s}, \overrightarrow{u}) := \mathbf{I}$
    pk $:= \mathbf{g}^x$
else
    parse $(b, \mathbf{y}, w, \overrightarrow{c}, \overrightarrow{r}, \overrightarrow{v}) := \mathbf{I}$
    pk $:= \mathbf{y}$
sid $:= 0$
$j := 0$
$(m_i, (\rho_i, \omega_i, \sigma_i, \delta_i))_{i=1}^{\ell+1} :=$ $\mathsf{M}^{\mathsf{sign}_1, \mathsf{sign}_2, \mathsf{H}, \mathsf{H}^*}$(pk; rand)
if $\forall i$: Verify(pk, $m_i, (\rho_i, \omega_i, \sigma_i, \delta_i)$)
    return $(m_i, (\rho_i, \omega_i, \sigma_i, \delta_i))_{i=1}^{\ell+1}$
else
    return $\perp$

H($\xi$)
$j$++
return $h_j$

H*(info)
if info = $\overline{\text{info}}$
    if $b = 0$ return $\mathbf{z}$
    else return $\mathbf{g}^w$
else return $\perp$

sign$_1$(info)
if info = $\overline{\text{info}}$
    sid++
    open(sid) := true
    if $b = 0$
        $\mathbf{a}_{\mathsf{sid}} := \mathbf{g}^{u_{\mathsf{sid}}}$
        $\mathbf{b}_{\mathsf{sid}} := \mathbf{g}^{s_{\mathsf{sid}}} \cdot \mathbf{z}^{d_{\mathsf{sid}}}$
    else
        $\mathbf{a}_{\mathsf{sid}} := \mathbf{g}^{r_{\mathsf{sid}}} \cdot \mathbf{y}^{c_{\mathsf{sid}}}$
        $\mathbf{b}_{\mathsf{sid}} := \mathbf{g}^{v_{\mathsf{sid}}}$
    return (sid, $\mathbf{a}_{\mathsf{sid}}, \mathbf{b}_{\mathsf{sid}}$)
else return $\perp$

sign$_2$(sid, $e_{\mathsf{sid}}$)
if open(sid)
    if $b = 0$
        $c_{\mathsf{sid}} := e_{\mathsf{sid}} - d_{\mathsf{sid}}$
        $r_{\mathsf{sid}} := u_{\mathsf{sid}} - c_{\mathsf{sid}} \cdot x$
    else
        $d_{\mathsf{sid}} := e_{\mathsf{sid}} - c_{\mathsf{sid}}$
        $s_{\mathsf{sid}} := v_{\mathsf{sid}} - d_{\mathsf{sid}} \cdot w$
else
    return $\perp$
open(sid) := false
return $(c_{\mathsf{sid}}, r_{\mathsf{sid}}, d_{\mathsf{sid}}, s_{\mathsf{sid}})$
```

Figure 4.2: Wrapper A that simulates the OMUF game to the adversary M

Transcripts.

We begin with the definition of the *query transcript*, which consists of the adversary's signing queries:

Definition 4.2.3 (Query Transcript)**.** *Consider the wrapper* A *running on input tuple* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$*. The* query transcript*, denoted* $\overrightarrow{e}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$*, is the vector of queries* e_{sid} *made to the* sign_2 *oracle (simulated by* A*) by the adversary* M*, ordered by* sid.

Next, we define (full) interaction *transcripts* between adversary M and wrapper A. These contain, in addition to $\overrightarrow{e}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$, also M's sign_1 queries and the signatures from the output of M. This will be useful to argue about A's behavior on different inputs $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$. Looking ahead, we will see that it is possible to deterministically transform $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ into a dual input $\Phi_{\mathsf{rand}, \overrightarrow{h}}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ that results in the same behavior as $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ (i.e., produces the same full transcript as $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$), but inverts the type of the witness $\mathbf{I}$ from $\mathbf{y}$-side to $\mathbf{z}$-side (or vice-versa).

Definition 4.2.4 (Full Transcripts)**.** *Consider the wrapper* A *running on input tuple* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$*. We denote by* $\mathsf{tr}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ *the transcript produced between* A *and the adversary* M*, i.e., all messages sent between the user (played by* M*) and the signer (played by* A*). Concretely,*

$$\mathsf{tr}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) = \left(\mathsf{info}, (\overrightarrow{\mathbf{a}}, \overrightarrow{\mathbf{b}}), \overrightarrow{e}, (\overrightarrow{c}, \overrightarrow{r}, \overrightarrow{d}, \overrightarrow{s}), \mathsf{sig}_1, \ldots \mathsf{sig}_{\ell+1}\right),$$

where $\mathsf{sig}_1, \ldots, \mathsf{sig}_{\ell+1}$ *are the signatures output by* M*. (If* M *aborts at any point during the protocol or outputs fewer than* $\ell + 1$ *signatures, we consider any undefined entry to be* $\perp$*.)*

Forking, Partners, and Triangles.

We next define what it means for two input tuples to *fork* successfully — this corresponds to all cases where the reduction would be able to compute at least one of the two witnesses from the resulting signatures. However, without further work, the witness that can be computed might be the one that the reduction already knows.

Definition 4.2.5 (Successful forking)**.** *We say two successful input tuples* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}), (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}') \in$ Succ fork *from each other at index* $i \in [\ell + 1]$ *if* $\overrightarrow{h}_{[i-1]} = \overrightarrow{h}'_{[i-1]}$ *but* $h_i \neq h_i$*. We denote the set of hash vector pairs* $(\overrightarrow{h}, \overrightarrow{h}')$ *such that* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$*,* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$ *fork at index* i *as* $F_i(\mathbf{I}, \mathsf{rand})$.

We now define *partners*, which will play a key role in our analysis. Informally, two tuples $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ and $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$ are partners at some index i if they fork from this index and produce the same query transcript (but not necessarily the same full transcript).

Definition 4.2.6 (Partners)**.** *We say two (successful) tuples* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}), (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$ *are* partners *at index* $i \in [\ell + 1]$ *if the followings hold:*

- $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ *and* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$ *fork at index* i
- $\overrightarrow{e}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) = \overrightarrow{e}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$

We denote the set of $(\overrightarrow{h}, \overrightarrow{h}')$ *such that* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ *and* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$ *are partners at index* i *by* $\mathrm{prt}_i(\mathbf{I}, \mathsf{rand})$*. We further denote by* $P_{\mathbf{I},\mathsf{rand}}$ *the following set:*

$$P_{\mathbf{I},\mathsf{rand}} = \left\{(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in \mathsf{Succ}_{\mathbf{I},\mathsf{rand}} \middle| \exists \overrightarrow{h}', i \in [\ell+1] \colon (\overrightarrow{h}, \overrightarrow{h}') \in \mathrm{prt}_i(\mathbf{I}, \mathsf{rand})\right\}$$

We define *triangles* in order to extend the nice properties of partners to more general forking tuples. Informally, a triangle consists of three vectors $\overrightarrow{h}, \overrightarrow{h'}, \overrightarrow{h''}$ which all fork from each other at the same index, and also have the property that $\overrightarrow{h}$ and $\overrightarrow{h'}$ are partners at this index. This way, it is natural to view these vectors as corners of the triangle and any pair of two vectors as the sides.

Definition 4.2.7 (Triangles)**.** *A* triangle *at index* $i \in [\ell + 1]$ *with respect to* $\mathbf{I}, \mathsf{rand}$ *is a tuple of three (successful) tuples in the following set:*

$$\triangle_i(\mathbf{I}, \mathsf{rand}) = \left\{ \begin{array}{c|c} ((\mathbf{I}, \mathsf{rand}, \overrightarrow{h}), & (\overrightarrow{h}, \overrightarrow{h}') \in \mathrm{prt}_i(\mathbf{I}, \mathsf{rand}) \\ (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}'), & (\overrightarrow{h}, \overrightarrow{h}'') \in F_i(\mathbf{I}, \mathsf{rand}) \\ (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}'')) & (\overrightarrow{h}', \overrightarrow{h}'') \in F_i(\mathbf{I}, \mathsf{rand}) \end{array} \right\}$$

For a triangle $((\mathbf{I}, \mathsf{rand}, \overrightarrow{h}), (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}'), (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}'')) \in \triangle_i(\mathbf{I}, \mathsf{rand})$*, we call the pair of tuples* $((\mathbf{I}, \mathsf{rand}, \overrightarrow{h}), (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}'))$ *the* base*, and* $((\mathbf{I}, \mathsf{rand}, \overrightarrow{h}), (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}''))$ *and* $((\mathbf{I}, \mathsf{rand}, \overrightarrow{h}'), (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}''))$ *the* sides*. We further refer to the tuples* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$*,* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$*,* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}'')$ *as* corners*, where the two corners incident to the base are called* base corners*, and the third corner is called the* top*. We will sometimes write* $(\overrightarrow{h}, \overrightarrow{h}', \overrightarrow{h}'') \in \triangle_i(\mathbf{I}, \mathsf{rand})$ *for compactness.*

Maximum Branching Index and Set.

In the following we define two important characteristics of partner tuples. We begin by defining the *maximum branching index*, which is the index at which a partner tuple $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in P_{\mathbf{I},\mathsf{rand}}$ has *the most partners*.

Definition 4.2.8 (Maximum Branching Index)**.** *Fix a pair* $\mathbf{I}, \mathsf{rand}$*. The* maximum branching index *of a partner tuple* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in P_{\mathbf{I},\mathsf{rand}}$ *is the index at which* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ *has the most partners, i.e.,*

$$\mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) = \mathrm{argmax}_{i \in [\ell+1]} \left| \left\{ \overrightarrow{h}' \middle| (\overrightarrow{h}, \overrightarrow{h}') \in \mathrm{prt}_i(\mathbf{I}, \mathsf{rand}) \right\} \right| .$$

In case of ties, we pick the lowest such index.

The maximum branching index naturally defines a partition of any non-empty set of partnered tuples $P_{\mathbf{I},\mathsf{rand}}$, where the i-th set of the partition contains all tuples with maximum branching index i. We define the *maximum branching set* as the largest part of this partition, i.e., the largest subset of tuples that share a common maximum branching index.

Definition 4.2.9 (Maximum Branching Set)**.** *For a pair* $\mathbf{I}, \mathsf{rand}$*, consider the partition of partner tuples according to their maximal branching indices:*

$$B_i(\mathbf{I}, \mathsf{rand}) = \left\{(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \middle| \mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) = i\right\} .$$

The maximum branching set *of* $\mathbf{I}$, rand *is defined as the largest set among them, i.e.,*

$$B_{max}(\mathbf{I}, \mathsf{rand}) = B_{i_{max}(\mathbf{I},\mathsf{rand})}(\mathbf{I}, \mathsf{rand}),$$

where

$$i_{max}(\mathbf{I}, \mathsf{rand}) = \mathrm{argmax}_{i\in[\ell+1]} \left|B_i(\mathbf{I}, \mathsf{rand})\right| .$$

In case of ties, we pick the lowest such index.

Note in particular that $B_{\mathrm{Br}_{\max}(\mathbf{I},\mathsf{rand},\overrightarrow{h})}(\mathbf{I}, \mathsf{rand})$ (henceforth $B_{\mathrm{Br}_{\max}}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ for simplicity) is the set of all tuples $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$ which have the same maximum branching index as $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ (so $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in B_{\mathrm{Br}_{\max}}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$). We define the *heavy row* of the set of successful tuples Succ as

$$HR(\mathsf{Succ}) := \left\{(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in \mathsf{Succ}\middle| |\mathsf{Succ}_{\mathbf{I},\mathsf{rand}}| \geq \frac{\epsilon}{2}\cdot q^{\ell+1}\right\}$$

By Lemma 2.6.3, $|HR(\mathsf{Succ})| \geq \frac{1}{2}\,|\mathsf{Succ}|$.

In the following we define a subset $P \subset HR(\mathsf{Succ})$ of "partner tuples" which have a partner at some index. We also define a "good" subset P_G of P and its "bad" complement P_B. Intuitively, P_G consists of those tuples $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ in P which have many partnering tuples at at least one index, i.e., for which $B_{\mathrm{Br}_{\max}}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ is large (relative to the number of all tuples $(\mathbf{I}, \mathsf{rand}, \cdot)$ in P).

Definition 4.2.10 (Partner Tuples). *We call* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in HR(\mathsf{Succ})$ *a* partner tuple *if* $\overrightarrow{h}$ *has a partner with respect to* $\mathbf{I}$, rand *(at any index* $i \in [\ell + 1]$*), i.e., if* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in P$ *where*

$$P = \left\{(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in HR(\mathsf{Succ})\middle|(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in P_{\mathbf{I},\mathsf{rand}}\right\}.$$

We further define the set of good partner tuples *as*

$$P_G = \left\{(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in P\middle| \left|B_{\mathrm{Br}_{\max}(\mathbf{I},\mathsf{rand},\overrightarrow{h})}\right| \geq \frac{1}{(\ell+1)^3}\left|P\cap P_{\mathbf{I},\mathsf{rand}}\right| \right\}.$$

The set of bad partner tuples *is defined as* $P_B = P \setminus P_G$.

Finally, we introduce the notion of S*-suffixes* (at some index j). For a successful tuple $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in S \subset \mathsf{Succ}_{\mathbf{I},\mathsf{rand}}$ we consider all hash vectors that share a j-prefix (i.e., up to index $j-1$) with $\overrightarrow{h}$ and also lie in S. We define the set $\Gamma_{j,S}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ of its S-suffixes at index j as the set of all the j-th entries h^* of such vectors.

Definition 4.2.11 (S-Suffixes). *Fix* $\mathbf{I}$, rand *and some* $S \subset \mathsf{Succ}_{\mathbf{I},\mathsf{rand}}$. *For a hash vector* $\overrightarrow{h}$ *with* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in S$ *and all* $j \in [\ell+1]$*, we define its set of* S-suffixes *at index* j *as*

$$\Gamma_{j,S}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) := \left\{ h^* \middle| \exists \overrightarrow{h}': \begin{array}{c} (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}') \in S \\ \overrightarrow{h}'_{[j-1]} = \overrightarrow{h}_{[j-1]} \\ h'_j = h^* \end{array} \right\}$$

4.2.3 Counting Partners and Triangles

Having defined our basic objects of interest, we now move to lower bounding their numbers. We start by considering the sizes of sets P and P_G.

The following lemma asserts that if the set $\mathsf{Succ}_{\mathbf{I},\mathsf{rand}}$ is sufficiently large for some fixed $\mathbf{I}, \mathsf{rand}$ (i.e., many different vectors $\overrightarrow{h}$ lead to success together with $\mathbf{I}, \mathsf{rand}$), then the set $P_{\mathbf{I},\mathsf{rand}}$ of tuples $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ that have some partner $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$, is large.

Lemma 4.2.12. *For $\mathbf{I}, \mathsf{rand}$ such that $|\mathsf{Succ}_{\mathbf{I},\mathsf{rand}}| > q^\ell$, there exist at least $|\mathsf{Succ}_{\mathbf{I},\mathsf{rand}}| - q^\ell + 1$ hash vectors $\overrightarrow{h}$ such that $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in P_{\mathbf{I},\mathsf{rand}}$.*

Proof. There are at most q^ℓ possible query transcripts. Thus, by pigeon hole-principle, for $|\mathsf{Succ}_{\mathbf{I},\mathsf{rand}}| > q^\ell$ there can be at most $q^\ell - 1$ hash vectors that do not have a partner. This yields the statement. □

We have proven above that the number of partner tuples with respect to a sufficiently good pair $\mathbf{I}, \mathsf{rand}$ (i.e., one for which many $\overrightarrow{h}$ lead to success) is large. The following simple corollaries combine the above with the properties of $HR(\mathsf{Succ})$ to ensure that the set P of heavy-row partner tuples (i.e., over all pairs $\mathbf{I}, \mathsf{rand} \in HR(\mathsf{Succ})$) is also large.

Corollary 4.2.13. *For $\mathbf{I}, \mathsf{rand}$ such that $\exists \overrightarrow{h} : (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in HR(\mathsf{Succ})$ and $\epsilon \geq \frac{4}{q}$, it holds that $|P_{\mathbf{I},\mathsf{rand}}| \geq \frac{1}{2} |\mathsf{Succ}_{\mathbf{I},\mathsf{rand}}|$.*

Corollary 4.2.14. *For ϵ as in Corollary 4.2.13, $|P| \geq \frac{1}{4} |\mathsf{Succ}|$.*

Proof. By Corollary 4.2.13, $|P_{\mathbf{I},\mathsf{rand}}| \geq \frac{1}{2} |\mathsf{Succ}_{\mathbf{I},\mathsf{rand}}|$ for $\mathbf{I}, \mathsf{rand}$ with $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in HR(\mathsf{Succ})$ for some $\overrightarrow{h}$. Summing over all such $(\mathbf{I}, \mathsf{rand})$ pairs yields that $|P| \geq \frac{1}{2} |HR(\mathsf{Succ})|$. As $|HR(\mathsf{Succ})| \geq \frac{1}{2} |\mathsf{Succ}|$, the statement follows. □

Next, we also show that the subset P_G of good tuples is large within P.

Lemma 4.2.15 (Many partner tuples are good)**.**

$$|P_G| \geq \left(1 - \frac{1}{(\ell+1)^2}\right) |P| .$$

Proof. Fix $\mathbf{I}, \mathsf{rand}$ such that $P_{\mathbf{I},\mathsf{rand}} \cap P \neq \emptyset$. For all $i \in [\ell+1]$, let $B_i = B_i(\mathbf{I}, \mathsf{rand})$ (as in Definition 4.2.9) and $\alpha = \frac{1}{(\ell+1)^2}$. We note here that $P \cap P_{\mathbf{I},\mathsf{rand}} = P_{\mathbf{I},\mathsf{rand}}$ for $\mathbf{I}, \mathsf{rand}$ as above and thus the B_i are a partition of $P \cap P_{\mathbf{I},\mathsf{rand}}$. By Lemma 2.6.4, there exists a subset $G(\mathbf{I}, \mathsf{rand})$ of size at least $\left(1 - \frac{1}{(\ell+1)^2}\right) |P \cap P_{\mathbf{I},\mathsf{rand}}|$, such that all tuples $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in G(\mathbf{I}, \mathsf{rand})$ lie in a set B_i of size at least $|B_i| \geq \frac{1}{(\ell+1)^3} |P \cap P_{\mathbf{I},\mathsf{rand}}|$, where by definition $i = \mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$. By definition of P_G, $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in P_G$. Since this holds for any $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in G(\mathbf{I}, \mathsf{rand})$, we have that $G(\mathbf{I}, \mathsf{rand}) \subset P_G$. Hence,

$$\bigcup_{\substack{\mathbf{I},\mathsf{rand}:\\ P \cap P_{\mathbf{I},\mathsf{rand}} \neq \emptyset}} G(\mathbf{I}, \mathsf{rand}) \subset P_G,$$

and since the sets $G(\mathbf{I}, \textsf{rand}) \subset P_{\mathbf{I},\textsf{rand}}$ are disjoint for distinct $\mathbf{I}, \textsf{rand}$,

$$\begin{aligned}|P_G| &\geq \sum_{\substack{\mathbf{I},\textsf{rand}:\\ P\cap P_{\mathbf{I},\textsf{rand}}\neq\emptyset}} |G(\mathbf{I}, \textsf{rand})| \\ &\geq \sum_{\substack{\mathbf{I},\textsf{rand}:\\ P\cap P_{\mathbf{I},\textsf{rand}}\neq\emptyset}} \left(1 - \frac{1}{(\ell+1)^2}\right) |P \cap P_{\mathbf{I},\textsf{rand}}| \\ &= \left(1 - \frac{1}{(\ell+1)^2}\right) |P| .\end{aligned}$$

□

We now want to argue that for sufficiently large sets of vectors, there must be a sufficiently large set of possible suffixes for many vectors within the set. In particular, this will help us find triangles.

Lemma 4.2.16 (Lower-bounding the amount of possible suffixes)**.** *Fix* $\mathbf{I}, \textsf{rand}$ *and* $\zeta \in [0,1]$. *Let* $H \subset \mathbb{Z}_q^{\ell+1}$ *with* $|H| \geq \zeta \cdot q^{\ell+1}$*, such that for all* $\overrightarrow{h} \in H$*,* $(\mathbf{I}, \textsf{rand}, \overrightarrow{h}) \in S \subset \textsf{Succ}$ *for some set* $S \supset H$*. Then for any constant* $c \in (0,1)$ *the following holds: for each index* $j \in [\ell+1]$*, there exists a subset* $H_j \subset H$ *with* $|H_j| > c \cdot |H|$*, such that for any* $\overrightarrow{h} \in H_j$*,*

$$\left|\Gamma_{j,S}(\mathbf{I}, \textsf{rand}, \overrightarrow{h})\right| \geq (1-c) \cdot \zeta \cdot q$$

Proof. Assume toward a contradiction that for some $c \in (0,1)$ and index $j \in [\ell+1]$, no such $H_j \subset H$ exists. This can be rephrased as: there exists a subset $F \subset H$ such that $|F| > (1-c) \cdot |H|$ and for all $\overrightarrow{h} \in F$, $\left|\Gamma_{j,S}(\mathbf{I}, \textsf{rand}, \overrightarrow{h})\right| < (1-c) \cdot \zeta \cdot q$. For any $\overrightarrow{h} \in F$, consider all successful vectors $\overrightarrow{h}'$ with $\overrightarrow{h}'_{[j-1]} = \overrightarrow{h}_{[j-1]}$. The j-th entry of $\overrightarrow{h}'$ takes $\left|\Gamma_{j,S}(\mathbf{I}, \textsf{rand}, \overrightarrow{h})\right|$ possible values, and all of the remaining $\ell - j + 1$ entries take (up to) q possible values. Therefore,

$$\left|\left\{\overrightarrow{h}' \in \mathbb{Z}_q{}^{\ell+1} \;\middle|\; \begin{matrix}(\mathbf{I}, \textsf{rand}, \overrightarrow{h}') \in \textsf{Succ} \\ \overrightarrow{h}'_{[j-1]} = \overrightarrow{h}_{[j-1]}\end{matrix}\right\}\right| \leq \left|\Gamma_{j,S}(\mathbf{I}, \textsf{rand}, \overrightarrow{h})\right| \cdot q^{\ell-j+1} < (1-c) \cdot \zeta \cdot q^{\ell-j+2}$$

Then we have

$$\begin{aligned}|F| &\leq \sum_{\substack{\overrightarrow{h}_{[j-1]}\\ \text{s.t. } \overrightarrow{h}\in F}} \left|\left\{\overrightarrow{h}' \in \mathbb{Z}_q{}^{\ell+1} \;\middle|\; \begin{matrix}(\mathbf{I}, \textsf{rand}, \overrightarrow{h}') \in \textsf{Succ} \\ \overrightarrow{h}'_{[j-1]} = \overrightarrow{h}_{[j-1]}\end{matrix}\right\}\right| \\ &\leq q^{j-1} \cdot \max_{\overrightarrow{h}\in F} \left|\left\{\overrightarrow{h}' \in \mathbb{Z}_q{}^{\ell+1} \;\middle|\; \begin{matrix}(\mathbf{I}, \textsf{rand}, \overrightarrow{h}') \in \textsf{Succ} \\ \overrightarrow{h}'_{[j-1]} = \overrightarrow{h}_{[j-1]}\end{matrix}\right\}\right| \\ &< (1-c) \cdot \zeta \cdot q^{(\ell+1)} \leq (1-c) \cdot |H| ,\end{aligned}$$

which is a contradiction to the assumption that $|F| > (1-c) \cdot |H|$. □

We apply the lower bound for suffixes from above to lower bound the number of triangle base corners that lie within P_G. We begin by proving the following technical lemma.

Lemma 4.2.17 (Many Good Partner Tuples are Triangle Base Corners)**.** *Assume* $\epsilon \geq \frac{72(\ell+1)^3}{q}$ *and fix* $\mathbf{I}, \mathsf{rand}$ *such that* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in HR(\mathsf{Succ})$ *for some* $\overrightarrow{h}$*. Then at least* $\frac{5}{6}$ *of tuples in* $P_G \cap P_{\mathbf{I},\mathsf{rand}}$ *are triangle base corners at index* $\mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$*. That is, there exists a subset* $T \subset P_G \cap P_{\mathbf{I},\mathsf{rand}}$ *with* $|T| \geq \frac{5}{6}|P_G \cap P_{\mathbf{I},\mathsf{rand}}|$ *such that all tuples* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}') \in T$ *are base corners of a triangle at index* $\mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$*.*

Proof. Take any $\mathbf{I}, \mathsf{rand}$ as in the lemma statement. Then

$$|P \cap P_{\mathbf{I},\mathsf{rand}}| \geq \frac{1}{2}|\mathsf{Succ}_{\mathbf{I},\mathsf{rand}}| \geq \frac{\epsilon}{4} \cdot q^{\ell+1},$$

where the first inequality is due to Corollary 4.2.13, and the second inequality is due to the definition of $HR(\mathsf{Succ})$. (Corollary 4.2.13 requires that $\epsilon \geq \frac{4}{q}$, which is implied by our assumption on ϵ here.)

Consider any index ν for which $B_\nu(\mathbf{I}, \mathsf{rand}) \cap P_G \neq \emptyset$. Then, by definition of P_G it holds that

$$|B_\nu(\mathbf{I}, \mathsf{rand})| \geq \frac{1}{(\ell+1)^3}|P \cap P_{\mathbf{I},\mathsf{rand}}| \geq \frac{\epsilon}{4(\ell+1)^3} \cdot q^{\ell+1}.$$

Applying Lemma 4.2.16 with $H = S = B_\nu(\mathbf{I}, \mathsf{rand})$, $\zeta = \frac{\epsilon}{4(\ell+1)^3}$, and $c = \frac{5}{6}$, we get: for any index $j \in [\ell+1]$, there exists a subset $T_j(\mathbf{I}, \mathsf{rand}) \subset B_\nu(\mathbf{I}, \mathsf{rand})$ with $|T_j(\mathbf{I}, \mathsf{rand})| \geq \frac{5}{6}|B_\nu|$ such that for all $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in T_j(\mathbf{I}, \mathsf{rand})$,

$$\left|\Gamma_{j,B_\nu}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})\right| \geq \left(1 - \frac{5}{6}\right) \cdot \frac{\epsilon}{4(\ell+1)^3} \cdot q \geq 3.$$

The set $T_\nu(\mathbf{I}, \mathsf{rand})$ yields a set of triangle corners at index ν, which can be seen as follows. First, for any tuple $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in T_\nu(\mathbf{I}, \mathsf{rand})$, there is a partner tuple $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$ at index ν (by definition of $B_\nu(\mathbf{I}, \mathsf{rand})$). Hence, $h_j, h'_j \in \Gamma_{j,B_\nu}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) = \Gamma_{j,B_\nu}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$. As $\left|\Gamma_{j,B_\nu}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})\right| \geq 3$, there exists at least one further entry h''_j which lies in $\Gamma_{j,B_\nu}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$. Thus, $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}), (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$, $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}'')$ mutually fork from each other at index ν. Moreover, the first two among them are partners and at least one of them lies in $T_\nu(\mathbf{I}, \mathsf{rand})$. Hence, the three of them satisfy the definition of a triangle at index ν and at least one of the triangle base corners lies in $T_\nu(\mathbf{I}, \mathsf{rand})$. Finally, by definition of the set $B_\nu(\mathbf{I}, \mathsf{rand})$, $\mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) = \nu$ as required.

Now define

$$T := \bigcup_{\nu\colon B_\nu(\mathbf{I},\mathsf{rand}) \cap P_G \neq \emptyset} T_\nu(\mathbf{I}, \mathsf{rand}).$$

Using that the sets $B_\nu(\mathbf{I}, \mathsf{rand})$ s.t. $B_\nu(\mathbf{I}, \mathsf{rand}) \cap P_G$ form a partition of $P_{\mathbf{I},\mathsf{rand}} \cap P_G$

$$\begin{aligned}|T| = \left|\bigcup_{\nu\colon B_\nu(\mathbf{I},\mathsf{rand}) \cap P_G \neq \emptyset} T_\nu\right| &= \sum_{\nu\colon B_\nu(\mathbf{I},\mathsf{rand}) \cap P_G \neq \emptyset} |T_\nu| \\ \geq \sum_{\nu\colon B_\nu(\mathbf{I},\mathsf{rand}) \cap P_G \neq \emptyset} \frac{5}{6}|B_\nu(\mathbf{I}, \mathsf{rand})| &\geq \frac{5}{6}|P_{\mathbf{I},\mathsf{rand}} \cap P_G|,\end{aligned}$$

where the second equality follows from disjointness of the sets T_ν. This yields the statement of the Lemma. □

Corollary 4.2.18. *Assume ϵ as in Lemma 4.2.17. Then at least $\frac{5}{6}$ of all tuples in P_G are triangle base corners.*

Proof. Consider $\mathbf{I}$, rand such that $(\mathbf{I}, \text{rand}, \overrightarrow{h}) \in P_G$ for some $\overrightarrow{h}$. By definition, $P_G \subset HR(\text{Succ})$, so $(\mathbf{I}, \text{rand}, \overrightarrow{h}) \in HR(\text{Succ})$ for some $\overrightarrow{h}$. By Lemma 4.2.17, at least $\frac{5}{6}$ of vectors in $P_G \cap \text{Succ}_{\mathbf{I},\text{rand}}$ are triangle base corners. Summing over all such $(\mathbf{I}, \text{rand})$ pairs yields the result. □

Lemma 4.2.19. *If $(\mathbf{I}, \text{rand}, \overrightarrow{h})$,$(\mathbf{I}, \text{rand}, \overrightarrow{h}')$ are a triangle base at index i, and $(\mathbf{I}, \text{rand}, \overrightarrow{h}'')$ is a partner of $(\mathbf{I}, \text{rand}, \overrightarrow{h})$ at index i, then $(\mathbf{I}, \text{rand}, \overrightarrow{h})$, $(\mathbf{I}, \text{rand}, \overrightarrow{h}'')$ are also a triangle base at index i.*

Proof. We distinguish between two cases:

Case $h'_i = h''_i$: Let $\overrightarrow{h}'''$ be such that $(\mathbf{I}, \text{rand}, \overrightarrow{h})$, $(\mathbf{I}, \text{rand}, \overrightarrow{h}')$, $(\mathbf{I}, \text{rand}, \overrightarrow{h}''')$ form a triangle at index i (such $\overrightarrow{h}'''$ must exist because $(\mathbf{I}, \text{rand}, \overrightarrow{h})$, $(\mathbf{I}, \text{rand}, \overrightarrow{h}')$ form a triangle base at index i). Then $(\mathbf{I}, \text{rand}, \overrightarrow{h})$, $(\mathbf{I}, \text{rand}, \overrightarrow{h}'')$, $(\mathbf{I}, \text{rand}, \overrightarrow{h}''')$ also form a triangle at index i.

Case $h'_i \neq h''_i$: in this case $(\mathbf{I}, \text{rand}, \overrightarrow{h}), (\mathbf{I}, \text{rand}, \overrightarrow{h}''), (\mathbf{I}, \text{rand}, \overrightarrow{h}')$ form a triangle at index i where $(\mathbf{I}, \text{rand}, \overrightarrow{h}')$ takes the role of the triangle top.

□

4.2.4 The Mapping Φ

For any successful tuple $(\mathbf{I}, \text{rand}, \overrightarrow{h})$, we now define the mapping $\Phi_{\text{rand},\overrightarrow{h}}$ and prove its transcript preserving properties in Lemma 4.2.21. We remark that this mapping is not efficiently computable and will merely serve as a technical tool in our analysis.

Definition 4.2.20 (Mapping instances via transcript)**.** *For $(\mathbf{I}, \text{rand}, \overrightarrow{h}) \in \text{Succ}$, we define $\Phi_{\text{rand},\overrightarrow{h}}(\mathbf{I})$ as follows. For a $\mathbf{y}$-side instance $\mathbf{I} = (1, w, \mathbf{y}, \overrightarrow{c}, \overrightarrow{r}, \overrightarrow{u})$, $\Phi_{\text{rand},\overrightarrow{h}}(\mathbf{I})$ is a $\mathbf{z}$-side instance that consists of*

$$b = 0 \qquad x = \operatorname{dlog} \mathbf{y} \qquad \mathbf{z} = \mathbf{g}^w \qquad \forall i \in [\ell]\colon d_i = e_i - c_i$$
$$\forall i \in [\ell]\colon s_i = u_i - d_i \cdot w \qquad \forall i \in [\ell]\colon v_i = c_i \cdot x + r_i$$

For a $\mathbf{z}$-side instance $\mathbf{I} = (0, x, \mathbf{z}, d, s, v)$, $\Phi_{\text{rand},\overrightarrow{h}}(\mathbf{I})$ is a $\mathbf{y}$-side instance that consists of

$$b = 1 \qquad w = \operatorname{dlog} \mathbf{z} \qquad \mathbf{y} = \mathbf{g}^x \qquad \forall i \in [\ell]\colon c_i = e_i - d_i$$
$$\forall i \in [\ell]\colon r_i = v_i - c_i \cdot x \qquad \forall i \in [\ell]\colon u_i = d_i \cdot w + s_i$$

(where $\overrightarrow{e}$ is the query vector produced by rand, $\overrightarrow{h}$ *using instance $\mathbf{I}$). We will sometimes use the notation $\Phi_{\overrightarrow{e}}$ instead of $\Phi_{\text{rand},\overrightarrow{h}}$ for a given $(\mathbf{I}, \text{rand}, \overrightarrow{h})$. We also define $\Phi(\mathbf{I}, \text{rand}, \overrightarrow{h}) = (\Phi_{\text{rand},\overrightarrow{h}}(\mathbf{I}), \text{rand}, \overrightarrow{h})$.*

Lemma 4.2.21 ($\Phi_{\text{rand},\overrightarrow{h}}$ is a bijection that preserves transcripts)**.** *Fix* rand, $\overrightarrow{h}$. *For all tuples $(\mathbf{I}, \text{rand}, \overrightarrow{h}) \in \text{Succ}$, $\Phi_{\text{rand},\overrightarrow{h}}$ is a self-inverse bijection and*

$$\text{tr}(\mathbf{I}, \text{rand}, \overrightarrow{h}) = \text{tr}(\Phi_{\text{rand},\overrightarrow{h}}(\mathbf{I}), \text{rand}, \overrightarrow{h})$$

Proof. We show the lemma for a $\mathbf{z}$-side instance $\mathbf{I} = (1, w, \mathbf{y}, \overrightarrow{c}, \overrightarrow{r}, \overrightarrow{u})$; the argument for $\mathbf{y}$-side instances works analogously.

Let $(0, x, \mathbf{z}, \overrightarrow{d}, \overrightarrow{s}, \overrightarrow{v}) = \Phi_{\mathsf{rand}, \overrightarrow{h}}(\mathbf{I})$ and $\overrightarrow{e}$ be the vector of queries to Sign$_2$ made by the adversary U on input $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$. We first show that $\mathbf{I}$ and $\Phi_{\mathsf{rand}, \overrightarrow{h}}(\mathbf{I})$ produce the same transcript. The bit b is not used actively in the simulation and thus does not affect the transcript. As the public key $\mathbf{y} = \mathbf{g}^x$ is the same in both instances from the view of U, and U is running on the same randomness rand, the info used will be the same for both instances. The tag key $\mathbf{z} = \mathbf{g}^w$ is the same in both instances. We now look at a single session of the protocol. For any i it holds that: The commitments $\mathbf{a}_i, \mathbf{b}_i$ are computed as $\mathbf{a}_i = \mathbf{g}^{v_i} = \mathbf{g}^{r_i + c \cdot x} = \mathbf{g}^{r_i} \cdot \mathbf{y}^{c_i}$ and $\mathbf{b}_i = \mathbf{g}^{s_i} \cdot \mathbf{z}^{d_i} = \mathbf{g}^{u_i - d_i \cdot w} \cdot \mathbf{g}^{d_i \cdot w} = \mathbf{g}^{u_i}$ which are the same group elements for both instances. We now use induction on the signing sessions in the order of the Sign$_2$ requests. Let therefore i_k be the session index of the kth closed session. As the instances provide the same response to Sign$_1$, the view up until the first query to Sign$_2$ is identical for U and thus it makes the same first Sign$_2$ query in both runs. Analogously, if the transcript is identical up to the kth request to Sign$_2$, the kth query e_{i_k} will also be identical. We now argue that for the kth closed session, if the views have been identical before, the kth response to Sign$_2$ is also identical. Thus, U makes the same query e_{i_k} to Sign$_2$. As $d_{i_k} = e_{i_k} - c_{i_k}$, it holds that $c'_{i_k} = e_{i_k} - (e_{i_k} - c_{i_k}) = c_{i_k}$, where c'_{i_k} is the c_{i_k} computed in the run with $\Phi_{\mathsf{rand}, \overrightarrow{h}}(\mathbf{I})$. Thus $r'_{i_k} = v_{i_k} - c_{i_k} \cdot x = (r_{i_k} + c_{i_k} \cdot x) - c_{i_k} \cdot x = r_{i_k}$ where r'_{i_k} is the r_{i_k} computed in the run with $\Phi_{\mathsf{rand}, \overrightarrow{h}}(\mathbf{I})$. Thus, the response $r_{i_k}, c_{i_k}, s_{i_k}, d_{i_k}$ of the oracle Sign$_2$ is identical. As the view as the adversary is identical for the entire run of the protocol, it must also output the same signatures in both runs. Thus, the two transcripts are identical.

We thus use $\overrightarrow{e}$ to denote the queries to Sign$_2$ in both runs. We now show that $\Phi_{\mathsf{rand}, \overrightarrow{h}}$ is a self-inverse bijection. For an instance $\mathbf{I}$, we show that $\Phi_{\mathsf{rand}, \overrightarrow{h}}(\Phi_{\mathsf{rand}, \overrightarrow{h}}(\mathbf{I})) = \mathbf{I}$ (denote with $'$ the components of $\Phi_{\mathsf{rand}, \overrightarrow{h}}(\Phi_{\mathsf{rand}, \overrightarrow{h}}(\mathbf{I}))$):

- $w' = \operatorname{dlog} \mathbf{z} = \operatorname{dlog} \mathbf{g}^w = w$
- $\mathbf{y}' = \mathbf{g}^x = \mathbf{y}$
- $\forall i \in [\ell] \colon c'_i = e_i - d_i = e_i - (e_i - c_i) = c_i$
- $\forall i \in [\ell] \colon r'_i = v_i - c_i \cdot x = (c_i \cdot x + r_i) - c_i \cdot x = r_i$
- $\forall i \in [\ell] \colon u'_i = d_i \cdot w + s_i = (e_i - c_i) \cdot w + [u_i - (e_i - c_i) \cdot w] = u_i$

Thus, $\Phi_{\mathsf{rand}, \overrightarrow{h}}$ is a bijection and its own inverse. □

The lemma above shows that the Abe-Okamoto scheme is *witness indistinguishable*, i.e., a simulator that uses the $\mathbf{z}$-side witness to sign (see Figure 4.3) creates a view identical to the real view to the adversary. In particular, this implies that the wrapper A simulates the ℓ-OMUF game to the adversary M perfectly.

Corollary 4.2.22. $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in \mathsf{Succ} \Leftrightarrow (\Phi_{\mathsf{rand}, \overrightarrow{h}}(\mathbf{I}), \mathsf{rand}, \overrightarrow{h}) \in \mathsf{Succ}$.

We look into the effect of the transcript mapping function on partner tuples. We have proven that $\Phi_{\mathsf{rand}, \overrightarrow{h}}$ preserves the transcript (and hence success) of $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$. However, note that this does not (by itself) imply that partnering tuples $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ and $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$ result in partnering tuples $(\Phi_{\mathsf{rand}, \overrightarrow{h}}(\mathbf{I}), \mathsf{rand}, \overrightarrow{h})$ and $(\Phi_{\mathsf{rand}, \overrightarrow{h}}(\mathbf{I}), \mathsf{rand}, \overrightarrow{h}')$, or $(\Phi_{\mathsf{rand}, \overrightarrow{h}'}(\mathbf{I}), \mathsf{rand}, \overrightarrow{h})$ and $(\Phi_{\mathsf{rand}, \overrightarrow{h}'}(\mathbf{I}), \mathsf{rand}, \overrightarrow{h}')$, respectively. Lemma 4.2.23 asserts that this is indeed the case.

$$
\begin{array}{lcr}
\textbf{Signer} & & \textbf{User} \\
\mathsf{pk} = (\mathbf{y}) & & \mathsf{pk} = \mathbf{y} \\
\mathbf{z} = H^*(\mathsf{info}) = \mathbf{g}^w & & m, \mathsf{info}, \mathbf{z} = H^*(\mathsf{info}) \\
\hline
c, r, v \xleftarrow{\$} \mathbb{Z}_q & & \\
\mathbf{a} := \mathbf{g}^r \mathbf{y}^c & & \\
\mathbf{b} := \mathbf{g}^v & & \\
& \xrightarrow{\mathbf{a},\mathbf{b}} & \\
& & t_1, t_2, t_3, t_4 \xleftarrow{\$} \mathbb{Z}_q \\
& & \boldsymbol{\alpha} := \mathbf{g}^{t_1} \cdot \mathbf{y}^{t_2} \cdot \mathbf{a} \\
& & \boldsymbol{\beta} := \mathbf{g}^{t_3} \cdot \mathbf{y}^{t_4} \cdot \mathbf{b} \\
& & h := H(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{z}, m) \\
& & e := h - t_2 - t_4 \\
& \xleftarrow{e} & \\
d := e - c & & \\
s := v - dw & & \\
& \xrightarrow{c,d,r,s} & \\
& & \rho := r + t_1 \\
& & \omega := c + t_2 \\
& & \sigma := s + t_3 \\
& & \delta := d + t_4 \\
& & \omega + \delta \overset{?}{=} H(\mathbf{g}^\rho \cdot \mathbf{y}^\omega, \mathbf{g}^\sigma \cdot \mathbf{z}^\delta, \mathbf{z}, m) \\
& & \Downarrow \\
& & \mathsf{sig} := (\rho, \omega, \sigma, \delta)
\end{array}
$$

Figure 4.3: How to use the $\mathbf{z}$-side witness to sign in the Abe-Okamoto scheme

Lemma 4.2.23 (Partners stay partners through Φ). *For all* $\mathbf{I}$, rand, *and vectors* $\overrightarrow{h}, \overrightarrow{h}'$,

$$
\begin{aligned}
(\overrightarrow{h}, \overrightarrow{h}') \in \mathrm{prt}_i(\mathbf{I}, \mathsf{rand}) &\Leftrightarrow (\overrightarrow{h}, \overrightarrow{h}') \in \mathrm{prt}_i(\Phi_{\mathsf{rand},\overrightarrow{h}}(\mathbf{I}), \mathsf{rand}) \\
&\Leftrightarrow (\overrightarrow{h}, \overrightarrow{h}') \in \mathrm{prt}_i(\Phi_{\mathsf{rand},\overrightarrow{h}'}(\mathbf{I}), \mathsf{rand})
\end{aligned}
$$

Proof. Suppose $(\overrightarrow{h}, \overrightarrow{h}') \in \mathrm{prt}_i(\mathbf{I}, \mathsf{rand}) \subset F_i(\mathbf{I}, \mathsf{rand})$; we have that $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}), (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}') \in \mathsf{Succ}$. Then by Corollary 4.2.22, $(\Phi_{\mathsf{rand},\overrightarrow{h}}(\mathbf{I}), \mathsf{rand}, \overrightarrow{h})$, $(\Phi_{\mathsf{rand},\overrightarrow{h}'}(\mathbf{I}), \mathsf{rand}, \overrightarrow{h}') \in \mathsf{Succ}$.

Furthermore, as $\overrightarrow{h}, \overrightarrow{h}'$ are partners for $\mathbf{I}$, rand, they produce the same query vector $\overrightarrow{e}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) = \overrightarrow{e}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$. Thus $\Phi_{\mathsf{rand},\overrightarrow{h}}(\mathbf{I}) = \Phi_{\overrightarrow{e}}(\mathbf{I}) = \Phi_{\mathsf{rand},\overrightarrow{h}'}(\mathbf{I})$. Using this fact, we obtain

$$
\begin{aligned}
&\overrightarrow{e}(\Phi_{\mathsf{rand},\overrightarrow{h}}(\mathbf{I}), \mathsf{rand}, \overrightarrow{h}) = \overrightarrow{e}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) = \overrightarrow{e}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}') \\
&= \overrightarrow{e}(\Phi_{\mathsf{rand},\overrightarrow{h}'}(\mathbf{I}), \mathsf{rand}, \overrightarrow{h}') = \overrightarrow{e}(\Phi_{\mathsf{rand},\overrightarrow{h}}(\mathbf{I}), \mathsf{rand}, \overrightarrow{h}')
\end{aligned}
$$

as follows. The first equality follows because $\overrightarrow{e}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ is contained in $\mathsf{tr}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ and by Lemma 4.2.21, we have that $\mathsf{tr}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) = \mathsf{tr}(\Phi_{\mathsf{rand},\overrightarrow{h}}(\mathbf{I}), \mathsf{rand}, \overrightarrow{h})$. The second equality holds because $\overrightarrow{h}$ and $\overrightarrow{h}'$ are partners. The third equality follows because $\overrightarrow{e}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$ is contained in tr$(\mathbf{I}, \text{rand}, \vec{h}\,')$ and from Lemma 4.2.21, we have that tr$(\mathbf{I}, \text{rand}, \vec{h}\,') = $ tr$(\Phi_{\text{rand}, \vec{h}}(\mathbf{I}), \text{rand}, \vec{h}\,')$. The fourth equality holds by another application of Lemma 4.2.21 which yields tr$(\mathbf{I}, \text{rand}, \vec{h}\,') = $ tr$(\Phi_{\text{rand}, \vec{h}\,'}(\mathbf{I}), \text{rand}, \vec{h}\,') = $ tr$(\Phi_{\text{rand}, \vec{h}}(\mathbf{I}), \text{rand}, \vec{h}\,')$.

Combining the two paragraphs above, we get $(\vec{h}, \vec{h}\,') \in \mathrm{prt}_i(\Phi_{\text{rand}, \vec{h}}(\mathbf{I}), \text{rand})$. Using a similar argument, $(\vec{h}, \vec{h}\,') \in \mathrm{prt}_i(\Phi_{\text{rand}, \vec{h}\,'}(\mathbf{I}), \text{rand})$. The inverse direction follows from the self-inverse property of $\Phi_{\text{rand}, \vec{h}}$. □

Corollary 4.2.24. $\mathrm{Br}_{\max}(\mathbf{I}, \text{rand}, \vec{h}) = \mathrm{Br}_{\max}(\Phi_{\text{rand}, \vec{h}}(\mathbf{I}), \text{rand}, \vec{h})$.

4.2.5 Counting the Image of Φ

In the following, we consider the image of the set P_G of all partners under Φ[1]. Recall that (roughly speaking) we defined P_G as the set of all 'good' partner tuples, i.e., tuples with many partner tuples. The goal of the next lemma is to lower bound the number of 'doubly good' tuples in P_G who retain a large number of partners after being mapped with Φ, i.e., good partner tuples whose image under Φ remains 'good'. Below, we implicitly use the fact that $\Phi(P)$ yields a set of partner tuples (due to Lemma 4.2.23).

Lemma 4.2.25 (Many good partner tuples have a good image)**.** *Let*

$$P'_G = \left\{ (\mathbf{I}, \text{rand}, \vec{h}) \in \Phi(P) \middle| \left| B_{\mathrm{Br}_{\max}(\mathbf{I}, \text{rand}, \vec{h})} \cap \Phi(P) \right| \geq \frac{1}{(\ell+1)^3} \left| \Phi(P) \cap \mathsf{Succ}_{\mathbf{I}, \text{rand}} \right| \right\}.$$

Then

$$|\Phi(P_G) \cap P'_G| \geq \left(1 - \frac{2}{(\ell+1)^2}\right) |P|.$$

Proof. Fix $\mathbf{I}$, rand with $\Phi(P) \cap \mathsf{Succ}_{\mathbf{I}, \text{rand}} \neq \emptyset$. Then it holds that each $(\mathbf{I}, \text{rand}, \vec{h}) \in \mathsf{Succ}_{\mathbf{I}, \text{rand}} \cap \Phi(P)$ lies in one set $B_i \cap \Phi(P)$ (namely $i = \mathrm{Br}_{\max}(\mathbf{I}, \text{rand}, \vec{h})$). As there are $(\ell + 1)$ hash queries, there are at most $(\ell + 1)$ such sets. Thus, by Lemma 2.6.4 with $\alpha = \frac{1}{(\ell+1)}$, there exists a set G_α such that for all $(\mathbf{I}, \text{rand}, \vec{h}) \in G_\alpha$ it holds that $\left| B_{\mathrm{Br}_{\max}(\mathbf{I}, \text{rand}, \vec{h})} \cap \Phi(P) \right| \geq \frac{1}{(\ell+1)^3} \left| \Phi(P) \cap \mathsf{Succ}_{\mathbf{I}, \text{rand}} \right|$ and $|G_\alpha| \geq \left(1 - \frac{1}{(\ell+1)}\right) \cdot |\mathsf{Succ}_{\mathbf{I}, \text{rand}}| \cap \Phi(P)$. Taking the G_α of all $\mathsf{Succ}_{\mathbf{I}, \text{rand}}$ with $\mathsf{Succ}_{\mathbf{I}, \text{rand}} \cap \Phi(P) \neq \emptyset$ yields that $|P'_G| \geq \left(1 - \frac{1}{(\ell+1)^2}\right) \cdot |P|$. Since $\Phi(P_G) \subset \Phi(P)$ and $P'_G \subset \Phi(P)$, it holds, using Lemma 4.2.15 and the inclusion-exclusion principle that

$$|\Phi(P_G) \cap P'_G| \geq \left(1 - \frac{2}{(\ell+1)^2}\right) \cdot |P|.$$

□

We now turn to lower bounding the number of triangle base corner within the set P'_G from Lemma 4.2.25. Together with the fact that P_G has many triangle base-corners, we will then be able to conclude that the images of many triangle base-corners remain base-corners in some other triangle at the same index.

[1]We have defined $\Phi(\mathbf{I}, \text{rand}, \vec{h}) = (\Phi_{\text{rand}, \vec{h}}(\mathbf{I}), \text{rand}, \vec{h})$, hence $\Phi(P)$ is well-defined.

Lemma 4.2.26 (Many images of good partner tuples are triangle base corners)**.** *Assume* $\epsilon \geq \frac{3\cdot 144 \cdot \frac{(\ell+1)^2-1}{(\ell+1)^2}}{q}$ *as well as* ϵ *as in Corollary 4.2.13 (whichever is larger). Then at least* $\frac{11}{18}$ *of tuples* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in P'_G$ *are base corners of a triangle at* $\mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$*.*

Proof. Let $\epsilon_{P'_G} = \frac{|P'_G|}{|\mathcal{I}\times\mathcal{R}\times\mathbb{Z}_q{}^{(\ell+1)}|}$ be the probability of getting a tuple in P'_G when sampling a tuple uniformly at random. Then

$$\epsilon_{P'_G} \geq \left(1 - \frac{1}{(\ell+1)^2}\right) \cdot \epsilon_P \geq \left(1 - \frac{1}{(\ell+1)^2}\right) \cdot \frac{\epsilon}{4}, \tag{$*$}$$

where the first inequality is due to the fact that $|P'_G| \geq \left(1 - \frac{1}{(\ell+1)^2}\right)|P|$ (see the proof of Lemma 4.2.25), and the second inequality is due to Corollary 4.2.14.

Define the heavy row of P'_G as

$$HR(P'_G) = \left\{(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in P'_G \middle| |P'_G \cap \mathsf{Succ}_{\mathbf{I},\mathsf{rand}}| \geq \frac{\epsilon_{P'_G}}{3} \cdot q^{(\ell+1)}\right\}.$$

By Lemma 2.6.3, $|HR(P'_G)| \geq \frac{2}{3}|P'_G|$. Now consider any tuple $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in HR(P'_G)$. From the definition of P'_G it follows that

$$\begin{aligned}\left|B_{\mathrm{Br}_{\max}(\mathbf{I},\mathsf{rand},\overrightarrow{h})} \cap P'_G\right| &\geq \frac{1}{(\ell+1)^3}\left|\Phi(P) \cap \mathsf{Succ}_{\mathbf{I},\mathsf{rand}}\right| \\ &\overset{P'_G \subset \Phi(P)}{\geq} \frac{1}{(\ell+1)^3}\left|P'_G \cap \mathsf{Succ}_{\mathbf{I},\mathsf{rand}}\right| \geq \frac{\epsilon_{P'_G}}{3(\ell+1)^3} \cdot q^{(\ell+1)}.\end{aligned}$$

Similar to the proof of Lemma 4.2.17, we apply Lemma 4.2.16 with $H = B_{\mathrm{Br}_{\max}(\mathbf{I},\mathsf{rand},\overrightarrow{h})} \cap P'_G$, $S = S_{\mathrm{Br}_{\max}(\mathbf{I},\mathsf{rand},\overrightarrow{h})} \cap P'_G$, $\zeta = \frac{\epsilon_{P'_G}}{3(\ell+1)^3}$, and $c = \frac{11}{12}$. This yields that for all indices $j \in [\ell+1]$, there exists a subset $H_j(\mathbf{I}, \mathsf{rand}) \subset B_{\mathrm{Br}_{\max}(\mathbf{I},\mathsf{rand},\overrightarrow{h})} \cap P'_G$ with $|H_j(\mathbf{I}, \mathsf{rand})| \geq \frac{11}{12}\left|B_{\mathrm{Br}_{\max}(\mathbf{I},\mathsf{rand},\overrightarrow{h})} \cap P'_G\right|$ such that for all $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}') \in H_j(\mathbf{I}, \mathsf{rand})$,

$$\left|\Gamma_{j,S}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')\right| \geq \left(1 - \frac{11}{12}\right) \cdot \frac{\epsilon_{P'_G}}{3(\ell+1)^3} \cdot q \overset{(*)}{\geq} \left(1 - \frac{1}{(\ell+1)^2}\right) \cdot \frac{\epsilon}{144(\ell+1)^3} \cdot q \geq 3$$

where the last step is obtained by plugging in ϵ as in the lemma statement. Setting $j = \mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$, we obtain a subset $H_{\mathrm{Br}_{\max}(\mathbf{I},\mathsf{rand},\overrightarrow{h})}(\mathbf{I}, \mathsf{rand})$ of triangle base corners at index $\mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ by a similar argument as in Lemma 4.2.17. (Henceforth we simplify it to $H_{\mathrm{Br}_{\max}(\mathbf{I},\mathsf{rand},\overrightarrow{h})}$.) Let T be the union of all such sets, i.e.,

$$T = \bigcup_{(\mathbf{I},\mathsf{rand},\overrightarrow{h}) \in HR(P'_G)} H_{\mathrm{Br}_{\max}(\mathbf{I},\mathsf{rand},\overrightarrow{h})}$$

Then all vectors in T are triangle base corners, and

$$|T| = \left|\bigcup_{(\mathbf{I},\mathsf{rand},\overrightarrow{h}) \in HR(P'_G)} H_{\mathrm{Br}_{\max}(\mathbf{I},\mathsf{rand},\overrightarrow{h})}\right| \geq \frac{11}{12}\left|\bigcup_{(\mathbf{I},\mathsf{rand},\overrightarrow{h})}\left(B_{\mathrm{Br}_{\max}(\mathbf{I},\mathsf{rand},\overrightarrow{h})} \cap P'_G\right)\right|$$

$$\overset{(**)}{\geq} \frac{11}{12}\left|HR(P'_G)\right| \geq \frac{11}{12}\cdot\frac{2}{3}\left|P'_G\right| = \frac{11}{18}\left|P'_G\right|,$$

where $(**)$ is because for any $(\mathbf{I}, \textsf{rand}, \overrightarrow{h}') \in HR(P'_G)$ it holds that

$$(\mathbf{I}, \textsf{rand}, \overrightarrow{h}') \in B_{\mathrm{Br}_{\max}(\mathbf{I}, \textsf{rand}, \overrightarrow{h}')} \cap P'_G \subset \bigcup_{(\mathbf{I}, \textsf{rand}, \overrightarrow{h}) \in HR(P'_G)} \left(B_{\mathrm{Br}_{\max}(\mathbf{I}, \textsf{rand}, \overrightarrow{h})} \cap P'_G\right),$$

hence

$$HR(P'_G) \subset \bigcup_{(\mathbf{I}, \textsf{rand}, \overrightarrow{h}) \in HR(P'_G)} \left(B_{\mathrm{Br}_{\max}(\mathbf{I}, \textsf{rand}, \overrightarrow{h})} \cap P'_G\right).$$

□

Having bounded the number of triangle corners within both P_G and P'_G, we now compute their overlap. More precisely, we show that there is a large set T such that all tuples $(\mathbf{I}, \textsf{rand}, \overrightarrow{h}) \in T$ are triangle base corners and, moreover, $\Phi(\mathbf{I}, \textsf{rand}, \overrightarrow{h})$ is also a triangle base-corner at the same index.

Lemma 4.2.27. *Assume ϵ as in Lemma 4.2.26. Then there exists a set $T \subset P$ with $|T| \geq \frac{1}{12}|P|$ such that for all $(\mathbf{I}, \textsf{rand}, \overrightarrow{h}) \in T$ it holds that both $(\mathbf{I}, \textsf{rand}, \overrightarrow{h})$ and $\Phi(\mathbf{I}, \textsf{rand}, \overrightarrow{h})$ are triangle base-corners at index $\mathrm{Br}_{\max}(\mathbf{I}, \textsf{rand}, \overrightarrow{h})$.*[2]

Proof. Let C denote the set of triangle base corners at their maximal branching index, i.e.,

$$C := \left\{(\mathbf{I}, \textsf{rand}, \overrightarrow{h}) \middle| \exists \overrightarrow{h}', \overrightarrow{h}'' \colon (\overrightarrow{h}, \overrightarrow{h}', \overrightarrow{h}'') \in \triangle_{\mathrm{Br}_{\max}(\mathbf{I}, \textsf{rand}, \overrightarrow{h})}\right\}.$$

Then

$$\left|P'_G \cap C\right| \overset{\text{Lemma 4.2.26}}{\geq} \frac{11}{18}\left|P'_G\right| \overset{\text{Lemma 4.2.25}}{\geq} \frac{11}{18}\cdot\left(1 - \frac{1}{(\ell+1)^2}\right)|P| \overset{\ell \geq 1}{\geq} \frac{11}{24}|P|;$$

$$\left|P_G \cap C\right| \overset{\text{Lemma 4.2.17}}{\geq} \frac{5}{6}\left|P_G\right| \overset{\text{Lemma 4.2.15}}{\geq} \left(1 - \frac{1}{(\ell+1)^2}\right)\cdot\frac{5}{6}|P| \overset{(\ell \geq 1}{\geq} \frac{5}{8}|P|.$$

Let $T = \Phi(P_G \cap C) \cap (P'_G \cap C)$. Clearly $T \subset C \cap \Phi(C)$, implying that the tuples in T satisfy the requirements of the lemma. Moreover, $T \subset \Phi(P_G) \cap P'_G \subset P$. By inclusion-exclusion, this yields

$$\begin{aligned}|T| \geq |\Phi(P_G \cap C)| + |P'_G \cap C| - |P| &= |P_G \cap C| + |P'_G \cap C| - |P| \\ &\geq \frac{5}{8}|P| + \frac{11}{24}|P| - |P| = \frac{1}{12}|P|\end{aligned}$$

□

We now relate the sets T from Lemma 4.2.27 and B_T. Recall that elements of the set T are triangle base-corners $(\mathbf{I}, \textsf{rand}, \overrightarrow{h})$ at $\mathrm{Br}_{\max}(\mathbf{I}, \textsf{rand}, \overrightarrow{h})$ s.t. $(\Phi_{\textsf{rand}, \overrightarrow{h}}(\mathbf{I}), \textsf{rand}, \overrightarrow{h})$ remains a triangle base-corner at index $\mathrm{Br}_{\max}(\mathbf{I}, \textsf{rand}, \overrightarrow{h})$. Concretely, we show that $T \subset B_T$. This establishes, for one, that B_T is large (because T is large, as we have shown).

[2]Note that $\mathrm{Br}_{\max}(\mathbf{I}, \textsf{rand}, \overrightarrow{h}) = \mathrm{Br}_{\max}(\Phi_{\textsf{rand}, \overrightarrow{h}}(\mathbf{I}), \textsf{rand}, \overrightarrow{h})$ due to Corollary 4.2.24.

Lemma 4.2.28. *Let T be as in Lemma 4.2.27. Then $T \subset B_T$.*

Proof. Fix some $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in T$. Then there exist $\overrightarrow{h}', \overrightarrow{h}'', \overrightarrow{h}''', \overrightarrow{h}''''$ such that $(\overrightarrow{h}, \overrightarrow{h}', \overrightarrow{h}''') \in \triangle_{\mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})}(\mathbf{I}, \mathsf{rand})$ and $(\overrightarrow{h}, \overrightarrow{h}'', \overrightarrow{h}'''') \in \triangle_{\mathrm{Br}_{\max}(\Phi_{\mathsf{rand}, \overrightarrow{h}}(\mathbf{I}), \mathsf{rand}, \overrightarrow{h})}(\Phi_{\mathsf{rand}, \overrightarrow{h}}(\mathbf{I}), \mathsf{rand})$. By Corollary 4.2.24, $\mathrm{Br}_{\max}(\Phi_{\mathsf{rand}, \overrightarrow{h}}(\mathbf{I}), \mathsf{rand}, \overrightarrow{h}) = \mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$; let this index be i. In the following, we also use that $(\overrightarrow{h}, \overrightarrow{h}') \in \mathrm{prt}_i(\Phi_{\mathsf{rand}, \overrightarrow{h}}(\mathbf{I}), \mathsf{rand})$ which follows from Lemma 4.2.23.

- If $h'_i = h''_i$, then we can replace $\overrightarrow{h}''$ by $\overrightarrow{h}'$ in the triangle $(\overrightarrow{h}, \overrightarrow{h}'', \overrightarrow{h}'''')$ as $h'_i = h''_i \neq h''''_i$. Since $(\overrightarrow{h}, \overrightarrow{h}') \in \mathrm{prt}_i(\Phi_{\mathsf{rand}, \overrightarrow{h}}(\mathbf{I}), \mathsf{rand})$, $(\overrightarrow{h}, \overrightarrow{h}', \overrightarrow{h}'''') \in \triangle_i(\Phi_{\mathsf{rand}, \overrightarrow{h}}(\mathbf{I}), \mathsf{rand})$ and thus, $\overrightarrow{h}, \overrightarrow{h}', \overrightarrow{h}''$ and $\overrightarrow{h}''''$ meet the definition of B_T. Hence, $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in B_T$.
- If $h'_i \neq h''_i$, then, since $(\overrightarrow{h}, \overrightarrow{h}') \in \mathrm{prt}_i(\Phi_{\mathsf{rand}, \overrightarrow{h}}(\mathbf{I}), \mathsf{rand})$, $(\overrightarrow{h}', \overrightarrow{h}'') \in \mathrm{prt}_i(\Phi_{\mathsf{rand}, \overrightarrow{h}}(\mathbf{I}), \mathsf{rand})$, and $h_i \neq h''_i$, it must also be the case that $(\overrightarrow{h}, \overrightarrow{h}'') \in \mathrm{prt}_i(\Phi_{\mathsf{rand}, \overrightarrow{h}}(\mathbf{I}), \mathsf{rand})$. This implies that $(\overrightarrow{h}, \overrightarrow{h}', \overrightarrow{h}'') \in \triangle_i(\Phi_{\mathsf{rand}, \overrightarrow{h}}(\mathbf{I}), \mathsf{rand})$ and thus $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in B_T$.

Either way, $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in B_T$, so $T \subset B_T$. □

Corollary 4.2.29. $|B_T| \geq \frac{1}{12}|P|$.

4.2.6 Extracting a Witness from a Fork

Witness Extraction. We briefly recall how the reduction can compute a witness from two signatures from forking runs of the wrapper A. We say a signature $(\rho, \omega, \sigma, \delta)$ on a message m in the output of A on input $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ *corresponds to* a hash value h_i, if $\mathsf{H}(\mathbf{g}^\rho \mathbf{y}^\omega, \mathbf{g}^\sigma \mathbf{z}^\delta, \mathbf{z}, m)$ was the i-th hash query made to the random oracle H in this run of A. Informally we say that a witness *can be extracted* from $\mathbf{I}$, rand, and a pair of forking hash vectors $(\overrightarrow{h}, \overrightarrow{h}') \in F_i(\mathbf{I}, \mathsf{rand})$, if it can be efficiently computed from the two signatures corresponding to h_i and h'_i. We make this formal in the following definition.

Definition 4.2.30 (Witness Extraction). *Fix* $\mathbf{I}$, rand *and let* $(\overrightarrow{h}, \overrightarrow{h}') \in F_i(\mathbf{I}, \mathsf{rand})$ *for some* $i \in [\ell + 1]$. *Moreover, denote* $\mathsf{sig}_i, \mathsf{sig}'_i$ *the signatures that correspond to* h_i *and* h'_i*, respectively. Consider the two witness extraction algorithms* $\mathsf{E}_\mathbf{y}, \mathsf{E}_\mathbf{z}$ *as described in Figure 4.4. For* $\times \in \{\mathbf{y}, \mathbf{z}\}$*, we say that the* $\times$*-side witness* can be extracted from $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ and $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$ *at index* i *if* $\mathsf{E}_\times$ *on input* $(\mathsf{sig}_i, \mathsf{sig}'_i)$ *does not return* $\perp$.

Lemma 4.2.31. *Let* $\mathbf{I}, \mathsf{rand}, i, (\overrightarrow{h}, \overrightarrow{h}') \in F_i(\mathbf{I}, \mathsf{rand}), \mathsf{sig}_i, \mathsf{sig}'_i$*, and algorithms* $\mathsf{E}_\mathbf{y}, \mathsf{E}_\mathbf{z}$ *be as in Definition 4.2.30. Then at least one of* $\mathsf{E}_\mathbf{y}$ *and* $\mathsf{E}_\mathbf{z}$ *outputs a correct witness on input the two signatures* $\mathsf{sig}_i = (\rho_i, \omega_i, \sigma_i, \delta_i)$ *and* $\mathsf{sig}'_i = (\rho'_i, \omega'_i, \sigma'_i, \delta'_i)$ *corresponding to* h_i *and* h'_i*. More specifically,* $\mathsf{E}_\mathbf{y}$ *outputs the* $\mathbf{y}$*-side witness if and only if* $\omega_i \neq \omega'_i$*, otherwise* $\mathsf{E}_\mathbf{z}$ *outputs the* $\mathbf{z}$*-side witness.*

Proof. Suppose $\omega_j \neq \omega'_j$. Let A make two runs, one on $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ and one on $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$. As the two runs were identical up to the point when A makes its j-th query to H, this query $\boldsymbol{\alpha}_j, \boldsymbol{\beta}_j, \mathbf{z}_j, m_j$ was also identical (note that rand is fixed and thus A is deterministic). Since $(\overrightarrow{h}, \overrightarrow{h}') \in F_j(\mathbf{I}, \mathsf{rand})$, we know that $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}), (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}') \in \mathsf{Succ}$, i.e., A outputs $\ell + 1$ valid signatures in both runs. This means that the two sigma protocol transcripts $(\boldsymbol{\alpha}_j, \omega_j, \rho_j)$ and $(\boldsymbol{\alpha}_j, \omega'_j, \rho'_j)$ are both accepting, so we

```
E_y((ρ_i, ω_i, σ_i, δ_i), (ρ'_i, ω'_i, σ'_i, δ'_i))
    if $(\omega_i \neq \omega'_i)$
        return $x := \frac{\rho_i - \rho'_i}{\omega'_i - \omega_i}$
    else
        return $\perp$
```

```
E_z((ρ_i, ω_i, σ_i, δ_i), (ρ'_i, ω'_i, σ'_i, δ'_i))
    if $(\delta_i \neq \delta'_i)$
        return $w := \frac{\sigma_i - \sigma'_i}{\delta'_i - \delta_i}$
    else
        return $\perp$
```

Figure 4.4: The two witness extraction algorithms from Definition 4.2.30

have $\boldsymbol{\alpha}_j = \mathbf{g}^{\rho_j} \cdot \mathbf{g}^{\omega_j \cdot x} = \mathbf{g}^{\rho'_j} \cdot \mathbf{g}^{\omega'_j \cdot x}$ Thus, x can be computed as $x = (\omega'_j - \omega_j)^{-1} \cdot (\rho_j - \rho'_j)$. Now suppose that $\omega_j = \omega'_j$. In this case, since $\omega_j + \delta_j = h_j \neq h'_j = \omega'_j + \delta'_j$, $\delta_j \neq \delta'_j$. For $\delta_j \neq \delta'_j$ we have $\boldsymbol{\beta}_j = \mathbf{g}^{\sigma_j} \cdot \mathbf{g}^{\delta_j \cdot w} = \mathbf{g}^{\sigma'_j} \cdot \mathbf{g}^{\delta'_j \cdot w}$ and thus $w = (\delta'_j - \delta_j)^{-1} \cdot (\sigma_j - \sigma'_j)$. □

Remark 4.2.32. *We note that the witness may be contained in the instance* $\mathbf{I}$*, in which case the witness can be trivially extracted. For the purposes of the lemma we only consider the more interesting case that the witness can be computed from the two signatures directly, regardless of which witness was used for simulating the signing oracles.*

Witnesses in triangles. We now show that if a witness can be extracted from the base of a triangle, it can also be extracted from at least one of the sides. This was previously shown in [AO00].

Corollary 4.2.33. *Fix* $\mathbf{I}, \mathsf{rand}$ *and let* $(\vec{h}, \vec{h}', \vec{h}'') \in \triangle_i(\mathbf{I}, \mathsf{rand})$ *for some* $i \in [\ell+1]$*. Moreover, suppose that the* $\mathbf{y}$*-side witness can be extracted from the base* $(\mathbf{I}, \mathsf{rand}, \vec{h}), (\mathbf{I}, \mathsf{rand}, \vec{h}')$ *of the triangle at index* i*. Then the* $\mathbf{y}$*-side witness can also be extracted from at least one of the sides* $(\mathbf{I}, \mathsf{rand}, \vec{h}), (\mathbf{I}, \mathsf{rand}, \vec{h}'')$ *or* $(\mathbf{I}, \mathsf{rand}, \vec{h}'), (\mathbf{I}, \mathsf{rand}, \vec{h}'')$ *at index* i*. An analogous statement holds for the* $\mathbf{z}$*-side witness.*

Proof. Toward a contradiction, suppose that the $\mathbf{y}$-side witness can be extracted from the base $(\mathbf{I}, \mathsf{rand}, \vec{h}), (\mathbf{I}, \mathsf{rand}, \vec{h}')$ at index i, but can not be extracted at index i for either of the sides $(\mathbf{I}, \mathsf{rand}, \vec{h}), (\mathbf{I}, \mathsf{rand}, \vec{h}'')$ or $(\mathbf{I}, \mathsf{rand}, \vec{h}'), (\mathbf{I}, \mathsf{rand}, \vec{h}'')$. Then, by Lemma 4.2.31, $\omega_i = \omega''_i$ and $\omega'_i = \omega''_i$, so $\omega_i = \omega'_i$. By Lemma 4.2.31 again, the $\mathbf{y}$-side witness can not be extracted from $(\mathbf{I}, \mathsf{rand}, \vec{h}), (\mathbf{I}, \mathsf{rand}, \vec{h}')$, a contradiction. An analogous argument can be made for the $\mathbf{z}$-side. □

We now define *both-sided triangle base corners* as triangle base corners $(\mathbf{I}, \mathsf{rand}, \vec{h})$ which remain base corners of some triangle at their maximal branching index when mapped via $\Phi_{\mathsf{rand}, \vec{h}}$. (Recall that by Corollary 4.2.24, the maximal branching index is preserved under Φ.) On top of this, if $(\mathbf{I}, \mathsf{rand}, \vec{h})$ is a both-sided triangle base corner, and forms a triangle base with $(\mathbf{I}, \mathsf{rand}, \vec{h}')$ at index $\mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \vec{h})$, then $(\Phi_{\mathsf{rand}, \vec{h}}(\mathbf{I}), \mathsf{rand}, \vec{h})$ and $(\Phi_{\mathsf{rand}, \vec{h}}, \mathsf{rand}, \vec{h}')$ also form a triangle base.

For every such tuple $(\mathbf{I}, \mathsf{rand}, \vec{h})$, we further define the set $D_i^{\mathbf{y}}(\mathbf{I}, \mathsf{rand}, \vec{h})$ of tuples that form a both-sided triangle base with $(\mathbf{I}, \mathsf{rand}, \vec{h})$ at index i from which the $\mathbf{y}$-side witness can be extracted, and an analogous set $D_i^{\mathbf{z}}(\mathbf{I}, \mathsf{rand}, \vec{h})$ for the $\mathbf{z}$-side witness. This allows us to then define sets $B_T^{\mathbf{y}}$ and $B_T^{\mathbf{z}}$ that contain tuples where the majority of both-sided triangle bases incident to the tuple allow for extraction of the $\mathbf{y}$-side or $\mathbf{z}$-side witness, respectively.

Definition 4.2.34 (Both-sided Triangle Base Corners). *We call elements of the set*

$$B_T := \left\{ (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \;\middle|\; \begin{array}{c} \exists \overrightarrow{h}', \\ \overrightarrow{h}'', \overrightarrow{h}''' \end{array} : \begin{array}{c} (\overrightarrow{h}, \overrightarrow{h}', \overrightarrow{h}'') \in \triangle_{\mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})}(\mathbf{I}, \mathsf{rand}) \\ (\overrightarrow{h}, \overrightarrow{h}', \overrightarrow{h}''') \in \triangle_{\mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})}(\Phi_{\mathsf{rand}, \overrightarrow{h}}(\mathbf{I}), \mathsf{rand}) \end{array} \right\}$$

both-sided triangle base corners. *For any index $i \in [\ell + 1]$, we define sets*

$$D_i^{\mathbf{y}}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) := \left\{ (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}') \;\middle|\; \begin{array}{c} \exists \overrightarrow{h}'', \\ \overrightarrow{h}''' \end{array} : \begin{array}{c} (\overrightarrow{h}, \overrightarrow{h}', \overrightarrow{h}'') \in \triangle_i(\mathbf{I}, \mathsf{rand}) \\ (\overrightarrow{h}, \overrightarrow{h}', \overrightarrow{h}''') \in \triangle_i(\Phi_{\mathsf{rand}, \overrightarrow{h}}(\mathbf{I}), \mathsf{rand}) \\ \text{The } \mathbf{y}\text{-side witness can be} \\ \text{extracted from } (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}), \\ (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}') \text{ at index } i \end{array} \right\}$$

and $B_T^{\mathbf{y}} \subset B_T$ as

$$B_T^{\mathbf{y}} := \left\{ (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \;\middle|\; \begin{array}{l} D^{\mathbf{y}}_{\mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \neq \emptyset \\ \left| D^{\mathbf{y}}_{\mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \right| \\ \geq \left| D^{\mathbf{z}}_{\mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \right| \end{array} \right\}$$

We define sets $D_i^{\mathbf{z}}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ and $B_T^{\mathbf{z}}$ analogously.

Lemma 4.2.35 (Both-sided triangle bases produce the same witness on both sides). *It holds that*

1. *$\Phi(B_T^{\mathbf{y}}) = B_T^{\mathbf{y}}$ and $\Phi(B_T^{\mathbf{z}}) = B_T^{\mathbf{z}}$;*
2. *$B_T^{\mathbf{y}} \cup B_T^{\mathbf{z}} = B_T$.*

Proof. 1. We show the equation for $B_T^{\mathbf{y}}$; the one for $B_T^{\mathbf{z}}$ can be proved similarly. By definition of B_T and the self-inverse property of Φ, it follows that $\Phi(B_T) = B_T$ and thus $\Phi(B_T^{\mathbf{y}}) \subset B_T$. Fix any $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in B_T^{\mathbf{y}}$. Further, fix a vector $\overrightarrow{h}'$ with $(\overrightarrow{h}, \overrightarrow{h}') \in \mathrm{prt}_{\mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})}(\mathbf{I}, \mathsf{rand})$ for which there exist $\overrightarrow{h}'', \overrightarrow{h}'''$ with $(\overrightarrow{h}, \overrightarrow{h}', \overrightarrow{h}'') \in \triangle_{\mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})}(\mathbf{I}, \mathsf{rand})$, $(\overrightarrow{h}, \overrightarrow{h}', \overrightarrow{h}''') \in \triangle_{\mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})}(\Phi_{\mathsf{rand}, \overrightarrow{h}}(\mathbf{I}), \mathsf{rand})$ and such that the $\mathbf{y}$-side witness can be extracted from $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ and $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$ at $\mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ (i.e., as guaranteed the definition of $B_T^{\mathbf{y}}$).

By Lemma 4.2.21, the signatures resulting from the tuple $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ and $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$ are the same as the signatures resulting from $(\Phi_{\mathsf{rand}, \overrightarrow{h}}(\mathbf{I}), \mathsf{rand}, \overrightarrow{h})$ and $(\Phi_{\mathsf{rand}, \overrightarrow{h}'}(\mathbf{I}), \mathsf{rand}, \overrightarrow{h}')$, respectively. As $\overrightarrow{h}$ and $\overrightarrow{h}'$ are partners, Lemma 4.2.23 implies that $(\Phi_{\mathsf{rand}, \overrightarrow{h}'}(\mathbf{I}), \mathsf{rand}, \overrightarrow{h}') = (\Phi_{\mathsf{rand}, \overrightarrow{h}}(\mathbf{I}), \mathsf{rand}, \overrightarrow{h}')$ and by Corollary 4.2.24, $\mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) = \mathrm{Br}_{\max}(\Phi_{\mathsf{rand}, \overrightarrow{h}}(\mathbf{I}), \mathsf{rand}, \overrightarrow{h})$. Hence, the signatures that result from $(\Phi_{\mathsf{rand}, \overrightarrow{h}}(\mathbf{I}), \mathsf{rand}, \overrightarrow{h}), (\Phi_{\mathsf{rand}, \overrightarrow{h}}(\mathbf{I}), \mathsf{rand}, \overrightarrow{h}')$ are the same signatures that result from $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ and $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$. As the witness that can be extracted from $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ and $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$ at index $\mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ is completely determined by the signatures corresponding to $h_{\mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})}$ and $h'_{\mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})}$, the same witness can be extracted from $(\Phi_{\mathsf{rand}, \overrightarrow{h}}(\mathbf{I}), \mathsf{rand}, \overrightarrow{h}), (\Phi_{\mathsf{rand}, \overrightarrow{h}}(\mathbf{I}), \mathsf{rand}, \overrightarrow{h}')$ at index i. By assumption, $(\overrightarrow{h}, \overrightarrow{h}', \overrightarrow{h}''') \in \triangle_{\mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})}(\Phi_{\mathsf{rand}, \overrightarrow{h}}(\mathbf{I}), \mathsf{rand})$ and $(\overrightarrow{h}, \overrightarrow{h}', \overrightarrow{h}'') \in$

$\triangle_{\mathrm{Br}_{\max}(\mathbf{I},\mathsf{rand},\vec{h})}(\mathbf{I},\mathsf{rand}) = \triangle_{\mathrm{Br}_{\max}(\mathbf{I},\mathsf{rand},\vec{h})}(\Phi_{\mathsf{rand},\vec{h}}(\Phi_{\mathsf{rand},\vec{h}}(\mathbf{I})),\mathsf{rand})$, where we have applied the self-inverse property of $\Phi_{\mathsf{rand},\vec{h}}$. So $(\Phi_{\mathsf{rand},\vec{h}}(\mathbf{I}),\mathsf{rand},\vec{h})$ meets the requirements of the definition of $B_T^{\mathbf{y}}$, and thus $(\Phi_{\mathsf{rand},\vec{h}}(\mathbf{I}),\mathsf{rand},\vec{h}) \in B_T^{\mathbf{y}}$. As $(\mathbf{I},\mathsf{rand},\vec{h}) \in B_T^{\mathbf{y}}$ was chosen arbitrarily, we obtain that $\Phi(B_T^{\mathbf{y}}) \subset B_T^{\mathbf{y}}$. Using the self-inverse property of the bijection Φ once more, we immediately obtain the converse inclusion $B_T^{\mathbf{y}} \subset \Phi(B_T^{\mathbf{y}})$. Thus $\Phi(B_T^{\mathbf{y}}) = B_T^{\mathbf{y}}$.

2. Consider any tuple $(\mathbf{I},\mathsf{rand},\vec{h}) \in B_T$. By Lemma 4.2.31, at least one witness can be extracted from each triangle base.Let $i = \mathrm{Br}_{\max}(\mathbf{I},\mathsf{rand},\vec{h})$. For any $\vec{h}'$ as in Definition 4.2.34, we know that $(\vec{h},\vec{h}') \in F_i(\mathbf{I},\mathsf{rand})$. By Lemma 4.2.31, at least one witness can be extracted from $(\mathbf{I},\mathsf{rand},\vec{h})$ and $(\mathbf{I},\mathsf{rand},\vec{h}')$, so at least one of $D_i^{\mathbf{y}}(\mathbf{I},\mathsf{rand},\vec{h}), D_i^{\mathbf{z}}(\mathbf{I},\mathsf{rand},\vec{h})$ is not $\emptyset$. Suppose $D_i^{\times}(\mathbf{I},\mathsf{rand},\vec{h})$ is the larger of the two sets; then $D_i^{\times}(\mathbf{I},\mathsf{rand},\vec{h}) \neq \emptyset$ and thus $(\mathbf{I},\mathsf{rand},\vec{h}) \in B_T^{\times}$. This shows that any tuple in B_T is in $B_T^{\mathbf{y}}$ or $B_T^{\mathbf{z}}$, so $B_T^{\mathbf{y}} \cup B_T^{\mathbf{z}} = B_T$. □

We define $B_T^{\times}$ as the larger set of $B_T^{\mathbf{y}}$ and $B_T^{\mathbf{z}}$. By the second item of Lemma 4.2.35, $|B_T^{\times}| \geq \frac{1}{2}|B_T|$.

Let $B_{T,\mathbf{y}}^{\times}$ (resp. $B_{T,\mathbf{z}}^{\times}$) be the subset of $B_T^{\times}$ with $\mathbf{y}$-side instances (resp. $\mathbf{z}$-side instances). We stress that $B_T^{\mathbf{y}}$ and $B_{T,\mathbf{y}}^{\times}$ are two different sets: $(\mathbf{I},\mathsf{rand},\vec{h}) \in B_T^{\mathbf{y}}$ means that more both-sided triangle bases (with $(\mathbf{I},\mathsf{rand},\vec{h})$ as one of its corners) allow for extracting the $\mathbf{y}$-side witness than the $\mathbf{z}$-side witness; whereas $(\mathbf{I},\mathsf{rand},\vec{h}) \in B_{T,\mathbf{y}}^{\times}$ means that $(\mathbf{I},\mathsf{rand},\vec{h}) \in B_T^{\times}$ and $\mathbf{I}$ is a $\mathbf{y}$-side witness.

Lemma 4.2.36. $\left|B_{T,\mathbf{y}}^{\times}\right| = \left|B_{T,\mathbf{z}}^{\times}\right| = \frac{1}{2}\left|B_T^{\times}\right|$.

Proof. By the first item of Lemma 4.2.35, Φ is a bijection within $B_T^{\times}$, and since Φ maps a tuple with a $\mathbf{y}$-side instance to a tuple with a $\mathbf{z}$-side instance (and vice versa), we know that Φ is a bijection between $B_{T,\mathbf{y}}^{\times}$ and $B_{T,\mathbf{z}}^{\times}$; therefore, $\left|B_{T,\mathbf{y}}^{\times}\right| = \left|B_{T,\mathbf{z}}^{\times}\right|$. Since $B_{T,\mathbf{y}}^{\times}$ and $B_{T,\mathbf{z}}^{\times}$ form a partition of $B_T^{\times}$, we know that $\left|B_{T,\mathbf{y}}^{\times}\right| + \left|B_{T,\mathbf{z}}^{\times}\right| = \left|B_T^{\times}\right|$, and the lemma follows. □

We now give a lower bound of the size of $B_T^{\times}$. Let $\epsilon_{B_T^{\times}}$ be the probability of getting a tuple in $B_T^{\times}$ while sampling uniformly at random, i.e.,

$$\epsilon_{B_T^{\times}} := \frac{\left|B_T^{\times}\right|}{\left|\mathcal{I} \times \mathcal{R} \times \mathbb{Z}_q{}^{\ell+1}\right|}.$$

Lemma 4.2.37 (Lower-bounding the size of $B_T^{\times}$)**.** *Assume* $\epsilon \geq \frac{432\left(1-\frac{1}{(\ell+1)^2}\right)}{q}$. *Then*

$$\epsilon_{B_T^{\times}} \geq \frac{\epsilon}{96}.$$

Proof.

$$\left|B_T^{\times}\right| \geq \frac{1}{2}|B_T| \geq \frac{1}{2}\cdot\frac{1}{12}|P| \geq \frac{1}{2}\cdot\frac{1}{12}\cdot\frac{1}{4}|\mathsf{Succ}| = \frac{\epsilon}{96}\left|\mathcal{I} \times \mathcal{R} \times \mathbb{Z}_q{}^{\ell+1}\right|,$$

where the steps follow (in this order) from Lemma 4.2.35, Corollary 4.2.29, Corollary 4.2.14, and the definition of ϵ. Thus, $\epsilon_{B_T^{\times}} = \frac{\left|B_T^{\times}\right|}{\left|\mathcal{I}\times\mathcal{R}\times\mathbb{Z}_q{}^{\ell+1}\right|} \geq \frac{\epsilon}{96}$. □

Finding triangle tops. In order for our security proof to go through, a key step is to compute the probability that the reduction hits a triangle side from which the $\times$-side witness can be extracted when forking the wrapper, independently of the witness that is being used by the reduction. This event is crucial in our proof because, assuming that the reduction samples one of these sides, it is likely that it did so with the witness opposite of $\times$, meaning that it extracts the witness $\times$ it *does not already know* with significant probability, hence solving the discrete logarithm problem. In order to lower bound the probability of the event above, we first define *relevant triangle tops* for a both-sided triangle base corner $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in B_T^\times$. These are all the tuples $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}'')$ such that $(\overrightarrow{h}, \overrightarrow{h}', \overrightarrow{h}'')$ forms triangles at index i (where $\overrightarrow{h}'$ is as in the definition of both-sided triangle tops (Definition 4.2.34)).

Definition 4.2.38 (Relevant triangle tops). *For a tuple* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$*, define its* relevant triangle tops *at index* i *as tuples in the following set:*

$$T_{T,i}^\times(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) := \left\{ (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}'') \middle| \exists \overrightarrow{h}': \begin{array}{c} (\overrightarrow{h}, \overrightarrow{h}', \overrightarrow{h}'') \in \triangle_i(\mathbf{I}, \mathsf{rand}) \\ \text{The } \times \text{-side witness} \\ \text{can be extracted from } (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}), \\ (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}') \text{ at } i \end{array} \right\}$$

We will mostly consider relevant triangle tops at the maximum branching index $\mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ and we thus define $T_T^\times(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) := T^\times_{T,\mathrm{Br}_{\max}(\mathbf{I},\mathsf{rand},\overrightarrow{h})}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$.

What remains to be shown is that many elements of $B_T^\times$ actually have many relevant triangle tops, regardless of whether they reside in $B_{T,\mathbf{y}}^\times$ or $B_{T,\mathbf{z}}^\times$, i.e., independently of the witness that they store. This ensures that when the reduction samples and then (partially) resamples the vectors during the forking process, it will hit a side from which the desired witness can be extracted with significant probability, as explained above.

Lemma 4.2.39 (There are enough relevant triangle tops). *There exists a subset* $G_\mathbf{y} \subset B_{T,\mathbf{y}}^\times$ *with* $|G_\mathbf{y}| \geq \frac{3}{8}\left|B_{T,\mathbf{y}}^\times\right|$ *such that for each* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in G_\mathbf{y}$,

$$\left|T_T^\times(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})\right| \geq \frac{\epsilon_{B_T^\times}}{16(\ell+1)} \cdot q^{\ell - \mathrm{Br}_{\max}(\mathbf{I},\mathsf{rand},\overrightarrow{h})+2} - 2q^{\ell-\mathrm{Br}_{\max}(\mathbf{I},\mathsf{rand},\overrightarrow{h})+1}.$$

An analogous statement holds for $B_{T,\mathbf{z}}^\times$.

Proof. We show this for $G_\mathbf{y}$; the proof for $G_\mathbf{z}$ works analogously. By Lemma 4.2.36 it holds that $\left|B_{T,\mathbf{y}}^\times\right| = \frac{1}{2}\left|B_T^\times\right|$.

For $i \in [\ell+1]$ we define a subset of $B_{T,\mathbf{y}}^\times$ that are both-sided triangle base corners at index i as follows:

$$G_{i,\mathbf{y}} := \left\{ (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in B_{T,\mathbf{y}}^\times \middle| \; i = \mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \right\}$$

It is easy to see that $G_{i,\mathbf{y}}$ $(i \in [\ell+1])$ form a partition of $B_{T,\mathbf{y}}^\times$. We note that membership in $G_{i,\mathbf{y}}$ is symmetrical, i.e., the other base corner of a both-sided triangle is always also contained in $G_{i,\mathbf{y}}$.

Denote the set of indices $i \in [\ell+1]$ such that $|G_{i,\mathbf{y}}| \geq \frac{1}{4(\ell+1)}\left|B_{T,\mathbf{y}}^\times\right|$ as $\mathcal{J}$. We now apply Lemma 2.6.4 with $B_i = G_{i,\mathbf{y}}$, $b = \ell+1$, $\alpha = \frac{1}{4}$, and $X = B_{T,\mathbf{y}}^\times$. Due to Lemma 2.6.4, at least $\frac{3}{4}$ of the tuples

$(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ in $B^{\times}_{T,\mathbf{y}}$ has the property that there exists $i \in \mathcal{J}$ such that $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in G_{i,\mathbf{y}}$. For each $G_{i,\mathbf{y}}$ with $i \in \mathcal{J}$, define

$$HR(G_{i,\mathbf{y}}) = \left\{ (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in G_{i,\mathbf{y}} \middle| \left| G_{i,\mathbf{y}} \cap \mathsf{Succ}_{\mathbf{I},\mathsf{rand},\overrightarrow{h}_{[i-1]}} \right| \geq \frac{1}{8} \cdot \frac{1}{\ell+1} \epsilon_{B^{\times}_{T,\mathbf{y}}} \cdot q^{\ell-i+2} \right\}$$

where $\epsilon_{B^{\times}_{T,\mathbf{y}}} := \frac{|B^{\times}_{T,\mathbf{y}}|}{|\mathcal{I} \times \mathcal{R} \times \mathbb{Z}_q{}^{\ell+1}|} = \frac{1}{2}\epsilon_{B^{\times}_T}$. Then, by Lemma 2.6.3, $|HR(G_{i,\mathbf{y}})| \geq \frac{1}{2}|G_{i,\mathbf{y}}|$ for each $G_{i,\mathbf{y}}$ with $i \in \mathcal{J}$. Now, fix some arbitrary $i \in \mathcal{J}$ and $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in HR(G_{i,\mathbf{y}}) \subset G_{i,\mathbf{y}}$. Furthermore, fix a partner $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}') \in D^{\times}_i(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})'$. Then, there exist at most $q^{\ell+1-i}$ vectors in $G_{i,\mathbf{y}} \cap \mathsf{Succ}_{\mathbf{I},\mathsf{rand},\overrightarrow{h}_{[i-1]}}$ that share the first i entries with $\overrightarrow{h}$ and at most $q^{\ell+1-i}$ vectors in $G_{i,\mathbf{y}} \cap \mathsf{Succ}_{\mathbf{I},\mathsf{rand},\overrightarrow{h}_{[i-1]}}$ that share the first i entries with its designated partner $\overrightarrow{h}'$. These vectors do not form triangles at index i with $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ and $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$. We denote this set of non-triangle-tops by $N(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}, \overrightarrow{h}')$ and by the above reasoning, $\left|N(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}, \overrightarrow{h}')\right| \leq 2 \cdot q^{\ell+1-i}$. We note that $\mathsf{Succ}_{\mathbf{I},\mathsf{rand},\overrightarrow{h}_{[i-1]}} \setminus N(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}, \overrightarrow{h}') \subset T^{\times}_{T,i}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$. Thus, the amount of triangle tops for $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ is at least

$$\begin{aligned} \left|T^{\times}_{T,i}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})\right| &\geq \left|\mathsf{Succ}_{\mathbf{I},\mathsf{rand},\overrightarrow{h}_{[i-1]}} \setminus N(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}, \overrightarrow{h}')\right| \\ &\geq \left|\mathsf{Succ}_{\mathbf{I},\mathsf{rand},\overrightarrow{h}_{[i-1]}}\right| - \left|N(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}, \overrightarrow{h}')\right| \\ &\geq \frac{1}{8} \cdot \frac{1}{\ell+1}\epsilon_{B^{\times}_{T,\mathbf{y}}} \cdot q^{\ell+1-i+1} - 2 \cdot q^{\ell+1-i} \\ &\geq \frac{1}{16} \cdot \frac{1}{\ell+1}\epsilon_{B^{\times}_T} \cdot q^{\ell+1-i+1} - 2 \cdot q^{\ell+1-i} \end{aligned}$$

Since $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in HR(G_{i,\mathbf{y}})$ was arbitrarily chosen, taking the union over all $HR(G_{i,\mathbf{y}})$ s.t. $i \in \mathcal{J}$ yields the statement.

□

Corollary 4.2.40. *Let $G_{\mathbf{y}}$ be as in Lemma 4.2.39. Then*

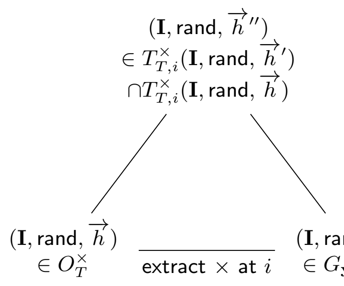
$$\Pr_{\substack{(\mathbf{I},\mathsf{rand},\overrightarrow{h}) \xleftarrow{\$} \mathcal{I}\times\mathcal{R}\times\mathbb{Z}_q{}^{\ell+1} \\ i \xleftarrow{\$} [\ell+1],\, \overrightarrow{h}' \xleftarrow{\$} \mathbb{Z}_q{}^{\ell+1}_{|\overrightarrow{h}_{[i-1]}}}} \left[(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}') \in T^{\times}_T(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \;\middle|\; \begin{array}{c} (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in G_{\mathbf{y}} \\ \mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) = i \end{array} \right]$$
$$\geq \frac{\epsilon_{B^{\times}_T}}{16(\ell+1)} - \frac{2}{q}.$$

An analogous statement holds for $G_{\mathbf{z}}$.

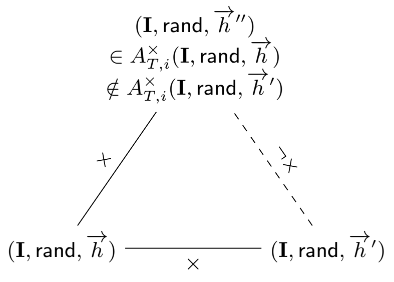
Proof. Suppose $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in G_{\mathbf{y}}$ and $\mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) = i$. Note that $\left|\mathbb{Z}_q{}^{\ell+1}_{|\overrightarrow{h}_{[i-1]}}\right| = q^{\ell-i+2}$. Therefore, the probability of sampling an $\overrightarrow{h}'$ such that $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}') \in T^{\times}_T(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ is

$$\frac{\left|T^{\times}_T(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})\right|}{q^{\ell-i+2}} \geq \frac{\frac{\epsilon_{B^{\times}_T}}{16(\ell+1)} \cdot q^{\ell-i+2} - 2q^{\ell-i+1}}{q^{\ell-i+2}} = \frac{\epsilon_{B^{\times}_T}}{16(\ell+1)} - \frac{2}{q}.$$

□


$(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}'')$
$\in T^{\times}_{T,i}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$
$\cap T^{\times}_{T,i}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$
$(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$
$\in O^{\times}_{T}$
extract $\times$ at i
$(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$
$\in G_{\mathbf{y}} \cup G_{\mathbf{z}}$

(a) Definition of $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in O^{\times}_{T}$ where $i = \mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$.


$(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}'')$
$\in A^{\times}_{T,i}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$
$\notin A^{\times}_{T,i}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$
$\times$
$\neg\times$
$(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$
$\times$
$(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$

(b) A useful triangle top for a base corner is one where the $\times$-side witness can be extracted. The $\times$ (resp. $\neg\times$) on an edge means that the $\times$-side witness can (resp. cannot) be extracted at index i.

Figure 4.5: Opposing base corners and useful triangle tops

Opposing Base Corners

By Corollary 4.2.33 we know that each triangle with a relevant base has at least one relevant side. We now want to consider the probability of finding such a relevant side in the forking proof.

To this end, we consider *opposing base corners* — corners of relevant bases whose partners are in $G_{\mathbf{y}}$ or $G_{\mathbf{z}}$. See Figure 4.5a for a graphic illustration. (Keep in mind that the sets $G_{\mathbf{y}}$ and $G_{\mathbf{z}}$ are the sets of both sided triangle base corners for which there exist many triangle tops.)

Definition 4.2.41 (Opposing base corners)**.**

$$O^{\times}_{T} := \left\{ (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \;\middle|\; \exists \overrightarrow{h}': \begin{array}{c} (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}') \in G_{\mathbf{y}} \cup G_{\mathbf{z}} \\ (\overrightarrow{h}, \overrightarrow{h}') \in \mathrm{prt}_{\mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')}(\mathbf{I}, \mathsf{rand}) \\ \text{the } \times \text{-side witness can be} \\ \text{extracted from } (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}), \\ (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}') \text{ at } \mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}') \end{array} \right\}$$

Good Corners with Useful Tops

For each tuple $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ in $O^{\times}_{T}$ or $B^{\times}_{T}$ we define *useful triangle tops* — triangle tops that allow for extraction of the $\times$-side witness when combined with the base corner $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ (see Figure 4.5b for a graphic illustration):

Definition 4.2.42 (Useful triangle tops)**.** *For any* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in O^{\times}_{T} \cup B^{\times}_{T}$*, define*

$$A^{\times}_{T,i}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) := \left\{ \begin{array}{c} (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}'') \\ \in T^{\times}_{T,i}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \end{array} \;\middle|\; \begin{array}{c} \text{the } \times \text{-side witness can be} \\ \text{extracted from } (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}), \\ (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}'') \text{ at index } i \end{array} \right\}$$

Recall that relevant base corners — those in $G_{\mathbf{y}}$ or $G_{\mathbf{z}}$ — are tuples in $B^{\times}_{T}$ for which many triangle tops are relevant (i.e., the corresponding $T^{\times}_{T}$ set is large). We now consider a subset of these relevant base corners for which a lot of the relevant triangle tops are useful (i.e., the corresponding $A_T^\times$ set is large). We call these base corners *good*.

Definition 4.2.43 (Good base corners). *We say that a base corner in $G_\mathbf{y} \cup G_\mathbf{z}$ is* good *if it lies within the following set:*

$$\widehat{B_T^\times} := \left\{ (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in G_\mathbf{y} \cup G_\mathbf{z} \;\middle|\; \begin{array}{c} \left|A_T^\times(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})\right| \\ \geq \frac{1}{2}\left|T_T^\times(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})\right| \\ -q^{\ell - \mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})+1} \end{array} \right\}$$

We now want to show that if the set of good base corners is small, then there exist a lot of opposing base corners — which we call *good opposing base corners* — that fulfill a property analogous to good base corners.

Definition 4.2.44 (Good opposing base corners).

$$\widehat{O_T^\times} := \left\{ (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \;\middle|\; \exists \overrightarrow{h}': \begin{array}{c} (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}') \in G_\mathbf{y} \cup G_\mathbf{z} \\ (\overrightarrow{h}, \overrightarrow{h}') \in \mathrm{prt}_{\mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')}(\mathbf{I}, \mathsf{rand}) \\ \text{the } \times \text{-side witness can be} \\ \text{extracted from } (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}), \\ (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}') \text{ at } \mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}') \\ \left|A^\times_{T, \mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})\right| \\ \geq \frac{1}{2}\left|T^\times_{T, \mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})\right| \\ -q^{\ell - \mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')+1} \end{array} \right\}$$

Let $\widehat{B_{T,\mathbf{y}}^\times} \subset \widehat{B_T^\times}$ and $\widehat{O_{T,\mathbf{y}}^\times} \subset \widehat{O_T^\times}$ be analogous to $B_{T,\mathbf{y}}^\times \subset B_T^\times$, i.e., the subset of tuples with $\mathbf{y}$-side instances. We define $\widehat{B_{T,\mathbf{z}}^\times}$ and $\widehat{O_{T,\mathbf{z}}^\times}$ similarly.

Lemma 4.2.45. *If $\left|\widehat{B_{T,\mathbf{y}}^\times}\right| < \frac{1}{2}|G_\mathbf{y}|$, then $\left|\widehat{O_{T,\mathbf{y}}^\times}\right| \geq \frac{1}{8(\ell+1)}|G_\mathbf{y}|$. An analogous statement holds for $\mathbf{z}$.*

Proof. Let $F = G_\mathbf{y} \setminus \widehat{B_{T,\mathbf{y}}^\times}$ (so $|F| \geq \frac{1}{2}|G_\mathbf{y}|$). Consider any $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}') \in F$, and let $i = \mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$. Then

$$\left|A_{T,i}^\times(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')\right| < \frac{1}{2}\left|T_{T,i}^\times(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')\right| - q^{\ell-i+1}.$$

By Corollary 4.2.33, for any $(\overrightarrow{h}, \overrightarrow{h}', \overrightarrow{h}'') \in \triangle_i(\mathbf{I}, \mathsf{rand})$ such that the $\times$-side witness can be extracted from the base $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}), (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$, if the $\times$-side witness cannot be extracted from $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}'), (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}'')$, then it can be extracted from $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}), (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}'')$. (All extractions mentioned above are at index i.) Therefore,

$$\left|A_{T,i}^\times(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})\right| + \left|A_{T,i}^\times(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')\right| \geq \left|T_{T,i}^\times(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')\right|.$$

We note that all but $q^{\ell-i+1}$ elements of $T^{\times}_{T,i}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ are also elements of $T^{\times}_{T,i}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$. This is because $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}^*) \in T^{\times}_{T,i}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \setminus T^{\times}_{T,i}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$ implies that $(\overrightarrow{h}, \overrightarrow{h}^*) \in F_i(\mathbf{I}, \mathsf{rand})$ but $(\overrightarrow{h}', \overrightarrow{h}^*) \notin F_i(\mathbf{I}, \mathsf{rand})$, which means that $\overrightarrow{h}^*$ must share its first i entries with $\overrightarrow{h}'$ (recall that $\overrightarrow{h}$ and $\overrightarrow{h}'$ share the first $i-1$ entries), so there are at most $q^{\ell-i+1}$ such vectors. We get that

$$\left|T^{\times}_{T,i}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')\right| \geq \left|T^{\times}_{T,i}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})\right| - q^{\ell-i+1}.$$

Combining all inequalities above, we get

$$\begin{aligned}
\left|A^{\times}_{T,i}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})\right| &\geq \left|T^{\times}_{T,i}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')\right| - \left|A^{\times}_{T,i}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')\right| \\
&> \left|T^{\times}_{T,i}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')\right| - \left(\frac{1}{2}\left|T^{\times}_{T,i}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')\right| - q^{\ell-i+1}\right) \\
&= \frac{1}{2}\left|T^{\times}_{T,i}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')\right| + q^{\ell-i+1} \\
&\geq \frac{1}{2}\left(\left|T^{\times}_{T,i}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})\right| - q^{\ell-i+1}\right) + q^{\ell-i+1} \\
&> \frac{1}{2}\left|T^{\times}_{T,i}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})\right| - q^{\ell-i+1}
\end{aligned}$$

I.e., if $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}') \in F$, then all of its partners $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ at index i with which it forms triangle bases from which the $\times$-side witness can be extracted, are in $\widehat{O^{\times}_{T,\mathbf{y}}}$.

We now lower-bound the number of such partners $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$. Define the set of tuples that yield the same query transcript with $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$ as

$$E(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}') = \{(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}^\star) | \overrightarrow{e}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}^\star) = \overrightarrow{e}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')\}.$$

Note that $E(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$ is the set of partners of $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ at any index. Consider a subset $E_i(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$ of all tuples that fork from $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$ at index i, i.e.,

$$E_i(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}') = \left\{(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}^\star) | (\overrightarrow{h}^\star, \overrightarrow{h}') \in \mathrm{prt}_i(\mathbf{I}, \mathsf{rand})\right\}.$$

Recall that $i = \mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$. By the definition of maximum branching index, we have

$$\left|E_i(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')\right| \geq \frac{1}{\ell+1}\left(\left|E(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')\right| - 1\right) \geq \frac{1}{2(\ell+1)}\left|E(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')\right|$$

(where the -1 comes from excluding $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$ itself). As $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}') \in B^{\times}_{T}$, it holds that at least half of the tuples in $E_i(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$, together with $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$, allow for the extraction of the $\times$-side witness. This means that at least half of the tuples in $E_i(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$ are in $\widehat{O^{\times}_{T,\mathbf{y}}}$.

We have shown that for any $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}') \in F$, at least $\frac{1}{4(\ell+1)}$ of tuples in $E(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$ are in $\widehat{O^{\times}_{T,\mathbf{y}}}$. Further note that for any $(\mathbf{I}_1, \mathsf{rand}_1, \overrightarrow{h}_1)$ and $(\mathbf{I}_2, \mathsf{rand}_2, \overrightarrow{h}_2)$, either $E(\mathbf{I}_1, \mathsf{rand}_1, \overrightarrow{h}_1) =$

$E(\mathbf{I}_2, \mathsf{rand}_2, \overrightarrow{h}_2)$ or $E(\mathbf{I}_1, \mathsf{rand}_1, \overrightarrow{h}_1) \cap E(\mathbf{I}_2, \mathsf{rand}_2, \overrightarrow{h}_2) = \emptyset$. [3] Summing over all $E(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$ for some $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}') \in F$, we get

$$\begin{aligned}
|O_T^\times| &\geq \frac{1}{4(\ell+1)} \sum_{\substack{E \text{ s.t. } E=E(\mathbf{I},\mathsf{rand},\overrightarrow{h}') \\ \text{for some } (\mathbf{I},\mathsf{rand},\overrightarrow{h}')\in F}} |E| \geq \frac{1}{4(\ell+1)} \sum_{\substack{E \text{ s.t. } E=E(\mathbf{I},\mathsf{rand},\overrightarrow{h}') \\ \text{for some } (\mathbf{I},\mathsf{rand},\overrightarrow{h}')\in F}} |E \cap F| \\
&= \frac{1}{4(\ell+1)} \left| \bigcup_{\substack{E \text{ s.t. } E=E(\mathbf{I},\mathsf{rand},\overrightarrow{h}') \\ \text{for some } (\mathbf{I},\mathsf{rand},\overrightarrow{h}')\in F}} (E \cap F) \right| \\
&= \frac{1}{4(\ell+1)} |F| \geq \frac{1}{8(\ell+1)} |G_{\mathbf{y}}| .
\end{aligned}$$

□

Remark 4.2.46. *We point out that it is at this point that we need to require the adversary to make exactly $\ell+1$ hash queries (and thus lose a $\binom{Q_h}{\ell+1}$ factor in advantage). The proof of Lemma 4.2.45 would not go through with $Q_h > \ell+1$ hash queries, as hash vectors in this case may fork at arbitrary indices that do not have a corresponding signature. Therefore, not every tuple in an E-set would also be a partner of every other tuple in the same E-set (with the definition of partners adapted to this setting, i.e., two tuples can only be partners if they both have a signature at their forking index).*

In the following, we want to avoid the case distinction of whether triangle corners come from the B-sets or the O-sets. We therefore define *good triangle corners*:

Definition 4.2.47. *Let $\widehat{G_{\mathbf{y}}}$ be the larger set of $\widehat{B_{T,\mathbf{y}}^\times}$ and $\widehat{O_{T,\mathbf{y}}^\times}$. Furthermore, for a tuple $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in \widehat{G_{\mathbf{y}}}$, let $t(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ be an index at which many relevant triangle tops exist, i.e.,*

$$t(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) = \begin{cases} \mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) & (\textit{if } \widehat{G_{\mathbf{y}}} = \widehat{B_{T,\mathbf{y}}^\times}) \\ \mathrm{Br}_{\max}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}') & (\textit{if } \widehat{G_{\mathbf{y}}} = \widehat{O_{T,\mathbf{y}}^\times}) \end{cases}$$

(where $\overrightarrow{h}'$ is as in the definition of $\widehat{O_{T,\mathbf{y}}^\times}$). If multiple such $\overrightarrow{h}'$ (and thus multiple choices for t) exist, choose one that results in the smallest value of t. Define set $\widehat{G_{\mathbf{z}}}$ analogously, and for a tuple $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in \widehat{G_{\mathbf{z}}}$, define $t(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ analogously.

It is easy to see that for a good opposing base corner, the number of triangle tops is the same as for the corresponding tuple from $G_{\mathbf{y}} \cup G_{\mathbf{z}}$. We state this as a lemma.

Lemma 4.2.48.

$$\Pr_{\substack{b \xleftarrow{\$} \{0,1\} \\ (\mathbf{I},\mathsf{rand},\overrightarrow{h}) \xleftarrow{\$} \mathcal{I}_b \times \mathcal{R} \times \mathbb{Z}_q^{\ell+1} \\ i \xleftarrow{\$} [\ell+1], \overrightarrow{h}' \xleftarrow{\$} \mathbb{Z}_q{}^{\ell+1}_{|\overrightarrow{h}_{[i-1]}}}} \left[\overrightarrow{h}' \in T_{T,i}^\times(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \;\middle|\; \begin{matrix} (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in \widehat{G_{\mathbf{y}}} \\ t(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) = i \end{matrix} \right] \geq \frac{\epsilon_{B_T^\times}}{16(\ell+1)} - \frac{2}{q}$$

[3]This is because $E(\mathbf{I}_1, \mathsf{rand}_1, \overrightarrow{h}_1) \cap E(\mathbf{I}_2, \mathsf{rand}_2, \overrightarrow{h}_2) \neq \emptyset$ implies that $\mathbf{I}_1 = \mathbf{I}_2$, $\mathsf{rand}_1 = \mathsf{rand}_2$, and $\overrightarrow{e}(\mathbf{I}_1, \mathsf{rand}_1, \overrightarrow{h}_1) = \overrightarrow{e}(\mathbf{I}_2, \mathsf{rand}_2, \overrightarrow{h}_2)$, which in turn implies that $E(\mathbf{I}_1, \mathsf{rand}_1, \overrightarrow{h}_1) = E(\mathbf{I}_2, \mathsf{rand}_2, \overrightarrow{h}_2)$.

An analogous statement holds for $\widehat{G_{\mathbf{z}}}$*.*

Proof. If $\widehat{G_{\mathbf{y}}} = \widehat{B^{\times}_{T,\mathbf{y}}}$, then the lower bound is implied by Corollary 4.2.40. If $\widehat{G_{\mathbf{y}}} = \widehat{O^{\times}_{T,\mathbf{y}}}$, setting the partner from the proof of Lemma 4.2.39 to the triangle corner from $\widehat{O^{\times}_{T,\mathbf{y}}}$ yields this lower bound. $\square$

We furthermore note the following regarding the probability of sampling a tuple in $\widehat{G_{\mathbf{y}}}$ and $\widehat{G_{\mathbf{z}}}$:

Lemma 4.2.49.

$$\Pr_{\substack{b \xleftarrow{\$} \{0,1\} \\ (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \xleftarrow{\$} \mathcal{I}_b \times \mathcal{R} \times {\mathbb{Z}_q}^{\ell+1} \\ i \xleftarrow{\$} [\ell+1],\, \overrightarrow{h}' \xleftarrow{\$} {\mathbb{Z}_q}^{\ell+1}_{|\overrightarrow{h}_{[i-1]}}}} \Pr\left[(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in \widehat{G_{\mathbf{y}}}\right] \geq \frac{3}{128(\ell+1)} \epsilon_{B^{\times}_{T}}$$

The same holds for $\widehat{G_{\mathbf{z}}}$*.*

Proof. We prove the lemma for $\widehat{G_{\mathbf{y}}}$; the argument for $\widehat{G_{\mathbf{z}}}$ is analogous. By Lemma 4.2.45, either $\left|\widehat{B^{\times}_{T,\mathbf{y}}}\right| \geq \frac{1}{2} |G_{\mathbf{y}}|$ or $\left|\widehat{O^{\times}_{T,\mathbf{y}}}\right| \geq \frac{1}{8(\ell+1)} |G_{\mathbf{y}}|$, so $\left|\widehat{G_{\mathbf{y}}}\right| = \max\left\{\left|\widehat{B^{\times}_{T,\mathbf{y}}}\right|, \left|\widehat{O^{\times}_{T,\mathbf{y}}}\right|\right\} \geq \frac{1}{8(\ell+1)} |G_{\mathbf{y}}|$. By Lemma 4.2.39, $|G_{\mathbf{y}}| \geq \frac{3}{8} \left|B^{\times}_{T,\mathbf{y}}\right|$; by Lemma 4.2.36, $\left|B^{\times}_{T,\mathbf{y}}\right| = \frac{1}{2} \left|B^{\times}_{T}\right|$. Combining these three inequalities yields

$$\left|\widehat{G_{\mathbf{y}}}\right| \geq \frac{3}{128(\ell+1)} \left|B^{\times}_{T}\right|,$$

and the lemma follows. $\square$

We will use the sets $\widehat{G_{\mathbf{y}}}$ and $\widehat{G_{\mathbf{z}}}$ for simplicity in the forking proof to avoid case distinctions over whether $\widehat{B^{\times}_{T,\mathbf{y}}}$ or $\widehat{O^{\times}_{T,\mathbf{y}}}$ (or $\widehat{B^{\times}_{T,\mathbf{z}}}$ or $\widehat{O^{\times}_{T,\mathbf{z}}}$) are larger.

4.3 Forking Proof for Concurrent OMUF

In this section, we show that the Abe-Okamoto partially blind signature scheme AO is single-tag one-more unforgeable. We extend the proof to multiple tags in Section 4.4.

Theorem 4.3.1 (OMUF security for single-tag adversaries)**.** *For all* $\ell \in \mathbb{N}$*, if there exists an adversary* U *that makes* Q_h *hash queries to random oracle* H *and* $(t_{\mathsf{U}}, \epsilon_{\mathsf{U}}, \ell)$*-breaks* 1-info-$\mathbf{OMUF}_{\mathsf{AO}}$ *with* $\epsilon_{\mathsf{U}} \geq \frac{432\left(1-\frac{1}{(\ell+1)^2}\right)}{q} \cdot \binom{Q_h}{\ell+1}$*, then there exists an algorithm* B *that* $(t_{\mathsf{B}}, \epsilon_{\mathsf{B}})$*-breaks* $\mathbf{DLOG}$ *with*

$$t_{\mathsf{B}} = 2t_{\mathsf{U}} + \mathrm{O}({Q_h}^2)$$

and

$$\epsilon_{\mathsf{B}} \approx \frac{3\epsilon_{\mathsf{U}}^2}{75423744 \cdot \binom{Q_h}{\ell+1}^2 \cdot (\ell+1)^3}.$$

Proof. We use the wrapper A as described in Figure 4.2. We now construct a reduction B that plays the $\mathbf{DLOG}$ game as follows.

After B receives its discrete logarithm challenge $\mathbf{U}$, it samples a bit $b \xleftarrow{\$} \{0,1\}$. It then samples an instance $\mathbf{I}$ of type b where it sets $\mathbf{z} := \mathbf{U}$ if $b = 0$ and $\mathbf{y} := \mathbf{U}$ if $b = 1$, and all other items uniformly at random from $\mathbb{Z}_q$. Furthermore, B samples a random tape rand for A and a random hash vector $\overrightarrow{h}$. After that, B runs A on $(\mathbf{I}, \text{rand}, \overrightarrow{h})$. If A returns a set of $\ell+1$ valid message-signature pairs, B chooses a random index $i \xleftarrow{\$} [\ell+1]$. B then re-samples the vector $\overrightarrow{h}' \xleftarrow{\$} \mathbb{Z}_q{}^{\ell+1}_{|\overrightarrow{h}[i-1]}$ and runs A on $(\mathbf{I}, \text{rand}, \overrightarrow{h}')$. If A outputs a second set of $\ell+1$ valid message-signature pairs, B identifies the signature matching the hash value h_i and h'_i respectively in both pair (it aborts if there exists no such signature for h'_i). Denote the corresponding signature components to the ith hash query by $\rho_i, \rho'_i, \omega_i, \omega'_i, \sigma_i, \sigma'_i, \delta_i, \delta'_i$ (see Figure 4.1).

If $\omega_i \neq \omega'_i$ and $b = 1$, B computes

$$x := (\omega_i - \omega'_i)^{-1} \cdot (\rho'_i - \rho_i)$$

as its output; if $\delta_i \neq \delta'_i$ and $b = 0$, B computes

$$w := (\delta_i - \delta'_i)^{-1} \cdot (\sigma'_i - \sigma_i)$$

as its output. Otherwise B aborts. (If A fails to return a set of $\ell+1$ valid message-signature pairs either time, B also aborts.)

B runs A twice, and performs $\Theta(\ell)$ additional computation (in particular, B verifies up to $2(\ell+1)$ signatures). Plugging in $t_{\mathsf{A}} = t_{\mathsf{U}} + \mathrm{O}({Q_h}^2)$, we get that

$$t_{\mathsf{B}} = 2t_{\mathsf{U}} + \mathrm{O}({Q_h}^2).$$

We now analyze the advantage of reduction B. Let ϵ_{U} be the advantage of U in the OMUF game, and ϵ be the probability that A outputs $\ell+1$ valid message-signature pairs. By Lemma 4.2.1 and subsequent analysis in Section 4.2.1,

$$\epsilon \geq \frac{\epsilon_{\mathsf{U}}}{\binom{Q_h}{\ell+1}}.$$

We can see that B internally runs the witness extracting algorithm $\mathsf{E}_{\mathbf{y}}$ or $\mathsf{E}_{\mathbf{z}}$ in Definition 4.2.30. Therefore, by Lemma 4.2.31, we have that

$$\begin{aligned}
\mathrm{Adv}_{\mathsf{B}}^{\mathbf{DLOG}} &= \Pr_{\substack{b \xleftarrow{\$} \{0,1\} \\ (\mathbf{I}, \text{rand}, \overrightarrow{h}) \xleftarrow{\$} \mathcal{I}_b \times \mathcal{R} \times {\mathbb{Z}_q}^{[\ell+1]} \\ i \xleftarrow{\$} [\ell+1], \overrightarrow{h}' \xleftarrow{\$} \mathbb{Z}_q{}^{[\ell+1]}_{|\overrightarrow{h}[i-1]}}} \left[\begin{array}{c} (\mathbf{I}, \text{rand}, \overrightarrow{h}) \in \mathsf{Succ} \wedge (\mathbf{I}, \text{rand}, \overrightarrow{h}') \in \mathsf{Succ} \\ (b = 0 \wedge \delta_i \neq \delta'_i) \vee (b = 1 \wedge \omega_i \neq \omega'_i) \end{array} \right] \\
&\geq \Pr \left[\begin{array}{c} (\mathbf{I}, \text{rand}, \overrightarrow{h}) \in \widehat{G_{\mathbf{y}}} \cup \widehat{G_{\mathbf{z}}} \\ (\mathbf{I}, \text{rand}, \overrightarrow{h}') \in T^{\times}_{\overrightarrow{T}, i}(\mathbf{I}, \text{rand}, \overrightarrow{h}) \\ t(\mathbf{I}, \text{rand}, \overrightarrow{h}) = i \\ \big((b = 0 \wedge \delta_i \neq \delta'_i) \\ \vee (b = 1 \wedge \omega_i \neq \omega'_i)\big) \end{array} \right] \geq \Pr \left[\begin{array}{c} (\mathbf{I}, \text{rand}, \overrightarrow{h}) \in \widehat{G_{\mathbf{y}}} \cup \widehat{G_{\mathbf{z}}} \\ (\mathbf{I}, \text{rand}, \overrightarrow{h}') \in T^{\times}_{\overrightarrow{T}, i}(\mathbf{I}, \text{rand}, \overrightarrow{h}) \\ t(\mathbf{I}, \text{rand}, \overrightarrow{h}) = i \\ \big((b = 0 \wedge \times = \mathbf{z} \wedge \delta_i \neq \delta'_i) \\ \vee (b = 1 \wedge \times = \mathbf{y} \wedge \omega_i \neq \omega'_i)\big) \end{array} \right] \\
&= \Pr \left[\begin{array}{c} (b = 0 \wedge \times = \mathbf{z} \wedge \delta_i \neq \delta'_i) \\ \vee (b = 1 \wedge \times = \mathbf{y} \wedge \omega_i \neq \omega'_i) \end{array} \middle| \begin{array}{c} (\mathbf{I}, \text{rand}, \overrightarrow{h}) \in \widehat{G_{\mathbf{y}}} \cup \widehat{G_{\mathbf{z}}} \\ (\mathbf{I}, \text{rand}, \overrightarrow{h}') \in T^{\times}_{\overrightarrow{T}, i}(\mathbf{I}, \text{rand}, \overrightarrow{h}) \\ t(\mathbf{I}, \text{rand}, \overrightarrow{h}) = i \end{array} \right]
\end{aligned}$$

$$\cdot \Pr\left[\begin{array}{c}(\mathbf{I},\mathsf{rand},\overrightarrow{h}) \in \widehat{G_{\mathbf{y}}} \cup \widehat{G_{\mathbf{z}}} \\ (\mathbf{I},\mathsf{rand},\overrightarrow{h}') \in T^{\times}_{T,i}(\mathbf{I},\mathsf{rand},\overrightarrow{h}) \\ t(\mathbf{I},\mathsf{rand},\overrightarrow{h}) = i\end{array}\right]$$

We now lower-bound the first term, where we abbreviate the event $(\mathbf{I},\mathsf{rand},\overrightarrow{h}) \in \widehat{G_{\mathbf{y}}} \cup \widehat{G_{\mathbf{z}}} \wedge (\mathbf{I},\mathsf{rand},\overrightarrow{h}') \in T^{\times}_{T,i}(\mathbf{I},\mathsf{rand},\overrightarrow{h}) \wedge t(\mathbf{I},\mathsf{rand},\overrightarrow{h}) = i$ as $\mathbf{E}(\mathbf{I},\mathsf{rand},\overrightarrow{h})$:

$$\begin{aligned}
&\Pr_{\substack{b \xleftarrow{\$} \{0,1\} \\ (\mathbf{I},\mathsf{rand},\overrightarrow{h}) \xleftarrow{\$} \mathcal{I}_b \times \mathcal{R} \times \mathbb{Z}_q^{[\ell+1]} \\ i \xleftarrow{\$} [\ell+1],\, \overrightarrow{h}' \xleftarrow{\$} \mathbb{Z}_q{}^{[\ell+1]}_{|\overrightarrow{h}_{[i-1]}}}} \left[\begin{array}{c}(b=0 \wedge \times = \mathbf{z} \wedge \delta_i \neq \delta_i') \\ \vee(b=1 \wedge \times = \mathbf{y} \wedge \omega_i \neq \omega_i')\end{array} \;\middle|\; \mathbf{E}(\mathbf{I},\mathsf{rand},\overrightarrow{h})\right] \\
=&\Pr\left[\begin{array}{c} b=1 \wedge \times = \mathbf{y} \\ \omega_i \neq \omega_i'\end{array} \;\middle|\; \mathbf{E}(\mathbf{I},\mathsf{rand},\overrightarrow{h})\right] + \Pr\left[\begin{array}{c} b=0 \wedge \times = \mathbf{z} \\ \delta_i \neq \delta_i'\end{array} \;\middle|\; \mathbf{E}(\mathbf{I},\mathsf{rand},\overrightarrow{h})\right] \\
=&\Pr[b=1]\cdot\Pr\left[\times = \mathbf{y} \wedge \omega_i \neq \omega_i' \;\middle|\; b=1 \wedge \mathbf{E}(\mathbf{I},\mathsf{rand},\overrightarrow{h})\right] \\
&+\Pr[b=0]\cdot\Pr\left[\times = \mathbf{z} \wedge \delta_i \neq \delta_i' \;\middle|\; b=0 \wedge \mathbf{E}(\mathbf{I},\mathsf{rand},\overrightarrow{h})\right] \\
=&\frac{1}{2}\Bigg(\Pr\left[\times = \mathbf{y} \wedge \omega_i \neq \omega_i' \;\middle|\; b=1 \wedge \mathbf{E}(\mathbf{I},\mathsf{rand},\overrightarrow{h})\right] \\
&+\Pr\left[\times = \mathbf{z} \wedge \delta_i \neq \delta_i' \;\middle|\; b=0 \wedge \mathbf{E}(\mathbf{I},\mathsf{rand},\overrightarrow{h})\right]\Bigg) \\
=&\frac{1}{2}\Bigg(\Pr[\times = \mathbf{y}]\cdot\Pr\left[\omega_i \neq \omega_i' \;\middle|\; b=1 \wedge \times = \mathbf{y} \wedge \mathbf{E}(\mathbf{I},\mathsf{rand},\overrightarrow{h})\right] \\
&+\Pr[\times = \mathbf{z}]\cdot\Pr\left[\delta_i \neq \delta_i' \;\middle|\; b=0 \wedge \times = \mathbf{z} \wedge \mathbf{E}(\mathbf{I},\mathsf{rand},\overrightarrow{h})\right]\Bigg) \\
=&\frac{1}{2}\Bigg(\Pr[\times = \mathbf{y}]\cdot\Pr\left[(\mathbf{I},\mathsf{rand},\overrightarrow{h}') \in A^{\times}_{T,i}(\mathbf{I},\mathsf{rand},\overrightarrow{h}) \;\middle|\; \begin{array}{c} b=1 \wedge \times = \mathbf{y} \\ \wedge\mathbf{E}(\mathbf{I},\mathsf{rand},\overrightarrow{h})\end{array}\right] \\
&+\Pr[\times = \mathbf{z}]\cdot\Pr\left[(\mathbf{I},\mathsf{rand},\overrightarrow{h}') \in A^{\times}_{T,i}(\mathbf{I},\mathsf{rand},\overrightarrow{h}) \;\middle|\; \begin{array}{c} b=0 \wedge \times = \mathbf{z} \\ \wedge\mathbf{E}(\mathbf{I},\mathsf{rand},\overrightarrow{h})\end{array}\right]\Bigg) \\
\geq&\frac{1}{2}\left(\Pr[\times = \mathbf{y}]\cdot\left(\frac{1}{2}-\frac{1}{q}\right) + \Pr[\times = \mathbf{z}]\cdot\left(\frac{1}{2}-\frac{1}{q}\right)\right) \\
=&\left(\frac{1}{4}-\frac{1}{2q}\right)\cdot\left(\Pr[\times = \mathbf{y}] + \Pr[\times = \mathbf{z}]\right) = \frac{1}{4}-\frac{1}{2q},
\end{aligned}$$

where the inequality is due to the following: since $(\mathbf{I},\mathsf{rand},\overrightarrow{h}) \in \widehat{G_{\mathbf{y}}} \cup \widehat{G_{\mathbf{z}}}$, we have that

$$\left|A^{\times}_{T,i}(\mathbf{I},\mathsf{rand},\overrightarrow{h})\right| \geq \frac{1}{2}\left|T^{\times}_{T,i}(\mathbf{I},\mathsf{rand},\overrightarrow{h})\right| - q^{\ell-i+1}.$$

Since we conditioned on $(\mathbf{I},\mathsf{rand},\overrightarrow{h}') \in T^{\times}_{T,i}(\mathbf{I},\mathsf{rand},\overrightarrow{h})$, the probability in question is

$$\frac{\left|A^{\times}_{T,i}(\mathbf{I},\mathsf{rand},\overrightarrow{h})\right|}{\left|T^{\times}_{T,i}(\mathbf{I},\mathsf{rand},\overrightarrow{h})\right|} \geq \frac{1}{2} - \frac{q^{\ell-i+1}}{\left|T^{\times}_{T,i}(\mathbf{I},\mathsf{rand},\overrightarrow{h})\right|} \geq \frac{1}{2} - \frac{q^{\ell-i+1}}{q^{\ell-i+2}} = \frac{1}{2} - \frac{1}{q}.$$

In the following we denote by $\widehat{G}_b$ the set $\widehat{G_{\mathbf{y}}}$ if $b = 1$ and the set $\widehat{G_{\mathbf{z}}}$ if $b = 0$. Plugging the result back into the previous lower bound of B's advantage yields

$$\begin{aligned}
\mathrm{Adv}_{\mathsf{B}}^{\mathbf{DLOG}} &\geq \left(\frac{1}{4} - \frac{1}{2q}\right) \cdot \Pr\left[\begin{array}{c} (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in \widehat{G}_b \\ (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}') \in T^{\times}_{T,i}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \\ t(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) = i \end{array}\right] \\
&= \left(\frac{1}{4} - \frac{1}{2q}\right) \cdot \Pr\left[(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}') \in T^{\times}_{T,i}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \middle| \begin{array}{c} (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in \widehat{G}_b \\ t(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) = i \end{array}\right] \\
&\quad \cdot \Pr\left[(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in \widehat{G}_b\right] \cdot \Pr\left[t(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) = i \middle| (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in \widehat{G}_b\right] \\
&\geq \left(\frac{1}{4} - \frac{1}{2q}\right) \cdot \left(\frac{\epsilon_{B_T^\times}}{16(\ell+1)} - \frac{2}{q}\right) \cdot \frac{3\epsilon_{B_T^\times}}{128(\ell+1)} \cdot \frac{1}{\ell+1}
\end{aligned}$$

(where the last inequality is due to Lemma 4.2.48 and Lemma 4.2.49). Plugging in $\epsilon_{B_T^\times} \geq \frac{\epsilon}{96}$ for $\epsilon \geq \frac{432\left(1 - \frac{1}{(\ell+1)^2}\right)}{q}$ (see Lemma 4.2.37) and $\epsilon = \frac{\epsilon_{\mathsf{U}}}{\binom{Q_h}{\ell+1}}$ yields the theorem statement.

□

4.4 Extension to Multiple Tags

Theorem 4.4.1. *Let* U *be an adversary against* ℓ-$\mathbf{OMUF}_{\mathsf{AO}}$ *that runs in time* t_{U}, *closes at most* ℓ_{info} *signing sessions per tag* info, *closes at most* ℓ *signing sessions in total, and queries at most* Q_{info} *tags* info *to oracle* H^*. *Let* $\mathrm{Adv}^{\mathbf{OMUF}_{\mathsf{AO}}}_{Q_{\mathsf{info}}, \ell_{\mathsf{info}}, \mathsf{U}}$ *be* U*'s advantage. Then there exists a reduction* B *against* 1-info-$\mathbf{OMUF}_{\mathsf{AO}}$ *that runs in time* $t_{\mathsf{B}} \approx t_{\mathsf{U}}$ *and makes at most* ℓ_{info} *signing queries and has advantage*

$$\mathrm{Adv}_{\mathsf{B}}^{\ell_{\mathsf{info}}\text{-}1\text{-}\mathsf{info}\text{-}\mathbf{OMUF}_{\mathsf{AO}}} \geq \frac{\mathrm{Adv}^{\ell - \mathbf{OMUF}_{\mathsf{AO}}}_{Q_{\mathsf{info}}, \ell_{\mathsf{info}}, \mathsf{A}} - \frac{\ell}{q}}{Q_{\mathsf{info}}}.$$

The proof of this theorem mostly follows that in [AO00].

Proof. Without loss of generality, assume that U's queries to H^* are all distinct. We first describe a game-hop.

$\mathbf{G}_0$**:** This is the original ℓ-$\mathbf{OMUF}_{\mathsf{AO}}$ game.

$\mathbf{G}_1$**:** This game outputs 0 if U outputs a valid tuple $(m, \mathsf{sig}, \mathsf{info})$ where info has never been queried to H^*.

Claim 4.4.2. $\left|\Pr[\mathbf{G}_0^{\mathsf{U}} = 1] - \Pr[\mathbf{G}_1^{\mathsf{U}} = 1]\right| \leq \frac{\ell}{q}$.

Proof. Let Valid be the event that U outputs a valid tuple $(m, \mathsf{sig} = (\rho, \omega, \sigma, \delta), \mathsf{info})$ where info has never been queried to H^*; $\mathbf{G}_0$ and $\mathbf{G}_1$ are identical unless Valid happens. For each output $(m, \mathsf{sig}, \mathsf{info})$ of U, if $H^*(\mathsf{info})$ has never been queried, $\mathbf{z} = H^*(\mathsf{info})$ is a random element in $\mathbb{G}$ independent of all other random variables in U's view. Hence, $H(\mathbf{g}^{\rho}\mathbf{y}^{\omega}, \mathbf{g}^{\sigma}\mathbf{z}^{\delta}, \mathbf{z}, m)$ is a random integer in $\mathbb{Z}_q$, and the probability that it equals $\omega + \delta$ is $1/q$. Since there are at most ℓ such output tuples in total, we have that

$$\Pr[\mathsf{Valid}] \leq \frac{\ell}{q},$$

and the claim follows. □

The reduction B against 1-info-$\mathbf{OMUF}_{\mathsf{AO}}$ behaves as follows.

Setup: On input a public key pk $= \mathbf{y}$, B forwards it to U and samples $J \xleftarrow{\$} [Q_{\mathsf{info}}]$ (a guess that U's J-th H^* query is part of its final output).

Online Phase: B answers signing and hash queries as follows.

Queries to H^*: For the J-th query info$_J$ to H^*, B forwards the query to its challenger and forwards the response back to U. For any other query info, B lazily samples $w_{\mathsf{info}} \xleftarrow{\$} \mathbb{Z}_q$ and sets $H^*(\mathsf{info}) := \mathbf{g}^{w_{\mathsf{info}}}$.

Queries to H: B forwards these queries to its challenger and forwards the responses back to U.

Queries to sign$_1$: On input info, if info $=$ info$_J$, B forwards the query to its challenger and forwards the response back to U. Otherwise B behaves as the $\mathbf{z}$-side signer. That is, it increments the session id sid, sets info$_{\mathsf{sid}}$:= info, samples $c_{\mathsf{sid}}, r_{\mathsf{sid}}, v_{\mathsf{sid}} \xleftarrow{\$} \mathbb{Z}_q$, and sets $\mathbf{a} := \mathbf{y}^{c_{\mathsf{sid}}} \cdot \mathbf{g}^{r_{\mathsf{sid}}}$ and $\mathbf{b} := \mathbf{g}^{v_{\mathsf{sid}}}$. It saves the internal state $c_{\mathsf{sid}}, r_{\mathsf{sid}}, v_{\mathsf{sid}}$ and outputs sid, $\mathbf{a}, \mathbf{b}$ to U.

Queries to sign$_2$: On input (sid, e), B checks if info$_{\mathsf{sid}}$ $=$ info$_J$. If so, it forwards the query to its challenger. Otherwise it computes $d_{\mathsf{sid}} := e - c_{\mathsf{sid}}$ and $s_{\mathsf{sid}} := v_{\mathsf{sid}} - d_{\mathsf{sid}} w_{\mathsf{info}}$. It outputs $c_{\mathsf{sid}}, r_{\mathsf{sid}}, d_{\mathsf{sid}}, s_{\mathsf{sid}}$ to U.

Output determination: When U outputs a list of signatures $(m_i, \mathsf{sig}_i, \mathsf{info}_i)_{i=1}^{\ell+1}$, B checks that all info were queried to H^* by U. If so, B outputs all $(m_i, \mathsf{sig}_i, \mathsf{info}_i)$ tuples with info$_i$ $=$ info$_J$. Otherwise B aborts.

We now analyze the advantage of the reduction B. Due to the witness-indistinguishability of the scheme (see Lemma 4.2.21 in Section 4.2.4), B simulates game $\mathbf{G}_1$ perfectly to U. If U wins $\mathbf{G}_1$, there must be one tag for which U has output more signatures than closed signing sessions. By the definition of $\mathbf{G}_1$, this tag was queried to H^* by U. Therefore, with probability $\frac{1}{Q_{\mathsf{info}}}$, this tag is info$_J$.

We conclude that

$$\mathrm{Adv}_{\mathsf{B}}^{\ell_{\mathsf{info}}\text{-}1\text{-info-}\mathbf{OMUF}_{\mathsf{AO}}} \geq \frac{\Pr[\mathbf{G}_1^{\mathsf{U}} = 1]}{Q_{\mathsf{info}}} \geq \frac{\Pr[\mathbf{G}_0^{\mathsf{U}} = 1] - \frac{\ell}{q}}{Q_{\mathsf{info}}} = \frac{\mathrm{Adv}_{Q_{\mathsf{info}}, \ell_{\mathsf{info}}, \mathsf{A}}^{\ell-\mathbf{OMUF}_{\mathsf{AO}}} - \frac{\ell}{q}}{Q_{\mathsf{info}}}.$$

□

Chapter 5

Abe's (Partially) Blind Signature Scheme

In this chapter, we revisit Abe's blind signature scheme [Abe01]. The original paper by Abe is known to contain a flaw in the forking-based proof of one-more unforgeability. This was pointed out by Ohkubo and Abe [OA03] who gave a proof in the GGM instead.

This chapter is mostly based on [KLX22a] with some improvements from a follow-up work [KLR23a]. We first describe a natural extension of Abe's scheme to the partially blind setting in Section 5.1. Namely, instead of obtaining the public key part $\mathbf{z}$ by hashing the other public key elements $\mathbf{g}, \mathbf{h}, \mathbf{y}$, we additionally include the tag info making $\mathbf{z}$ a tag key as in the scheme by Abe and Okamoto discussed in Chapter 4. The original scheme can be obtained by leaving the tag empty. Like the scheme by Abe and Okamoto [AO00], Abe's scheme (and our partially blind variant) heavily relies on the OR-proof technique. However, to avoid vulnerability to the ROS attack, the scheme uses what we call *linking components*, that is, in each session the signer samples a new value $\mathbf{z}_1$ through a random oracle and computes $\mathbf{z}_2 = \mathbf{z}/\mathbf{z}_1$. From the signer's side, the protocol is a proof of knowledge of the discrete logarithm of the public key part $\mathbf{y}$ to $\mathbf{g}$ (this is the branch used during normal signing) or of $\mathbf{z}_1$ to $\mathbf{g}$ and $\mathbf{z}_2$ to the public key part $\mathbf{h}$ (this branch can only be used by a reduction that has control over the random oracles used to generate $\mathbf{z}$ and $\mathbf{z}_1$). To achieve (computational) blindness, the user blinds the values $\mathbf{z}$ and $\mathbf{z}_1$ into $\zeta = \mathbf{z}^\gamma$ and $\zeta_1 = \mathbf{z}_1^\gamma$ for some $\gamma \xleftarrow{\$} q$. This also sets $\zeta_2 = \mathbf{z}_2^\gamma = \zeta/\zeta_1$. With the help of the signer, the user generates a signature that serves as a Fiat-Shamir style proof of knowledge of the discrete logarithm of $\mathbf{y}$ to $\mathbf{g}$ or ζ_1 to $\mathbf{g}$ and ζ_2 to $\mathbf{h}$ and ζ to $\mathbf{z}$ (the last part the user generates without the signer's help as it knows γ).

We then turn to proving partial blindness of the scheme in Section 5.2. Partial blindness holds unter the Decisional Diffie-Hellman assumption (DDH) in the underlying group in the ROM - the algebraic group model is not needed for this proof.

The main result of this chapter is the proof of one-more unforgeability which we provide in Section 5.3. We tightly prove OMUF under the Discrete Logarithm assumption in the AGM + ROM. Like the proof by Abe [Abe01], we proceed in two steps. First, we prove the so-called *restrictive blinding lemma* in Section 5.3.3 which states that it is infeasible for the adversary to come up with its own values for ζ and ζ_1 such that they are not linked to a signing session We then turn to the main theorem in Section 5.3.4 which shows full one-more unforgeability of Abe's blind signature scheme where now we can rely on the fact that since the adversary cannot come up with its own linking components, there must be two signatures linked to the same session.

After having proven the full concurrent security in the AGM, we briefly sketch how the forking-based technique from Chapter 4 can be applied to the partially blind variant of Abe's scheme in Section 5.4. We note that the same strategy for extending a single tag proof to a multi-tag proof as in the previous chapter does not work, and thus we turn to a slightly adapted proof strategy where the deterministic wrapper directly deals with all the tags.

5.1 Adaption of Abe's Blind Signature Scheme to Allow Partial Blindness

We begin by describing an adaption of Abe's blind signature scheme BSA [Abe01] to the partially blind setting. A figure depicting an interaction between signer and user can be found in Figure 5.1 . Let again $\mathbb{G}$ be a group of order q with generator $\mathbf{g}$ described by public parameters pp. Let $H_1 \colon \{0,1\}^* \to \mathbb{G} \setminus \{\epsilon\}$, $H_2 \colon \{0,1\}^* \to \mathbb{G} \setminus \{\epsilon\}$, $H_3 \colon \{0,1\}^* \to \mathbb{Z}_q$ be hash functions.

- KeyGen: On input pp, KeyGen samples $\mathbf{h} \xleftarrow{\$} \mathbb{G}$, $x \xleftarrow{\$} \mathbb{Z}_q$ and sets $\mathbf{y} \leftarrow \mathbf{g}^x$. It sets sk $\leftarrow x$, pk $\leftarrow (\mathbf{g}, \mathbf{h}, \mathbf{y})$ and returns (sk, pk).
- Sign$_1$: On input sk, info, Sign$_1$ samples $\mathrm{rnd} \xleftarrow{\$} \{0,1\}^\lambda$ and $u, d, s_1, s_2 \xleftarrow{\$} \mathbb{Z}_q$. It computes $\mathbf{z} \leftarrow H_1(\mathsf{pk}, \mathsf{info})$, $\mathbf{z}_1 \leftarrow H_2(\mathrm{rnd})$, $\mathbf{z}_2 \leftarrow \mathbf{z}/\mathbf{z}_1$, $\mathbf{a} \leftarrow \mathbf{g}^u$, $\mathbf{b_1} \leftarrow \mathbf{g}^{s_1} \cdot \mathbf{z}_1^d$, $\mathbf{b_2} \leftarrow \mathbf{h}^{s_2} \cdot \mathbf{z}_2^d$. It returns a commitment $(\mathrm{rnd}, \mathbf{a}, \mathbf{b_1}, \mathbf{b_2})$ and a state $\mathsf{st}_S = (u, d, s_1, s_2, \mathsf{info})$.
- Sign$_2$: On input a secret key sk, a challenge e, and state $\mathsf{st}_S = (u, d, s_1, s_2, \mathsf{info})$, Sign$_2$ computes $c \leftarrow e - d \mod q, r \leftarrow u - c \cdot \mathsf{sk} \mod q$ and returns the response (c, d, r, s_1, s_2).
- User$_1$: On input a public key pk and a commitment $(\mathrm{rnd}, \mathbf{a}, \mathbf{b_1}, \mathbf{b_2})$, a tag info, and message m, User$_1$ does the following. It samples $\gamma \xleftarrow{\$} \mathbb{Z}_q^*$ and $\tau, t_1, t_2, t_3, t_4, t_5 \xleftarrow{\$} \mathbb{Z}_q$. Then, it computes $\mathbf{z} \leftarrow H_1(\mathsf{pk}, \mathsf{info})$, $\mathbf{z}_1 \leftarrow H_2(\mathrm{rnd})$, $\alpha \leftarrow \mathbf{a} \cdot \mathbf{g}^{t_1} \cdot \mathbf{y}^{t_2}$, $\zeta \leftarrow \mathbf{z}^\gamma$, $\zeta_1 \leftarrow \mathbf{z}_1^\gamma$, $\zeta_2 \leftarrow \zeta/\zeta_1$. Next, it sets $\beta_1 \leftarrow \mathbf{b}_1^\gamma \cdot \mathbf{g}^{t_3} \cdot \zeta_1^{t_4}$, $\beta_2 \leftarrow \mathbf{b}_2^\gamma \cdot \mathbf{h}^{t_5} \cdot \zeta_2^{t_4}$, $\eta \leftarrow \mathbf{z}^\tau$, and $h \leftarrow H_3(\zeta, \zeta_1, \alpha, \beta_1, \beta_2, \eta, m, \mathsf{info})$. Finally, it computes a challenge $e \leftarrow h - t_2 - t_4 \mod q$, the state $St_U \leftarrow (\gamma, \tau, t_1, t_2, t_3, t_4, t_5, m)$ and returns e, St_U.
- User$_2$: On input a public key pk, a response (c, d, r, s_1, s_2) and a state $(\gamma, \tau, t_1, t_2, t_3, t_4, t_5, m)$, User$_2$ first computes $\rho \leftarrow r + t_1$, $\omega \leftarrow c + t_2$, $\sigma_1 \leftarrow \gamma \cdot s_1 + t_3$, $\sigma_2 \leftarrow \gamma \cdot s_2 + t_5$, and $\delta \leftarrow d + t_4$. Then, it computes $\mu \leftarrow \tau - \delta \cdot \gamma$ and $h \leftarrow H_3(\zeta, \zeta_1, \mathbf{g}^\rho \mathbf{y}^\omega, \mathbf{g}^{\sigma_1} \zeta_1^\delta, \mathbf{h}^{\sigma_2} \zeta_2^\delta, \mathbf{z}^\mu \zeta^\delta, m)$. It returns the signature $\sigma \leftarrow (\zeta, \zeta_1, \rho, \omega, \sigma_1, \sigma_2, \delta, \mu)$ if $\delta + \omega = h$; otherwise, it returns $\perp$.[1]
- Verify: On input a public key pk, a signature $(\zeta, \zeta_1, \rho, \omega, \sigma_1, \sigma_2, \delta, \mu)$ and a message m, Verify returns 0 if $\zeta = \zeta_1 = \epsilon$.[2] It then computes first $\mathbf{z} \leftarrow H_1(\mathsf{pk}, \mathsf{info})$ and then $h := H_3(\zeta, \zeta_1, \mathbf{g}^\rho \mathbf{y}^\omega, \mathbf{g}^{\sigma_1} \zeta_1^\delta, \mathbf{h}^{\sigma_2} \zeta_2^\delta, \mathbf{z}^\mu \zeta^\delta, m, \mathsf{info})$. It returns 1 if $\delta + \omega = h$; otherwise, it returns 0.

We note that the only change we made to Abe's scheme is that in our variant, the $\mathbf{z}$ part of the public key is derived as a hash of pk and a tag info instead of as a hash of the other elements of the public key. It is easy to see that by using an empty info this yields the original scheme and thus our proofs about the adapted scheme also apply to the original.

[1]We note that the check for $h = \omega + \delta$ implicitly checks that $c + d = e$ as well as $\mathbf{a} = \mathbf{y}^c \mathbf{g}^r, \mathbf{b}_1 = \mathbf{z}_1^d \mathbf{g}^{s_1}, \mathbf{b}_2 = \mathbf{z}_2^d \mathbf{h}^{s_2}$, i.e. it checks that the output of Sign$_2$ was valid.

[2]This is necessary as otherwise there is a trivial attack: pick $\omega, \rho \xleftarrow{\$} \mathbb{Z}_q$, set $\alpha = \mathbf{g}^\rho \mathbf{y}^\omega$, then pick $\sigma_1, \sigma_2, \mu \xleftarrow{\$} \mathbb{Z}_q$ compute $h = H_3(\zeta, \zeta_1, \alpha, \mathbf{g}^{\sigma_1}, \mathbf{h}^{\sigma_2}, \mathbf{z}^\mu, \mathsf{info})$. Then set $\delta = h - \omega$ and output the signature $(\zeta, \zeta_1, \rho, \omega, \sigma_1, \sigma_2, \delta, \mu)$

$$
\begin{array}{lcr}
\textbf{Signer} & & \textbf{User} \\
\mathsf{sk} = x & & \\
\mathsf{pk} = (\mathbf{g}, \mathbf{h}, \mathbf{y} = \mathbf{g}^x) & & \mathsf{pk} = (\mathbf{g}, \mathbf{h}, \mathbf{y}) \\
\mathsf{info} & & m, \mathsf{info} \\
\mathbf{z} \leftarrow H_1(\mathsf{pk}, \mathsf{info}) & & \\
u, d, s_1, s_2 \xleftarrow{\$} \mathbb{Z}_q & & \\
\mathrm{rnd} \xleftarrow{\$} \{0,1\}^\lambda & & \\
\mathbf{z}_1 \leftarrow H_2(\mathrm{rnd}),\ \mathbf{z}_2 \leftarrow \mathbf{z}/\mathbf{z}_1 & & \\
\mathbf{a} \leftarrow \mathbf{g}^u & & \\
\mathbf{b_1} \leftarrow \mathbf{g}^{s_1} \cdot \mathbf{z}_1^d & & \\
\mathbf{b_2} \leftarrow \mathbf{h}^{s_2} \cdot \mathbf{z}_2^d & \xrightarrow{\mathbf{a},\mathbf{b_1},\mathbf{b_1}} & \tau, t_1, t_2, t_3, t_4, t_5 \xleftarrow{\$} \mathbb{Z}_q \\
 & & \mathbf{z}_1 \leftarrow H_2(\mathrm{rnd}) \\
 & & \alpha \leftarrow \mathbf{a} \cdot \mathbf{g}^{t_1} \cdot \mathbf{y}^{t_2} \\
 & & \zeta \leftarrow \mathbf{z}^\gamma,\ \zeta_1 \leftarrow \mathbf{z}_1^\gamma,\ \zeta_2 \leftarrow \zeta/\zeta_1 \\
 & & \beta_1 \leftarrow \mathbf{b}_1^\gamma \cdot \mathbf{g}^{t_3} \cdot \zeta_1^{t_4} \\
 & & \beta_2 \leftarrow \mathbf{b}_2^\gamma \cdot \mathbf{h}^{t_5} \cdot \zeta_2^{t_4} \\
 & & \eta \leftarrow \mathbf{z}^\tau \\
 & & h \leftarrow H_3(\zeta, \zeta_1, \alpha, \beta_1, \beta_2, \eta, m, \mathsf{info}) \\
 & \xleftarrow{x} & e \leftarrow h - t_2 - t_4 \\
c \leftarrow e - d & & \\
r \leftarrow u - c \cdot x & & \\
 & \xrightarrow{c,r,d,s_1,s_2} & \rho \leftarrow r + t_1,\ \omega \leftarrow c + t_2 \\
 & & \sigma_1 \leftarrow \gamma \cdot s_1 + t_3 \\
 & & \sigma_2 \leftarrow \gamma \cdot s_2 + t_5 \\
 & & \delta \leftarrow d + t_4 \\
 & & \mu \leftarrow \tau - \delta \cdot \gamma \\
 & & \delta + \omega \stackrel{?}{=} H_3(\zeta, \zeta_1, \mathbf{g}^\rho \mathbf{y}^\omega, \mathbf{g}^{\sigma_1} \zeta_1^\delta, \mathbf{h}^{\sigma_2} \zeta_2^\delta, \mathbf{z}^\mu \zeta^\delta, m, \mathsf{info}) \\
 & & \Downarrow \\
 & & (m, (\zeta, \zeta_1, \rho, \omega, \sigma_1, \sigma_2, \delta, \mu))
\end{array}
$$

Figure 5.1: Interaction between Signer and User for the partially blind version of BSA

We note that Abe refers to $\mathbf{z}, \mathbf{z}_1, \zeta, \zeta_1$ as the tags of a signing session or signature. However, as we are considering partial blindness, we will refer to them as the *linking components*. By [Abe01], the original scheme is computationally blind under the Decisional Diffie-Hellman assumption. For completeness, we provide a detailed proof of the partial computational blindness of our variant in Section 5.2.

5.2 Partial Blindness of the Adapted Abe Scheme

We provide a formal proof of partial blindness under chosen keys for the Abe blind signature scheme. Abe [Abe01] proved the scheme to be blind for keys selected by the challenger.

Lemma 5.2.1. *If the Decisional Diffie-Hellman problem is (t, ϵ)-hard in $\mathbb{G}$, the partially blind variant of Abe's blind signature scheme* BSA *is $(t, \approx 2\epsilon)$ computationally partially blind in the random oracle model.*

Proof. We use similar techniques as [BL13a].
Game $\mathbf{G_1}$ The first game is identical to the blindness game from Definition 2.4.3 for Abe's blind signature scheme.

Setup. $\mathbf{G_1}$ samples $b \xleftarrow{\$} \{0,1\}$.

Simulation of oracle H_1. $\mathbf{G_1}$ simulates H_1 by lazy sampling of group elements.

Online Phase. When M outputs a public key $(\mathbf{g}, \mathbf{y}, \mathbf{h})$ and messages $\widetilde{m}_0$ and $\widetilde{m}_1$, and tags info$_0$, info$_1$, $\mathbf{G_1}$ verifies info$_0$ = info$_1$ assigns $m_0 = \widetilde{m}_b$ and $m_1 = \widetilde{m}_{b-1}$

Oracle user$_1$**.** works the same as described in Definition 2.4.3

Oracle user$_2$**.** works the same as described in Definition 2.4.3

Simulation of H_2. H_2 is simulated through lazy sampling

Simulation of H_3. H_3 is simulated through lazy sampling

Output determination. as described in Definition 2.4.3

The second game replaces the signature for m_0 by a signature that is independent of the run with the signer.
Game $\mathbf{G_2}$ The second game generates the signature on m_0 independently of the corresponding signing session.

Setup. $\mathbf{G_2}$ samples $b \xleftarrow{\$} \{0,1\}$.

Simulation of oracle H_1. $\mathbf{G_2}$ simulates H_1 by lazy sampling of group elements.

Online Phase. When M outputs a public key $(\mathbf{g}, \mathbf{y}, \mathbf{h})$ and messages $\widetilde{m}_0$ and $\widetilde{m}_1$ and $\widetilde{\mathsf{info}}_0, \widetilde{\mathsf{info}}_1$, $\mathbf{G_2}$ verifies that the key is well-formed and that $\widetilde{\mathsf{info}}_0 = \widetilde{\mathsf{info}}_1$ and aborts with output 0 if this check fails. It further assigns $m_0 = \widetilde{m}_b$ and $m_1 = \widetilde{m}_{b-1}$ as well as info$_0$ = $\widetilde{\mathsf{info}}_0$ and info$_1$ = $\widetilde{\mathsf{info}}_1$.

Oracle user$_1$**.** For message m_1, the oracle behaves the same as in $\mathbf{G_1}$. For message m_0, it checks that session 0 is not open yet and opens session 0. Then the game picks $\delta, \omega, \sigma_1, \sigma_2, \rho, \mu$ uniformly at random from $\mathbb{Z}_q$. It further draws two random group elements ζ and ζ_1 and sets $\zeta_2 := \zeta/\zeta_1$. It then sets $H_3(\mathbf{y}^\omega \cdot \mathbf{g}^\rho, \zeta_1^\delta \cdot \mathbf{g}^{\sigma_1}, \zeta_2^\delta \cdot \mathbf{h}^{\sigma_2}, \zeta^\delta \cdot \mathbf{z}^\mu, m_0, \mathsf{info}_0) := \delta + \omega$. It draws $e \xleftarrow{\$} \mathbb{Z}_q$ uniformly at random and returns e as a challenge to the adversary.

Oracle user$_2$. For message m_1, the oracle behaves the same as in $\mathbf{G_1}$. For message m_0, on input c, d, r, s_1, s_2, the game does the following checks[3]: $e = d + c$, $\mathbf{a}_0 = \mathbf{g}^r \cdot \mathbf{y}^c$, $\mathbf{b}_{1,0} = \mathbf{g}^{s_1} \cdot \mathbf{z}_{1,0}^d$, $\mathbf{b}_{2,0} = \mathbf{h}^{s_2} \cdot \mathbf{z}_{2,0}^d$. It considers the produced signature to be the one generated in user$_1$.

Simulation of H_2. H_2 is simulated through lazy sampling

Simulation of H_3. For values not programmed in user$_1$, $\mathbf{G_2}$ simulates H_3 via lazy sampling

Output determination. as described in Definition 2.4.3

Claim 5.2.2. *The advantage of an adversary* B *to tell the difference between* $\mathbf{G_1}$ *and* $\mathbf{G_2}$ *is* $\mathrm{Adv}_{\mathsf{B}}^{\mathbf{G_1},\mathbf{G_2}} = \left|\Pr\left[\mathbf{G_1}^{\mathsf{B}} = 1\right] - \Pr\left[\mathbf{G_2}^{\mathsf{B}} = 1\right]\right| \leq \mathrm{Adv}_{\mathsf{B}'}^{\mathbf{DDH}}$.

Proof. We provide a reduction B$'$ that receives a random-generator DDH challenge $(\mathbf{W}, \mathbf{X}, \mathbf{Y}, \mathbf{Z})$ and simulates either $\mathbf{G_1}$ or $\mathbf{G_2}$ to the adversary. During the first phase of the online phase, the reduction programs the random oracle H_1 to return values $\mathbf{W}^{f_i}$ $f_i \in \mathbb{Z}_q$. For simulation of H_2, the reduction chooses exponents $g_i \xleftarrow{\$} \mathbb{Z}_q$ and returns values $\mathbf{X}^{g_i}$, yielding uniformly random values from the group $\mathbb{G}$. In user$_1$ for m_0, when the adversary sends the commitment which contains a random string rnd to be queried to the oracle H_2, the reduction identifies the $g = g_i$ that was used as the random exponent for $\mathbf{z}_1 = \mathbf{X}^g$. Denote further by f the f_i used for generation of $\mathbf{z} = H_1(\mathsf{pk}, \mathsf{info}_1)$. It sets $\zeta = \mathbf{Y}^f$ and $\zeta_1 = \mathbf{Z}^{f \cdot g}$. The reduction then proceeds to generate a signature by programming the random oracle H_3 as described in $\mathbf{G_2}$. For m_1, the reduction participates honestly in the signing protocol. In user$_2$, for m_0, the reduction checks that the adversary produces a valid signing transcript as described in $\mathbf{G_2}$. If both interactions yield valid signatures (i.e. the adversary produced a valid transcript for m_0 and a valid signature for m_1), the reduction outputs both signatures, otherwise $\perp$. If the adversary outputs it was playing game $\mathbf{G_1}$, the reduction outputs 0, otherwise it outputs 1.

We argue that if the challenge is a Diffie-Hellman tuple, the reduction simulates $\mathbf{G_1}$ perfectly. For a tuple $\mathbf{W}, \mathbf{W}^a, \mathbf{W}^b, \mathbf{W}^{ab}$, the tuple $\mathbf{z} = \mathbf{W}^f, \mathbf{z}_1 = \mathbf{W}^{a \cdot f \cdot \frac{g}{f}}, \zeta = \mathbf{W}^{b \cdot f}, \zeta_1 = \mathbf{W}^{a \cdot b \cdot f \cdot g}$ is a valid Diffie-Hellman tuple w.r.t generator $\mathbf{W}^f$. Furthermore, the user tags ζ and ζ_1 can be computed from $\mathbf{z}$ and $\mathbf{z}_1$ using blinding factor $\gamma = b$. Furthermore, for any c, d, r, s_1, s_2 and signature components $\omega, \delta, \rho, \sigma_1, \sigma_2, \mu$ there are unique choices of $t_1 = \rho - r, t_2 = \omega - c, t_3 = \sigma_1 - \gamma \cdot s_1, t_4 = \delta - d, t_5 = \sigma_2 - \gamma \cdot s_2, \tau = \mu + \delta \cdot \gamma$ that explain the signature in combination with the transcript. Thus, the produced combination of signature and transcript is identically distributed as an honestly generated signature.

If the challenge is not a Diffie-Hellman tuple, then the reduction simulates $\mathbf{G_2}$ perfectly as the linking components $\zeta_i, \zeta_{1,i}$ look like random group elements and the reduction computes the same steps as $\mathbf{G_2}$ to generate the signatures and its outputs to the adversary. □

We describe the final game $\mathbf{G_3}$ where both signatures are independent from the runs with the signer.
Game $\mathbf{G_3}$

Setup. $\mathbf{G_3}$ samples $b \xleftarrow{\$} \{0, 1\}$.

Simulation of oracle H_1. $\mathbf{G_3}$ simulates H_1 by lazy sampling of group elements.

Online Phase. When M outputs a public key $(\mathbf{g}, \mathbf{y}, \mathbf{h})$ and messages $\widetilde{m}_0$ and $\widetilde{m}_1$, $\mathbf{G_3}$ verifies that the key is well-formed and checks that $\mathsf{info}_0 = \mathsf{info}_1$ and aborts with output 0 if this check fails. It further assigns $m_0 = \widetilde{m}_b$ and $m_1 = \widetilde{m}_{b-1}$

[3]We note that these checks need to be done explicitly here, as they are no longer implicitly performed through checking that $h = \omega + \delta$,

Oracle user_1**.** For session b', the game checks that session b' is not open yet and opens session b'. It sets $\mathbf{z} \leftarrow H_1(\mathsf{info})$. Then the game picks $\delta, \omega, \sigma_1, \sigma_2, \rho, \mu$ uniformly at random from $\mathbb{Z}_q$. It further draws two random group elements ζ and ζ_1 and sets $\zeta_2 := \zeta/\zeta_1$. It then sets $H_3(\mathbf{y}^\omega \cdot \mathbf{g}^\rho, \zeta_1^\delta \cdot \mathbf{g}^{\sigma_1}, \zeta_2^\delta \cdot \mathbf{h}^{\sigma_2}, \zeta^\delta \cdot \mathbf{z}^\mu, m_{b'}, \mathsf{info}_{b'}) := \delta + \omega$. It draws $e \xleftarrow{\$} \mathbb{Z}_q$ uniformly at random and returns e as a challenge to the adversary.

Oracle user_2**.** For both sessions (denoted by $i = 0, 1$), on input $c_i, d_i, r_i, s_{1,i}, s_{2,i}$, the game does the following checks: $e_i = d_i + c_i$, $\mathbf{a}_i = \mathbf{g}^{r_i} \cdot \mathbf{y}^{c_i}$, $\mathbf{b}_{1,i} = \mathbf{g}^{s_{1,i}} \cdot \mathbf{z}_{1,i}^{d_i}$, $\mathbf{b}_{2,i} = \mathbf{h}^{s_{2,i}} \cdot \mathbf{z}_{2,i}^{d_i}$. It considers the output signature to be the one generated for this session in user_1.

Simulation of H_2**.** H_2 is simulated through lazy sampling

Simulation of H_3**.** For values not programmed in user_1, $\mathbf{G_2}$ simulates H_3 via lazy sampling

Output determination. as described in Definition 2.4.3

Claim 5.2.3. *The advantage of an adversary* B *to tell the difference between* $\mathbf{G_1}$ *and* $\mathbf{G_2}$ *is* $\mathrm{Adv}_{\mathsf{B}'''}^{\mathbf{G_2},\mathbf{G_3}} = \Pr\left[\mathbf{G_2}^{\mathsf{B}'''} = 1\right] - \Pr\left[\mathbf{G_3}^{\mathsf{B}'''} = 1\right] \leq \mathrm{Adv}_{\mathsf{B}''}^{\mathbf{DDH}}$.

Proof. Follows along the same lines as Claim 5.2.2, embedding the $\mathbf{DDH}$ challenge in the signature for m_1 this time. □

In game $\mathbf{G_3}$, the adversary cannot win, as both signatures are completely independent from the two runs. As game $\mathbf{G_3}$ needs to program the random oracle H_3 twice to generate the signatures (this fails with probability at most $\frac{2Q_h}{q^4 \cdot 2^{|m_0|}}$, i.e. if the adversary has made the exact same requests before), we get the following overall advantage of

$$\mathrm{Adv}_{\mathsf{M}}^{\mathbf{BLIND}_{\mathsf{BSA}}} = \frac{2 \cdot Q_h}{q^4 \cdot 2^{|m_0|}} + \mathrm{Adv}_{\mathsf{B}'}^{\mathbf{DDH}} + \mathrm{Adv}_{\mathsf{B}''}^{\mathbf{DDH}}$$

□

5.3 One-More-Unforgeability

In the following, we provide a proof for the one-more-unforgeability. Similar to [Abe01] we do this in two steps. First, we show that it is infeasible for an adversary to generate a signature that does not use a tag that corresponds to a closed signing session. (Note that the scheme is only computationally blind, and an unbounded algorithm can link signatures and sessions since $(\mathbf{z}, \mathbf{z}_1, \zeta, \zeta_1)$ forms a DDH tuple. We call such tuples *linking components*, and refer to $\mathbf{z}, \mathbf{z}_1$ as "signer-side" and ζ, ζ_1 as "user-side".) This corresponds to Abe's restrictive blinding lemma. Then, as the main theorem, we show that it is also infeasible for an adversary to win ℓ-$\mathbf{OMUF}$ by providing two signatures corresponding to the same closed signing session.

Our techniques. The main idea for both the lemma and the theorem is to use the algebraic representations of the group elements submitted to the random oracle H_3 in combination with the corresponding signature to compute the discrete logarithm of either $\mathbf{y}$ or $\mathbf{h}$. This fails either when the adversary has not made a hash query for the signature in question, or when the representation of the hash query does not contain more information than the signature, i.e., the exponents in the representation already match the signature. We show that both of these cases only occur with a negligible probability. We simulate the protocol in two different ways. One way is to use the secret key x like an honest signer and try to extract the discrete logarithm of $\mathbf{h}$ or one of the $\mathbf{z}$ (as we will compute all the $\mathbf{z}$ using $\mathbf{h}$ the latter two cases are equivalent). The other way is to program the random oracles H_1 and H_2 so that the reduction can use the discrete logarithms of $\mathbf{z}, \mathbf{z}_1, \mathbf{z}_2$ to simulate the other side of the OR-proof for extraction of the secret key. We will show that the reduction always has an efficient way to identify the 'forgery', i.e. a signature that was not generated using the honest singing algorithm. We elaborate on this in the next part. This efficient identification, in combination with not having to run the protocol twice for forking, renders a tight proof.

Comparison to the original standard model proof by Abe [Abe01]. We briefly recall that similar to our proof, the original proof also shows the restrictive blinding lemma first, which, shows that an adversary that wins the $\mathbf{OMUF}$ game and at the same time produces a signature where $\mathrm{dlog}_\zeta \zeta_1 \neq \mathrm{dlog}_{\mathbf{z}} \mathbf{z}_{1,i}$ for all sessions i, can be used to solve the discrete logarithm problem. The proof uses the forking technique, i.e. it rewinds the adversary to obtain a second set of signatures with different hash responses to H_3. The original proof of the restrictive blinding lemma also uses two signers, one that embeds in $\mathbf{y}$ and signs using the $\mathbf{z}$-side witness, another that embeds in $\mathbf{h}$ and signs using the secret key x. These two signers are indistinguishable for a single run, however, two forking runs using the same witness reveal the witness being used internally. In particular, a forking pair of runs using the secret key x to sign, cannot be reproduced by a signer that does not know the x-side witness. Therefore, the distribution of signatures obtained from forking runs, in particular the components δ and ω may depend on which witness was used internally. We note that for example in 'honestly generated' signatures (i.e. when the adversary followed the User$_1$ and User$_2$ algorithms to generate signatures), the a pair of signatures at the forking hash query reveals exactly the same witness as the signer used to sign while forking, so it is not clear why a similar thing may not also hold for 'dishonestly generated' signatures.

As our reduction for the restrictive blinding lemma works in the AGM, we can avoid the rewinding step. The adversary submits representations of all the group elements contained in a hash query, which gives the reduction information that would otherwise be obtained from the previous run. As the scheme is perfectly witness indistinguishable, the representations submitted by the adversary are independent of the witness used internally. We show in Claim 5.3.10, that even a so-called *reduced representation* that does use factors that are only determined after all signing sessions were closed, is likely to reveal enough information for the reduction to be able to solve the discrete logarithm problem.

5.3.1 The $\mathbf{z}$-side Signer

We describe an alternate signing procedure for the scheme from Section 5.1 that will come in handy during the proof of one-more unforgeability. This $\mathbf{z}$-side signer has knowledge of the discrete logarithms of all $\mathbf{z}_i, \mathbf{z}_{1,i}$ as well as of $\mathbf{h}$, but it does not need the secret key x to generate signatures. We will specify later on how our reduction(s) can obtain knowledge of these values and just write dlog here to denote the various discrete logarithms.

Sign$_1$ On input info, sample $c, r, v_1, v_2 \xleftarrow{\$} \mathbb{Z}_q$, sample $\mathrm{rnd} \xleftarrow{\$} \{0,1\}^\lambda$, set $\mathbf{a} = \mathbf{g}^r \mathbf{y}^c$, $\mathbf{b_1} = \mathbf{g}^{v_1}$, $\mathbf{b_2} = \mathbf{g}^{v_2}$, $\mathbf{z}_1 = H_2(\mathrm{rnd})$, $\mathbf{z} = H_1(\mathsf{pk}, \mathsf{info})$. Return $(\mathrm{rnd}, \mathbf{a}, \mathbf{b_1}, \mathbf{b_2})$ $\mathsf{st}_S = (c, r, v_1, v_2, \mathbf{z}, \mathbf{z}_1)$.

Sign$_2$ on input $\mathsf{st}_S = (c, r, v_1, v_2, \mathbf{z}, \mathbf{z}_1)$ along with e, compute $d = e - c$, $s_1 = v_1 - d \cdot \mathrm{dlog}_{\mathbf{g}} \mathbf{z}_1$ and $s_2 = v_2 - d \cdot \mathrm{dlog}_{\mathbf{h}}(\mathbf{z}/\mathbf{z}_1)$. Output (c, d, r, s_1, s_2).

We show also an interaction between the $\mathbf{z}$-side signer and the user in Figure 5.2.

Signer		**User**
$\mathsf{pk} = (\mathbf{g}, \mathbf{h}, \mathbf{y})$		$\mathsf{pk} = (\mathbf{g}, \mathbf{h}, \mathbf{y})$
info		m, info
$\mathbf{z} \leftarrow H_1(\mathsf{pk}, \mathsf{info})$		$\mathbf{z} \leftarrow H_1(\mathsf{pk}, \mathsf{info})$
$c, r, v_1, v_2 \xleftarrow{\$} \mathbb{Z}_q$		
$\mathrm{rnd} \xleftarrow{\$} \{0,1\}^\lambda$		
$\mathbf{z}_1 \leftarrow H_2(\mathrm{rnd})$, $\mathbf{z}_2 \leftarrow \mathbf{z}/\mathbf{z}_1$		
$\mathbf{a} \leftarrow \mathbf{g}^r \mathbf{y}^c$		
$\mathbf{b_1} \leftarrow \mathbf{g}^{v_1}$		
$\mathbf{b_2} \leftarrow \mathbf{h}^{v_2}$	$\xrightarrow{\mathbf{a}, \mathbf{b_1}, \mathbf{b_1}}$	$\tau, t_1, t_2, t_3, t_4, t_5 \xleftarrow{\$} \mathbb{Z}_q$
		$\mathbf{z}_1 \leftarrow H_2(\mathrm{rnd})$
		$\alpha \leftarrow \mathbf{a} \cdot \mathbf{g}^{t_1} \cdot \mathbf{y}^{t_2}$
		$\zeta \leftarrow \mathbf{z}^\gamma$, $\zeta_1 \leftarrow \mathbf{z}_1^\gamma$, $\zeta_2 \leftarrow \zeta/\zeta_1$
		$\beta_1 \leftarrow \mathbf{b}_1^\gamma \cdot \mathbf{g}^{t_3} \cdot \zeta_1^{t_4}$
		$\beta_2 \leftarrow \mathbf{b}_2^\gamma \cdot \mathbf{h}^{t_5} \cdot \zeta_2^{t_4}$
		$\eta \leftarrow \mathbf{z}^\tau$
		$h \leftarrow H_3(\zeta, \zeta_1, \alpha, \beta_1, \beta_2, \eta, m, \mathsf{info})$
	$\xleftarrow{x}$	$e \leftarrow h - t_2 - t_4$
$c \leftarrow e - c$		
$s_1 \leftarrow v_1 - d \cdot \mathrm{dlog}_{\mathbf{g}}\, \mathbf{z}_1$		
$s_2 \leftarrow v_2 - d \cdot \mathrm{dlog}_{\mathbf{h}}\, \mathbf{z}_2$		
	$\xrightarrow{c, r, d, s_1, s_2}$	$\rho \leftarrow r + t_1$, $\omega \leftarrow c + t_2$
		$\sigma_1 \leftarrow \gamma \cdot s_1 + t_3$
		$\sigma_2 \leftarrow \gamma \cdot s_2 + t_5$
		$\delta \leftarrow d + t_4$
		$\mu \leftarrow \tau - \delta \cdot \gamma$
		$\delta + \omega \stackrel{?}{=} H_3(\zeta, \zeta_1, \mathbf{g}^\rho \mathbf{y}^\omega, \mathbf{g}^{\sigma_1} \zeta_1^\delta, \mathbf{h}^{\sigma_2} \zeta_2^\delta, \mathbf{z}^\mu \zeta^\delta, m, \mathsf{info})$
		$\Downarrow$
		$(m, (\zeta, \zeta_1, \rho, \omega, \sigma_1, \sigma_2, \delta, \mu))$

Figure 5.2: How to sign using the $\mathbf{z}$-side witnesses in BSA. We will describe how the reduction can obtain the necessary discrete logarithms in the games where it needs them.

5.3.2 Useful Lemmata

Before we turn to the proof of one-more unforgeability, we want to prove some lemmata that will come in handy in the following.

The first is the 'template proof' claim adapted from [KLR23a, Definition D.1 and Claim D.1].

Lemma 5.3.1 (Template Proof Lemma). *Let $\mathcal{X}\colon \mathbb{Z}_q \to \mathbb{Z}_q$ and $\mathcal{Y}\colon \mathbb{Z}_q \to \mathbb{Z}_q$ be two functions defined as*

$$\mathcal{X}(\Omega) := \mathcal{C}_0 + \mathcal{C}_1 \cdot \Omega$$

and

$$\mathcal{Y}(\Delta) := \frac{\mathcal{C}_2 + \mathcal{C}_4 \cdot \Delta}{\mathcal{C}_3 + \mathcal{C}_5 \cdot \Delta}$$

such that $\mathcal{C}_4/\mathcal{C}_3 \neq \mathcal{C}_1$ or $\mathcal{C}_5 \neq 0$.

Let $\xi' \xleftarrow{\$} \mathbb{Z}_q$ be a value sampled uniformly at random after the values $\mathcal{C}_{1,\dots,5}$ are fixed. It then holds for any value $\xi \in \mathbb{Z}_q$ that

$$\Pr_{\substack{\Omega,\Delta \xleftarrow{\$} \mathbb{Z}_q \\ s.t.\ \Omega+\Delta=\xi}} [\mathcal{X}(\Omega) + \mathcal{Y}(\Delta) = \xi'] \leq \frac{5}{q}.$$

Proof. We distinguish two cases depending on which one of the values Ω and Δ is chosen uniformly at random. However, since these cases are analogous, we only prove the claim assuming that Ω is random. By putting the definitions of $\mathcal{X}$ and $\mathcal{Y}$, we get

$$\mathcal{C}_0 + \mathcal{C}_1 \cdot \Omega + \frac{\mathcal{C}_2 + \mathcal{C}_4 \cdot \Delta}{\mathcal{C}_3 + \mathcal{C}_5 \cdot \Delta} = \xi'.$$

As $\Delta = \xi - \Omega$, we have

$$\mathcal{C}_0 + \mathcal{C}_1 \cdot \Omega + \frac{\mathcal{C}_2 + \mathcal{C}_4 \cdot (\xi - \Omega)}{\mathcal{C}_3 + \mathcal{C}_5 \cdot (\xi - \Omega)} = \xi'.$$

We make the following case distinction

1. $\mathcal{C}_1 = 0$. In this case, we obtain that the event $\mathcal{X}(\Omega)+\mathcal{Y}(\Delta) = \xi'$ occurs iff $(\mathcal{C}_0\mathcal{C}_5 + \mathcal{C}_4 - \xi'\mathcal{C}_5)\cdot\Delta = \xi'\mathcal{C}_3 - \mathcal{C}_2 - \mathcal{C}_0\mathcal{C}_3$. As $\mathcal{C}_{0,\dots,5}$ are fixed before ξ' is sampled, the probability for the righthand side becoming 0 is $\frac{1}{q}$. Therefore, with probability $1 - \frac{1}{q}$, this linear equation has exactly one solution. As Δ gets sampled uniformly at random after ξ' is sampled, the probability of $\mathcal{X}(\Omega) + \mathcal{Y}(\Delta) = \xi'$ in this case is $\leq \frac{2}{q}$. We therefore assume in the following that $\mathcal{C}_1 \neq 0$.

2. $\mathcal{C}_5 = 0$. In this case, we can simplify the event to the linear equation $(\mathcal{C}_1 - \mathcal{C}_4/\mathcal{C}_3)\Omega + \mathcal{C}_0 + \mathcal{C}_2/\mathcal{C}_3 + \mathcal{C}_4/\mathcal{C}_3 \cdot \xi = \xi'$. This linear equation has one solution for Ω after ξ, ξ' have been fixed as $\mathcal{C}_5 = 0$ implies $\mathcal{C}_4/\mathcal{C}_3 \neq \mathcal{C}_1$ As Ω is sampled after ξ' and ξ have been fixed, the probability of the event $\mathcal{X}(\Omega) + \mathcal{Y}(\Delta) = \xi'$ occurring in this case is $\frac{1}{q}$.

3. For the last case, we know $\mathcal{C}_1 \neq 0$ and $\mathcal{C}_5 \neq 0$. Thus By rearranging this equation, we get

$$\mathcal{V}_1 \cdot \Omega^2 + \mathcal{V}_2 \cdot \Omega + \mathcal{V}_3 = 0,$$

where $\mathcal{V}_1 = -\mathcal{C}_1 \cdot \mathcal{C}_5, \mathcal{V}_2 = \mathcal{C}_1 \cdot (\mathcal{C}_3 + \mathcal{C}_5 \cdot \xi) - \mathcal{C}_4 - \mathcal{C}_5 \cdot (\mathcal{C}_0 - \xi'),\ \mathcal{V}_3 = (\mathcal{C}_0 - \xi') \cdot (\mathcal{C}_3 + \mathcal{C}_5 \cdot \xi) + \mathcal{C}_2 + \mathcal{C}_4 \cdot \xi$. This quadratic equation has at most two solutions for Ω. In particular, it must hold that either

$$\Omega = \frac{-\mathcal{V}_2 + \sqrt{\mathcal{V}_2^2 - 4 \cdot \mathcal{V}_1 \cdot \mathcal{V}_3}}{2 \cdot \mathcal{V}_1}, \tag{5.1}$$

or

$$\Omega = \frac{-\mathcal{V}_2 - \sqrt{\mathcal{V}_2^2 - 4 \cdot \mathcal{V}_1 \cdot \mathcal{V}_3}}{2 \cdot \mathcal{V}_1}. \tag{5.2}$$

Thus, if $\mathcal{X}(\Omega) + \mathcal{Y}(\Delta) = \xi'$ occurs, the values $\mathcal{C}_i$ for $i \in \{0, \dots, 5\}$ ξ, and ξ' must be chosen, such that at least one of the equations (5.1) and (5.2) holds. However, the value $\mathcal{C}_0, \dots, \mathcal{C}_5$, ξ, and ξ' are fixed and cannot be influenced after Ω gets fixed, hence satisfying these equations is equivalent to guessing the value of Ω in two attempts, which is achievable with probability at most $\frac{2}{q}$, because Ω is chosen uniformly at random in $\mathbb{Z}_q$.

In summary, this yields that $\mathcal{X}(\Omega) + \mathcal{Y}(\Delta) = \xi'$ occurs with probability at most $\frac{5}{q}$ □

Additionally, we introduce a second 'template proof' style lemma. This will be useful when arguing about hidden algebraic decompositions of group elements.

Lemma 5.3.2 (Hidden Decomposition Lemma)**.** *Let $0 \neq c_1, c_2 \in \mathbb{Z}_q$. Fix linear polynomials L_1, L_2 over $\mathbb{Z}_q$ in two variables $\mathfrak{W}_1, \mathfrak{W}_2$. It holds that either*

1. $\mathfrak{W}_1 = L_1(\mathfrak{W}_1, \mathfrak{W}_2) \mod (c_1 \cdot \mathfrak{W}_1 + \mathfrak{W}_2 - c_2)$
   or $\mathfrak{W}_2 = L_2(\mathfrak{W}_1, \mathfrak{W}_2) \mod (c_1 \cdot \mathfrak{W}_1 + \mathfrak{W}_2 - c_2)$ *(as polynomials).*
2. *or*
$$\Pr_{\substack{\mathfrak{w}_1, \mathfrak{w}_2 \xleftarrow{\$} \mathbb{Z}_q \\ c_1 \mathfrak{w}_1 + \mathfrak{w}_2 = c_2}} [\mathfrak{w}_1 = L(\mathfrak{w}_1, \mathfrak{w}_2) \vee \mathfrak{w}_2 = L(\mathfrak{w}_1, \mathfrak{w}_2)] \leq \frac{2}{q}$$

Proof. Assume case 1 does not occur (otherwise we're done). We want to apply the Schwartz-Zippel Lemma (see Lemma 2.6.6). We show this for $\mathfrak{W}_1$, the other case is symmetrical. To this end, we replace $\mathfrak{W}_2$ by $c_2 - c_1 \cdot \mathfrak{W}_1$ in L_1, name this polynomial L_1' (L_1' is a polynomial only in $\mathfrak{W}_1$). As we assumed case 1 does not occur, it how holds that still $\mathfrak{W}_1 \neq L_1(\mathfrak{W}_1)$ as polynomials. It holds that

$$\Pr_{\mathfrak{w}_1 \xleftarrow{\$} \mathbb{Z}_q} [\mathfrak{w}_1 = L_1(\mathfrak{w}_1)] = \Pr_{\substack{\mathfrak{w}_1, \mathfrak{w}_2 \xleftarrow{\$} \mathbb{Z}_q \\ c_1 \mathfrak{w}_1 + \mathfrak{w}_2 = c_2}} [\mathfrak{w}_1 = L(\mathfrak{w}_1, \mathfrak{w}_2)]$$

and by Lemma 2.6.6 it further holds that

$$\Pr_{\mathfrak{w}_1 \xleftarrow{\$} \mathbb{Z}_q} [\mathfrak{w}_1 = L_1(\mathfrak{w}_1)] = \frac{1}{q}.$$

A union bound yields case 2. □

5.3.3 The Restrictive Blinding Lemma

We first provide a reduction for the restrictive blinding lemma in the AGM + ROM. We therefore define the game ℓ-$\mathbf{RB\text{-}OMUF}_{\mathsf{BSA}}$ as follows:

Setup: Sample keys via $(\mathsf{sk} = x, \mathsf{pk} = (\mathbf{g}, \mathbf{h}, \mathbf{y})) \xleftarrow{\$} \mathsf{BSA.KeyGen}(\mathsf{pp})$.

Online Phase: M is given access to oracles $\mathsf{sign}_1, \mathsf{sign}_2$ that emulate the behavior of the honest signer in BSA. It is allowed to arbitrarily many calls to sign_1 and allowed to make ℓ queries to sign_2. In addition, it is given access to random oracles H_1, H_2, H_3. Let ℓ_{info} denote the number of interactions that M completes with oracle sign_2 in this phase for each tag info.

Output Determination: When M outputs a list L of tuples $(m_1, \mathsf{sig}_1, \mathsf{info}_1), \ldots, (m_k, \mathsf{sig}_k, \mathsf{info}_k)$, proceed as follows:

- If the list contains a tuple $(m, \mathsf{sig}, \mathsf{info})$ s.t. $\mathsf{Verify}(\mathsf{pk}, m, \mathsf{sig}, \mathsf{info}) = 0$, or does not contain $\ell_{\overline{\mathsf{info}}} + 1$ pairwise-distinct tuples for some tag $\overline{\mathsf{info}}$, return 0.
- Let $\mathbf{z}_j, \mathbf{z}_{1,j}$ denote the values of $\mathbf{z}$ and $\mathbf{z}_1$ used in the j-th invocation of sign_1. If there exists $(m, \mathsf{sig}, \overline{\mathsf{info}}) \in L$ with signature components $\zeta \neq \zeta_1$ (equivalently, $\zeta_2 \neq \epsilon$), s.t. for all j with $H_1(\mathsf{pk}, \overline{\mathsf{info}}) = \mathbf{z}_j$ whose sessions were closed with an invocation of sign_2, $\zeta^{\mathrm{dlog}_{\mathbf{z}_j} \mathbf{z}_{1,j}} \neq \zeta_1$, then return 1. Otherwise, return 0. We call the first signature in L with these mismatched linking components the *special signature*.

Define $\mathrm{Adv}^{\mathbf{RB\text{-}OMUF}}_{\mathsf{M},\ell,\mathsf{BSA}} \leftarrow \Pr[\ell\text{-}\mathbf{RB\text{-}OMUF}^{\mathsf{M}}_{\mathsf{BSA}} = 1]$. We show that an algebraic forger M that wins ℓ-$\mathbf{RB\text{-}OMUF}_{\mathsf{BSA}}$ can be used to solve the discrete logarithm problem. This reduction is tight and does not require rewinding of the adversary.

Lemma 5.3.3 (Restrictive Blinding, see Lemma 3 in [Abe01])**.** *Let* M *be an algebraic algorithm that runs in time t_{M}, makes at most ℓ queries to oracle* Sign_2 *in* $\mathbf{RB\text{-}OMUF}_{\mathsf{BSA}}$ *and at most (total) Q_h queries to H_1, H_2, H_3. Then, in the random oracle model, there exists an algorithm* B *s.t.*

$$\begin{aligned}\mathrm{Adv}^{\mathbf{DLOG}}_{\mathsf{B}} \geq \left(1 - \frac{4}{q}\right)\frac{1}{2}\mathrm{Adv}^{\mathbf{RB\text{-}OMUF}}_{\mathsf{M},\ell,\mathsf{BSA}} - \frac{\ell+1}{q} \\ - \mathrm{Adv}^{\mathbf{DLOG}}_{\mathsf{R}_1} - \mathrm{Adv}^{\mathbf{DLOG}}_{\mathsf{R}_2} + \mathrm{Adv}^{\mathbf{DLOG}}_{\mathsf{R}_3} - \frac{15Q_h + 20}{q}\end{aligned}$$

Proof. Let M be as in the lemma statement. As before, we assume w.l.o.g. that M makes exactly ℓ queries to Sign_2 and outputs a list of $\ell + 1$ tuples. The proof goes by a series of games, which we describe below.

Game 0. This is ℓ-$\mathbf{RB\text{-}OMUF}_{\mathsf{BSA}}$.

Game 1. To define Game 1, we first define the following event E_1. E_1 happens if M returns a list L of $\ell + 1$ valid signatures on distinct messages $m_1, ..., m_\ell$ and there exists $(m, \mathsf{sig}, \mathsf{info}) = (m, (\zeta, \zeta_1, \rho, \omega, \sigma_1, \sigma_2, \delta, \mu), \mathsf{info}) \in L$ s.t. for all j whose sessions were closed with an invocation of sign_2, $\zeta^{\mathrm{dlog}_{\mathbf{z}_j} \mathbf{z}_{1,j}} \neq \zeta_1$ and M did not make a query of the form $H_3(\zeta, \zeta_1, \mathbf{g}^\rho \mathbf{y}^\omega, \mathbf{g}^{\sigma_1} \zeta_1^\delta, \mathbf{h}^{\sigma_2} \zeta_2^\delta, \mathbf{z}^\mu \zeta^\delta, m, \mathsf{info})$. In the following, we refer to the first tuple $(m, \mathsf{sig}, \mathsf{info}) \in L$ as *the special tuple* for convenience. Game 1 is identical to Game 0, except that it aborts when E_1 happens.

Claim 5.3.4. $\Pr[E_1] = \frac{\ell+1}{q}$

Proof. The only way for an adversary to succeed without querying H_3 for the signature is by guessing the hash value $h = \omega + \delta$. Since there are $\ell + 1$ valid signatures in L, the probability of guessing h correctly for one of them is $\frac{\ell+1}{q}$. □

By the claim, we have that $\mathrm{Adv}^{\mathsf{Game\ 1}}_{\mathsf{M}} \geq \mathrm{Adv}^{\mathsf{Game\ 0}}_{\mathsf{M}} - \frac{\ell+1}{q}$.

Game 2. Game 2 is identical to Game 1, except that it keeps track of the algebraic representations of group elements submitted to H_3 by M and aborts if a certain condition applies that, looking forward, will make it impossible for the reduction to extract the discrete logarithm.

Simplifying Notations. For each query to H_3, the adversary M submits a set of group elements $\zeta, \zeta_1, \alpha, \beta_1, \beta_2, \eta$ along with a message m and info.

As M is algebraic, it also provides a representation of these group elements to the basis of elements $\mathbf{g}, \mathbf{h}, \mathbf{y}, \overrightarrow{\mathbf{z}}, \overrightarrow{\mathbf{a}}, \overrightarrow{\mathbf{b}_1}, \overrightarrow{\mathbf{b}_2}, \overrightarrow{\mathbf{z}_1}$ that it has previously obtained via calls to H_1, H_2, sign$_1$, or sign$_2$. We note that by programming the oracles H_1 and H_2 the game (as well as the reduction later on) knows a representation of its responses $\mathbf{z}_i$ and $\mathbf{z}_{1,i}$ to the base $\mathbf{g}$ and $\mathbf{h}$. Any element $\mathbf{a}, \mathbf{b}_1, \mathbf{b}_2$ that was returned as reply to a query to sign$_1$ can be represented as $\mathbf{a} = \mathbf{y}^c \cdot \mathbf{g}^r, \mathbf{b}_1 = \mathbf{z}_1^d \cdot \mathbf{g}^{s_1}, \mathbf{b}_2 = \mathbf{z}_2^d \cdot \mathbf{h}^{s_2}$. For elements $\mathbf{a}, \mathbf{b}_1, \mathbf{b}_1$ coming from sessions that are never closed, the game closes the session on its own in the end by sampling a random value $e \xleftarrow{\$} \mathbb{Z}_q$ and defining c and d accordingly.

Here, $\mathbf{z}_1, \mathbf{z}_2 = \mathbf{z}/\mathbf{z}_1$ correspond to the call $H_2(\mathrm{rnd})$ made as part of answering this query to sign$_1$. This allows us to convert any representation provided by M into a *reduced representation* in the (simpler) basis $\mathbf{g}, \mathbf{h}, \mathbf{y}$. For a group element $\mathbf{o}$, we denote this reduced representation by $[\mathbf{o}]_{\vec{I}}$ and its components as $g_{[\mathbf{o}]_{\vec{I}}}, h_{[\mathbf{o}]_{\vec{I}}}, y_{[\mathbf{o}]_{\vec{I}}}$, respectively, where $\vec{I} \leftarrow (\mathbf{g}, \mathbf{h}, \mathbf{y})$. If M wins, we define the following values for each message tag signature tuple $(m, \text{info}, (\zeta, \zeta_1, \rho, \omega, \sigma_1, \sigma_2, \delta, \mu))$ in the output of M where the algebraic representations are those submitted when the first hash query to H_3 corresponding to this message tag signature tuple was made.:

$$\begin{aligned}
\omega' &:= y_{[\alpha]_{\vec{I}}} \\
\delta' &:= \frac{h_{[\beta_1]_{\vec{I}}}}{h_{[\zeta_1]_{\vec{I}}}} \\
\delta'' &:= \frac{g_{[\beta_2]_{\vec{I}}} + x \cdot y_{[\beta_2]_{\vec{I}}}}{x \cdot y_{[\zeta_2]_{\vec{I}}} + g_{[\zeta_2]_{\vec{I}}}} \\
\delta''' &:= \begin{cases} \frac{g_{[\eta]_{\vec{I}}} + x \cdot y_{[\eta]_{\vec{I}}} - h_{[\eta]_{\vec{I}}} \cdot w_{0,i,g}/w_{0,i,h}}{g_{[\zeta]_{\vec{I}}} + x \cdot y_{[\zeta]_{\vec{I}}} - h_{[\zeta]_{\vec{I}}} \cdot w_{0,i,g}/w_{0,i,h}} & \text{if } w_{0,i,h} \neq 0 \\ \frac{h_{[\eta]_{\vec{I}}}}{h_{[\zeta]_{\vec{I}}}} & \text{if } w_{0,i,h} = 0 \end{cases}.
\end{aligned}$$

Looking forward, the 'preliminary values' will be used like values from a second signature would be used in a forking based proof. That is, if there is a signature with the value $\omega' \neq \omega$, a reduction will be able to efficiently compute the discrete logarithm of $\mathbf{y}$ and if one of the $\delta', \delta'', \delta''' \neq \delta$, the reduction will be able to compute the discrete logarithm of the $\mathbf{z}$-side witness $\mathbf{h}$.

However, we first note that it is not obvious to see that the values $\delta', \delta'', \delta'''$ are defined. In fact, an earlier version of this proof (the one published in [KLX22a]) had to make the additional requirement here that $\zeta_2 \neq \epsilon$ where ϵ is the neutral element of the group $\mathbb{G}$. We avoid this by sampling the values for $\mathbf{z}, \mathbf{z}_1$ slightly different, but it means that the proof that the preliminary values for δ are defined is a bit more involved.

We define the following non-exclusive boolean variables that tell us which of the $\delta', \delta'', \delta'''$ is defined.

$$\begin{aligned}
C_1 &\leftarrow h_{[\zeta_1]_{\vec{I}}} \neq 0 \\
C_2 &\leftarrow g_{[\zeta_2]_{\vec{I}}} + x \cdot y_{[\zeta_2]_{\vec{I}}} \neq 0 \\
C_3 &\leftarrow \nexists \gamma \in \mathbb{Z}_q \colon g_{[\zeta]_{\vec{I}}} + x \cdot y_{[\zeta]_{\vec{I}}} = \gamma \cdot w_{0,i,g} \wedge h_{[\zeta]_{\vec{I}}} = \gamma \cdot w_{0,i,h}
\end{aligned}$$

Claim 5.3.5.

$$\Pr_{\substack{w_{0,i,g}, w_{0,i,h} \xleftarrow{\$} \mathbb{Z}_q \\ w_{0,i,g} + \mathrm{dlog}_{\mathbf{g}} \mathbf{h} \cdot w_{0,i,h} = \mathrm{dlog}_{\mathbf{g}} \mathbf{z}_i}} \left[\bigvee_i C_i = 1 \right] \geq 1 - \frac{4}{q}.$$

Proof. First of all, we note that the case $\zeta = \zeta_1 = \epsilon$ is excluded by the verification algorithm of the scheme. Thus, we can rule out the case that $h_{[\zeta_1]_{\vec{I}}} = 0 \wedge g_{[\zeta_2]_{\vec{I}}} + x \cdot y_{[\zeta_2]_{\vec{I}}} = 0 \wedge \gamma = 0$. We next consider the case that $\exists \gamma \in \mathbb{Z}_q \backslash \colon g_{[\zeta]_{\vec{I}}} + x \cdot y_{[\zeta]_{\vec{I}}} = \gamma \cdot w_{0,i,g} \wedge h_{[\zeta]_{\vec{I}}} = \gamma \cdot w_{0,i,h}$. If this is not the case we are done as $C_3 = 1$.

We distinguish two cases. First, if $\gamma = 0$, then $\zeta = \epsilon$, but as we require then $\zeta_1 \neq \epsilon$, it must hold that either $h_{[\zeta_1]_{\vec{I}}} \neq 0$ or $g_{[\zeta_2]_{\vec{I}}} + x \cdot y_{[\zeta_2]_{\vec{I}}} = -\left(g_{[\zeta_1]_{\vec{I}}} + x \cdot y_{[\zeta_1]_{\vec{I}}} \right) \neq 0$ (recall that the representation of ζ_2 is computed as the difference of the representations of ζ and ζ_1).

The second case is $\gamma \neq 0$. In this case, we want to apply Lemma 5.3.2. We again imagine that the game samples the values $w_{0,j,h}$ and $w_{0,i,g}$ at the end of the game such that they match the used element $\mathbf{z}_i$. In particular, we set $\mathfrak{W}_1$ to be a formal variable representing $w_{0,i,h}$ and $\mathfrak{W}_2$ to be $w_{0,i,g}$, and $c_1 = \mathrm{dlog}_{\mathbf{g}} \mathbf{h}$, and $c_2 = \mathrm{dlog}_{\mathbf{g}} \mathbf{z}_i$. In order for $g_{[\zeta_2]_{\vec{I}}} + x \cdot y_{[\zeta_2]_{\vec{I}}} = 0$ to occur, it must hold that $g_{[\zeta_1]_{\vec{I}}} + x \cdot y_{[\zeta_1]_{\vec{I}}} = g_{[\zeta]_{\vec{I}}} + x \cdot y_{[\zeta]_{\vec{I}}} = \gamma \cdot w_{0,i,g}$. We thus set $L_1 = g_{[\zeta_1]_{\vec{I}}} + x \cdot y_{[\zeta_1]_{\vec{I}}} / \gamma$ where we interpret $g_{[\zeta_1]_{\vec{I}}} + x \cdot y_{[\zeta_1]_{\vec{I}}}$ as a polynomial in $w_{0,i,g}$. Applying Lemma 5.3.2 yields that either $g_{[\zeta_1]_{\vec{I}}} + x \cdot y_{[\zeta_1]_{\vec{I}}}$ is of the form $w_{0,i,g} \cdot \gamma$, i.e. the representation of ζ_1 contains $\mathbf{z}_i^{\gamma}$ and then only $\mathbf{h}$-components. Applying the same lemma again using $L_2 = \frac{h_{[\zeta_1]_{\vec{I}}} - \gamma \cdot \mathfrak{W}_2}{\gamma}$ yields that again, either $h_{[\zeta_1]_{\vec{I}}} = 2 \cdot \gamma \cdot \mathfrak{W}_2$ which would imply that actually ζ_1 contains $\mathbf{z}_i^{2\gamma}$, a contradiction to what we already found out above, or that the probability of 'compensating' for the $\mathbf{h}$-component of $\mathbf{z}_i$ using other group elements is at most $\frac{2}{q}$. $\square$

Game 2 aborts if none of the values $\delta', \delta'', \delta'''$ is defined. According to Claim 5.3.5, this happens with probability at most $\frac{4}{q}$.

Game 3. This game aborts if there exists a signature among the $\ell + 1$ forgeries where the preliminary value δ''' is defined and $\delta''' = \delta$.

Claim 5.3.6. *If there exists a signature for which δ''' is defined, then there exists a reduction* R_1 *such that.*

$$\Pr\left[\delta''' = \delta\right] \leq \mathrm{Adv}_{\mathsf{R}_1}^{\mathbf{DLOG}} + \frac{5Q_h}{q}$$

Proof. We want to apply Lemma 5.3.1 to ω' and δ'''. To this end, we need to define the values $\mathcal{C}_1, \ldots, \mathcal{C}_5$ of Lemma 5.3.1 and show that either $\mathcal{C}_4/\mathcal{C}_3 \neq \mathcal{C}_1$ or $\mathcal{C}_5 \neq 0$.

Let sid* denote the number of the last[4] session that is open at time of the hash query and eventually closed such that $\mathbf{a}_{\mathsf{sid}^*}$, $\mathbf{b}_{1,\mathsf{sid}^*}$ or $\mathbf{b}_{2,\mathsf{sid}^*}$ appear with a non-zero coefficient in the representations of $\alpha, \beta_1, \beta_2, \eta, \zeta, \zeta_1$. For a group element $\mathbf{o}$, we denote by $\overline{a}(\mathbf{o})$ the coefficient of $\mathbf{a}_{\mathsf{sid}^*}$ in the representation of $\mathbf{o}$, by $\overline{b_1}(\mathbf{o})$ the coefficient of $\mathbf{b}_{1,\mathsf{sid}^*}$ and by $\overline{b_2}(\mathbf{o})$ the coefficient of $\mathbf{b}_{2,\mathsf{sid}^*}$. We set $\mathcal{X}(\Omega) \leftarrow \omega'$, that is $\Omega = c_{\mathsf{sid}^*}$ and $\mathcal{C}_0 \leftarrow y_{[\alpha]_{\vec{I}}} - \overline{a}(\alpha) \cdot c_{\mathsf{sid}^*}$ and $\mathcal{C}_1 \leftarrow \overline{a}(\alpha)$ where by $\overline{a}(\cdot)$ we denote exponent of $\mathbf{a}_{\mathsf{sid}^*}$ in the representation of the group element in brackets. For δ''' we set

$$\mathcal{C}_2 := h_{[\eta]_{\vec{I}}} - \overline{b_1}(\eta) \cdot d_{\mathsf{sid}^*} \cdot w_{1,\mathsf{sid}^*,h} + \overline{b_2}(\eta) \, \mathrm{dlog}_{\mathbf{h}}(\mathbf{b_1})$$

[4]by closing time

and

$$\mathcal{C}_3 := \overline{b_1}(\eta) \cdot w_{1,\mathsf{sid}^*,h} - \operatorname{dlog}_{\mathbf{h}}(\mathbf{g})\overline{b_2}(\eta)(w_{0,\mathsf{sid}^*,g} - w_{1,\mathsf{sid}^*,g})$$

and

$$\mathcal{C}_4 := h_{[\zeta]_{\overrightarrow{I}}} - \overline{b_1}(\eta) \cdot d_{\mathsf{sid}^*} \cdot w_{1,\mathsf{sid}^*,h} + \overline{b_2}(\zeta) \operatorname{dlog}_{\mathbf{h}}(\mathbf{b_1})$$

and

$$\mathcal{C}_5 := \overline{b_1}(\zeta) \cdot w_{1,\mathsf{sid}^*,h} - \operatorname{dlog}_{\mathbf{h}}(\mathbf{g})\overline{b_2}(\zeta)(w_{0,\mathsf{sid}^*,g} - w_{1,\mathsf{sid}^*,g})$$

in the case that $w_{0,\mathsf{sid}^*,g} = 0$. We note that if $\overline{b_1}(\zeta) = 0$ and $\overline{b_2}(\zeta) = 0$, $\mathcal{C}_5 = 0$ and $\mathcal{C}_4 = h_{[\zeta_2]_{\overrightarrow{I}}}$. and otherwise

$$\mathcal{C}_2 := g_{[\eta]_{\overrightarrow{I}}} + x \cdot y_{[\eta]_{\overrightarrow{I}}} - d \cdot \overline{b_2}(\eta) \cdot (w_{0,\mathsf{sid}^*,\mathbf{g}} - w_{1,\mathsf{sid}^*,\mathbf{g}}) + \overline{b_1}(\eta) \operatorname{dlog}_{\mathbf{g}}(\mathbf{b_1}) \\ + \frac{w_{0,\mathsf{sid}^*,g}}{w_{0,\mathsf{sid}^*,h}} \cdot \left(h_{[\eta]_{\overrightarrow{I}}} - \overline{b_1}(\eta) \cdot d_{\mathsf{sid}^*} \cdot w_{1,\mathsf{sid}^*,h} + \overline{b_2}(\eta) \operatorname{dlog}_{\mathbf{h}}(\mathbf{b_1})\right)$$

and

$$\mathcal{C}_4 := \overline{b_1}(\eta) \cdot w_{1,\mathsf{sid}^*,h} - \operatorname{dlog}_{\mathbf{h}}(\mathbf{g})\overline{b_2}(\eta)(w_{0,\mathsf{sid}^*,g} - w_{1,\mathsf{sid}^*,g}) \\ + \frac{w_{0,\mathsf{sid}^*,g}}{w_{0,\mathsf{sid}^*,h}} \left(\overline{b_2}(\eta)\left(w_{0,\mathsf{sid}^*,g} - w_{1,\mathsf{sid}^*,g}\right) - \overline{b_1}(\eta) \operatorname{dlog}_{\mathbf{g}} \mathbf{h} \cdot w_{1,\mathsf{sid}^*,h}\right)$$

and

$$\mathcal{C}_3 := g_{[\zeta]_{\overrightarrow{I}}} + x \cdot y_{[\zeta]_{\overrightarrow{I}}} - d \cdot \overline{b_2}(\zeta) \cdot (w_{0,\mathsf{sid}^*,\mathbf{g}} - w_{1,\mathsf{sid}^*,\mathbf{g}}) + \overline{b_1}(\zeta) \operatorname{dlog}_{\mathbf{g}}(\mathbf{b_1}) \\ + \frac{w_{0,\mathsf{sid}^*,g}}{w_{0,\mathsf{sid}^*,h}} \cdot \left(h_{[\zeta]_{\overrightarrow{I}}} - \overline{b_1}(\zeta) \cdot d_{\mathsf{sid}^*} \cdot w_{1,\mathsf{sid}^*,h} + \overline{b_2}(\zeta) \operatorname{dlog}_{\mathbf{h}}(\mathbf{b_1})\right)$$

and

$$\mathcal{C}_5 := \overline{b_1}(\zeta) \cdot w_{1,\mathsf{sid}^*,h} - \operatorname{dlog}_{\mathbf{h}}(\mathbf{g})\overline{b_2}(\zeta)(w_{0,\mathsf{sid}^*,g} - w_{1,\mathsf{sid}^*,g}) \\ + \frac{w_{0,\mathsf{sid}^*,g}}{w_{0,\mathsf{sid}^*,h}} \left(\overline{b_2}(\zeta)\left(w_{0,\mathsf{sid}^*,g} - w_{1,\mathsf{sid}^*,g}\right) - \overline{b_1}(\zeta) \operatorname{dlog}_{\mathbf{g}} \mathbf{h} \cdot w_{1,\mathsf{sid}^*,h}\right)$$

We focus on the latter case that $\mathcal{C}_5 = 0$, otherwise we can directly apply Lemma 5.3.1 and be done.

We show that this can occur with probability at most $\mathrm{Adv}^{\mathbf{DLOG}}_{\mathsf{R}_1}$ for a reduction R_1 to the discrete logarithm problem that succeeds in this case. We provide the reduction in the following. The reduction embeds its discrete logarithm challenge in $\mathbf{h}$, samples a secret key value $x \xleftarrow{\$} \mathbb{Z}_q$ and simulates all oracles as Game 3 using the secret key x to simulate. If $\mathcal{C}_4/\mathcal{C}_3 = \mathcal{C}_1$, the reduction solves the equation $\mathcal{C}_4/\mathcal{C}_3 = \mathcal{C}_1$ for the $\operatorname{dlog}_{\mathbf{g}} \mathbf{h}$ expressions appearing in $\mathcal{C}_4$ (we note that $\operatorname{dlog}_{\mathbf{g}} \mathbf{h} = (\operatorname{dlog}_{\mathbf{h}} \mathbf{g})^{-1}$). We note that as $\mathcal{C}_5 = 0$, it actually holds that $\mathcal{C}_3 = g_{[\zeta]_{\overrightarrow{I}}} + x \cdot y_{[\zeta]_{\overrightarrow{I}}} + \frac{w_{0,\mathsf{sid}^*,g}}{w_{0,\mathsf{sid}^*,h}} \cdot h_{[\zeta]_{\overrightarrow{I}}}$.

If $\mathcal{C}_1 \neq \mathcal{C}_4/\mathcal{C}_3$ or $\mathcal{C}_5 \neq 0$, we apply Lemma 5.3.1 to obtain that $\Pr[\delta''' = \delta] \leq \frac{5}{q}$ for a single signature. Applying a union bound over all signatures yields the statement.

□

Corollary 5.3.7.

$$\Pr[\textit{Game 3} = 1] - \Pr[\textit{Game 2} = 1] \leq \mathrm{Adv}^{\mathbf{DLOG}}_{\mathsf{R}_1} + \frac{5}{q}$$

Proof. Follows directly from Claim 5.3.6. □

We furthermore prove the following claim:

Claim 5.3.8. *The probability that δ''' is undefined is $\frac{4}{q}$ if the representation of ζ is not of the form $\mathbf{z}_i^\gamma$.*

Proof. This follows from Lemma 5.3.2. To see this, we imagine the following game: Instead of fixing the values $w_{0,i,h}$ and $w_{0,i,g}$ at the beginning of the game like Game 3, the game samples $\mathrm{dlog}_{\mathbf{g}}\,\mathbf{z}_i \xleftarrow{\$} \mathbb{Z}_q$ at the beginning of the game and then later on, after the adversary's run, samples $w_{0,i,g} \xleftarrow{\$} \mathbb{Z}_q$ and sets $w_{0,i,h}$ accordingly. This game is identically distributed to Game 3, however we can apply Lemma 5.3.2 as follows. We set $c_1 = \mathrm{dlog}_{\mathbf{g}}\,\mathbf{h}$ and $c_2 = \mathrm{dlog}_{\mathbf{g}}\,\mathbf{z}_i$. We further replace all occurrences of $w_{0,i,h}$ in the representation of ζ by the formal variable $\mathfrak{W}_1$ and all occurrences of $w_{0,i,g}$ by the formal variable $\mathfrak{W}_2$. We set L_1 to be $h_{[\zeta]_{\overrightarrow{I}}}/\gamma$ where γ is the value from the definition of δ''', and where we interpret $h_{[\zeta]_{\overrightarrow{I}}}$ as a polynomial in $\mathfrak{W}_1$ (as we are considering the $\mathbf{h}$-component, $\mathfrak{W}_2$ does not appear.). Lemma 5.3.2 yields that either $h_{[\zeta]_{\overrightarrow{I}}}/\gamma$ must be of the form $\mathfrak{W}_1$, or the probability that $h_{[\zeta]_{\overrightarrow{I}}}/\gamma = w_{0,i,h}$ is at most $\frac{2}{q}$. Applying the same argument to $(g_{[\zeta]_{\overrightarrow{I}}} + x \cdot y_{[\zeta]_{\overrightarrow{I}}})/\gamma$ yields the claim. □

Remark 5.3.9 (On how to identify a 'special signature' if $\mathrm{dlog}_{\mathbf{g}}\,\mathbf{h}$ is unknown)**.** *Throughout this section, reductions may need to identify the special signature while they have embedded a discrete logarithm challenge in $\mathbf{h}$.*

This is however still possible by the above claim. For any signature, if the corresponding δ''' is undefined, there exists an value γ that can be computed without the knowledge of $\mathrm{dlog}_{\mathbf{g}}\,\mathbf{h}$ as $\gamma = h_{[\zeta]_{\overrightarrow{I}}}/h_{[\mathbf{z}_i]_{\overrightarrow{I}}} = g_{[\zeta]_{\overrightarrow{I}}}/g_{[z_i]_{\overrightarrow{I}}}$.

So , by Claim 5.3.8, the reduction can always either solve for $\mathrm{dlog}_{\mathbf{g}}\,\mathbf{h}$ or identify the special signature efficiently by computing γ and comparing the corresponding ζ_1 to the values $\mathbf{z}_1^\gamma$.

Game 4. Game 4 aborts in case the adversary 'predicted' the hash response to the random oracle, i.e. if the values ω and δ from all signatures are equal to the corresponding preliminary values. However, as some parts of the 'internal' representation may be influenced by the adversary closing signing sessions that were open at the time of the hash query, proving that this influence is infeasible is non-trivial.

We describe this as an event. We define E_2 as the following event: $\omega' = \omega$, and for any of $\delta', \delta'', \delta'''$, as long as its denominator is not 0 (i.e., it is well-defined), then it is equal to δ. That is,

$$\begin{aligned} E_2 :=& (\omega' = \omega) \wedge (C_1 = 0 \vee (C_1 = 1 \wedge (\delta' = \delta))) \\ & \wedge (C_2 = 0 \vee (C_2 = 1 \wedge (\delta'' = \delta))) \wedge (C_3 = 0 \vee (C_3 = 1 \wedge (\delta''' = \delta))). \end{aligned}$$

Claim 5.3.10.

$$\Pr[E_2] \leq \mathrm{Adv}^{\mathbf{DLOG}}_{\mathsf{R}_2} + \mathrm{Adv}^{\mathbf{DLOG}}_{\mathsf{R}_3} + \frac{10 \cdot Q_h + 20}{q}$$

Proof Strategy. We give a brief overview over the proof strategy. The main idea of this proof is that the adversary fixes most parts of the representation of the group elements at the time it makes the hash query to H_3. After the hash query is made, the only way to influence the representation is by the values c and d included in the internal representations of $\mathbf{a}$ and $\mathbf{b}_1, \mathbf{b}_2$, which can be influenced by the adversary only through the value e it sends to close a signing session. Thus, we need to consider how these c and d values of sessions open at time of the hash query can impact the representations.

In the end, we will want to employ Lemma 5.3.1 for each of the defined δ. To do this, we must define the values $\mathcal{C}_1, \ldots, \mathcal{C}_5$ and then show that for each defined δ either $\mathcal{C}_1 \neq \mathcal{C}_4/\mathcal{C}_3$ or $\mathcal{C}_5 \neq 0$.

This is a difference to our proof in [KLX22a], as we there used a manual case distinction over all possible representations that the adversary could submit to the hash oracle.

Proof. We want to apply Lemma 5.3.1. To this end, we need to define what the constants $\mathcal{C}_1, \ldots, \mathcal{C}_5$ are for each of the (potentially) defined δ. Let sid* be defined as before.

In the following $\overline{b_1}(\mathbf{o})$ denotes the exponent of $\mathbf{b}_{1,\mathsf{sid}^*}$ in the representation of $\mathbf{o}$ and $\overline{b_2}(\mathbf{o})$ denotes the exponent of $\mathbf{b}_{2,\mathsf{sid}^*}$ in the representation of $\mathbf{o}$ For all δ, we set $\Delta := d_{\mathsf{sid}^*}$.

For δ', we set $\mathcal{Y}(\Delta) := \delta'$ and

$$\mathcal{C}_2 := h_{[\beta_1]_{\vec{I}}} - \overline{b_1}(\beta_1) \cdot d_{\mathsf{sid}^*} \cdot w_{1,\mathsf{sid}^*,h} + \overline{b_2}(\beta_1)\,\mathrm{dlog}_{\mathbf{h}}(\mathbf{b_1})$$

and

$$\mathcal{C}_3 := \overline{b_1}(\beta_1) \cdot w_{1,\mathsf{sid}^*,h} - \mathrm{dlog}_{\mathbf{h}}(\mathbf{g})\overline{b_2}(\beta_1)(w_{0,\mathsf{sid}^*,g} - w_{1,\mathsf{sid}^*,g})$$

and

$$\mathcal{C}_4 := h_{[\zeta_1]_{\vec{I}}} - \overline{b_1}(\zeta_1) \cdot d_{\mathsf{sid}^*} \cdot w_{1,\mathsf{sid}^*,h} + \overline{b_2}(\zeta_1)\,\mathrm{dlog}_{\mathbf{h}}(\mathbf{b_1})$$

and

$$\mathcal{C}_5 := \overline{b_1}(\zeta_1) \cdot w_{1,\mathsf{sid}^*,h} - \mathrm{dlog}_{\mathbf{h}}(\mathbf{g})\overline{b_2}(\zeta_1)(w_{0,\mathsf{sid}^*,g} - w_{1,\mathsf{sid}^*,g}).$$

We note that if $\overline{b_1}(\zeta_1) = 0$ and $\overline{b_2}(\zeta_1) = 0$, $\mathcal{C}_5 = 0$ and $\mathcal{C}_4 = h_{[\zeta_1]_{\vec{I}}}$.

For δ'' we set

$$\mathcal{C}_2 := g_{[\beta_2]_{\vec{I}}} + x \cdot y_{[\beta_2]_{\vec{I}}} - d \cdot \overline{b_2}(\beta_2) \cdot (w_{0,\mathsf{sid}^*,\mathbf{g}} - w_{1,\mathsf{sid}^*,\mathbf{g}}) + \overline{b_1}(\beta_2)\,\mathrm{dlog}_{\mathbf{g}}(\mathbf{b_1})$$

and

$$\mathcal{C}_3 := \overline{b_2}(\beta_2)\,(w_{0,\mathsf{sid}^*,g} - w_{1,\mathsf{sid}^*,g}) - \overline{b_1}(\beta_2)\,\mathrm{dlog}_{\mathbf{g}}\,\mathbf{h} \cdot w_{1,\mathsf{sid}^*,h}$$

and

$$\mathcal{C}_4 := g_{[\zeta_2]_{\vec{I}}} + x \cdot y_{[\zeta_2]_{\vec{I}}} - d \cdot \overline{b_2}(\zeta_2) \cdot (w_{0,\mathsf{sid}^*,\mathbf{g}} - w_{1,\mathsf{sid}^*,\mathbf{g}}) + \overline{b_1}(\zeta_2)\,\mathrm{dlog}_{\mathbf{g}}(\mathbf{b_1})$$

and

$$\mathcal{C}_5 := \overline{b_2}(\zeta_2)\,(w_{0,\mathsf{sid}^*,g} - w_{1,\mathsf{sid}^*,g}) - \overline{b_1}(\zeta_2)\,\mathrm{dlog}_{\mathbf{g}}\,\mathbf{h} \cdot w_{1,\mathsf{sid}^*,h}.$$

We note that if $\overline{b_1}(\zeta_2) = 0$ and $\overline{b_2}(\zeta_2) = 0$, $\mathcal{C}_5 = 0$ and $\mathcal{C}_4 = g_{[\zeta_2]_{\vec{I}}} + x \cdot y_{[\zeta_2]_{\vec{I}}}$.

Remark 5.3.11. *We note that the values $\mathcal{C}_2, \ldots, \mathcal{C}_5$ may not always be efficiently computable without knowledge of $\mathrm{dlog}_{\mathbf{g}}\,\mathbf{h}$. Looking forward, this will allow some reductions to solve the discrete logarithm problem.*

In the following, we always assume that we are dealing with the so-called *special signature* which can be efficiently identified both by the game as well as by the reduction(s) we will provide due to Remark 5.3.9 and because we introduced an abort condition in Game 3 if δ''' is defined for any signature and $\delta''' = \delta$.

We now turn to the next case.

Case 1: δ' is defined and $\overline{b_2}(\beta_1) \neq 0$. We want to apply Lemma 5.3.1 to δ'. We again assume that $\mathcal{C}_5 = 0$ as otherwise we can apply Lemma 5.3.1 directly. In this case, we again provide a reduction solving for $\operatorname{dlog}_{\mathbf{g}} \mathbf{h}$. The reduction R_2 has the same strategy as in the previous case except that it considers the $\mathcal{C}$ values from δ'. The reduction embeds its discrete logarithm challenge in $\mathbf{h}$ and simulates using the secret key x to sign. It answers random oracle queries as described in Game 4. If the current case occurs, it uses the equation $\mathcal{C}_3/\mathcal{C}_4 = \mathcal{C}_1$ to solve for $\operatorname{dlog}_{\mathbf{h}} \mathbf{g}$. As we assumed that $\mathcal{C}_5 = 0$, it holds that $\mathcal{C}_4 = h_{[\zeta_1]_{\vec{\gamma}}}$ which the reduction can compute efficiently without knowledge of $\operatorname{dlog}_{\mathbf{g}} \mathbf{h}$. As $\operatorname{dlog}_{\mathbf{g}} \mathbf{h} = (\operatorname{dlog}_{\mathbf{h}} \mathbf{g})^{-1} \bmod q$, this allows the reduction to solve the discrete logarithm problem.

Case 2: δ'' is defined and $\overline{b_1}(\beta_2) \neq 0$. In this case we want to show that either Lemma 5.3.1 can be applied to δ'' or a reduction can solve for the discrete logarithm of $\mathbf{h}$. We describe how to deal with the latter case in the following: Analogously to the previous two cases we assume $\mathcal{C}_5 = 0$ and use a reduction R_3 to solve for $\operatorname{dlog}_{\mathbf{g}} \mathbf{h}$. The reduction can embed the discrete logarithm in the same way as in the previous case and solve for $\operatorname{dlog}_{\mathbf{g}} \mathbf{h}$.

Case 3: δ'' is undefined. In this case, we want to apply Lemma 5.3.1 to δ'. We want to show that in this case, $\mathcal{C}_4/\mathcal{C}_3 \neq \mathcal{C}_1$ for the $\mathcal{C}_3, \mathcal{C}_4$ derived from β_1 and ζ_1. This follows from Lemma 5.3.2 in the following way: First of all, we note that if δ'' is undefined, then $g_{[\zeta_1]_{\vec{\gamma}}} + x \cdot y_{[\zeta_1]_{\vec{\gamma}}} = g_{[\zeta]_{\vec{\gamma}}} + x \cdot y_{[\zeta]_{\vec{\gamma}}}$. On the other hand, we want to consider the case of $\mathcal{C}_4/\mathcal{C}_3 = \mathcal{C}_1$. As $\mathcal{C}_5 = 0$ and we already know δ' is defined, we know that $\mathcal{C}_3 \neq 0$. On the other hand, we know that $\mathcal{C}_4 = \overline{b_1}(\beta_1) \cdot w_{1,\mathsf{sid}^*,h}$, and thus we can set $L_2 = \overline{a}(\alpha) \cdot h_{[\zeta_1]_{\vec{\gamma}}}/\overline{b_1}(\beta_1)$ where we consider $h_{[\zeta_1]_{\vec{\gamma}}}$ to be a polynomial in $w_{1,\mathsf{sid}^*,h}$ as a formal variable $W_{1,\mathsf{sid}^*,h}$. We further replace all occurrences of $w_{1,\mathsf{sid}^*,g}$ with a formal variable $W_{1,\mathsf{sid}^*,g}$ Then, Lemma 5.3.2 with $\mathfrak{W}_2 = W_{1,\mathsf{sid}^*,h}$ and L_2 as described above, and $c_1 = \operatorname{dlog}_{\mathbf{g}} \mathbf{h}$ and $c_2 = \operatorname{dlog}_{\mathbf{g}} \mathbf{z}_{1,\mathsf{sid}^*}$, yields that either $L_2 = W_{1,\mathsf{sid}^*,h} \mod \operatorname{dlog}_{\mathbf{g}} \mathbf{h} \cdot W_{1,\mathsf{sid}^*,h} + W_{1,\mathsf{sid}^*,g} = \operatorname{dlog}_{\mathbf{g}} \mathbf{z}_{1,\mathsf{sid}^*}$ or $\Pr[w_{1,\mathsf{sid}^*,h} = L_2(w_{1,\mathsf{sid}^*,h})] \leq \frac{2}{q}$. As L_2 is a polynomial in $W_{1,\mathsf{sid}^*,h}$ only , it must hold that if $L_2 = W_{1,\mathsf{sid}^*,h} \mod \operatorname{dlog}_{\mathbf{g}} \mathbf{h} \cdot W_{1,\mathsf{sid}^*,h} + W_{1,\mathsf{sid}^*,g} = \operatorname{dlog}_{\mathbf{g}} \mathbf{z}_{1,\mathsf{sid}^*}$ already $L_2 = W_{1,\mathsf{sid}^*,h}$. Thus, the probability of $L_2 = w_{1,\mathsf{sid}^*,h}$ is $\frac{2}{q}$ (we also note that as the internal decomposition is perfectly hidden from the adversary at all times, we can actually pretend that the game samples the internal decompositions at the end, like in the proof of Claim 5.3.8). Thus, we in the following assume that $h_{[\zeta_1]_{\vec{\gamma}}}$ is of the form $\mathbf{z}_{1,\mathsf{sid}^*}^{\gamma_1}$ for a suitable value γ_1. We now want to consider the probability that at the same time, $g_{[\zeta_1]_{\vec{\gamma}}} + x \cdot y_{[\zeta_1]_{\vec{\gamma}}} = g_{[\zeta]_{\vec{\gamma}}} + x \cdot y_{[\zeta]_{\vec{\gamma}}}$. As we consider the case that δ''' is undefined, it holds that $g_{[\zeta]_{\vec{\gamma}}} + x \cdot y_{[\zeta]_{\vec{\gamma}}}$ is of the form $w_{0,\mathsf{sid}^*,g}$, but we also know that $g_{[\zeta_1]_{\vec{\gamma}}} + x \cdot y_{[\zeta_1]_{\vec{\gamma}}}$ contains a non-zero $\mathbf{z}_{1,\mathsf{sid}^*}$-component. Thus, applying Lemma 5.3.2 again, this time with $W_{0,\mathsf{sid}^*,g}$ as a replacement of $w_{0,\mathsf{sid}^*,g}$ and $L_2(W_{0,\mathsf{sid}^*,g}) = g_{[\zeta_1]_{\vec{\gamma}}} + x \cdot y_{[\zeta_1]_{\vec{\gamma}}}$, we obtain that again, the probability is $\frac{2}{q}$.

Case 4: δ' is undefined. This case is symmetrical to the previous case and can thus occur with the same probability.

Case 5: both δ' and δ'' are defined and none of the previous cases occurred. We show that, using Lemma 5.3.2, it is infeasible for the adversary to provoke the case that $\mathcal{C}_4/\mathcal{C}_3 = \mathcal{C}_1$ for both δ' and δ''. Using the same argument as for the two previous cases, we come to the conclusion that if $\mathcal{C}_4/\mathcal{C}_3 = \mathcal{C}_1$ for both, then $h_{[\zeta_1]_{\vec{\gamma}}}$ must be of the form $w_{1,\mathsf{sid}^*,h} \cdot \gamma_1$ whereas $g_{[\zeta_2]_{\vec{\gamma}}} + x \cdot y_{[\zeta_2]_{\vec{\gamma}}}$ must be of the form $(w_{0,i,g} - w_{1,\mathsf{sid}^*,g}) \cdot \gamma_2$. Thus, in particular, also $g_{[\zeta_1]_{\vec{\gamma}}}$ consists of $w_{0,i,g} \cdot \gamma_1$ and $h_{[\zeta_2]_{\vec{\gamma}}}$ consists of $(w_{0,\mathsf{sid}^*,h} - w_{1,\mathsf{sid}^*,h}) \cdot \gamma_2$. We note that since the representation of ζ_2 is computed from that of ζ_1 and ζ, an occurrence of $w_{1,\mathsf{sid}^*,g}$ or $w_{1,\mathsf{sid}^*,h}$ there can only come from ζ_1, and thus it must hold that $\gamma_1 = \gamma_2$.

Furthermore, since we assumed that the representation of ζ is of the form $\mathbf{z}_i^{\gamma}$ it must hold that $\mathbf{z}_i = \mathbf{z}_{\mathsf{sid}^*}$ or $g_{[\zeta_2]_{\overrightarrow{I}}} + x \cdot y_{[\zeta_2]_{\overrightarrow{I}}}$ will be of the form above only with probability $\frac{4}{q}$. In particular, any additional occurrence of $\mathbf{z}_{\mathsf{sid}^*}$ in the representation of ζ_1 to cancel out some of the $w_{0,\mathsf{sid}^*,g}$ in the representation of ζ_2 would lead to additional $\mathbf{h}$-components in the representation of ζ_1.

As δ''' is undefined, we know that the representation of ζ is of the form $\mathbf{z}_{\mathsf{sid}^*}$ by Claim 5.3.8. We consider the 'leftover' parts of the representations of ζ_1 and ζ_2, i.e. the parts of the representation that are not $\mathbf{z}_{1,\mathsf{sid}^*}^{\gamma_1}$ in ζ_1 and $\mathbf{z}_{2,\mathsf{sid}^*}^{\gamma_1}$ in ζ_2. We note that as we are looking at the special signature, these leftover components must exist as otherwise $\gamma = \gamma_1, = \gamma_2$ and $\zeta = \mathbf{z}_i^{\gamma}$ and $\zeta_1 = \mathbf{z}_{1,\mathsf{sid}^*}^{\gamma}$. We further note, that in the case of ζ_1, these components must amount to $(\gamma - \gamma_1) \cdot w_{0,\mathsf{sid}^*,g}$.

Thus, we can again apply Lemma 5.3.2 to the leftover component to find that the probability that $g_{[\zeta_1]_{\overrightarrow{I}}} + x \cdot y_{[\zeta_1]_{\overrightarrow{I}}} - \gamma_1 \cdot w_{1,\mathsf{sid}^*,g} = (\gamma - \gamma_1) \cdot w_{0,\mathsf{sid}^*,g}$ is $\frac{2}{q}$.

We are now ready to apply Lemma 5.3.1. Using the value δ', δ'' for which either $\mathcal{C}_5 \neq 0$ or $\mathcal{C}_4/\mathcal{C}_3 \neq \mathcal{C}_1$, we can obtain that in the remaining cases the probability of E_2 is at most $\frac{2 \cdot Q_h \cdot 5}{q}$ where the factor 2 comes from union bounding over the two δ', δ'' and the factor Q_h comes from union bounding over the Q_h queries made to H_3.

We note there that since Lemma 5.3.1 makes an information-theoretical argument, it is not necessary that the values $\mathcal{C}_1, \ldots, \mathcal{C}_5$ are efficiently computable for any reduction that may have embedded a discrete logarithm challenge somewhere.

We now union bound over all the cases and their different subcases.

$$\Pr[E_2] \leq \mathrm{Adv}_{\mathsf{R}_2}^{\mathbf{DLOG}} + \mathrm{Adv}_{\mathsf{R}_3}^{\mathbf{DLOG}} + \frac{6}{q} + \frac{6}{q} + \frac{8}{q} + \frac{5 \cdot 2 \cdot Q_h}{q}$$

□

In the following, we explain how the reduction can simulate Game 4 to the adversary M and win the discrete logarithm game.

Simulation of H_1, H_2, H_3. We begin by describing how S_0, S_1 simulate the random oracles H_1, H_2, H_3. These simulations are common to both S_ι and are performed in the straightforward way using lazy sampling. We assume that the oracles keep respective lists L_i for bookkeeping, where L_i stores input/output pairs. More specifically.

- H_1 and H_2: on each fresh input ξ, H_i samples $w_g, w_h \xleftarrow{\$} \mathbb{Z}_q$ and returns $\mathbf{g}^{w_g}\mathbf{h}^{w_h}$. It stores $(\xi, \mathbf{g}^{w_g}\mathbf{h}^{w_h}, w_g, w_h)$ in L_i.
- H_3 : on each fresh input ξ, H_3 samples $h \xleftarrow{\$} \mathbb{Z}_q$ and returns h. It stores $(\xi, \overrightarrow{\mathrm{rep}}, h)$ in L_i.
- On repeated inputs H_i returns whatever it returned the first time that ξ was queried.

Scheduling of Signing Sessions. We assume that each S_i internally schedules sessions with the oracles sign_1 and sign_2 as required by Game 4 . This can be easily implemented by using a fresh session identifier for each new session.

Extracting Equations from Forgery. Suppose that M wins Game 4. Recall that in this case, M produces a one-more forgery of at least $\ell + 1$ valid signatures, after having completed at most ℓ sessions with oracle sign_2. In addition, we have required that one of the returned tuples $(m, \mathsf{info}, \mathsf{sig})$ be special, i.e., that $\zeta^{\mathrm{dlog}_{\mathbf{z}_j} \mathbf{z}_{1,j}} \neq \zeta_1$ for all $\mathbf{z}_j$ and $\mathbf{z}_{1,j}$ (where again $\mathbf{z}_j$ and $\mathbf{z}_{1,j}$ corresponds to the value of $\mathbf{z}$ and $\mathbf{z}_1$, respectively, derived during the j-th interaction with oracle sign_1).

From the verification equation of the special signature $(m, \mathsf{info}, \mathsf{sig})$, one obtains the equations $\alpha = \mathbf{g}^\rho \cdot \mathbf{y}^\omega$, $\beta_1 = \zeta_1^\delta \cdot \mathbf{g}^{\sigma_1}$, $\beta_2 = \zeta_2^\delta \cdot \mathbf{h}^{\sigma_2}$, $\eta = z_j^\mu \cdot \zeta^\delta$. Denoting $w_{0,j} \leftarrow \mathrm{dlog}\, \mathbf{z}_j$, $w \leftarrow \mathrm{dlog}\, \mathbf{h}$, we obtain the reduced equations

$$g_{[\alpha]_{\vec{I}}} + x \cdot y_{[\alpha]_{\vec{I}}} + w \cdot h_{[\alpha]_{\vec{I}}} = \rho + x \cdot \omega \tag{5.3}$$

$$g_{[\beta_1]_{\vec{I}}} + x \cdot y_{[\beta_1]_{\vec{I}}} + w \cdot h_{[\beta_1]_{\vec{I}}} = (g_{[\zeta_1]_{\vec{I}}} + w \cdot h_{[\zeta_1]_{\vec{I}}} + x \cdot y_{[\zeta_1]_{\vec{I}}}) \cdot \delta + \sigma_1 \tag{5.4}$$

$$g_{[\beta_2]_{\vec{I}}} + x \cdot y_{[\beta_2]_{\vec{I}}} + w \cdot h_{[\beta_2]_{\vec{I}}} = (g_{[\zeta_2]_{\vec{I}}} + w \cdot h_{[\zeta_2]_{\vec{I}}} + x \cdot y_{[\zeta_2]_{\vec{I}}}) \cdot \delta + \sigma_2 \cdot w \tag{5.5}$$

$$g_{[\eta]_{\vec{I}}} + w \cdot h_{[\eta]_{\vec{I}}} + x \cdot y_{[\eta]_{\vec{I}}} = w_{0,j} \cdot \mu + (g_{[\zeta]_{\vec{I}}} + w \cdot h_{[\zeta]_{\vec{I}}} + x \cdot y_{[\zeta]_{\vec{I}}}) \cdot \delta. \tag{5.6}$$

We continue by describing simulators S_0 which covers case C_0, and S_1 which covers C_1, C_2, C_3. As we will see, the values c, r, d, s_1, s_2 inside a signature issued as part of a signing query are all known to S_i. Together with the above observations, it is easy for each simulator to convert a query to H_3 into reduced representation. Moreover, the winning tuple in M's output can be identified through knowledge of the logarithms of all $\mathbf{z}_i$ and all $\mathbf{z}_{1,i}$ efficiently.

Case $C_0 = 1$. We describe simulator S_0, which simulates Game 4 using knowledge of $w = \mathrm{dlog}_{\mathbf{g}}\, \mathbf{h}$. On input a discrete logarithm instance $\mathbf{U} \leftarrow \mathbf{g}^x$, it behaves as follows:

Setup: S_0 samples $w \xleftarrow{\$} \mathbb{Z}_q$ and computes the public key pk as pk $\leftarrow (\mathbf{g}, \mathbf{h} \leftarrow \mathbf{g}^w, \mathbf{y} \leftarrow \mathbf{U})$, which implicitly sets sk $\leftarrow x$.

Online Phase. S_0 runs M on input pp, pk and simulates the oracles sign_1, sign_2 as described below. In addition, it simulates the oracles H_1, H_2, H_3 as outlined above.

Queries to sign_1. When M queries $\mathsf{sign}_1(\mathsf{info})$ to open session sid, it calls $H_1(\mathsf{pk}, \mathsf{info})$ and $H_2(\mathrm{rnd}_{\mathsf{sid}})$ for $\mathrm{rnd} \xleftarrow{\$} \{0,1\}^\lambda$. It then uses th $\mathbf{z}$-side signing algorithm from Section 5.3.1 to generate $\mathbf{a}_{\mathsf{sid}}, \mathbf{b}_{1,\mathsf{sid}}, \mathbf{b}_{2,\mathsf{sid}}$ and outputs them along with $\mathrm{rnd}_{\mathsf{sid}}$ to the adversary.

Queries to sign_2. When M queries $\mathsf{sign}_2(\mathsf{sid}, e_{\mathsf{sid}})$, S_0 uses the $\mathbf{z}$-side signer from Section 5.3.1 to compute $c_{\mathsf{sid}}, d_{\mathsf{sid}}, r_{\mathsf{sid}}, s_{1,\mathsf{sid}}, s_{2,\mathsf{sid}}$ using $\mathrm{dlog}_{\mathbf{g}}\, \mathbf{h}$ along with the values $w_{0,\mathsf{sid},g}, w_{0,\mathsf{sid},h}, w_{1,\mathsf{sid},g}, w_{1,\mathsf{sid},h}$ stored in the lists L_1, L_2. It is straightforward to verify that the above simulation of Game 4 is perfect.

Solving the DLOG instance. When M returns $\ell + 1$ message signature pairs, S_0 identifies a special signature using the exponents stored in L_2 along with $\mathrm{dlog}_{\mathbf{g}}\, \mathbf{h}$. It retrieves the corresponding hash query to H_3 from L_3 together with representations of $\alpha, \beta_1, \beta_2, \eta$. S_0 uses Eq. (5.3) and the fact that $C_0 = 1 \Leftrightarrow \omega \neq y_{[\alpha]_{\vec{I}}}$, to (efficiently) compute and output the value x as $x = (\rho - g_{[\alpha]_{\vec{I}}} - w \cdot h_{[\alpha]_{\vec{I}}})/(y_{[\alpha]_{\vec{I}}} - \omega)$. (In case $C_0 = 0$, or there is no hash query corresponding to the special signature, it aborts.)

If $C_0 = 1$, then S_0's simulation of Game 4 is perfect.

Case $C_0 = 0$. We describe simulator S_1, which simulates Game 4 using x. On input a discrete logarithm instance $\mathbf{U} \leftarrow \mathbf{g}^w$, it behaves as follows.

Setup. S_1 samples $x \xleftarrow{\$} \mathbb{Z}_q$. It sets pk $\leftarrow (\mathbf{g}, \mathbf{h} \leftarrow \mathbf{U}, \mathbf{y} \leftarrow \mathbf{g}^x)$, sk $\leftarrow x$.

Online Phase. S_1 runs M on input pp, pk and simulates the oracles sign$_1$, sign$_2$ as described below. In addition, it simulates the oracles H_1, H_2, H_3 as outlined above.

Queries to sign$_1$**.** When M queries sign$_1$(info) to open session sid, it calls $H_1(\mathsf{pk}, \mathsf{info})$ and $H_2(\mathrm{rnd}_{\mathsf{sid}})$ for $\mathrm{rnd} \xleftarrow{\$} \{0,1\}^\lambda$. It computes $\mathbf{a}, \mathbf{b}_1, \mathbf{b}_2$ like the normal signer.

Queries to sign$_2$**.** When M queries sign$_2$ on input (sid, e_{sid}), S_1 uses the normal signing algorithm with the secret key x to compute c, d, r, s_1, s_2 and returns them.

Solving the DLOG instance. When M returns $\ell + 1$ message signature pairs, S_1 identifies the special signature using the exponents stored in L_2 and the method described in Remark 5.3.9. It retrieves the corresponding hash query to H_3 from L_3 together with representations of $\alpha, \beta_1, \beta_2, \eta$. If there is no hash query to H_3 corresponding to the special signature, it aborts. Since $C_0 = 0$ it holds that $C_1 = 1 \vee C_2 = 1 \vee C_3 = 1$. S_1 uses one of the following extraction strategies.

If $C_1 = 1$. S_1 uses the equation Eq. (5.4) to compute a value

$$\sigma_1' := g_{[\beta_1]_{\vec{I}}} + x \cdot y_{[\beta_1]_{\vec{I}}} - \left(g_{[\zeta_1]_{\vec{I}}} + x \cdot y_{[\zeta_1]_{\vec{I}}}\right) \cdot \delta'.$$

It then solves the equation

$$\mathrm{dlog}_{\mathbf{g}}\, \zeta_1 = \frac{\sigma - \sigma'}{\delta' - \delta}$$

we have ruled out that $\delta = \delta'$ in a previous game, this value is indeed computable and well-defined. Using $\mathrm{dlog}_{\mathbf{g}}\, \zeta_1 = g_{[\zeta_1]_{\vec{I}}} + x \cdot y_{[\zeta_1]_{\vec{I}}} + w \cdot h_{[\zeta_1]_{\vec{I}}}$ with $\zeta_1 \neq 0$ allows S_1 to solve for

$$w = \frac{\mathrm{dlog}_{\mathbf{g}}(\zeta_1) - \left(g_{[\zeta_1]_{\vec{I}}} + x \cdot y_{[\zeta_1]_{\vec{I}}}\right)}{h_{[\zeta_1]_{\vec{I}}}}.$$

If $C_1 = 0$ and $C_2 = 1$. In this case, the reduction S_1 uses Eq. (5.5) to compute an alternate

$$\sigma_2' := h_{[\beta_2]_{\vec{I}}} - h_{[\zeta_2]_{\vec{I}}} \cdot \delta''.$$

It then computes

$$\mathrm{dlog}_{\mathbf{h}}(\zeta_2) := \frac{\sigma_2 - \sigma_2'}{\delta'' - \delta}$$

and from this

$$w = \left(\frac{\mathrm{dlog}_{\mathbf{h}}\, \zeta_2 - h_{[\zeta_2]_{\vec{I}}}}{g_{[\zeta_2]_{\vec{I}}} + x \cdot y_{[\zeta_2]_{\vec{I}}}}\right)^{-1}.$$

If $C_1 = C_2 = 0$ and $C_3 = 1$. In this case, the simulator can compute $\mathrm{dlog}_{\mathbf{z}_i} \zeta$. First, it computes

$$\mu' := \frac{g_{[\eta]_{\overrightarrow{I}}} + x \cdot y_{[\eta]_{\overrightarrow{I}}} - \delta'''(g_{[\zeta]_{\overrightarrow{I}}} + x \cdot y_{[\zeta]_{\overrightarrow{I}}})}{w_{0,i,g}} = \frac{h_{[\eta]_{\overrightarrow{I}}} - \delta''' h_{[\zeta]_{\overrightarrow{I}}}}{w_{0,i,h}}.$$

This allows the reduction S_1 to also compute the value $\mathrm{dlog}_{\mathbf{z}_i} \zeta = \frac{\mu' - \mu}{\delta''' - \delta}$ and then

$$\mathrm{dlog}_{\mathbf{g}} \mathbf{h} = \frac{g_{[\zeta]_{\overrightarrow{I}}} + x \cdot y_{[\zeta]_{\overrightarrow{I}}} - \mathrm{dlog}_{\mathbf{z}_i} \zeta \cdot w_{0,i,g}}{h_{[\zeta]_{\overrightarrow{I}}} - \mathrm{dlog}_{\mathbf{z}_i} \zeta \cdot w_{0,i,h}}$$

where i is an index corresponding to info.

Since both simulators provide a perfect simulation (in their respective cases) and cover all cases that can happen whenever M wins Game 4, B can run the correct simulator to extract the discrete logarithm with advantage $\mathrm{Adv}_{\mathsf{B}}^{\mathbf{DLOG}} \geq \mathrm{Adv}_{\mathsf{M}}^{\mathsf{Game\ 4}}/2$. Hence, $t_{\mathsf{B}} \approx t_{\mathsf{M}}$ and summing up over the game hops leading to Game 4 yields

$$\begin{aligned}\mathrm{Adv}_{\mathsf{B}}^{\mathbf{DLOG}} \geq \left(1 - \frac{4}{q}\right) \frac{1}{2} \mathrm{Adv}_{\mathsf{M},\ell,\mathsf{BSA}}^{\mathbf{RB\text{-}OMUF}} - \frac{\ell + 1}{q} \\ - \mathrm{Adv}_{\mathsf{R}_1}^{\mathbf{DLOG}} - \mathrm{Adv}_{\mathsf{R}_2}^{\mathbf{DLOG}} + \mathrm{Adv}_{\mathsf{R}_3}^{\mathbf{DLOG}} - \frac{15Q_h + 20}{q}\end{aligned}$$

□

5.3.4 The Main Theorem

In the following, we show that Abe's blind signature scheme has full one-more-unforgeability. We make use of the restrictive blinding lemma to identify the forged signature.

Theorem 5.3.12. *Let* M *be an algebraic algorithm that runs in time t_{M}, makes at most ℓ queries to oracle* sign_2 *in ℓ-$\mathbf{OMUF}_{\mathsf{BSA}}$ and at most (total) Q_h queries to H_1, H_2, H_3. Then, in the random oracle model, there exists an algorithm* C *and reductions* $\mathsf{R}_1, \ldots \mathsf{R}_5$ *such that*

$$\begin{aligned}\mathrm{Adv}_{\mathsf{C}}^{\mathbf{DLOG}} \geq \frac{1}{4} \cdot \left(1 - \frac{4}{q}\right) \cdot \Big(\mathrm{Adv}_{\mathsf{M},\ell,\mathsf{BSA}}^{\mathbf{OMUF}} - \frac{2\ell + 41Q_h + 25}{q} - 2\mathrm{Adv}_{\mathsf{R}_1}^{\mathbf{DLOG}} - 2\mathrm{Adv}_{\mathsf{R}_2}^{\mathbf{DLOG}} \\ -2\mathrm{Adv}_{\mathsf{R}_3}^{\mathbf{DLOG}} - \mathrm{Adv}_{\mathsf{R}_4}^{\mathbf{DLOG}} - \mathrm{Adv}_{\mathsf{R}_5}^{\mathbf{DLOG}} / \left(1 - \frac{4}{q}\right)\Big)\end{aligned}$$

Proof Overview. The proof strategy is very similar to that of Lemma 5.3.3. The strategy of the main reduction is mostly the same, except that we assume that for every signature the value δ''' is well defined (we can use Claim 5.3.8 to ensure that), and that there are two instead of one 'special signatures', namely there are two signatures linked to the same session via a DDH-style relation by the ζ, ζ_1 values in the signature and the $\mathbf{z}, \mathbf{z}_1$ values from the signer.

We then prove an analogous claim to Claim 5.3.10, showing that even though there are intertwined dependencies between the values d and δ and c and ω, it is infeasible for the adversary to provoke a scenario where the preliminary $\delta', \delta'', \delta'''$ and ω' are equal to the actual signature components δ and ω for both of the special singatures.

Proof of Theorem 5.3.12. We prove this using a series of games.

Game 0. This is the standard one-more unforgeability game for the partially blind scheme.

Game 1. This game aborts if there are no two signatures linked to the same session.

Claim 5.3.13.

$$\Pr[\textit{Game 1} = 1] - \Pr[\textit{Game 0} = 1] = \Pr[\mathbf{RB\text{-}OMUF} = 1]$$

Proof. As there are $\ell+1$ valid signatures and ℓ closed signing sessions, there must be either two signatures that are linked to the same session, or one that is not linked to any session. □

Game 2. This game aborts if there is a signature among the $\ell + 1$ signatures in the adversary's output where no hash query to H_3 was made.

Claim 5.3.14. *It holds that* $\Pr[\textit{Game 2} = 1] - \Pr[\textit{Game 1} = 1] \leq \frac{\ell+1}{q}$.

Proof. This is because the probability that $\omega + \delta = \varepsilon$ for such a signature is $\frac{1}{q}$ as the value ε (the output of H_3) would be sampled after the values ω and δ are already fixed. Union bounding over all signatures yields the above claim.

□

Game 3. In this game, we introduce a conceptual change to how we sample the values $\mathbf{z}_i$ and $\mathbf{z}_{1,i}$ in the RO responses for H_1, H_2. Namely, instead of sampling from $\mathbb{G}$ directly, we sample $w_{0,i,g}, w_{0,i,h} \xleftarrow{\$} \mathbb{Z}_q$ in the case of $\mathbf{z}_i$ and $w_{1,i,g}, w_{1,i,h} \xleftarrow{\$} Z_q$ in the case of $\mathbf{z}_{1,i}$. We then set $\mathbf{z}_i = \mathbf{g}^{w_{0,i,g}} \cdot \mathbf{h}^{w_{0,i,h}}$ and $z_{1,i} = \mathbf{g}^{w_{1,i,g}} \mathbf{h}^{w_{1,i,h}}$.

In the following, we use the definitions of the reduced representations as in the proof of Lemma 5.3.3 as well as the definitions of $\delta', \delta'', \delta''', \omega'$ and of the cases C_0, C_1, C_2, C_3 for the two special signatures instead of one special signature.

Game 4. This game aborts if there exists a signature among the $\ell + 1$ forgeries where the preliminary value δ''' is defined and $\delta''' = \delta$.

Claim 5.3.15.

$$\Pr[\textit{Game 4} = 1] - \Pr[\textit{Game 3} = 1] \leq \mathrm{Adv}^{\mathrm{dlog}}_{\mathsf{R}_1} + \frac{5Q_h}{q}$$

Proof. This follows from Claim 5.3.6 as δ''' is defined in the same way here as in the proof of Lemma 5.3.3. □

Game 5. In this game, we introduce an abort condition that will be analogous to that of E_2 in the proof of Lemma 5.3.3.

Namely, we define the event E_i for the ith signature as follows:

$$\begin{aligned} E_i :=& (\omega_i = \omega_i') \wedge ((\delta_i' = \bot) \vee (\delta_i' = \delta_i)) \\ & \wedge ((\delta_i'' = \bot) \vee (\delta_i'' = \delta_i)) \wedge ((\delta_i''' = \bot) \vee \delta_i''' = \delta_i) \end{aligned}$$

Game 5 aborts if for the special signatures (indexed by i^*, i^{**}) it holds that $E_{i^{**}} \wedge E_{i^*}$.

Claim 5.3.16.

$$\Pr[E_{i^*} \wedge E_{i^{**}}] \leq \frac{21Q_h + 4}{q} + \mathrm{Adv}^{\mathbf{DLOG}}_{\mathsf{R}_2} + \mathrm{Adv}^{\mathbf{DLOG}}_{\mathsf{R}_3} + \mathrm{Adv}^{\mathbf{DLOG}}_{\mathsf{R}_4} + \mathrm{Adv}^{\mathbf{DLOG}}_{\mathsf{R}_5}/(1 - \frac{4}{q})$$

Proof. We first note that we already ruled out $\delta_i''' \neq \perp \wedge \delta_i''' = \delta_i$ in Game 4. Thus, we consider here only the cases where $\delta''' = \perp$.

Case 1: any of the $\mathcal{C}_5 \neq 0$. In this case, we can apply Lemma 5.3.1 and find that the probability of $E_{i^{**}} \wedge E_{i^*}$ is at most $\frac{5Q_h}{q}$.

Case 2: $\delta'_{i^*} \neq \perp \wedge \overline{b_2}(\beta_{1,i^*}) \vee \delta'_{i^{**}} \neq \perp \wedge \overline{b_2}(\beta_{1,i^{**}})$. We want to apply Lemma 5.3.1 to the values δ'_{i^*} and δ'_{i^*}. In this case, it either holds that $\mathcal{C}_4/\mathcal{C}_3 \neq \mathcal{C}_1$ for at least one of the two special signatures, and we can apply Lemma 5.3.1 to upper bound the probability of $E_{i^{**}} \wedge E_{i^*}$ to $\frac{5}{q}$. Or, if $\mathcal{C}_4/\mathcal{C}_3 = \mathcal{C}_1$ for both special signatures, we apply the same reduction strategy as in the analogous case of Claim 5.3.10 to solve the discrete logarithm problem.

Case 3: $\delta''_{i^*} \neq \perp \wedge \overline{b_1}(\beta_{2,i^*}) \neq 0 \vee \delta''_{i^{**}} \neq \perp \wedge \overline{b_1}(\beta_{2,i^{**}}) \neq 0$. We want to show that here we can either we can apply Lemma 5.3.1 for either δ''_{i^*} or $\delta''_{i^{**}}$, or we can again solve a discrete logarithm problem. In this case, either $\mathcal{C}_4/\mathcal{C}_3 \neq \mathcal{C}_1$ for at least one of the special signatures and we can apply Lemma 5.3.1 to upper bound the probability of the corresponding event E_{i^*} or $E_{i^{**}}$ as $\frac{5}{q}$ or (in the case that $\mathcal{C}_4/\mathcal{C}_3 \neq \mathcal{C}_1$ for the δ'' of both special signatures) we can apply the same reduction strategy as in the corresponding case in Claim 5.3.10 to solve for the discrete logarithm of the group element $\mathbf{h}$ in the public key.

Case 4: $\exists s \in \{*, **\} \colon \delta'_{i^s} \neq \perp \wedge \delta''_{i^s} = \perp \vee \delta'_{i^s} = \perp \wedge \delta''_{i^s} \neq \perp$. Since we assumed that for both special signatures, δ''' is undefined, we know that $h_{[\zeta_{i^s}]_{\overrightarrow{l}}} = \gamma_{i^s} w_{0,i,h} \wedge g_{[\zeta_{i^s}]_{\overrightarrow{l}}} = \gamma_{i^s} w_{0,i,g}$ for $s \in \{*, **\}$. Furthermore, it holds that $\zeta_{1,i^s} = \mathbf{z}_{1,i}^{\gamma_{i^s}}$ and thus $h_{[\zeta_{1,i^s}]_{\overrightarrow{l}}} = \gamma_{i^s} \cdot w_{1,i,h}$ and $g_{[\zeta_{1,i^s}]_{\overrightarrow{l}}} = \gamma_{i^s} \cdot w_{1,i,g}$ (if this is not the case then we can employ a reduction R_4 that embeds its discrete logarithm challenge in $\mathbf{h}$, simulates using the secret key, and solves for $\mathrm{dlog}_{\mathbf{g}} \mathbf{h}$ using that $\gamma_{i^s} \cdot (w_{1,i,g} + \mathrm{dlog}_{\mathbf{g}} \mathbf{h} \cdot w_{1,i,h}) = g_{[\zeta_{1,i^s}]_{\overrightarrow{l}}} + \mathrm{dlog}_{\mathbf{g}} \mathbf{h} \cdot h_{[\zeta_{1,i^s}]_{\overrightarrow{l}}}$).

Case 5: $\delta'_{i^*} \neq \perp \wedge \delta''_{i^*} \neq \perp \wedge \delta'_{i^{**}} \neq \perp \wedge \delta''_{i^{**}} \neq \perp$. In this case, we want to show that provoking the event E_{i^*} or $E_{i^{**}}$ is equivalent to solving a $2 - 1$-ROS problem which is information-theoretically hard. First, we would like to note that $\delta'_{i^*} = \delta''_{i^*}$ and $\delta'_{i^{**}} = \delta''_{i^{**}}$ as otherwise at least one of the δ', δ'' will be unequal to δ from the corresponding signatures and we are done with the proof.

First, we argue that it is infeasible for the adversary to provoke the events E_{i^*} or $E_{i^{**}}$ if it does not use $\mathbf{z}_{\mathsf{sid}^*}, \mathbf{z}_{1,\mathsf{sid}^*}$ such that $\mathrm{dlog}_{\mathbf{z}_{\mathsf{sid}^*}} \zeta_{i^*} = \mathrm{dlog}_{\mathbf{z}_{1,\mathsf{sid}^*}} \zeta_{1,i^*}$ and $\mathrm{dlog}_{\mathbf{z}_{\mathsf{sid}^*}} \zeta_{i^{**}} = \mathrm{dlog}_{\mathbf{z}_{1,\mathsf{sid}^*}} \zeta_{1,i^{**}}$. First, we note that we have already ruled out the case that $\mathcal{C}_5 \neq 0$. If for any of the $\delta'_{i^*}, \delta''_{i^*}, \delta'_{i^{**}}, \delta''_{i^{**}}$ the corresponding values $\mathcal{C}_4/\mathcal{C}_3 \neq \mathcal{C}_1$, we can apply Lemma 5.3.1 to bound the probability of the corresponding event E_{i^*} or $E_{i^{**}}$ to $\frac{5}{q}$.

We thus consider the probability of $\mathcal{C}_4/\mathcal{C}_3 = \mathcal{C}_1$ while at the same time using $\mathbf{z}_{\mathsf{sid}^*}, \mathbf{z}_{1,\mathsf{sid}^*}$ not linked to $\zeta_{i^*}, \zeta_{i^{**}}, \zeta_{1,i^*}$ and $\zeta_{1,i^{**}}$.

We note that we are considering the special signatures here, and thus there exists a session i that is linked to both i_* and i^{**} via $\mathrm{dlog}_{\mathbf{z}_i} \zeta_{i^*} = \mathrm{dlog}_{\mathbf{z}_{1,i}} \zeta_{1,i^*} =: \gamma_{i^*}$ and $\mathrm{dlog}_{\mathbf{z}_i} \zeta_{i^{**}} = \mathrm{dlog}_{\mathbf{z}_{1,i}} \zeta_{1,i^{**}} =: \gamma_{i^{**}}$.

As we assumed that δ'''_{i^*} and $\delta'''_{i^{**}}$ are undefined, the values $\gamma_{i^{**}}$ and γ_{i^*} can be efficiently computed even without knowledge of $\mathrm{dlog}_{\mathbf{g}}\,\mathbf{h}$ via Remark 5.3.9 with probability $1-\frac{4}{q}$. Thus, if the representations of ζ_{1,i^*} and $\zeta_{1,i^{**}}$ are not of the form $(\mathbf{g}^{w_{1,i,g}}\cdot\mathbf{h}^{w_{1,i,h}})^{\gamma_{i^*}}$ and $(\mathbf{g}^{w_{1,i,g}}\cdot\mathbf{h}^{w_{1,i,h}})^{\gamma_{i^{**}}}$ (i.e. $h_{[\zeta_{1,i^s}]_{\vec{I}}} \neq w_{1,i,h}\cdot\gamma_{i^s}$), a reduction R_5 embedding its discrete logarithm challenge in $\mathbf{h}$ and simulating signatures using x can solve for $\mathrm{dlog}_{\mathbf{g}}\,\mathbf{h}$ using the knowledge that $g_{[\zeta_{1,i^s}]_{\vec{I}}}+x\cdot y_{[\zeta_{1,i^s}]_{\vec{I}}}+\mathrm{dlog}_{\mathbf{g}}\,\mathbf{h}\cdot h_{[\zeta_{1,i^s}]_{\vec{I}}} = w_{1,i,g}\cdot\gamma_{i^s}+w_{1,i,h}\cdot\gamma_{i^s}$.

Applying Lemma 5.3.2 in a similar fashion to Claim 5.3.8 yields that the representations of ζ_{1,i^*} and $\zeta_{1,i^{**}}$ must be of the form $\mathbf{z}_{1,i}^{\gamma_{i^*}}$ and $\mathbf{z}_{1,i}^{\gamma_{i^{**}}}$.

We therefore obtain that $\mathcal{C}_4$ must be of the form $\gamma_{i^s}\cdot w_{1,i,h}$ in the case of δ'_{i^s} and $\gamma_{i^s}\cdot w_{1,i,g}$ in the case of δ''_{i^s}.

To bound the probability of $\mathcal{C}_4/\mathcal{C}_3 = \mathcal{C}_1$ if $i \neq \mathsf{sid}^*$, we again want to apply Lemma 5.3.2, this time with $\mathfrak{W}_1$ replacing $w_{1,\mathsf{sid}^*,h}$ and $\mathfrak{W}_2$ replacing $w_{1,\mathsf{sid}^*,g}$, and $L_1 = \mathcal{C}'_4/(\mathcal{C}'_1\cdot\overline{b_1}(\beta_{1,i^s}))$ where all occurrences of $w_{1,\mathsf{sid}^*,h}$ are replaced with the corresponding variable (in the case of δ', using the values $\mathcal{C}'_4,\mathcal{C}_1$ from δ'). We apply the same argument for $\mathcal{C}''_4,\mathcal{C}''_3$ with $\mathcal{C}_1$ for δ''_{i^s} using $\mathfrak{W}_1$ as a replacement for $w_{1,\mathsf{sid}^*,g}$.

This bounds the probability in case $i\neq\mathsf{sid}^*$ to $\frac{4}{q}+\frac{5Q_h}{q}$.

We therefore assume that indeed $\mathrm{dlog}_{\mathbf{z}}\,\zeta_{i^s} = \mathrm{dlog}_{\mathbf{z}_1}\,\zeta_{1,i^s}$ for $s\in\{*,**\}$. We write the verification equations with respect to ω',δ':

$$\begin{aligned} h_{i^*} &= \omega'_{i^*}(c_{\mathsf{sid}^*}) + \delta'_{i^*}(d_{\mathsf{sid}^*}) \\ h_{i^{**}} &= \omega'_{i^{**}}(c_{\mathsf{sid}^*}) + \delta'_{i^{**}}(d_{\mathsf{sid}^*}) \end{aligned}$$

As $d_{\mathsf{sid}^*}+c_{\mathsf{sid}^*} = e_{\mathsf{sid}^*}$, and the only way the adversary can influence $c_{\mathsf{sid}^*}, d_{\mathsf{sid}^*}$ is through e_{sid^*}, this corresponds to solving a 1-ROS problem. The probability of success for this is $\frac{1}{q}$ for a single hash query and thus $\frac{Q_h}{q}$ if we union-bound over all hash queries.

Summing up over all the potential cases yields

$$\begin{aligned} \Pr[E_{i^*}\wedge E_{i^{**}}] \leq \frac{5Q_h}{q} + \frac{5Q_h}{q} + \mathrm{Adv}^{\mathbf{DLOG}}_{\mathsf{R}_2} + \frac{5Q_h}{q} + \mathrm{Adv}^{\mathbf{DLOG}}_{\mathsf{R}_3} + \mathrm{Adv}^{\mathbf{DLOG}}_{\mathsf{R}_4} \\ + \frac{5Q_h}{q} + \frac{4}{q} + \mathrm{Adv}^{\mathbf{DLOG}}_{\mathsf{R}_5}/(1-\frac{4}{q}) + \frac{Q_h}{q} \end{aligned}$$

□

We can now apply the almost same reduction strategy as for Lemma 5.3.3 to solve the discrete logarithm problem.

In particular, the reduction first flips a coin whether to run one of the two simulators S_0 or S_1 described in the following.

We first describe the simulation parts that S_0 and S_1 have in common:

Hash Oracles H_1, H_2**:** given their respective public keys (whose generation we will describe later), both simulators implement H_1, H_2 via lazy sampling, but instead of sampling uniformly at random from $\mathbb{G}$, they sample two exponents $w_g, w_h \xleftarrow{\$} \mathbb{Z}_q$ and output $\mathbf{g}^{w_g}\mathbf{h}^{w_h}$. They keep lists L_1, L_2 (for H_1, H_2) of the hash queries along with w_g, w_h and the hash output.

Hash Oracle H_3**:** both S_0 and S_1 simulate H_3 via lazy sampling from $\mathbb{Z}_q$. They both keep a list of the algebraic representations submitted by the adversary.

Case $C_0 = 1$. We now turn to the specifics of simulator S_0 which simulates Game 5 and extracts $\mathrm{dlog}_{\mathbf{g}}\, \mathbf{y}$ in case $C_0 = 1$:

Setup: S_0 samples $w \xleftarrow{\$} \mathbb{Z}_q$ and computes the public key pk as pk $\leftarrow (\mathbf{g}, \mathbf{h} \leftarrow \mathbf{g}^w, \mathbf{y} \leftarrow \mathbf{U})$, which implicitly sets sk $\leftarrow x$.

Online Phase. S_0 runs M on input pp, pk and simulates the oracles sign_1, sign_2 as described below. In addition, it simulates the oracles H_1, H_2, H_3 as outlined above.

Queries to sign_1**.** When M queries sign_1(info) to open session sid, it calls H_1(pk, info) and $H_2(\mathrm{rnd}_{\mathsf{sid}})$ for $\mathrm{rnd} \xleftarrow{\$} \{0,1\}^\lambda$. It then uses th $\mathbf{z}$-side signing algorithm from Section 5.3.1 to generate $\mathbf{a}_{\mathsf{sid}}, \mathbf{b}_{1,\mathsf{sid}}, \mathbf{b}_{2,\mathsf{sid}}$ and outputs them along with $\mathrm{rnd}_{\mathsf{sid}}$ to the adversary.

Queries to sign_2**.** When M queries $\text{sign}_2(\text{sid}, e_{\mathsf{sid}})$, S_0 uses the $\mathbf{z}$-side signer from Section 5.3.1 to compute $c_{\mathsf{sid}}, d_{\mathsf{sid}}, r_{\mathsf{sid}}, s_{1,\mathsf{sid}}, s_{2,\mathsf{sid}}$ using $\mathrm{dlog}_{\mathbf{g}}\, \mathbf{h}$ along with the values $w_{0,\mathsf{sid},g}, w_{0,\mathsf{sid},h}$, $w_{1,\mathsf{sid},g}, w_{1,\mathsf{sid},h}$ stored in the lists L_1, L_2. It is straightforward to verify that the above simulation of Game 5 is perfect.

Solving the $\mathbf{DLOG}$ instance. When M returns $\ell + 1$ message signature pairs, S_0 identifies the two special signatures using the exponents stored in L_2 along with $\mathrm{dlog}_{\mathbf{g}}\, \mathbf{h}$. It retrieves the corresponding hash queries to H_3 from L_3 together with representations of $\alpha, \beta_1, \beta_2, \eta$. S_0 uses Eq. (5.3) and the fact that $C_0 = 1 \Leftrightarrow \omega \neq y_{[\alpha]_{\vec{I}}}$ for at least one of the special signatures, to (efficiently) compute and output the value x as $x = (\rho - g_{[\alpha]_{\vec{I}}} - w \cdot h_{[\alpha]_{\vec{I}}})/(y_{[\alpha]_{\vec{I}}} - \omega)$. (In case $C_0 = 0$, or there is no hash query corresponding to the special signature, it aborts.)

If $C_0 = 1$, then S_0's simulation of Game 5 is perfect.

Case $C_0 = 0$. We now describe the simulator S_1 which simulates Game 5and extracts $\mathrm{dlog}_{\mathbf{g}}\, \mathbf{h}$ in case $C_1 = 1 \vee C_2 = 1 \vee C_3 = 1$. We capture these cases with one simulator as the strategy during the online-phase is the same and only extraction differs.

Setup. S_1 samples $x \xleftarrow{\$} \mathbb{Z}_q$. It sets pk $\leftarrow (\mathbf{g}, \mathbf{h} \leftarrow \mathbf{U}, \mathbf{y} \leftarrow \mathbf{g}^x)$, sk $\leftarrow x$.

Online Phase. S_1 runs M on input pp, pk and simulates the oracles sign_1, sign_2 as described below. In addition, it simulates the oracles H_1, H_2, H_3 as outlined above.

Queries to sign_1**.** When M queries sign_1(info) to open session sid, it calls H_1(pk, info) and $H_2(\mathrm{rnd}_{\mathsf{sid}})$ for $\mathrm{rnd} \xleftarrow{\$} \{0,1\}^\lambda$. It computes $\mathbf{a}, \mathbf{b}_1, \mathbf{b}_2$ like the normal signer.

Queries to sign_2**.** When M queries sign_2 on input $(\text{sid}, e_{\mathsf{sid}})$, S_1 uses the normal signing algorithm with the secret key x to compute c, d, r, s_1, s_2 and returns them.

Solving the $\mathbf{DLOG}$ Instance. When M returns $\ell + 1$ message signature pairs, S_1 identifies the special signatures using the exponents stored in L_2 and the method described in Remark 5.3.9 or a signature for which $\delta''' \neq \perp$. It retrieves the corresponding hash queries to H_3 from L_3 together with representations of $\alpha, \beta_1, \beta_2, \eta$. If there is no hash query to H_3 corresponding to the special signature, it aborts. By Claim 5.3.5 it holds that $C_1 \vee C_2 \vee C_3$ with probability $1 - \frac{4}{q}$. S_1 uses one of the following extraction strategies, where again the two special signatures are indexed by i^s for $s \in \{*, **\}$.

If $C_1 = 1$ for i^s with $s \in \{*, **\}$. S_1 uses the equation Eq. (5.4) to compute a value

$$\sigma'_{1,i^s} := g_{[\beta_{1,i^s}]_{\vec{I}}} + x \cdot y_{[\beta_{1,i^s}]_{\vec{I}}} - \left(g_{[\zeta_{1,i^s}]_{\vec{I}}} + x \cdot y_{[\zeta_{1,i^s}]_{\vec{I}}}\right) \cdot \delta'_{i^s}.$$

It then solves the equation

$$\mathrm{dlog}_{\mathbf{g}}\, \zeta_{1,i^s} = \frac{\sigma_{1,i^s} - \sigma'_{1,i^s}}{\delta'_{i^s} - \delta_{i^s}}$$

we have ruled out that $\delta = \delta'$ in a previous game, this value is indeed computable and well-defined. Using $\mathrm{dlog}_{\mathbf{g}}\, \zeta_{1,i^s} = g_{[\zeta_{1,i^s}]_{\vec{I}}} + x \cdot y_{[\zeta_{1,i^s}]_{\vec{I}}} + w \cdot h_{[\zeta_{1,i^s}]_{\vec{I}}}$ with $\zeta_1 \neq 0$ allows S_1 to solve for

$$w = \frac{\mathrm{dlog}_{\mathbf{g}}(\zeta_{1,i^s}) - \left(g_{[\zeta_{1,i^s}]_{\vec{I}}} + x \cdot y_{[\zeta_{1,i^s}]_{\vec{I}}}\right)}{h_{[\zeta_{1,i^s}]_{\vec{I}}}}.$$

If $C_1 = 0$ and $C_2 = 1$ for i^s with $s \in \{*, **\}$. In this case, the reduction S_1 uses Eq. (5.5) to compute an alternate

$$\sigma'_{2,i^s} := h_{[\beta_{2,i^s}]_{\vec{I}}} - h_{[\zeta_{2,i^s}]_{\vec{I}}} \cdot \delta''_{i^s}.$$

It then computes

$$\mathrm{dlog}_{\mathbf{h}}(\zeta_{2,i^s}) := \frac{\sigma_{2,i^s} - \sigma'_{2,i^s}}{\delta''_{i^s} - \delta_{i^s}}$$

and from this

$$w = \left(\frac{\mathrm{dlog}_{\mathbf{h}}\, \zeta_{2,i^s} - h_{[\zeta_{2,i^s}]_{\vec{I}}}}{g_{[\zeta_{2,i^s}]_{\vec{I}}} + x \cdot y_{[\zeta_{2,i^s}]_{\vec{I}}}}\right)^{-1}.$$

If $C_1 = C_2 = 0$ and $C_3 = 1$ for i^s with $s \in \{*, **\}$. In this case, the simulator can compute $\mathrm{dlog}_{\mathbf{z}_i}\, \zeta_{i^s}$. First, it computes

$$\mu'_{i^s} := \frac{g_{[\eta_{i^s}]_{\vec{I}}} + x \cdot y_{[\eta_{i^s}]_{\vec{I}}} - \delta'''_{i^s}(g_{[\zeta_{i^s}]_{\vec{I}}} + x \cdot y_{[\zeta_{i^s}]_{\vec{I}}})}{w_{0,i,g}} = \frac{h_{[\eta_{i^s}]_{\vec{I}}} - \delta'''_{i^s} h_{[\zeta]_{\vec{I}}}}{w_{0,i,h}}.$$

This allows the reduction S_1 to also compute the value $\mathrm{dlog}_{\mathbf{z}_i}\, \zeta_{i^s} = \frac{\mu'_{i^s} - \mu_{i^s}}{\delta'''_{i^s} - \delta_{i^s}}$ and then

$$\mathrm{dlog}_{\mathbf{g}}\, \mathbf{h} = \frac{g_{[\zeta_{i^s}]_{\vec{I}}} + x \cdot y_{[\zeta_{i^s}]_{\vec{I}}} - \mathrm{dlog}_{\mathbf{z}_i}\, \zeta \cdot w_{0,i,g}}{h_{[\zeta_{i^s}]_{\vec{I}}} - \mathrm{dlog}_{\mathbf{z}_i}\, \zeta \cdot w_{0,i,h}}$$

where i is an index corresponding to info.

This yields in total that

$$\begin{aligned}\mathrm{Adv}^{\mathbf{DLOG}}_{\mathsf{C}} \geq \frac{1}{4} \cdot \left(1 - \frac{4}{q}\right) \cdot \Bigg(&\mathrm{Adv}^{\mathbf{OMUF}}_{\mathsf{M},\ell,\mathsf{BSA}} - \frac{\ell+1}{q} - \mathrm{Adv}^{\mathbf{DLOG}}_{\mathsf{R}_1} - \mathrm{Adv}^{\mathbf{DLOG}}_{\mathsf{R}_2} \\ &- \mathrm{Adv}^{\mathbf{DLOG}}_{\mathsf{R}_3} - \frac{15Q_h + 20}{q} - \frac{\ell+1}{q}\mathrm{Adv}^{\mathbf{DLOG}}_{\mathsf{R}_1} - \frac{5Q_h}{q} - \frac{21Q_h + 4}{q} - \mathrm{Adv}^{\mathbf{DLOG}}_{\mathsf{R}_2} + \mathrm{Adv}^{\mathbf{DLOG}}_{\mathsf{R}_3} \\ &- \mathrm{Adv}^{\mathbf{DLOG}}_{\mathsf{R}_4} - \mathrm{Adv}^{\mathbf{DLOG}}_{\mathsf{R}_5} / \left(1 - \frac{4}{q}\right)\Bigg)\end{aligned}$$

□

5.4 Applying the Forking-Based Proof from Chapter 4 to Abe's Scheme

This section is based on the full version [KLX22b] of [KLX22c].

In this section, we briefly sketch how the technique described in Chapter 4 can be applied to the (partially) blind signature scheme by Abe [Abe01]. We note that Abe's blind signature scheme is immune to the ROS-attack. Therefore, it may be possible to prove security also outside of the AGM with respect to polynomially many concurrent signing sessions. However, the forking-based strategy from Chapter 4 yields a superpolynomial loss in such a setting. It remains an open question whether Abe's scheme can be proven secure for polynomially many concurrent signing sessions in the plain ROM (i.e. without using the AGM).

5.4.1 Adapting the Proof to Partial Blindness

In [KLX22b], we described how to use the forking reduction to solve for the discrete logarithm of the public key group element $\mathbf{h}$ (which is part of the $\mathbf{z}$-side proofs), or for the discrete logarithm of $\mathbf{y}$. This reduction was applied to the original version of Abe's blind signature scheme, i.e. without the adaption to partial blindness described in Section 5.1.

Here, we extend this reduction to also capture the partially blind variant. However, as the public key of the partially blind variant contains a $\mathbf{z}$-side component $\mathbf{h}$ that is shared over all tags, we cannot directly apply the same strategy as in Theorem 4.4.1. Instead, we alter the main reduction in such a way that it alternates between witnesses only for one selected tag $\overline{\mathsf{info}}$, whereas it uses the $\mathbf{z}$-side signer for all other tags. This strategy is necessary as, unlike for the scheme by Abe and Okamoto, the key part $\mathbf{h}$ is the same for all signatures to all tags. We briefly sketch some other approaches to overcome this issue and why they are less promising or do not work at all.

- Using the same techniques as in Theorem 4.4.1. While we could still re-program the random oracle H_1 and H_2 for all sessions that do not use the challenge tag, the reduction does not know the discrete logarithm of $\mathbf{h}$ and therefore also not the discrete logarithms of the values $\mathbf{z}_{2,i}$ to $\mathbf{h}$.

- proving a different variant of Theorem 4.4.1 by re-randomizing the values $\mathbf{z}$ and $\mathbf{z}_{1,i}$ from a single-tag challenger. One could attempt to take a single-tag challenger and whenever the adversary asks for signatures with different tags, the reduction would re-randomize the single tag key $\mathbf{z}$ to $\mathbf{z}' = \mathbf{z} \cdot \mathbf{g}^{\kappa}$ for some uniformly random value κ. The corresponding session keys $\mathbf{z}_{1,i}$ could be re-randomized to $\mathbf{z}'_{1,i} = \mathbf{z}_{1,i} \cdot \mathbf{g}^{\kappa}$ using the same κ for the same info, making it possible to also adapt the $\mathbf{z}_1$-parts of the signing queries accordingly. The $\mathbf{z}_{2,i}$ parts of the signing queries would stay the same, allowing the reduction to re-use the value $\mathbf{b}_2$ as well as d, s_2 from the single-tag signer. In principle, this strategy works, however, the drawback is that then the sum of all concurrent open sessions can only be the same as for one single session which would be even worse than for the Abe-Okamoto scheme.

- Directly applying the reduction for all tags but with alternating the witnesses for all tags at once as well. This on the one hand has the same issue as the previous approach, and on the other hand it would require a one-more forgery with respect to the total count of signing sessions instead of with respect to the number of sessions per tag.

5.4.2 The Deterministic Wrapper

We describe the reduction strategy. Mostly this proof follows the steps from Chapter 4. However, due to the structure of Abe's scheme, part of the $\mathbf{z}$-side witness, namely the public key part $\mathbf{h}$ is fix throughout all signing sessions. This prevents us from using the $\mathbf{z}$-side simulator in a one-tag-to-many-tag theorem. We therefore instead prove security for all tags directly. To this end, we will pick one of the tags info where the simulation will differ between $\mathbf{y}$-side and $\mathbf{z}$-side, and for all other tags we simulate using the $\mathbf{z}$-side signer.

Guessing the Challenge Tag. The instances will contain an index j^* that corresponds to a hash query made by the adversary to H_1. We consider the adversary to be successfull only if its one-more forgery corresponds to the tag $\overline{\mathsf{info}}$ such that $(\mathsf{pk}, \overline{\mathsf{info}})$ is the j^*th fresh hash query made to H_1. If Q_{H_1} is an upper bound on the number of distinct queries that U makes to H_1, this introduces a loss of $\frac{1}{Q_{H_1}}$. Using a game hop we can also rule out the case that the adversary never queries the challenge tag to the RO H_1, as guessing the challenge tag key without querying it to the RO succeeds with probability at most $\frac{1}{q-1}$.

Restricting the Hash Queries to $\ell + 1$. For an adversary U that makes at most ℓ signing queries per tag info (i.e. closes ℓ signing sessions) and Q_h hash queries we make use of the same hash query guessing strategy as before, i.e. we use a wrapper M that restricts the adversary to exactly $\ell + 1$ hash queries and introduces a loss of $\frac{1}{\binom{Q_h}{\ell+1}}$. In this case, the wrapper guesses the hash queries used for the $\ell + 1$ signatures that correspond to the guessed challenge tag $\overline{\mathsf{info}}$.

The Deterministic Wrapper We describe the inputs of the deterministic wrapper A.

Different to before, the wrapper will have an index j^* that corresponds to a hash query to the hash oracle H_1 that maps $(\mathsf{pk}, \mathsf{info})$ to the tag key $\mathbf{z}$. We denote the tag corresponding to this hash query by $\overline{\mathsf{info}}$. We again want to define $\mathbf{y}$-side instances and $\mathbf{z}$-side instances, but now both types of instances will simulate using the $\mathbf{z}$-side witness whenever the tag is not $\overline{\mathsf{info}}$.

First, we define $\mathbf{y}$-side instances, i.e. instances that use the $\mathbf{y}$-side witness x for simulation when the tag is $\overline{\mathsf{info}}$.

- $b = 0$
- $x \in \mathbb{Z}_q$
- $w \in \mathbb{Z}_q$
- $\mathbf{z}_{\overline{\mathsf{info}}} \in \mathbb{G}; j^* \in [Q_{H_1}]$
- for $j \in [Q_{H_1}] \setminus \{j^*\}$: $w_{0,j} \in \mathbb{Z}_q$
- for $i \in [(Q_{\mathsf{info}} - 1) \cdot \ell]$: $\mathrm{rnd}_i \in \{0,1\}^\lambda$
- for $i' \in [\ell]$: $\mathrm{rnd}_{i'} \in \{0,1\}^\lambda$
- for $i' \in [\ell]$: $\mathbf{z}_{1,i'} = \mathbf{g}^{w_{1,i'}}$ with $w_{1,i'} \in \mathbb{Z}_q$
- for $i \in [(Q_{\mathsf{info}} - 1) \cdot \ell]$: $\mathbf{z}_{1,i} = \mathbf{g}^{w_{1,i}}$ with $w_{1,i} \in \mathbb{Z}_q$

- for $i' \in [\ell]$: $u_{i'}, d_{i'}, s_{1,i'}, s_{2,i'} \in \mathbb{Z}_q$
- for $i \in [(Q_{\mathsf{info}} - 1) \cdot \ell]$: $c_i, r_i, v_{1,i}, v_{2,i} \in \mathbb{Z}_q$

We further describe the $\mathbf{z}$-side instances, i.e. instances that alway simulate using the $\mathbf{z}$-side signer:

- $b = 1$
- $\mathbf{y} \in \mathbb{G}$
- $w, w_{0,\overline{\mathsf{info}}} \in \mathbb{Z}_q$; $j^* \in Q_{H_1}$
- for $j \in [Q_{H_1}] \setminus \{j^*\}$: $w_{0,j} \in \mathbb{Z}_q$
- for $i \in [(Q_{\mathsf{info}} - 1) \cdot \ell]$: $\mathrm{rnd}_i \in \{0,1\}^\lambda$
- for $i' \in [\ell]$: $\mathrm{rnd}_{i'} \in \{0,1\}^\lambda$
- for $i \in [(Q_{\mathsf{info}} - 1) \cdot \ell]$: $w_{1,i} \in \mathbb{Z}_q$
- for $i' \in [\ell]$: $w_{1,i'} \in \mathbb{Z}_q$
- for $i' \in [\ell]$: $c_{i'}, r_{i'}, v_{1,i'}, v_{2,i'} \in \mathbb{Z}_q$
- for $i \in [(Q_{\mathsf{info}} - 1) \cdot \ell]$: $c_i, r_i, v_{1,i}, v_{2,i} \in \mathbb{Z}_q$

The wrapper additionally takes as input a set of random coins $\mathsf{rand} = (\mathsf{r_B}, \mathsf{r_A})$ as well as a vector of hash responses $\overrightarrow{h} \in \mathbb{Z}_q^{Q_h}$. We show the wrapper as pseudocode in Figure 5.3

Analogous to before, we can define forking tuples, partners, and triangles. For this, we consider only the partial query transcript w.r.t. the challenge tag $\overline{\mathsf{info}}$. We denote this partial transcript corresponding to the sessions where $\overline{\mathsf{info}}$ was used by $\overrightarrow{\overline{e}}$.

Definition 5.4.1 (Query Transcript)**.** *Consider the wrapper* A *running on input tuple* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$. *The* query transcript for $\overline{\mathsf{info}}$, *denoted* $\overrightarrow{\overline{e}}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$, *is the vector of queries* e_{sid} *made to the* sign_2 *oracle for sessions with the tag* $\overline{\mathsf{info}}$ *(simulated by* A*) by the adversary* M, *ordered by* sid.

We use the same definition of forking as in Definition 4.2.5 and redefine partners w.r.t. the partial transcript of $\overline{\mathsf{info}}$.

Definition 5.4.2 (Partners)**.** *We say two (successful) tuples* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}), (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$ *are* partners *w.r.t.* $\overline{\mathsf{info}}$ *at index* $i \in [\ell + 1]$ *if the followings hold:*

- $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ *and* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$ *fork at index* i
- $\overrightarrow{\overline{e}}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) = \overrightarrow{\overline{e}}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$

We denote the set of $(\overrightarrow{h}, \overrightarrow{h}')$ *such that* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ *and* $(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}')$ *are partners at index* i *by* $\mathrm{prt}_i(\mathbf{I}, \mathsf{rand})$. *We further denote by* $P_{\mathbf{I},\mathsf{rand}}$ *the following set:*

$$P_{\mathbf{I},\mathsf{rand}} = \left\{ (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in \mathsf{Succ}_{\mathbf{I},\mathsf{rand}} \middle| \exists \overrightarrow{h}', i \in [\ell+1] \colon (\overrightarrow{h}, \overrightarrow{h}') \in \mathrm{prt}_i(\mathbf{I}, \mathsf{rand}) \right\}$$

We can use the same definition for the maximum branching index.

In order to count P_G like in Lemma 4.2.12, we consider that the number of query transcripts corresponding to $\overline{\mathsf{info}}$ is of size at most q^ℓ whereas there need to be $\ell + 1$ hash queries made w.r.t. $\overline{\mathsf{info}}$. Thus, the same lemma applies.

```
$\mathsf{A}^{\mathsf{M}}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ :
  $L_1 = L_2 = L_3 = \emptyset$
  $j_{\mathsf{info}} = 0$
  $\mathsf{sid}, i, i', \mathsf{hind} \leftarrow 0$
  parse $\mathbf{I} = (b, \ldots)$
  if $b = 0$
     parse $\mathbf{I} = (0, x, \mathbf{h}, w_1, w_2, \mathbf{z}, \overrightarrow{\mathrm{rnd}}, \overrightarrow{\mathbf{z}_1}, \overrightarrow{u}, \overrightarrow{d}, \overrightarrow{s_1}, \overrightarrow{s_2})$
     $\mathsf{pk} \leftarrow (\mathbf{g}, \mathbf{y} = \mathbf{g}^x, \mathbf{h}$
     $L_1 \leftarrow L_1 \cup \{((\mathsf{pk}, \mathsf{info}), \mathbf{z})\}$
     for $i \in [\ell]$
        $L_2 \leftarrow L_2 \cup \{(\mathrm{rnd}_i, \mathbf{z}_{1,i})\}$
  else
     parse $\mathbf{I} = (1, \mathbf{y}, w, w_0, \overrightarrow{\mathrm{rnd}}, \overrightarrow{w_1}, \overrightarrow{c}, \overrightarrow{r}, \overrightarrow{v_1}, \overrightarrow{v_2})$
     $\mathsf{pk} \leftarrow (\mathbf{g}, \mathbf{y}, \mathbf{h} = \mathbf{g}^w, \mathbf{z} = \mathbf{g}^{w_0})$
     $L_1 \leftarrow L_1 \cup \{((\mathsf{pk}, \mathsf{info}), \mathbf{z})\}$
     for $i \in [\ell]$
        $L_2 \leftarrow L_2 \cup \{(\mathrm{rnd}_i, \mathbf{g}^{w_{1,i}})\}$
  $(m_i, \mathsf{sig}_i)_{i=1}^{\ell+1} \xleftarrow{\$} \mathsf{M}^{\mathsf{Sign}_1, \mathsf{Sign}_2, H_1, H_2, H_3}(\mathsf{pk})$
  for $i = 1 \ldots \ell + 1$
     if $\mathsf{Verify}(\mathsf{pk}, m_i, \mathsf{sig}_i) = 0$
        output $\perp$
  output $(m_i, \mathsf{sig}_i)_{i=1}^{\ell+1}$

$\mathsf{H}_1(\xi)$ :
  if $(\xi, \widetilde{\mathbf{z}}, w) \notin L_1$
     $j_{\mathsf{info}}{+}{+}$
     if $j^* == j_{\mathsf{info}}$
        if $b = 0$
           $L_1 \leftarrow L_1 \cup \{(\xi, \mathbf{z}, \perp)\}$
        else
           $L_1 \leftarrow L_1 \cup \{(\xi, \mathbf{g}^{w_0}, w_0)\}$
        parse $\xi = \mathsf{pk}, \mathsf{info}$
        set $\overline{\mathsf{info}} = \mathsf{info}$
     else
        $\widetilde{w} \xleftarrow{\$} \mathbb{Z}_q$
        $\widetilde{\mathbf{z}} \leftarrow \mathbf{g}^w$
        $L_1 \leftarrow L_1 \cup \{(\xi, \widetilde{z}, \widetilde{w})\}$
  lookup $(\xi, \widetilde{\mathbf{z}}, \cdot) \in L_1$
  return $\widetilde{\mathbf{z}}$

$\mathsf{H}_2(\xi)$ :
  if $(\xi, \cdot) \notin L_2$
     $\widetilde{w} \xleftarrow{\$} \mathbb{Z}_q$
     $L_2 \leftarrow L_2 \cup \{(\xi, \mathbf{g}^{\widetilde{w}}, \widetilde{w})\}$
  lookup $(\xi, \widetilde{\mathbf{z}}, \cdot) \in L_2$
  return $\widetilde{\mathbf{z}}$

$\mathsf{H}_3(\xi)$ :
  if $(\xi, \cdot) \notin L_3$
     $\mathsf{hind}{+}{+}$
     $L_3 \leftarrow L_3 \cup \{(\xi, h_{\mathsf{hind}})\}$
  lookup $(\xi, \widetilde{h}) \in L_3$
  return $\widetilde{h}$

$\mathsf{Sign}_1(\mathsf{info})$ :
  $\mathsf{sid}{+}{+}$
  $\mathsf{sid.open} = \mathtt{true}$
  $\mathbf{z}_{\mathsf{sid}} := H_1(\mathsf{pk}, \mathsf{info})$
  $\mathsf{info}_{\mathsf{sid}} = \mathsf{info}$
  if $\mathsf{info} = \overline{\mathsf{info}}$
     $i'{+}{+}$
     if $b = 0$
        $\mathbf{a} \leftarrow \mathbf{g}^{u_{i'}}$
        $\mathbf{b}_1 \leftarrow \mathbf{g}^{s_{1,i'}} \cdot \mathbf{z}_{1,i'}^{d_{i'}}$
        $\mathbf{b}_2 \leftarrow \mathbf{h}^{s_{2,i'}} \cdot (\mathbf{z}/\mathbf{z}_{1,i'})^{d_{i'}}$
        $(u_{\mathsf{sid}}, d_{\mathsf{sid}}, s_{1,\mathsf{sid}}, s_{2,\mathsf{sid}}) := (u_{i'}, d_{i'}, s_{1,i'}, s_{2,i'})$
     else
        $\mathbf{a} \leftarrow \mathbf{g}^{r_{i'}} \cdot \mathbf{y}^{c_{i'}}$
        $\mathbf{b}_1 \leftarrow \mathbf{g}^{v_{1,i'}}$
        $\mathbf{b}_2 \leftarrow \mathbf{h}^{v_{2,i'}}$
        $(c_{\mathsf{sid}}, r_{\mathsf{sid}}, v_{1,\mathsf{sid}}, v_{2,\mathsf{sid}}) := (c_{i'}, r_{i'}, v_{1,i'}, v_{2,i'})$
  else
     $i{+}{+}$
     $\mathbf{a} \leftarrow \mathbf{g}^{r_i} \cdot \mathbf{y}^{c_i}$
     $\mathbf{b}_1 \leftarrow \mathbf{g}^{v_{1,i}}$
     $\mathbf{b}_2 \leftarrow \mathbf{h}^{v_{2,i}}$
     $(c_{\mathsf{sid}}, r_{\mathsf{sid}}, v_{1,\mathsf{sid}}, v_{2,\mathsf{sid}}) := (c_i, r_i, v_{1,i}, v_{2,i})$
  return $\mathrm{rnd}_{\mathsf{sid}}, \mathbf{a}, \mathbf{b}_1, \mathbf{b}_2$

$\mathsf{Sign}_2(\widetilde{\mathsf{sid}}, e)$ :
  if $\widetilde{\mathsf{sid}}.\mathsf{open} = \mathtt{false}$
     return $\perp$
  $\widetilde{\mathsf{sid}}.\mathsf{open} \leftarrow \mathtt{false}$
  if $b = 0 \wedge \mathsf{info}_{\mathsf{sid}} = \overline{\mathsf{info}}$
     $c_{\widetilde{\mathsf{sid}}} \leftarrow e - d_{\widetilde{\mathsf{sid}}}$
     $r_{\mathsf{sid}} \leftarrow u_{\widetilde{\mathsf{sid}}} - c_{\widetilde{\mathsf{sid}}} \cdot x$
  else
     retrieve $((\mathsf{pk}, \mathsf{info}), \mathbf{z}_{\mathsf{sid}}, w_{0,\mathsf{sid}}) \in L_1$
     retrieve $(\mathrm{rnd}_{\mathsf{sid}}, \mathbf{z}_{1,\mathsf{sid}}, w_{1,\mathsf{sid}}) \in L_2$
     $d_{\widetilde{\mathsf{sid}}} \leftarrow e - c_{\widetilde{\mathsf{sid}}}$
     $s_{1,\widetilde{\mathsf{sid}}} \leftarrow v_{1,\widetilde{\mathsf{sid}}} - d_{\widetilde{\mathsf{sid}}} \cdot w_{1,\widetilde{\mathsf{sid}}}$
     $w_{2,\widetilde{\mathsf{sid}}} \leftarrow (w_{0,\mathsf{sid}} - w_{1,\widetilde{\mathsf{sid}}})/w$
     $s_{2,\widetilde{\mathsf{sid}}} \leftarrow v_{2,\widetilde{\mathsf{sid}}} - d_{\widetilde{\mathsf{sid}}} \cdot w_{2,\widetilde{\mathsf{sid}}}$
  return $c_{\widetilde{\mathsf{sid}}}, r_{\widetilde{\mathsf{sid}}}, d_{\widetilde{\mathsf{sid}}}, s_{1,\widetilde{\mathsf{sid}}}, s_{2,\widetilde{\mathsf{sid}}}$
```

Figure 5.3: Deterministic wrapper for Abe's blind signature scheme

5.4.3 The Transcript Mapping Function and Its Image

We describe the transcript mapping function Φ for Abe's scheme where by $e_{i'}$ we denote the value e_{sid} from the session where $i' \in [\ell]$ was used.

Definition 5.4.3 (Mapping Instances of Abe's scheme)**.** *The transcript mapping function* Φ *is defined like this. For an instance* $\mathbf{I} = (0, x, \mathbf{h}, w_1, w_2, \mathbf{z}, \overrightarrow{\mathrm{rnd}}, \overrightarrow{\mathbf{z_1}}, \overrightarrow{u}, \overrightarrow{d}, \overrightarrow{s}_1, \overrightarrow{s}_2)$*, mapping* $\Phi(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ *with the query transcript* $\overrightarrow{e}$ *generated by running the wrapper* $\mathsf{A}^{\mathsf{U}}(\mathbf{I}, \mathsf{rand}, \overrightarrow{h})$ *does the following:*

- $b \leftarrow 1$
- $\mathbf{y} \leftarrow \mathbf{g}^x$
- $w \leftarrow \mathrm{dlog}_{\mathbf{g}}\, \mathbf{h}$
- $w_1' \leftarrow w_1$
- $w_2' \leftarrow w_2$
- $\overrightarrow{\mathrm{rnd}}' \leftarrow \overrightarrow{\mathrm{rnd}}$

For sessions that use $\overline{\mathsf{info}}$*:*

- $\forall i' \in [\ell]\colon w_{1,i} \leftarrow \mathrm{dlog}_{\mathbf{g}}\, \mathbf{z}_{1,i}$
- $\forall i' \in [\ell]\colon c_i \leftarrow e_{i'} - d_{i'}$
- $\forall i' \in [\ell]\colon r_i \leftarrow u_{i'} - c_{i'} \cdot x$
- $\forall i' \in [\ell]\colon v_{1,i} \leftarrow w_{1,i'} \cdot d_{i'} + s_{1,i'}$
- $\forall i' \in [\ell]\colon v_{2,i} \leftarrow w_{2,i'} \cdot d_{i'} + s_{2,i'}$

For sessions that do not use $\overline{\mathsf{info}}$

- *for* $i \in [(Q_{\mathsf{info}} - 1) \cdot \ell]$ *keep all values as is*

For an instance $\mathbf{I} = (1, \mathbf{y}, w, w_1, w_2, \overrightarrow{\mathrm{rnd}}, \overrightarrow{w}_1, \overrightarrow{c}, \overrightarrow{r}, \overrightarrow{v_1}, \overrightarrow{v_2})$ *the mapping does the following:*

- $b \leftarrow 0$
- $x \leftarrow \mathrm{dlog}_{\mathbf{g}}\, \mathbf{y}$
- $\mathbf{h} \leftarrow \mathbf{g}^w$
- $w_1' \leftarrow w_1$
- $w_2' \leftarrow w_2$
- $\overrightarrow{\mathrm{rnd}}' \leftarrow \overrightarrow{\mathrm{rnd}}$

For sessions that use $\overline{\mathsf{info}}$

- $\forall i' \in [\ell]\colon \mathbf{z}_{1,i'} \leftarrow \mathbf{g}^{w_{1,i'}}$

- $\forall i' \in [\ell]\colon u_{i'} \leftarrow c_{i'} \cdot x + r_{i'}$
- $\forall i' \in [\ell]\colon d_{i'} \leftarrow e_{i'} - c_{i'}$
- $\forall i' \in [\ell]\colon s_{1,i'} \leftarrow v_{1,i'} - d_{i'} \cdot w_{1,i'}$
- $\forall i' \in [\ell]\colon s_{2,i'} \leftarrow v_{2,i'} - d_{i'} \cdot w_{2,i'}$

For sessions that do not use $\overline{\mathsf{info}}$

- *for* $i \in [(Q_{\mathsf{info}} - 1) \cdot \ell]$ *keep all values as is*

Analogously to before, we can show that Φ is a self-inverse bijection that preserves the partner relation. We can then lower-bound the sizes of relevant sets in the image of Φ to finally show that there is a large enough set of both-sided triangle corners.

Using the analogous definition of $B_T^{\times}$ and $\widehat{B_T^{\times}}$, as well as $\widehat{O_T^{\times}}$, we can obtain that there must be a 'good set' G for which forking is likely to result in the desired witness.

5.4.4 Forking Reduction

Theorem 5.4.4 (OMUF security of Abe's scheme)**.** *For all* $\ell \in \mathbb{N}$*, if there exists an adversary* U *that requests at signatures w.r.t. at most* Q_{info} *tags from the signer with at most* ℓ *singatures per tag, makes* Q_{H_1} *queries to random oracle* H_1*,* Q_h *hash queries to random oracle* H_3 *and* $(t_{\mathsf{U}}, \epsilon_{\mathsf{U}}, \ell)$*-breaks* $\mathbf{OMUF}_{\mathsf{Abe}}$ *with* $\epsilon_{\mathsf{U}} \geq \frac{432\left(1-\frac{1}{(\ell+1)^2}\right)}{q} \cdot \binom{Q_h}{\ell+1} \cdot Q_{H_1}$*, then there exists an algorithm* B *that* $\left(t_{\mathsf{B}} \approx 2t_{\mathsf{U}} + \mathrm{O}(\ell+1) + \mathrm{O}({Q_h}^2), \epsilon_{\mathsf{B}} \approx \frac{3\epsilon_{\mathsf{U}}^2}{75423744 \cdot \binom{Q_h}{\ell+1}^2 \cdot (\ell+1)^3 \cdot Q_{H_1}^2}\right)$*-breaks* $\mathbf{DLOG}$*.*

Proof. We give a sketch of the main parts of the proof that work slightly different from the AO scheme.

We describe the reduction R. On input of a discrete logarithm challenge $\mathbf{U}$, R first samples a bit b. If $b = 0$ it samples a $\mathbf{y}$-side instance $\mathbf{I}$ with $\mathbf{h} = \mathbf{U}$, otherwise it samples a $\mathbf{z}$-side instance $\mathbf{I}$ with $\mathbf{y} = \mathbf{U}$. It furthermore samples rand and a hash vector $\vec{h} \xleftarrow{\$} {\mathbb{Z}_q}^{\ell+1}$. It then runs the wrapper $\mathsf{B}^{\mathsf{M}}(\mathbf{I}, \mathsf{rand}, \vec{h})$. Where M is the hash query reduction wrapper around U as described in the previous subsection.

R then samples $i \xleftarrow{\$} [\ell+1]$ and re-samples $\vec{h}' \xleftarrow{\$} {\mathbb{Z}_q}^{\ell+1}_{|\vec{h}_{[i]}}$. The reduction R then re-runs $\mathsf{B}^{\mathsf{A}}(\mathbf{I}, \mathsf{rand}, \vec{h}')$.

We denote by $(\zeta_i, \zeta_{1,i}, \omega_i, \delta_i, \rho_i, \sigma_{1,i}, \sigma_{2,i}, \mu_i)$ and $(\zeta'_i, \zeta'_{1,i}, \omega'_i, \delta'_i, \rho'_i, \sigma'_{1,i}, \sigma'_{2,i}, \mu'_i)$ the signature at hash index i in the first and second run respectively. If both runs are successful and produced a signature for index i, the reduction attempts to solve its discrete logarithm challenge as follows:

If the reduction chose $b = 0$: First compute

$$\operatorname{dlog}_{\mathbf{g}} \zeta_{1,i} = \operatorname{dlog}_{\mathbf{g}} \zeta'_{1,i} = \frac{\sigma'_{1,i} - \sigma_{1,i}}{\delta_i - \delta'_i}$$

and

$$\operatorname{dlog}_{\mathbf{h}} \zeta_{2,i} = \operatorname{dlog}_{\mathbf{h}} \zeta'_{2,i} = \frac{\sigma'_{2,i} - \sigma_{2,i}}{\delta_i - \delta'_i}$$

and

$$\operatorname{dlog}_{\mathbf{z}_{j^*}} \zeta_i = \operatorname{dlog}_{\mathbf{z}} \zeta'_i = \frac{\mu'_i - \mu_i}{\delta_i - \delta'_i}.$$

Then, compute

$$\mathrm{dlog}_{\mathbf{g}}\,\mathbf{z}_1^* = \frac{\mathrm{dlog}_{\mathbf{g}}\,\zeta_{1,i}}{\mathrm{dlog}_{\mathbf{z}_{j^*}}\,\zeta_i}$$

and

$$\mathrm{dlog}_{\mathbf{h}}\,\mathbf{z}_2^* = \frac{\mathrm{dlog}_{\mathbf{h}}\,\zeta_{2,i}}{\mathrm{dlog}_{\mathbf{z}_{j^*}}\,\zeta_i}$$

to finally obtain

$$\mathrm{dlog}_{\mathbf{g}}\,\mathbf{z} = \mathrm{dlog}_{\mathbf{g}}\,\mathbf{z}_1^* + w \cdot \mathrm{dlog}_{\mathbf{h}}\,\mathbf{z}_2^*$$

If the reduction chose $b = 1$ it merely computes

$$\mathrm{dlog}_{\mathbf{g}}\,\mathbf{y} = \frac{\rho_i' - \rho_i}{\omega_i' - \omega_i}.$$

We note that if $b = 0$, extraction works if $\delta_i \neq \delta_i'$. We further note that this event is only dependent on the first run, as this run fixes $\zeta_i, \zeta_{1,i}$, and $\zeta_{2,i}$ already (it is only that the reduction needs the second run to compute).

In the case that $b = 1$, the reduction succeeds in solving its dlog challenge if $\omega_i \neq \omega_i'$.

We can thus apply a similar analysis as for the Abe-Okamoto scheme and obtain the following.

$$\begin{aligned}
\mathrm{Adv}_{\mathsf{R}}^{\mathbf{DLOG}} &\geq \left(\frac{1}{4} - \frac{1}{2q}\right) \cdot \Pr\left[\begin{array}{c} (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in \widehat{G}_b \\ (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}') \in T^{\times}_{T,i}\mathbf{I}, \mathsf{rand}, \overrightarrow{h} \\ t(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) = i \end{array}\right] \\
&= \left(\frac{1}{4} - \frac{1}{2q}\right) \cdot \Pr\left[(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}') \in T^{\times}_{T,i}\mathbf{I}, \mathsf{rand}, \overrightarrow{h} \,\middle|\, \begin{array}{c} (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in \widehat{G}_b \\ t(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) = i \end{array}\right] \\
&\quad \cdot \Pr\left[(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in \widehat{G}_b\right] \cdot \Pr\left[t(\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) = i \,\middle|\, (\mathbf{I}, \mathsf{rand}, \overrightarrow{h}) \in \widehat{G}_b\right] \\
&\geq \left(\frac{1}{4} - \frac{1}{2q}\right) \cdot \left(1 - \frac{\ell+1}{q}\right)\left(\frac{\epsilon_{B_T^{\times}}}{16(\ell+1)} - \frac{2}{q}\right) \cdot \frac{3\epsilon_{B_T^{\times}}}{128(\ell+1)} \cdot \frac{1}{\ell+1}
\end{aligned}$$

(where the last inequality is due to Lemma 4.2.48 and Lemma 4.2.49). Plugging in $\epsilon_{B_T^{\times}} \geq \frac{\epsilon_{\mathsf{M}}}{96}$ for $\epsilon_{\mathsf{M}} \geq \frac{432\left(1 - \frac{1}{(\ell+1)^2}\right)}{q}$ and $\epsilon_{\mathsf{M}} = \frac{\epsilon_{\mathsf{U}}}{\binom{Q_h}{\ell+1}} \cdot \frac{1}{Q_{H_1}}$ (see Lemma 4.2.37) yields the theorem statement. □

5.4.5 Discussion

We note that Abe's scheme is immune to the ROS-attack[5] and thus the bounds induced by the recent polynomial-time ROS-solver [Ben+21] do not apply to this scheme. This means that unlike for the Abe-Okamoto scheme, the gap between our proof of security and the best possible bound one could hope for is rather large. It therefore remains an open question whether this gap can be closed without relying on the AGM.

[5]The ROS attack by [Ben+21] is algebraic and therefore would contradict our proof in the AGM.

Chapter 6

Sequential Security of Blind Schnorr Signatures in the AGM

This chapter is based on [KLX22a]. In this chapter, we look into the sequential security of Blind Schnorr Signatures in the AGM based on the One-More Discrete Logarithm Assumption (OMDL). In Section 6.1 we revisit the Blind Schnorr Signature scheme and show that it can be proven secure in the AGM under OMDL. This proof largely follows the proof of [FPS20], however avoiding the ROS-problem as there are no concurrent sessions. The reduction strategy is as follows: The reduction embeds its DL challenges in the public key $\mathbf{x}$ as well as in the first signer messages $\mathbf{r}$. For every signing query the adversary closes, the reduction queries its DL oracle on $r \cdot \mathbf{x}^c$ to obtain the signature part s. This allows the reduction to sign without knowing the secret key.

The reduction hopes to learn the secret key from the signatures submitted by the adversary along with the algebraic representations submitted during random oracle calls for signature generation. As the signing happens in a sequential manner, each hash query can be uniquely attributed to one of the following timestamps:

- before any signing session is opened
- during the ith signing session for some i, i.e. after Sign_1 was called the ith time and before Sign_2 is called the ith time
- in between the ith and $i+1$st signing session, i.e. after Sign_2 is called for the ith time but before Sign_1 is called for the $i+1$st time
- after the last signing session was closed or while the last (unclosed) signing session is still open

The key idea here is that most of the algebraic representation of the group element $\mathbf{r}'$ submitted to the random oracle H for a signature is actually already completely fixed w.r.t. $\mathbf{x}$ and $\mathbf{g}$, with only at most one 'unresolved' value $\mathbf{r}$ present in the representation for which a Sign_2-query can be made and a response s can still be learned. Thus, either the representation is entirely fixed at the time of the hash query, in which case the hash response and resulting signature will allow the reduction to extract the secret key with overwhelming probability, or there is dependent on value c to be queried to Sign_2 and the response. In the latter case, there must be two hash queries that eventually result in signatures being made during the same signing session, i.e. depending on the same value c. We show that in this case, the probability of linear dependence of the two algebraic representations is small and thus extraction can happen using at least one of the two signatures resulting from these hash queries.

In the end, the reduction uses the knowledge of the secret key to solve all other challenges using the values s, c it learned during its signing sessions.

The second part of this chapter (Section 6.2) is devoted to showing that this reduction is optimal with respect to the number of discrete logarithm queries made. In particular, we provide a meta-reduction that shows that if there exists a reduction that makes a smaller number of discrete logarithm queries, this reduction can be used to break the OMDL assumption efficiently. The meta-reduction's strategy is to provoke the reduction to output a system of linear equations (through the algebraic representations) that can be solved for the discrete logarithms of the challenges or directly for the secret key of the reduction. This in turn allows the meta-reduction to generate signatures and thus use the reduction to solve OMDL.

6.1 Sequential Unforgeability of Blind Schnorr Signatures

In this section we show that Schnorr's blind signature scheme satisfies sequential one-more unforgeability under the one-more DL assumption in the AGM. We first recall Schnorr's blind signature scheme BSS below. A figure depicting an interaction can be found in Figure 6.1.[1]

$$\begin{array}{lcr}
\textbf{Signer} & & \textbf{User} \\
\mathsf{sk} = x & & \mathsf{pk} = \mathbf{x} = \mathbf{g}^x \\
 & & \mathsf{m} \\
\hline
r \xleftarrow{\$} \mathbb{Z}_q & \xrightarrow{\mathbf{r}=\mathbf{g}^r} & \alpha, \beta \xleftarrow{\$} \mathbb{Z}_q \\
 & & \mathbf{r}' := \mathbf{r} \cdot \mathbf{g}^\alpha \cdot \mathbf{x}^\beta \\
 & & c' := H(\mathbf{r}', m) \\
 & \xleftarrow{c'} & c := c' + \beta \\
s := c \cdot x + r & & \mathbf{g}^s \overset{?}{=} \mathbf{r} \cdot \mathbf{x}^c \\
 & & s' := s + \alpha \\
 & & \Downarrow \\
 & & (m, \sigma = (\mathbf{r}', s'))
\end{array}$$

Figure 6.1: Interaction between Signer and User for BSS

Let $H \colon \{0,1\}^* \to \mathbb{Z}_q$ be a hash function.

- KeyGen: On input pp, KeyGen samples $x \xleftarrow{\$} \mathbb{Z}_q$ and sets $\mathbf{x} \leftarrow \mathbf{g}^x$. It sets $\mathsf{sk} \leftarrow x, \mathsf{pk} \leftarrow \mathbf{x}$ and returns $(\mathsf{sk}, \mathsf{pk})$.

- Sign_1: On input sk, Sign_1 samples $r \xleftarrow{\$} \mathbb{Z}_q$ and returns the commitment $\mathbf{r} := \mathbf{g}^r$ and the state $St_S := r$.

- Sign_2: On input a secret key sk, a state $St_S = r$ and a challenge c, Sign_2 computes $s \leftarrow c \cdot \mathsf{sk} + r \mod q$ and returns the response s.

[1]We use different letters to denote the variables in the scheme than what we used in the previous section. Our choices are in line with the standard notation for this scheme.

- User_1: On input a public key pk, a commitment $\mathbf{r}$, and a message m, User_1 does the following. It samples first samples $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_q$. Then, it computes $\mathbf{r}' \leftarrow \mathbf{r} \cdot \mathbf{g}^{\alpha} \cdot \mathsf{pk}^{\beta}$ and $c' \leftarrow H(\mathbf{r}', m), c \leftarrow c' + \beta \mod q$. It returns the challenge c and the state $St_U \leftarrow (\mathbf{r}, c, \alpha, \beta, m)$.
- User_2: On input a public key pk, a state $St_U = (\mathbf{r}, c, \alpha, \beta, m)$, and a response s, User_2 first checks if $\mathbf{g}^s = \mathbf{r} \cdot \mathbf{x}^c$ and returns $\perp$ if not. Otherwise, it computes $\mathbf{r}' \leftarrow \mathbf{r} \cdot \mathbf{g}^{\alpha} \cdot \mathsf{pk}^{\beta}$ and $s' \leftarrow s + \alpha$ and returns the signature $\sigma \leftarrow (\mathbf{r}', s')$.
- Verify: On input a public key pk, a signature $\sigma = (\mathbf{r}', s')$ and a message m, Verify computes $c' \leftarrow H(\mathbf{r}', m)$ and checks whether $g^{s'} = \mathbf{r}' \cdot \mathsf{pk}^{c'}$. If so, it returns 1; otherwise, it returns 0.

Theorem 6.1.1. *Let* M *be an algebraic adversary that runs in time* t_{M}*, makes at most* ℓ *queries to* sign_2 *in* ℓ-$\mathbf{SEQ\text{-}OMUF}_{\mathsf{BSS}}$*, and at most* Q_h *random oracle queries to* H*. Then there exists an adversary* B *such that*

$$\mathrm{Adv}^{\mathbf{OMDL}}_{\mathsf{B},\ell} \geq \mathrm{Adv}^{\mathbf{SEQ\text{-}OMUF}}_{\mathsf{M},\ell,\mathsf{BSS}} - \frac{q_h^2 + Q_h + 2}{2q},$$

and B *runs in time* $t_{\mathsf{B}} = t_{\mathsf{M}} + O(\ell + Q_h)$.

We briefly explain the proof idea. Many of the ideas and notations are reused from [FPS20]; we include them for completeness. Since M can query sign_2 a total of ℓ times, it is allowed to open $\ell+1$ sessions and close the first ℓ of them (the last session is never closed). Let $\mathbf{x}$ be the public key, and $\mathbf{r}_1, \ldots, \mathbf{r}_{\ell+1}$ be the group elements returned by sign_1. Let $(m_1^*, (\mathbf{r}_1^*, s_1^*)), \ldots, (m_{\ell+1}^*, (\mathbf{r}_{\ell+1}^*, s_{\ell+1}^*))$ be M's final outputs, i.e., $(\mathbf{r}_i^*, s_i^*)$ (where $i \in [\ell+1]$) is M's forgery on message m_i^*. Since M is algebraic, it also outputs $\mathbf{r}_i^*$'s algebraic representation $(\gamma_i^*, \xi_i^*, \rho_{i,1}^*, \ldots, \rho_{i,\ell+1}^*)$ based on $\mathbf{g}, \mathbf{x}, \mathbf{r}_1, \ldots, \mathbf{r}_{\ell+1}$, i.e.,

$$\mathbf{r}_i^* = \mathbf{g}^{\gamma_i^*} \cdot \mathbf{x}^{\xi_i^*} \cdot \prod_{j=1}^{\ell} \mathbf{r}_j^{\rho_{i,j}^*} \cdot \mathbf{r}_{\ell+1}^{\rho_{i,\ell+1}^*} = \mathbf{g}^{\gamma_i^* + \rho_{i,\ell+1}^* r_{\ell+1}} \cdot \mathbf{x}^{\xi_i^*} \cdot \prod_{j=1}^{\ell} \mathbf{r}_j^{\rho_{i,j}^*}$$

(where $r_{\ell+1} = \operatorname{dlog} \mathbf{r}_{\ell+1}$). Suppose M wins ℓ-$\mathbf{SEQ\text{-}OMUF}_{\mathsf{BSS}}$, i.e., $(\mathbf{r}_i^*, s_i^*)$ is a valid forgery on message m_i^* and we have that

$$\mathbf{g}^{s_i^*} = \mathbf{r}_i^* \cdot \mathbf{x}^{c_i^*}, \tag{6.1}$$

where $c_i^* = H(\mathbf{r}_i^*, m_i^*)$, for all $i \in [\ell+1]$. The two equations above combined yield

$$\mathbf{x}^{c_i^* + \xi_i^*} \cdot \prod_{j=1}^{\ell} \mathbf{r}_j^{\rho_{i,j}^*} = \mathbf{g}^{s_i^* - \gamma_i^* - \rho_{i,\ell+1}^* r_{\ell+1}}. \tag{6.2}$$

The reduction to ℓ-$\mathbf{OMDL}$, B, works as follows. B queries its challenge oracle $\ell+1$ times to obtain $\mathbf{x}, \mathbf{r}_1, \ldots, \mathbf{r}_\ell$, samples $r_{\ell+1} \xleftarrow{\$} \mathbb{Z}_q$ and sets $\mathbf{r}_{\ell+1} \leftarrow \mathbf{g}^{r_{\ell+1}}$, and simulates $\mathsf{sign}_1()$ by returning $\mathbf{r}_i$. To simulate $\mathsf{sign}_2(c_j)$ queries, B queries its dlog oracle and returns $s_j \leftarrow \operatorname{dlog}(\mathbf{r}_j \cdot \mathbf{x}^{c_j})$. Substituting the definition of s_j into Eq. (6.2), we get

$$\mathbf{x}^{c_i^* + \xi_i^* - \sum_{j=1}^{\ell} \rho_{i,j}^* c_j} = \mathbf{g}^{s_i^* - \gamma_i^* - \sum_{j=1}^{\ell} \rho_{i,j}^* s_j - \rho_{i,\ell+1}^* r_{\ell+1}},$$

which can be used to compute $x = \operatorname{dlog} \mathbf{x}$ as long as $\chi_i = c_i^* + \xi_i^* - \sum_{j=1}^{\ell} \rho_{i,j}^* c_j \neq 0$ for some i.

Now we need to upper bound the probability that $\chi_i = 0$ for *all* $i = 1, \ldots, \ell+1$. Recall that in [FPS20], this is reduced to the ℓ-ROS problem (which can be solved in polynomial time when $\ell \geq \lambda$, as shown in the recent work of [Ben+21]). Here, since we are in the sequential setting where the adversary must close one session before opening another, we can make a statistical argument instead.

For each message/forgery pair $(m_i^*, (\mathbf{r}_i^*, s_i^*))$, there is a corresponding random oracle query $H(\mathbf{r}_i^*, m_i^*)$. (If M does not make such a query, then $\Pr[\chi_i = 0] = 1/q$.) Call this query the *i-th special query.* Any special query is made during a session which is eventually closed (i.e., between M's j-th sign$_1$ query and j-th sign$_2$ query for some $j \in [\ell]$), or between two sessions (including before the first session), or during the last session which is never closed. If there is *any* special query (say the i-th) made between two sessions or during the last session, then it is not hard to see that all coefficients in χ_i's expression, except c_i^*, are fixed when M makes its i-th special query. On the other hand, c_i^* is a uniformly random integer in $\mathbb{Z}_q$. Therefore, $\Pr[\chi_i = 0] = 1/q$ for a single $H(\mathbf{r}_i^*, m_i^*)$ query. Otherwise, i.e., if *all* special queries are made during some session which is eventually closed, since there are ℓ such sessions and $\ell + 1$ special queries, there is at least one session (say the j_0-th) with at least two special queries (say the i-th and $(i+1)$-th) during it. At the time when M makes its $(i+1)$-th special query, i.e., when c_{i+1}^* is chosen at random from $\mathbb{Z}_q$, all coefficients in both χ_i and χ_{i+1}'s expression, except c_{i+1}^* and c_{j_0}, are fixed. Therefore, at this time whether M can come up with a c_{j_0} s.t. $\chi_i = \chi_{i+1} = 0$ is already determined, and it depends on the random choice of c_{i+1}^*. It can be shown (see the full proof) that there is at most one c_{i+1}^* s.t. the linear system $\chi_i = \chi_{i+1} = 0$ (with unknown c_{j_0}) has a solution; therefore, $\Pr[\chi_i = \chi_{i+1} = 0] \leq 1/q$ for a single pair of $H(\mathbf{r}_i^*, m_i^*)$ and $H(\mathbf{r}_{i+1}^*, m_{i+1}^*)$ queries.[2]

Proof. Let M be as in the theorem statement. Without loss of generality, we assume that M makes exactly $\ell + 1$ many sign$_1()$ and exactly ℓ many sign$_2$ queries, and returns exactly $\ell + 1$ valid signatures $(\mathbf{r}_1^*, s_1^*), \ldots, (\mathbf{r}_{\ell+1}^*, s_{\ell+1}^*)$ of messages $m_1^*, \ldots, m_{\ell+1}^*$.[3] We further assume that pairs $(m_1^*, \mathbf{r}_1^*), \ldots, (m_{\ell+1}^*, \mathbf{r}_{\ell+1}^*)$ are all distinct; otherwise M could not win ℓ-$\mathbf{SEQ\text{-}OMUF}_{\mathsf{BSS}}$ as we prove in the following simple claim.

Claim 6.1.2. *The pairs $(m_i^*, \mathbf{r}_i^*), \ldots, (m_j^*, \mathbf{r}_j^*)$ are pairwise distinct for all $i, j \in [\ell + 1]$.*

Proof. Suppose $(m_i^*, \mathbf{r}_i^*) = (m_j^*, \mathbf{r}_j^*)$ for $i \neq j \in [\ell + 1]$. If $s_i^* = s_j^*$ then M outputs two identical message/signature pairs, violating the winning condition. Otherwise it cannot be the case that both $(\mathbf{r}_i^*, s_i^*)$ and $(\mathbf{r}_i^*, s_j^*)$ are both valid signatures of m_i^*, since given m_i^* and $\mathbf{r}_i^*$, s_i^* as in the valid signature is uniquely defined (as in Eq. (6.1)). □

Let $\mathbf{x}$ be the public key, $\mathbf{r}_1, \ldots, \mathbf{r}_{\ell+1}$ be the group elements returned by sign$_1$, and M's sign$_2$ queries be sign$_2(c_1), \ldots,$ sign$_2(c_\ell)$. The proof goes by a sequence of games, which we describe below. For convenience, we set $\mathrm{Adv}_{\mathsf{M}}^{\mathbf{G_i}} := \Pr[\mathbf{G_i^{\mathsf{M}}} = 1]$.

Game $\mathbf{G_0}$. This is the ℓ-SEQ-OMUF game. We have that

$$\mathrm{Adv}_{\mathsf{M}}^{\mathbf{G_0}} = \mathrm{Adv}_{\mathsf{M},\ell,\mathsf{BSS}}^{\mathbf{SEQ\text{-}OMUF}}.$$

[2]We remark that this is essentially the 1-ROS problem, which is statistically hard.

[3]Since the security game is *sequential* OMUF, and M can make at most ℓ many sign$_2$ queries, this implies that M can make at most $\ell + 1$ many sign$_1$ queries. Obviously, any adversary who makes less than $\ell + 1$ many sign$_1$ queries, or less than ℓ many sign$_2$ queries, or returns more than $\ell + 1$ valid signatures, can be turned into an adversary who makes exactly $\ell + 1$ many sign$_1$ and exactly ℓ many sign$_2$ queries, and returns exactly $\ell + 1$ valid signatures, with the same advantage and roughly the same running time.

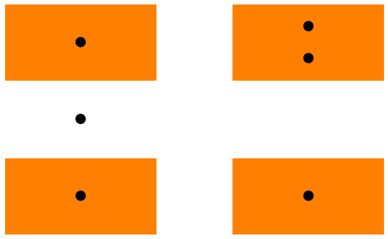


Figure 6.2: Depiction of the two cases for our reduction with hash queries denoted by • and signing sessions denoted by orange boxes. Case C_1 is on the left with a hash query made in between two signing sessions, and case C_2 on the right with two hash queries made within the same signing session.

Game $\mathbf{G_1}$. In $\mathbf{G_1}$ we make the following change. When M returns its final outputs $(m_1^*, (\mathbf{r}_1^*, s_1^*)), \ldots, (m_{\ell+1}^*, (\mathbf{r}_{\ell+1}^*, s_{\ell+1}^*))$, together with $\mathbf{r}_i^*$'s algebraic representation $(\gamma_i^*, \xi_i^*, \rho_{i,1}^*, \ldots, \rho_{i,\ell+1}^*)$ based on $\mathbf{g}, \mathbf{x}, \mathbf{r}_1, \ldots, \mathbf{r}_{\ell+1}$, for each $i \in [\ell+1]$ for which $H(\mathbf{r}_i^*, m_i^*)$ is undefined, we emulate a query $c_i^* \leftarrow H(\mathbf{r}_i^*, m_i^*)$ via lazy sampling. (If M has not seen a certain $\mathbf{r}_j$ when outputting $\mathbf{r}_i^*$, then the game naturally sets $\rho_{i,j}^* = 0$, as M is not allowed to use $\mathbf{r}_j$ as a base.) After that, we define $\chi_i \leftarrow c_i^* + \xi_i^* - \sum_{j=1}^{\ell} \rho_{i,j}^* c_j$, and abort if $\chi_i = 0$ for all i. (Note that $\rho_{i,\ell+1}^*$ does not appear in the definition of χ_i.)

$\mathbf{G_1}$ and $\mathbf{G_0}$ are identical unless $\chi_i = 0$ for all $i \in [\ell+1]$. Call this event E.

Claim 6.1.3. $\Pr[E] \leq \frac{q_h^2 + Q_h + 2}{2q}$

Proof. If M does not query $H(\mathbf{r}_i^*, m_i^*)$ for some i, then c_i^* is a uniformly random element of $\mathbb{Z}_q$ in M's view, so $\Pr[\chi_i = 0] = 1/q$.

Next we assume that M queries $H(\mathbf{r}_i^*, m_i^*)$ for all i; call such query the *i-th special query.* Since $(m_i^*, \mathbf{r}_i^*)$ pairs are all distinct, $c_i^* = H(\mathbf{r}_i^*, m_i^*)$ is a uniformly random element in $\mathbb{Z}_q$ (independent of everything else) when M makes the i-th special query. Also, $\mathbf{r}_i^*$'s algebraic representation $(\gamma_i^*, \xi_i^*, \rho_{i,1}^*, \ldots, \rho_{i,\ell+1}^*)$ is already determined when M makes its i-th special query. Any special query is made either during a session which is eventually closed (i.e., between M's j-th sign$_1$ query and j-th sign$_2$ query for some $j \in [\ell]$), or between two sessions (including before the first session), or during the last session which is never closed (i.e., after M's $(\ell+1)$-th sign$_1$ query). A depiction of the two cases can be seen in Figure 6.2 We consider these cases separately:

Case C_1. Suppose that there is *any* special query (say the i-th) made (a) between two sessions (including before the first session); say the i-th special query is made after the j_0-th sign$_2$ query and before the (j_0+1)-th sign$_1$ query, or (b) after the $(\ell+1)$-th sign$_1$ query. Consider the time when M makes its i-th special query $H(\mathbf{r}_i^*, m_i^*)$. In case (a), at this point all group elements M has seen are $\mathbf{g}, \mathbf{x}, \mathbf{r}_1, \ldots, \mathbf{r}_{j_0}$, so $\rho_{i,j_0+1}^* = \ldots = \rho_{i,\ell}^* = 0$; furthermore, the algebraic coefficients (for $\mathbf{r}_i^*$) $\xi_i^*, \rho_{i,1}^*, \ldots, \rho_{i,j_0}^*$ are all fixed. Finally, c_j (where $j \in [j_0]$) is fixed when M makes its j-th sign$_2$ query, which happens *before* M's i-th special query. Similarly, in case (b), at this point the algebraic coefficients (for $\mathbf{r}_i^*$) $\xi_i^*, \rho_{i,1}^*, \ldots, \rho_{i,\ell+1}^*$ are all fixed, and $c_1, \ldots, c_\ell$ are fixed when M makes its ℓ-th sign$_2$ query, which happens *before* M's i-th special query. This means that in both cases (a) and (b), all coefficients in χ_i's expression, except c_i^*, are fixed when M makes its i-th special query. On the other hand, c_i^* is a uniformly random element in $\mathbb{Z}_q$. Therefore, $\Pr[\chi_i = c_i^* + \xi_i^* - \sum_{j=1}^{j_0} \rho_{i,j}^* c_j = 0] = \frac{1}{q}$, for a single $H(\mathbf{r}_i^*, m_i^*)$ query. Since M makes Q_h random oracle queries in total, we have that $\Pr[\chi_i = 0 \wedge C_1] \leq \frac{Q_h}{q}$, and hence $\Pr[E \wedge C_1] \leq \frac{Q_h}{q}$.

Case C_2. Suppose that *all* special queries are made during some session which is eventually closed. Since there are ℓ such sessions and $\ell+1$ special queries, there is at least one session with at least two special queries during it; say the i-th and $(i+1)$-th special queries are made during the j_0-th session. Consider the time when M makes its $(i+1)$-st special query. At this point all group elements M has seen are $\mathbf{g}, \mathbf{x}, \mathbf{r}_1, \ldots, \mathbf{r}_{j_0}$, so $\rho^*_{i,j_0+1} = \ldots = \rho^*_{i,\ell} = 0$; furthermore, the algebraic coefficients (for $\mathbf{r}^*_i$ and $\mathbf{r}^*_{i+1}$) $\xi^*_i, \rho^*_{i,1}, \ldots, \rho^*_{i,j_0}, \xi^*_{i+1}, \rho^*_{i+1,1}, \ldots, \rho^*_{i+1,j_0}$ are all fixed. The output of M's i-th special query c^*_i is also fixed right after M makes its i-th special query, which happens *before* M's $(i+1)$-th special query. Finally, c_j (where $j \in [j_0-1]$) is fixed when M makes its j-th sign_2 query, which again happens *before* M's $(i+1)$-th special query. (This is because M's $(i+1)$-th special query is made during the j_0-th session, which is started after the j-th session is closed.) This means that all coefficients in χ_i and χ_{i+1}'s expressions, except c_{j_0} and c^*_{i+1}, are fixed when M makes its $(i+1)$-th special query.
Next consider the time when M makes its j_0-th sign_2 query (i.e., when the j_0-th session is closed). At this point c^*_{i+1} is also fixed, so the only coefficient in χ_i and χ_{i+1}'s expressions which is not fixed is c_{j_0} (to be chosen by M). In sum, the last coefficient fixed is c_{j_0} (chosen by M), and the second last coefficient fixed is c^*_{i+1} (uniformly random in $\mathbb{Z}_q$).
Consider the linear system with unknown c_{j_0}

$$\begin{cases} \chi_i = c^*_i + \xi^*_i - \sum_{j=1}^{j_0} \rho^*_{i,j} c_j = 0, \\ \chi_{i+1} = c^*_{i+1} + \xi^*_{i+1} - \sum_{j=1}^{j_0} \rho^*_{i+1,j} c_j = 0. \end{cases} \tag{6.3}$$

Denote $A := \begin{pmatrix} \rho^*_{i,j_0} & c^*_i + \xi^*_i - \sum_{j=1}^{j_0-1} \rho^*_{i,j} c_j \\ \rho^*_{i+1,j_0} & c^*_{i+1} + \xi^*_{i+1} - \sum_{j=1}^{j_0-1} \rho^*_{i+1,j} c_j \end{pmatrix}$ and $B := \begin{pmatrix} \rho^*_{i,j_0} \\ \rho^*_{i+1,j_0} \end{pmatrix}$ the augmented matrix and coefficient matrix, respectively, of (6.3). We first note that if $\rho^*_{i,j_0} = \rho^*_{i+1,j_0} = 0$ all factors in Eq. (6.3) are fixed when M makes his query. Thus, the probability that $\chi_i = \chi_{i+1} = 0$ is at most $\frac{1}{q}$ over the choice of c^*_i and c^*_{i+1}. In the following we assume that $\rho^*_{i,j_0} \neq 0$ or $\rho^*_{i+1,j_0} \neq 0$. Then

$$\begin{aligned} &\Pr[\chi_i = \chi_{i+1} = 0] = \Pr[c_{j_0} \text{ is the solution of (6.3)}] \leq \Pr[\text{(6.3) has a solution}] \\ &= \Pr[\operatorname{rank}(A) = \operatorname{rank}(B)] \leq \Pr[\operatorname{rank}(A) \leq 1] = \Pr[\det(A) = 0] \\ &= \Pr\left[\begin{array}{c} \rho^*_{i,j_0} c^*_{i+1} + \rho^*_{i,j_0}(\xi^*_{i+1} - \sum_{j=1}^{j_0-1} \rho^*_{i+1,j} c_j) \\ -\rho^*_{i+1,j_0}(c^*_i + \xi^*_i - \sum_{j=1}^{j_0-1} \rho^*_{i,j} c_j) = 0 \end{array} \right] = \frac{1}{q}, \end{aligned}$$

for a single pair of $H(\mathbf{r}^*_i, m^*_i)$ and $H(\mathbf{r}^*_{i+1}, m^*_{i+1})$ queries. (The last equation is true because when M makes its $(i+1)$-th special query, c^*_{i+1} is a uniformly random element of $\mathbb{Z}_q$, and all other coefficients are fixed.) Since M makes Q_h random oracle queries in total, we have that $\Pr[\chi_i = \chi_{i+1} = 0 \wedge C_2] \leq \frac{\binom{Q_h}{2}}{q}$, and hence $\Pr[E \wedge C_2] \leq \frac{\binom{Q_h}{2}}{q}$.

In sum, we have that (let case C_0 be "M does not make the i-th special query for some $i \in [\ell+1]$")

$$\begin{aligned} \Pr[E] &= \Pr[E \wedge C_0] + \Pr[E \wedge C_1] + \Pr[E \wedge C_2] \\ &\leq \frac{1}{q} + \frac{Q_h}{q} + \frac{\binom{Q_h}{2}}{q} = \frac{q_h^2 + Q_h + 2}{2q}. \end{aligned}$$

□

By the claim, $\mathrm{Adv}_{\mathsf{M}}^{\mathbf{G_1}} \leq \mathrm{Adv}_{\mathsf{M}}^{\mathbf{G_0}} - \frac{q_h^2 + Q_h + 2}{2q}$.

Reduction to ℓ-OMDL. We now upper bound $\mathrm{Adv}_{\mathsf{M}}^{\mathbf{G_1}}$ via a reduction B from ℓ-$\mathbf{OMDL}$. B runs on input $(\mathbb{G}, \mathbf{g}, q)$, and is given oracle access to chal and dlog. B first queries $\mathbf{x} \leftarrow$ chal() and runs M$(\mathbb{G}, \mathbf{g}, q, \mathbf{x})$. B runs the code of $\mathbf{G_1}$ except that (1) on M's j-th sign$_1$ query ($j \in [\ell]$), B returns $\mathbf{r}_j \leftarrow$ chal(); (2) on M's j-th sign$_2$ query, B returns $s_j \leftarrow$ dlog$(\mathbf{g}, \mathbf{r}_j \cdot \mathbf{x}^{c_j})$. (B answers M's $(\ell+1)$-th sign$_1$ query just as in $\mathbf{G_1}$, i.e., by sampling $r_{\ell+1} \xleftarrow{\$} \mathbb{Z}_q$ and returning $\mathbf{r}_{\ell+1} := \mathbf{g}^{r_{\ell+1}}$.) Finally, when M returns its final outputs, if there exists an $i \in [\ell+1]$ s.t. $\chi_i \neq 0$, B computes

$$x \leftarrow \frac{s_i^* - \gamma_i^* - \sum_{j=1}^{\ell} \rho_{i,j}^* s_j - \rho_{i,\ell+1}^* r_{\ell+1}}{\chi_i}$$

and

$$r_j \leftarrow s_j - c_j x,$$

and outputs $(x, r_1, \ldots, r_\ell)$. (If $\chi_i = 0$ for all i, B aborts.)

Clearly, B runs in time $t_{\mathsf{M}} + O(\ell + Q_h)$. We claim that B wins ℓ-$\mathbf{OMDL}$ if M wins $\mathbf{G_1}$. Since M is algebraic, we have that

$$\mathbf{r}_i^* = \mathbf{g}^{\gamma_i^*} \cdot \mathbf{x}^{\xi_i^*} \cdot \prod_{j=1}^{\ell} \mathbf{r}_j^{\rho_{i,j}^*} \cdot \mathbf{r}_{\ell+1}^{\rho_{i,\ell+1}^*} = \mathbf{g}^{\gamma_i^* + \rho_{i,\ell+1}^* r_{\ell+1}} \cdot \mathbf{x}^{\xi_i^*} \cdot \prod_{j=1}^{\ell} \mathbf{r}_j^{\rho_{i,j}^*}.$$

On the other hand, since M wins $\mathbf{G_1}$, i.e., $(\mathbf{r}_i^*, s_i^*)$ is a valid forgery on message m_i^*, we have that

$$\mathbf{g}^{s_i^*} = \mathbf{r}_i^* \cdot \mathbf{x}^{c_i^*}.$$

The two equations above combined yield

$$\mathbf{x}^{c_i^* + \xi_i^*} \cdot \prod_{j=1}^{\ell} \mathbf{r}_j^{\rho_{i,j}^*} = \mathbf{g}^{s_i^* - \gamma_i^* - \rho_{i,\ell+1}^* r_{\ell+1}}. \tag{6.4}$$

By definition of s_j, we have that

$$\mathbf{r}_j = \frac{\mathbf{g}^{s_j}}{\mathbf{x}^{c_j}}, \tag{6.5}$$

substituting (6.5) into (6.4), we get

$$\mathbf{x}^{\chi_i} = \mathbf{x}^{c_i^* + \xi_i^* - \sum_{j=1}^{\ell} \rho_{i,j}^* c_j} = \mathbf{g}^{s_i^* - \gamma_i^* - \sum_{j=1}^{\ell} \rho_{i,j}^* s_j - \rho_{i,\ell+1}^* r_{\ell+1}},$$

so $x = \mathrm{dlog}\,\mathbf{x}$. By (6.5) again, $r_j = \mathrm{dlog}\,\mathbf{r}_j$. This means that B wins ℓ-$\mathbf{OMDL}$. We have that

$$\mathrm{Adv}_{\mathsf{B},\ell}^{\mathbf{OMDL}} = \mathrm{Adv}_{\mathsf{M}}^{\mathbf{G_1}}.$$

We conclude that

$$\mathrm{Adv}_{\mathsf{B},\ell}^{\mathbf{OMDL}} \geq \mathrm{Adv}_{\mathsf{M},\ell,\mathsf{BSS}}^{\mathbf{SEQ\text{-}OMUF}} - \frac{q_h^2 + Q_h + 2}{2q},$$

completing the proof. □

6.2 Optimality of Our Reduction

In this section, we show an impossibility result which states (roughly) that reducing ℓ-sequential one-more unforgeability of Schnorr's blind signature scheme from ℓ-$\mathbf{OMDL}$ (as shown in section 6.1) is the best one can hope for. Concretely, we show that any algebraic reduction B that solves $(\ell-1)$-$\mathbf{OMDL}$ when provided with black-box access to a successful algebraic forger A in ℓ-$\mathbf{SEQ\text{-}OMUF}_{\mathsf{BSS}}$, can be turned into an efficient adversary M against $(\ell-1)$-$\mathbf{OMDL}$.

Algebraic Black Boxes. We consider a type of algebraic adversary that, apart from providing algebraic representations for each of its output group elements to the reduction, does not provide any further access (beyond black-box access). In particular, the reduction does not get access to the code of the adversary. This notion was previously put forth and used by Bauer et al. [BFL20].

Theorem 6.2.1. [4] *Let* B *be an algebraic reduction that satisfies the following: if algorithm* A *is an algebraic black-box algorithm that runs in time* t_{A} *then*

$$\mathrm{Adv}^{\mathbf{OMDL}}_{\mathsf{B},\ell-1} = \epsilon_{\mathsf{B}}\left(\mathrm{Adv}^{\mathbf{SEQ\text{-}OMUF}}_{\mathsf{A},\ell,\mathsf{BSS}}\right)$$

and B *runs in time* $t_{\mathsf{B}}(t_{\mathsf{A}})$*. (Here,* ϵ_{B} *and* t_{B} *are functions in the success probability and running time of* A*). Then there exists an algorithm* M *(the meta-reduction) such that*

$$\mathrm{Adv}^{\mathbf{OMDL}}_{\mathsf{M},\ell-1} \geq \epsilon_{\mathsf{B}}\left(\left(1-\frac{1}{q}\right)^{\ell}\right)$$

and M *runs in time* $t_{\mathsf{M}} = t_{\mathsf{B}}(O(\ell^3))$.

Proof Idea. We employ the meta-reduction technique [Cor02]. Our meta-reduction provides the reduction with interfaces from the one-more discrete logarithm game as well as an algebraic black box forger for blind Schnorr signatures. It plays the OMDL game itself and forwards all oracle queries and responses, thereby providing the reduction with the interfaces of an OMDL challenger. The meta-reduction (in the role of the forger) first opens and closes all signing sessions before it makes its first hash query. We note that up to this point the only outputs made by the meta-reduction in the role of the forger have been uniformly random queries to the sign_2 oracle provided by the reduction, and thus independent of the algebraic representations output by the meta-reduction during the process. It then uses the algebraic representations output by the reduction as well as the responses from sign_2 to compute the secret key through means of linear algebra. The meta-reduction then starts making queries to the random oracle provided by the reduction and generating signatures, providing the discrete logarithm of its random commitments as a representation. Thus, all representations as well as all queries made by the reduction are independent from the algebraic representations that the reduction provides to the meta-reduction but not a to a real adversary. When the meta-reduction has output its signatures to the reduction, the reduction solves the OMDL challenge. The meta-reduction at this point only forwards the solution to its own OMDL challenger and wins whenever the reduction wins.

[4]This theorem even holds for a weaker version of ℓ-$\mathbf{SEQ\text{-}OMUF}_{\mathsf{BSS}}$ where the adversary A is required to output signatures for $\ell+1$ *distinct* messages.

Doesn't This Also Contradict Section 5.3? One may ask if it is possible to apply a similar meta-reduction technique to Abe's blind signature scheme or our partially blind variant, which would contradict our result from Section 5.3. However, this is not possible as the algebraic representations output by the reduction break the witness-indistinguishability of the scheme. The meta-reduction would only be able to compute the witness used by the reduction. Thus, the combination of representations provided by the adversary and signatures provided by the adversary would be dependent on the algebraic representations provided by the reduction.

Proof of Theorem 6.2.1. We briefly sketch an unbounded algebraic adversary U that wins the game ℓ-$\mathbf{SEQ}$-$\mathbf{OMUF}$. The adversary receives the public key $\mathbf{x}$. It then opens and closes ℓ singing sessions, using a freshly sampled $c \xleftarrow{\$} \mathbb{Z}_q$. The adversary then brute-forces the discrete logarithm $\operatorname{dlog}_{\mathbf{g}} \mathbf{x}$. Finally, the adversary U picks values $r_1, \ldots, r_{\ell+1} \xleftarrow{\$} \mathbb{Z}_q$. It queries $H(\mathbf{g}^{\mathbf{r}_i}, m_i)$ with uniformly random messages m_i to the random oracle H provided by the reduction for $i = 1, \ldots, \ell + 1$. For this, it includes the algebraic representation merely as r_i. It generates signatures using the normal signing key $\operatorname{dlog}_{\mathbf{g}} \mathbf{x}$ it previously retrieved. This adversary succeeds in the $\mathbf{SEQ}$-$\mathbf{OMUF}$ game with probability 1.

We will now provide a meta-reduction that simulates the adversary U to the reduction with probability $\left(1 - \frac{1}{q}\right)^{\ell}$ over its own internal random choices.

The meta-reduction M needs to provide the oracles from $(\ell - 1)$-$\mathbf{OMDL}$ as well as simulate an adversary A playing the ℓ-$\mathbf{SEQ}$-$\mathbf{OMUF}$.

Interactions with $(\ell - 1)$-OMDL. M forwards all queries to the oracles dlog and chal made by B to the corresponding oracles provided by the interface of its own game $(\ell - 1)$-$\mathbf{OMDL}$. We denote the ith challenge value returned by chal in $(\ell - 1)$-$\mathbf{OMDL}$ with $\mathbf{U}_i$.

Public Key. When B outputs a public key $\mathbf{x}$, M obtains its algebraic representation $\overrightarrow{z_0}$ such that

$$\mathbf{x} = \mathbf{g}^{z_{0,0}} \cdot \prod_{i=1}^{\ell} \mathbf{U}_i^{z_{0,i}}.$$

Signing Sessions. To simulate the behaviour of an adversary in signing sessions $j = 1, \ldots, \ell$ of ℓ-$\mathbf{SEQ}$-$\mathbf{OMUF}$, M does as follows. It first queries $\mathbf{r}_j \leftarrow \mathsf{sign}_1()$ (as a query to B made by an algebraic forger), and obtains $\mathbf{r}_j$'s algebraic representation $\overrightarrow{z_j}$ such that

$$\mathbf{r}_j = \mathbf{g}^{z_{j,0}} \cdot \prod_{i=1}^{\ell} \mathbf{U}_i^{z_{j,i}}.$$

Then M picks $c_j \xleftarrow{\$} \mathbb{Z}_q$ and queries $s_j := \mathsf{sign}_2(c_j)$. M knows that

$$\mathbf{g}^{s_j} = \mathbf{r}_j \cdot \mathbf{x}^{c_j}.$$

If the above equation does not hold, M aborts. This is consistent with a real adversary A as then the verification equation does not hold.

Obtaining the Secret Key. Combining all equations above, M obtains a linear system of ℓ equations

$$s_j - c_j \cdot \underbrace{\left(z_{0,0} + \sum_{i=1}^{\ell} z_{0,i} \cdot \operatorname{dlog} \mathbf{U}_i \right)}_{x = \operatorname{dlog} \mathbf{x}} = \underbrace{z_{j,0} + \sum_{i=1}^{\ell} z_{j,i} \cdot \operatorname{dlog} \mathbf{U}_i}_{\operatorname{dlog} \mathbf{r}_j} \qquad (*)$$

for $j = 1, \ldots, \ell$. We will show below that either (1) with probability at least $(1 - \frac{1}{q})^\ell$ this linear system yields a solution for $\mathrm{dlog}\, \mathbf{U}_i$ $(i = 1, \ldots, \ell)$, or (2) M can compute the secret key $x = \mathrm{dlog}\, \mathbf{x}$ without solving $\mathrm{dlog}\, \mathbf{U}_i$. In case (1), M can compute $x = z_{0,0} + \sum_{i=1}^{\ell} \mathrm{dlog}\, \mathbf{U_i}^{z_{0,i}}$.

Either way, M now knows the secret key x with probability at least $(1 - \frac{1}{q})^\ell$, conditioned on B being able to answer all signing queries.

Forging Signatures. After obtaining x, M runs the standard Schnorr signing protocol $\ell + 1$ times. Concretely, for $k = 1, \ldots \ell + 1$, M picks random distinct $m_k \xleftarrow{\$} \{0,1\}^\lambda$ and $r_k \xleftarrow{\$} \mathbb{Z}_q$, computes $\mathbf{r}'_k = \mathbf{g}^{r_k}$, and queries $H(\mathbf{r}'_k, m_k)$ (using r_k the algebraic representation of $\mathbf{r}'_k$). Then M computes $s_k := r_k + x \cdot H(\mathbf{r}'_k, m_k)$ and outputs $(m_1, (\mathbf{r}'_1, s_1)), \ldots, (m_{\ell+1}, (\mathbf{r}'_{\ell+1}, s_{\ell+1}))$ to B.

Solving OMDL. Once B outputs the final outputs (supposed to be $\mathrm{dlog}\, \mathbf{U}_i$ for $i = 1, \ldots, \ell$), M forwards them to its own challenger. We note that this step is necessary because M may not have received all of $\mathrm{dlog}\, \mathbf{U}_1, \ldots, \mathrm{dlog}\, \mathbf{U}_\ell$ in the "obtaining the secret key" step.

Analysis of success probability. We now analyze the linear system $(*)$ in step 'obtaining the secret key' (recall that the unknowns are $\mathrm{dlog}\, \mathbf{U}_1, \ldots, \mathrm{dlog}\, \mathbf{U}_\ell$). Its augmented matrix is

$$A = \left(\begin{array}{ccc|c} (-c_1 \cdot z_{0,1} - z_{1,1}) & \cdots & (-c_1 \cdot z_{0,\ell} - z_{1,\ell}) & z_{1,0} - s_1 + c_1 \cdot z_{0,0} \\ \vdots & & & \vdots \\ (-c_\ell \cdot z_{0,1} - z_{\ell,1}) & \cdots & (-c_\ell \cdot z_{0,\ell} - z_{\ell,\ell}) & z_{\ell,0} - s_\ell + c_\ell \cdot z_{0,0} \end{array} \right)$$

Claim 6.2.2. *If the first $j - 1$ rows of A are linearly independent, then either (1) the j-th row is linearly independent of the previous rows with probability at least $1 - \frac{1}{q}$ (over the choice of c_j), or (2)* M *can compute the secret key x from the j-th row. (In the case of $j = 1$ or 2, a single vector is linearly independent iff it is non-zero.)*

Proof. Suppose that the first $j - 1$ rows of A are linearly independent. The algebraic representation $\overrightarrow{z_j}$ of $\mathbf{r}_j$ is provided by the reduction B, and the algebraic representation $\overrightarrow{z_0}$ of $\mathbf{x}$ has been provided by B at the beginning of the game. As ℓ-**SEQ-OMUF** is played in a sequential manner, in session j, all parameters in the first $j - 1$ rows of A (and thus their corresponding equations) are known to both B and M. We want to analyze for which possible choices of c_j the j-th row of A can be linearly expressed by the first $j - 1$ rows. This is equivalent to asking for which parameters $d_1, \ldots, d_{j-1}, c_j$ it is possible, for any $i \in [j]$, to express $(-c_j \cdot z_{0,i} - z_{j,i})$ as $d_1 \cdot (-c_1 \cdot z_{0,i} - z_{1,i}) + \ldots + d_{j-1} \cdot (-c_{j-1} \cdot z_{0,i} - z_{j-1,i})$. Thus, we are led to analyze the following linear system with unknowns $d_1, \ldots, d_{j-1}, c_j$ (where we only consider the left-hand side of the matrix A):

$$\begin{pmatrix} (-c_1 \cdot z_{0,1} - z_{1,1}) & \ldots & \cdot(-c_{j-1} \cdot z_{0,1} - z_{j-1,1}) & z_{0,1} \\ & \vdots & & \\ (-c_1 \cdot z_{0,\ell} - z_{1,\ell}) & \ldots & \cdot(-c_{j-1} \cdot z_{0,\ell} - z_{j-1,\ell}) & z_{0,\ell} \end{pmatrix} \cdot \begin{pmatrix} d_1 \\ \vdots \\ d_{j-1} \\ c_j \end{pmatrix} = \begin{pmatrix} -z_{j,1} \\ \vdots \\ -z_{j,\ell} \end{pmatrix} \qquad (**)$$

There are three possibilities:

$(**)$ **has no solution.** In this case, for any possible choice of c_j, the j-th row of A is linearly independent of the first $j - 1$ rows.

$(**)$ **has a solution and the kernel is trivial.** In this case, there is exactly one c_j such that the j-th row of A is *not* linearly independent of the first $j-1$ rows. Note however, that the coefficient matrix in $(**)$ is independent and fixed *before* M returns c_j to B. Therefore, the probability that the j-th row of A is linearly independent of the first $j-1$ rows is $1-\frac{1}{q}$.

$(**)$ **has a solution and the kernel has dimension 1.** We argue that in this case the meta-reduction M can compute the secret key x. M first solves $(**)$ and puts c_j as the variable term; that is, the values for d_i are expressed dependent on c_j. Fixing any $c_j \in \mathbb{Z}_q$, M can thus compute the corresponding $d_1, \ldots, d_{j-1}$ (which are uniquely defined). Plugging this back into $(*)$ yields that

$$z_{j,0} - s_j + c_j \cdot z_{0,0} = \sum_{i=1}^{j-1} d_i \cdot (z_{i,0} - s_i + c_i \cdot z_{0,0}),$$

so M can compute

$$s_j = -\sum_{i=1}^{j-1} d_i \cdot (z_{i,0} - s_i + c_i \cdot z_{0,0}) + z_{j,0} + c_j \cdot z_{0,0}.$$

M can thus choose two arbitrary $c_j, c_j' \in \mathbb{Z}_q$ with $c_j \neq c_j'$. It sends c_j as a challenge to B and obtains s_j. (It does this only in order to close the current session with B). It computes s_j' for c_j' according to the above formula. It obtains

$$s_j - x \cdot c_j = s_j' - x \cdot c_j' = \operatorname{dlog} \mathbf{r}_j,$$

hence $x = \frac{s_j - s_j'}{c_j - c_j'}$.

□

Given the claim, we analyze the probability that M is able to compute the secret key x. Let E_j be the event that the first j rows of A are linearly independent or that the meta-reduction can compute the secret key after round j by means of case three. We have that

$$\begin{aligned}
\Pr[\text{M can compute } x] &\geq \Pr[E_\ell] \\
&= \Pr\left[\begin{matrix}\ell\text{-th row lies outside the span} \\ \text{of the previous rows} \\ \vee \\ \text{case three applies in rows } \leq \ell\end{matrix} \quad \wedge E_{\ell-1}\right] \\
&= \Pr\left[\begin{matrix}\ell\text{-th row lies outside the span} \\ \text{of the previous rows} \\ \vee \\ \text{case three applies in rows } \leq \ell\end{matrix} \,\middle|\, E_{\ell-1}\right] \cdot \Pr[E_{\ell-1}] \\
&\geq \left(1 - \frac{1}{q}\right) \cdot \Pr[E_{\ell-1}] \geq \ldots \geq \left(1 - \frac{1}{q}\right)^{\ell-1} \cdot \Pr_{c_1 \xleftarrow{\$} \mathbb{Z}_q}[E_1] \\
&= \left(1 - \frac{1}{q}\right)^{\ell-1} \cdot \Pr_{c_1 \xleftarrow{\$} \mathbb{Z}_q}[\exists i\colon -c_1 \cdot z_{0,1} - z_{1,i} \neq 0]
\end{aligned}$$

$$\geq \left(1 - \frac{1}{q}\right)^{\ell}$$

We thus obtain that M simulates a successful algebraic adversary in ℓ-$\mathbf{SEQ\text{-}OMUF}_{\mathsf{BSS}}$ to B with probability at least $(1 - \frac{1}{q})^{\ell}$ over the choice of $c_1, \ldots, c_\ell$. Furthermore, M wins $(\ell - 1)$-$\mathbf{OMDL}$ whenever B wins $(\ell - 1)$-$\mathbf{OMDL}$. Since B solves $(\ell - 1)$-$\mathbf{OMDL}$ with probability $\epsilon_{\mathsf{B}}\left(\mathrm{Adv}^{\mathbf{SEQ\text{-}OMUF}}_{\mathsf{A},\ell,\mathsf{BSS}}\right)$ for any adversary algebraic black box adversary A against ℓ-$\mathbf{SEQ\text{-}OMUF}_{\mathsf{BSS}}$, M has advantage

$$\mathrm{Adv}^{\mathbf{OMDL}}_{\mathsf{M},\ell-1} = \epsilon_{\mathsf{B}}\left(\left(1 - \frac{1}{q}\right)^{\ell}\right)$$

Running Time. M needs to solve the linear system of equations A or the system of equations given in Claim 6.2.2. This takes time $O(\ell^3)$, for example using Gaussian elimination. The signatures (in case M is successful in computing the secret key) can be generated in time $O(\ell)$. Thus the running time of M for signature generation is $O(\ell^3)$. For a reduction B that takes time $t_{\mathsf{B}}(t_{\mathsf{A}})$ where t_{A} is the running time of an adversary, M thus takes time $t_{\mathsf{M}} = t_{\mathsf{B}}(O(\ell^3))$. □

Chapter 7

The Algebraic Wrapper

In this section, we discuss the applications of the 'algebraic wrapper', a partial instantiation of the AGM. This chapter is based on [AHK20]. We then turn to our main construction, namely we introduce what it means for a group scheme to be an algebraic wrapper in Section 7.1. A group scheme is a way to model cryptographic groups, especially those where group elements may have multiple encodings as a set of algorithms that give users a clear interface of how to interact with the group. The algebraic wrapper provides additional functionalities, such as an alternative setup algorithm that take in group elements from the base group, i.e. the group that is to be 'wrapped' by the algebraic wrapper, as well as an algorithm for unwrapping group elements (i.e. getting back the element from the base group) and extracting an algebraic explanation (the key feature that likens this group variant to the AGM). The algebraic wrapper also allows for re-randomization of group elements into other group elements with the same algebraic explanation.

We then briefly recall how to construct an algebraic wrapper using falsifiable assumptions in Section 7.2 and sketch the proofs of its key properties. We give a short overview of the construction and an intuition for why the key properties hold.

Finally, in Section 7.3 we show how the algebraic wrapper can be applied to transfer several proofs from the AGM into the standard model. We first briefly describe some common techniques that come in handy when using the algebraic wrapper in Section 7.3.1. We then warm up with some Diffie-Hellman type assumptions in Section 7.3.2 before we turn to the proof of security of Schnorr signatures from [FPS20] in Section 7.3.4 and Signed ElGamal in Section 7.3.5.

7.1 How to Simulate Extraction – Algebraic Wrappers

In order to instantiate the AGM, we need to first find a way to conceptualize what it means to be a group in a cryptographic sense. This is captured by the notion of a *group scheme* or *encoding scheme*, [GGH13]. In a nutshell, a group scheme provides an interface of algorithms abstracting the handling of a cryptographic group. As we want to prove hardness of certain problems based on hardness assumptions in an already existing base group, we incorporate this existing group into our group scheme.

More specifically, we introduce the concept of an *algebraic wrapper*, i.e. a group scheme that allows to extract a representation which – similar to the AGM – can be used in a security reduction. A similar approach has already been taken by [KP19]. [KP19] define their group scheme as a linear subspace of $\mathbb{G} \times \mathbb{G}$ for an existing group $\mathbb{G}$ in such a way that the Generalized Knowledge of Exponent

Assumption (GKEA) can be used to extract a representation (membership can for instance be tested via a symmetric pairing). Hence, that group scheme can also be viewed as an extension, or a *wrapper*, for the underlying base group. However, [KP19] relies on GKEA in the base group which more or less directly yields an equivalence between algebraic groups and GKEA. The existence of algebraic groups, however, implies the existence of extractable one-way functions with unbounded auxiliary input (since the AGM allows an additional unstructured input from $\{0,1\}^*$) which in turn conflicts with the existence of indistinguishability obfuscation, [Bit+14]. Due to this contradiction and the difficulty to assess the plausibility of knowledge-type assumptions, we strive for a weaker model which can purely be based on falsifiable assumptions.

Extraction Trapdoors. In [KP19], extraction is possible as long as the code and the randomness which where used to produce a group element are known. Since we strive to avoid knowledge-type assumptions, we need to find a different mechanism of what enables extraction. We observe that in order to reproduce proof strategies from the algebraic group model, extraction is only necessary during security reductions. Since the reduction to some assumption in the base group is in control of the group parameters of the wrapper, the reduction may use corresponding trapdoor information which we define to enable extraction. We call this notion *private extractability*.

7.1.1 Group Schemes

A group scheme or encoding scheme [GGH13] abstracts the properties of mathematical groups used in cryptography. Group schemes have recently been studied in [Alb+16; AH18; Far+18; KP19]. In contrast to traditional groups, group elements are not bound to be represented by a unique bitstring (henceforth referred to as encoding). This allows to encode auxiliary information inside group elements.

Formally, a group scheme $\mathbb{H}$ consists of the algorithms $(\mathsf{Setup}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}, \mathsf{Val}_{\mathbb{H}}, \mathsf{Add}_{\mathbb{H}}, \mathsf{Eq}_{\mathbb{H}}, \mathsf{GetID}_{\mathbb{H}})$. A group generation algorithm $\mathsf{Setup}_{\mathbb{H}}$, which given 1^λ, samples group parameters $\mathsf{pp}_{\mathbb{H}}$. A sampling algorithm $\mathsf{Sam}_{\mathbb{H}}$, given the group parameters and an additional parameter determining the exponent of the desired group element, produces an encoding corresponding to that exponent. A validation algorithm $\mathsf{Val}_{\mathbb{H}}$, given the group parameters and a bitstring, decides whether the given bitstring is a valid encoding. The algorithm $\mathsf{Add}_{\mathbb{H}}$ implements the group operation, i.e. expects the group parameters and two encodings as input and produces an encoding of the resulting group element. Since group elements do not necessarily possess unique encodings, the equality testing algorithm $\mathsf{Eq}_{\mathbb{H}}$ enables to test whether two given encodings correspond to the same group element (with respect to the given group parameters). Note that $\mathsf{Eq}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \cdot)$ defines an equivalence relation on the set of valid bitstrings. Finally, again compensating for the non-unique encodings, a group scheme describes a "get-identifier" algorithm which given the group parameters and an encoding of a group element, produces a bitstring which is unique for all encodings of the same group element.[1] Note that $\mathsf{Eq}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, a, b)$ can be implemented using $\mathsf{GetID}_{\mathbb{H}}$ by simply comparing $\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, a)$ and $\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, b)$ as bitstrings. The "get-identifier" algorithm compensates for the potential non-uniqueness of encodings and allows to extract, for instance, symmetric keys from group elements.

For a group scheme it is required that the quotient set

$$\{a \in \{0,1\}^* \mid \mathsf{Val}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, a) = 1\}/\mathsf{Eq}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \cdot)$$

[1] Previous work refers to this algorithm as "extraction algorithm". However, in order not to overload the word "extraction", we rename this algorithm in this work.

equipped with the operation defined via $\mathsf{Add}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \cdot, \cdot)$ defines a mathematical group (with overwhelming probability over the choice of $\mathsf{pp}_{\mathbb{H}} \leftarrow \mathsf{Setup}_{\mathbb{H}}(1^{\lambda})$). We say that an a is (an encoding of) a group element (relative to $\mathsf{pp}_{\mathbb{H}}$), written as $a \in \mathbb{H}$, if and only if $\mathsf{Val}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, a) = 1$.

A group scheme requires that encodings corresponding to the same group element are computationally indistinguishable as formalized by the "Switching Lemma(s)" in [Alb+16; AH18; Far+18].

Due to the non-uniqueness of encodings, we henceforth use the notation $\widehat{h}$ to denote an encoding of a group element.

7.1.2 An Algebraic Wrapper

Given a cyclic group, an algebraic wrapper is a group scheme which equips a given group $\mathbb{G}$ with a notion of extractability while preserving its group structure and complexity theoretic hardness guarantees. In particular, we achieve a property which we refer to as "private extractability" with respect to a given set of group elements in the base group. More precisely, the group generation algorithm expects group parameters $\mathsf{pp}_{\mathbb{G}}$ of the base group together with a set of group elements $\left[\overrightarrow{b}\right]_{\mathbb{G}} \in \mathbb{G}^n$ in that base group, henceforth referred to as *basis*, and produces group parameters $\mathsf{pp}_{\mathbb{H}}$ of the wrapper group together with a corresponding trapdoor $\tau_{\mathbb{H}}$. This trapdoor enables to extract a representation with respect to the basis $\left[\overrightarrow{b}\right]_{\mathbb{G}}$ from every encoding. Looking ahead, this property will allow to implement proof strategies of the algebraic group model, [FKL18].

More precisely, encodings can be seen to always carry computationally hidden representation vectors with respect to the basis $\left[\overrightarrow{b}\right]_{\mathbb{G}}$. The private extraction recovers this representation vector. Given the trapdoor, we require that it is possible to "privately" sample encodings which carry a specific dictated representation vector. We require that publicly sampled encodings and privately sampled encodings are computationally indistinguishable. We refer to this property as "switching". In order to preserve tightness of security reductions when implementing AGM proofs with our algebraic wrapper, we require a statistical re-randomization property. Furthermore, we require that representation vectors compose additively (in $\mathbb{Z}_q^n$) with the group operation and do not change when encodings are re-randomized.

Let $\mathcal{B}^n_{\mathsf{pp}_{\mathbb{G}}} := \{([1]_{\mathbb{G}}, [x_2]_{\mathbb{G}}, \ldots, [x_n]_{\mathbb{G}})^{\intercal} \in \mathbb{G}^n \mid x_2, \ldots, x_n \in \mathbb{Z}_p^{\times}\}$ be the set of what we call "legitimate basis vectors". Note that we require the first group element to be the generator of the group. This is necessary to allow public sampling.

Definition 7.1.1 (Algebraic wrapper for $\mathbb{G}$)**.** *An* algebraic wrapper $\mathbb{H}$ for $\mathbb{G}$ *is a tuple of PPT algorithms* $(\mathsf{Setup}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}, \mathsf{Val}_{\mathbb{H}}, \mathsf{Add}_{\mathbb{H}}, \mathsf{Eq}_{\mathbb{H}}, \mathsf{GetID}_{\mathbb{H}}, \mathsf{Rerand}_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}, \mathsf{PrivExt}_{\mathbb{H}}, \mathsf{Unwrap}_{\mathbb{H}})$ *such that* $(\mathsf{Setup}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}, \mathsf{Val}_{\mathbb{H}}, \mathsf{Add}_{\mathbb{H}}, \mathsf{Eq}_{\mathbb{H}}, \mathsf{GetID}_{\mathbb{H}})$ *constitutes a group scheme and the following properties are satisfied.*

$\mathbb{G}$**-wrapping.** *The algorithm* $\mathsf{Unwrap}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \cdot)$ *is deterministic and for all* $\mathsf{pp}_{\mathbb{G}} \in \mathsf{supp}(\mathsf{Setup}_{\mathbb{G}}(1^{\lambda}))$, *all* $\left[\overrightarrow{b}\right]_{\mathbb{G}} \in \mathcal{B}^n_{\mathsf{pp}_{\mathbb{G}}}$, *all* $(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}) \in \mathsf{supp}(\mathsf{Setup}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, \left[\overrightarrow{b}\right]_{\mathbb{G}}))$, $\mathsf{Unwrap}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \cdot)$ *defines a* group isomorphism *from* $\mathbb{H}$ *to* $\mathbb{G}$.

Extractability. *The algorithm* $\mathsf{PrivExt}_{\mathbb{H}}$ *is deterministic. Furthermore, for all* $\mathsf{pp}_{\mathbb{G}} \in \mathsf{supp}(\mathsf{Setup}_{\mathbb{G}}(1^{\lambda}))$, *all* $\left[\overrightarrow{b}\right]_{\mathbb{G}} \in \mathcal{B}^n_{\mathsf{pp}_{\mathbb{G}}}$, *all* $(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}) \in \mathsf{supp}(\mathsf{Setup}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, \left[\overrightarrow{b}\right]_{\mathbb{G}}))$, *all* $\widehat{h} \in \mathbb{H}$, *we require that* $\mathsf{PrivExt}_{\mathbb{H}}$ *always extracts a representation of* $[x]_{\mathbb{G}}$ *with respect to* $\left[\overrightarrow{b}\right]_{\mathbb{G}}$, *i.e. for* $\overrightarrow{z} := \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{h})$, $\left[\overrightarrow{z}^{\intercal} \cdot \overrightarrow{b}\right]_{\mathbb{G}} = \mathsf{Unwrap}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{h})$.

Correctness of extraction. *For all* $\mathsf{pp}_{\mathbb{G}} \in \mathsf{supp}(\mathsf{Setup}_{\mathbb{G}}(1^\lambda))$*, all* $\left[\overrightarrow{b}\right]_{\mathbb{G}} \in \mathcal{B}^n_{\mathsf{pp}_{\mathbb{G}}}$*, all* $(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}) \in \mathsf{supp}(\mathsf{Setup}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, \left[\overrightarrow{b}\right]_{\mathbb{G}}))$*, all* $\widehat{h_0}, \widehat{h_1} \in \mathbb{H}$*, we require that private extraction respects the group operation in the sense that for all* $\widehat{h_2} \in \mathsf{supp}(\mathsf{Add}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{h_0}, \widehat{h_1}))$*,* $\overrightarrow{z}^{(i)} := \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{h_i})$ *satisfy* $\overrightarrow{z}^{(2)} = \overrightarrow{z}^{(0)} + \overrightarrow{z}^{(1)}$*. Furthermore, for all* $\mathsf{pp}_{\mathbb{G}} \in \mathsf{supp}(\mathsf{Setup}_{\mathbb{G}}(1^\lambda))$*, all* $\left[\overrightarrow{b}\right]_{\mathbb{G}} \in \mathcal{B}^n_{\mathsf{pp}_{\mathbb{G}}}$*, all* $(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}) \in \mathsf{supp}(\mathsf{Setup}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, \left[\overrightarrow{b}\right]_{\mathbb{G}}))$*, all* $\widehat{h} \in \mathbb{H}$*, we require that re-randomization does not interfere with private extraction in the sense that for all* $\widehat{h'} \in \mathsf{supp}(\mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{h}))$*,* $\mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{h}) = \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{h'})$*.*

Correctness of sampling. *For all* $\mathsf{pp}_{\mathbb{G}} \in \mathsf{supp}(\mathsf{Setup}_{\mathbb{G}}(1^\lambda))$*, all* $\left[\overrightarrow{b}\right]_{\mathbb{G}} \in \mathcal{B}^n_{\mathsf{pp}_{\mathbb{G}}}$*, all* $(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}) \in \mathsf{supp}(\mathsf{Setup}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, \left[\overrightarrow{b}\right]_{\mathbb{G}}))$*, we require that*

- *for all* $\overrightarrow{v} \in \mathbb{Z}_q^n$*,* $\Pr[\mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, \overrightarrow{v})) = \overrightarrow{v}] = 1$*, and*
- *for all* $x \in \mathbb{Z}_p$*,* $\Pr[\mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, x \cdot \overrightarrow{e_1})) = x \cdot \overrightarrow{e_1}] = 1$*.*

k**-Switching.** *We say a PPT adversary* A *is a legitimate* k*-switching adversary if on input of base group parameters* $\mathsf{pp}_{\mathbb{G}}$*,* A *outputs two bases* $(\left[\overrightarrow{b}\right]^{(j)}_{\mathbb{G}})_{j \in \{0,1\}}$ *and two lists comprising* k *representation vectors* $(\overrightarrow{v}^{(j),(i)})_{i \in [k], j \in \{0,1\}}$ *(and an internal state* st*) such that* $\left[\overrightarrow{b}\right]^{(0)}_{\mathbb{G}}, \left[\overrightarrow{b}\right]^{(1)}_{\mathbb{G}} \in \mathcal{B}^n_{\mathsf{pp}_{\mathbb{G}}}$ *and* $\overrightarrow{v}^{(0),(i)}, \overrightarrow{v}^{(1),(i)} \in \mathbb{Z}_q^n$ *for some* $n \in \mathbb{N}$ *and all* $i \in [k]$ *and* $\left[(\overrightarrow{v}^{(0),(i)})^\intercal \cdot \overrightarrow{b}^{(0)}\right]_{\mathbb{G}} = \left[(\overrightarrow{v}^{(1),(i)})^\intercal \cdot \overrightarrow{b}^{(1)}\right]_{\mathbb{G}}$ *for all* $i \in [k]$*.*

For all legitimate k*-switching PPT adversaries* A*,*

$$\mathrm{Adv}^{k\text{-}\mathtt{switching}}_{\mathbb{H},\mathsf{A}}\lambda := \left|\Pr[\mathrm{Exp}^{k\text{-}\mathtt{switching}}_{\mathbb{H},\mathsf{A},0}(\lambda) = 1] - \Pr[\mathrm{Exp}^{k\text{-}\mathtt{switching}}_{\mathbb{H},\mathsf{A},1}(\lambda) = 1]\right|$$

is negligible, where $\mathrm{Exp}^{k\text{-}\mathtt{switching}}_{\mathbb{H},\mathsf{A},b}(\lambda)$ *(for* $b \in \{0,1\}$*) is defined in Figure 7.1.*

Statistically re-randomizable. *We say an unbounded adversary* A *is a legitimate re-randomization adversary if on input of base group parameters* $\mathsf{pp}_{\mathbb{G}}$*,* A *outputs* $\left[\overrightarrow{b}\right]_{\mathbb{G}}$ *and a state* st *such that* $\left[\overrightarrow{b}\right]_{\mathbb{G}} \in \mathcal{B}^n_{\mathsf{pp}_{\mathbb{G}}}$ *and, in a second phase,* A *on input of* $(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}, st)$ *outputs two valid encodings* $\widehat{h_0}, \widehat{h_1}$ *(and a state* st*) such that* $\mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{h_0}) = \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{h_1})$*.*

For all unbounded legitimate re-randomization adversaries A*,*

$$\mathrm{Adv}^{\mathtt{rerand}}_{\mathbb{H},\mathsf{A}}\lambda := \left|\Pr[\mathrm{Exp}^{\mathtt{rerand}}_{\mathbb{H},\mathsf{A},0}(\lambda) = 1] - \Pr[\mathrm{Exp}^{\mathtt{rerand}}_{\mathbb{H},\mathsf{A},1}(\lambda) = 1]\right| \leq \frac{1}{2^\lambda},$$

where $\mathrm{Exp}^{\mathtt{rerand}}_{\mathbb{H},\mathsf{A},b}(\lambda)$ *(for* $b \in \{0,1\}$*) is defined in Figure 7.1.*

For simplicity we require that encodings are always in $\{0,1\}^{p_{\mathsf{enc}}(\lambda)}$ for a fixed polynomial $p_{\mathsf{enc}}(\lambda)$.

The k-switching property allows to simultaneously switch the representation vectors of multiple group element encodings. It is necessary to switch all encodings simultaneously since private sampling can only be simulated knowing the trapdoor $\tau_{\mathbb{H}}$ which is not the case in $\mathrm{Exp}^{k\text{-}\mathtt{switching}}_{\mathbb{H},\mathsf{A},b}(\lambda)$.

```
$\underline{\mathrm{Exp}^{\texttt{rerand}}_{\mathbb{H},\mathsf{A},b}(\lambda)}$
$\mathsf{pp}_{\mathbb{G}} \leftarrow \mathsf{Setup}_{\mathbb{G}}(1^\lambda)$
$(\left[\overrightarrow{b}\right]_{\mathbb{G}}, st) \leftarrow \mathsf{A}(1^\lambda, \mathsf{pp}_{\mathbb{G}})$
$(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{Setup}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, \left[\overrightarrow{b}\right]_{\mathbb{G}})$
$(\widehat{h_0}, \widehat{h_1}, st) \leftarrow \mathsf{A}(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}, st)$
$\widehat{h} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{h}_b)$
return $\mathsf{A}(\widehat{h}, st)$
```

```
$\underline{\mathrm{Exp}^{k\texttt{-switching}}_{\mathbb{H},\mathsf{A},b}(\lambda)}$
$\mathsf{pp}_{\mathbb{G}} \leftarrow \mathsf{Setup}_{\mathbb{G}}(1^\lambda)$
$\big((\left[\overrightarrow{b}\right]^{(j)}_{\mathbb{G}})_{j\in\{0,1\}},$
    $(\overrightarrow{v}^{(j),(i)})_{i\in[k],j\in\{0,1\}}, st\big) \leftarrow \mathsf{A}(1^\lambda, \mathsf{pp}_{\mathbb{G}})$
$(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{Setup}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, \left[\overrightarrow{b}\right]^{(b)}_{\mathbb{G}})$
$\widehat{h^*_i} \leftarrow \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, \overrightarrow{v}^{(b),(i)})$
return $\mathsf{A}(\mathsf{pp}_{\mathbb{H}}, (\widehat{h^*_i})_{i\in[k]}, st)$
```

Figure 7.1: The re-randomization and k-switching games.

7.2 Construction

Our construction follows the ideas from [Alb+16; AH18; Far+18]. Let $\mathsf{Setup}_{\mathbb{G}}$ be a group generator for a cyclic group $\mathbb{G}$. Let $\mathcal{TD}$ be a family of hard subset membership problems. Let $\mathsf{FHE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval}, \mathsf{Rerand})$ be a perfectly correct and perfectly re-randomizable fully homomorphic public-key encryption scheme. Let $\mathsf{pp}_{\mathbb{G}}$ be group parameters for $\mathbb{G}$ and $\left[\overrightarrow{\Omega}\right]_{\mathbb{G}} \in \mathbb{G}^n$ for some $n \in \mathbb{N}$. Let $\mathtt{TD} \subseteq X$ be a subset membership problem from $\mathcal{TD}$ and $y \leftarrow X \setminus \mathtt{TD}$ and pk be a public key for FHE. For ease of notation, we define $\mathsf{pars} := (\mathsf{pp}_{\mathbb{G}}, \mathtt{TD}, y, \mathsf{pk}, \left[\overrightarrow{\Omega}\right]_{\mathbb{G}})$. Let $\Pi := (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{HSetup}, \mathsf{Ext})$ be a perfectly complete, perfectly sound and perfectly witness-indistinguishable dual-mode NIZK proof system for the language

$$\mathcal{L} := \left\{ y := (\mathsf{pars}, [x]_{\mathbb{G}}, C) \;\middle|\; \exists w\colon (y, w) \in \mathcal{R} := \mathcal{R}_1 \vee \mathcal{R}_2 \vee \mathcal{R}_3 \right\}.$$

The relations $\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3$ are defined as follows.

$$\mathcal{R}_1 = \left\{ \Big((\mathsf{pars}, [x]_{\mathbb{G}}, C), (\mathsf{sk}, \overrightarrow{v})\Big) \;\middle|\; \begin{array}{lrcl} & \mathsf{KGen}(1^\lambda; \mathsf{sk}) & = & (\mathsf{pk}, \mathsf{sk}) \\ \wedge & \mathsf{Dec}(\mathsf{sk}, C) & = & \overrightarrow{v} \\ \wedge & \left[\overrightarrow{\Omega}^{\intercal} \cdot \overrightarrow{v}\right]_{\mathbb{G}} & = & [x]_{\mathbb{G}} \end{array} \right\}$$

$$\mathcal{R}_2 = \left\{ \Big((\mathsf{pars}, [x]_{\mathbb{G}}, C), (r, \overrightarrow{v})\Big) \;\middle|\; \begin{array}{lrcl} & \mathsf{Enc}(\mathsf{pk}, \overrightarrow{v}; r) & = & C \\ \wedge & \left[\overrightarrow{\Omega}^{\intercal} \cdot \overrightarrow{v}\right]_{\mathbb{G}} & = & [x]_{\mathbb{G}} \end{array} \right\}$$

$$\mathcal{R}_3 = \left\{ \Big((\mathsf{pars}, [x]_{\mathbb{G}}, C), (w_y)\Big) \;\middle|\; (y, w_y) \in R_{\mathtt{TD}} \right\}$$

With $m'(\lambda)$ we denote a polynomial upper bound on the number of random bits $\mathsf{FHE}.\mathsf{Rerand}(1^\lambda, \cdot, \cdot)$ expects and with $m''(\lambda)$ we denote a polynomial upper bound on the number of random bits $\Pi.\mathsf{Prove}(1^\lambda, \cdot, \cdot, \cdot)$ expects. Let $\ell(\lambda) := m'(\lambda) + m''(\lambda) + 2(\lambda + 1) + 3$. Let piO be a pIO scheme for the class of samplers $\mathcal{S}^{X\text{-ind}}$ and let $\mathsf{piO}^\star_\ell$ be an ℓ-expanding pIO scheme for the class of samplers $\mathcal{S}^{X\text{-}(\star)\text{-ind}}_\ell$. Further, let $p_{\mathsf{add}}(\lambda)$ denote a polynomial upper bound on the size of addition circuits and $p_{\mathsf{rerand}}(\lambda)$ denote a polynomial upper bound on the size of re-randomization circuits which are used during the proof, see fig:cadds for details.

Our algebraic wrapper $\mathbb{H}$ is composed of the PPT algorithms $(\mathsf{Setup}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}, \mathsf{Val}_{\mathbb{H}}, \mathsf{Add}_{\mathbb{H}}, \mathsf{Eq}_{\mathbb{H}}, \mathsf{Rerand}_{\mathbb{H}}, \mathsf{PrivExt}_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}, \mathsf{GetID}_{\mathbb{H}}, \mathsf{Unwrap}_{\mathbb{H}})$ which are defined in Figures 7.2a and 7.2b. We note

```
$\mathsf{Setup}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, [\vec{b}]_{\mathbb{G}} = [(b_1, \ldots, b_n)^\intercal]_{\mathbb{G}})$
$\alpha_1 := 1, \alpha_2, \ldots, \alpha_n \leftarrow \mathbb{Z}_q^\times$
$[\vec{\Omega}]_{\mathbb{G}} := ([b_1]_{\mathbb{G}}^{\alpha_1}, \ldots, [b_n]_{\mathbb{G}}^{\alpha_n})^\intercal$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.KGen}(1^\lambda)$
$\mathsf{crs} \leftarrow \Pi.\mathsf{Setup}(1^\lambda), \mathtt{TD} \leftarrow \mathcal{TD}, y \leftarrow \overline{\mathtt{TD}}$
$\Lambda_{\mathsf{Add}} \leftarrow \mathsf{piO}(1^{p_{\mathsf{Add}}(\lambda)}, C_{\mathsf{Add}})$
$\Lambda_{\mathsf{rerand}} \leftarrow \mathsf{piO}_\ell^\star(1^{p_{\mathsf{rerand}}(\lambda)}, C_{\mathsf{rerand}})$
$\mathsf{pars} := (\mathsf{pp}_{\mathbb{G}}, \mathtt{TD}, y, \mathsf{pk}, [\vec{\Omega}]_{\mathbb{G}})$
$\mathsf{pp}_{\mathbb{H}} := (\mathsf{crs}, \mathsf{pars}, \Lambda_{\mathsf{Add}}, \Lambda_{\mathsf{rerand}})$
$\tau_{\mathbb{H}} := (\mathsf{pp}_{\mathbb{H}}, \mathsf{sk}, \alpha_1, \ldots, \alpha_n, [\vec{b}]_{\mathbb{G}})$
return $(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}})$

$\mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \vec{v} \in \mathbb{Z}_q^n)$
$C = \mathsf{Enc}(\mathsf{pk}, \vec{v}; r)$
$[x]_{\mathbb{G}} := [\vec{\Omega}^\intercal \cdot \vec{v}]_{\mathbb{G}}$
$\pi = \mathsf{Prove}(\mathsf{crs}, (\mathsf{pars}, [x]_{\mathbb{G}}, C), (r, \vec{v}))$
return $\widehat{h} := ([x]_{\mathbb{G}}, C, \pi)_{\mathbb{H}}$

$\mathsf{Val}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{h})$
parse $\widehat{x} =: ([x]_{\mathbb{G}}, C, \pi)_{\mathbb{H}}$
return $\Pi.\mathsf{Verify}(\mathsf{crs}, (\mathsf{pars}, [x]_{\mathbb{G}}, C), \pi)$

$\mathsf{Unwrap}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{h})$
if $\neg \mathsf{Val}_{\mathbb{H}}((\mathsf{crs}, \mathsf{pars}), \widehat{h})$ then
  return $\perp$
parse $\widehat{h} =: ([x]_{\mathbb{G}}, C, \pi)_{\mathbb{H}}$
return $[x]_{\mathbb{G}}$

$\mathsf{Eq}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{h_1}, \widehat{h_2})$
if $\exists j \in [2] \colon \neg \mathsf{Val}_{\mathbb{H}}((\mathsf{crs}, \mathsf{pars}), \widehat{h_j})$ then
  return $\perp$
parse $\widehat{h_i} =: ([x_i]_{\mathbb{G}}, C_i, \pi_i)_{\mathbb{H}}$
return $[x_1]_{\mathbb{G}} = [x_2]_{\mathbb{G}}$

$\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{h})$
if $\neg \mathsf{Val}_{\mathbb{H}}((\mathsf{crs}, \mathsf{pars}), \widehat{h})$ then
  return $\perp$
parse $\widehat{h} =: ([x]_{\mathbb{G}}, C, \pi)_{\mathbb{H}}$
return $[x]_{\mathbb{G}}$

$\mathsf{Add}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{h_1}, \widehat{h_2})$
return $\Lambda_{\mathsf{Add}}(\widehat{h_1}, \widehat{h_2})$

$C_{\mathsf{Add}}[\mathsf{pars}, \mathsf{crs}, \mathsf{sk}](\widehat{h_1}, \widehat{h_2}; r)$
if $\exists j \in [2] \colon \neg \mathsf{Val}_{\mathbb{H}}((\mathsf{crs}, \mathsf{pars}), \widehat{h_j})$ then
  return $\perp$
parse $\widehat{h_i} =: ([x_i]_{\mathbb{G}}, C_i, \pi_i)_{\mathbb{H}}$
$[x_{\mathsf{out}}]_{\mathbb{G}} := [x_1]_{\mathbb{G}} \cdot [x_2]_{\mathbb{G}}$
$C_{\mathsf{out}} \leftarrow \mathsf{FHE.Eval}(\mathsf{pk}, C^{(+)}[\mathbb{Z}_q^n], C_1, C_2)$
// $C^{(+)}[\mathbb{Z}_q^n]$ computes addition in $\mathbb{Z}_q^n$
$\vec{v}_i \leftarrow \mathsf{Dec}(\mathsf{sk}, C_i)$
$\vec{v}_{\mathsf{out}} := \vec{v}_1 + \vec{v}_2$
$\pi_{\mathsf{out}} \leftarrow \mathsf{Prove}(\mathsf{crs},$
    $(\mathsf{pars}, [x_{\mathsf{out}}]_{\mathbb{G}}, C_{\mathsf{out}}), (\mathsf{sk}, \vec{v}_{\mathsf{out}}))$
return $\widehat{h_{\mathsf{out}}} := ([x_{\mathsf{out}}]_{\mathbb{G}}, C_{\mathsf{out}}, \pi_{\mathsf{out}})$
```

(a) Definition of the algorithms $\mathsf{Setup}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}, \mathsf{Val}_{\mathbb{H}}, \mathsf{Eq}_{\mathbb{H}}, \mathsf{GetID}_{\mathbb{H}}, \mathsf{Add}_{\mathbb{H}}, \mathsf{Unwrap}_{\mathbb{H}}$ and the circuit C_{Add}.

```
$\mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, \vec{v} \in \mathbb{Z}_q^n)$
$\vec{v}^* := (v_1 \cdot \alpha_1^{-1}, \ldots, v_n \cdot \alpha_n^{-1})^\intercal$
$[x]_{\mathbb{G}} := [\vec{b}^\intercal \cdot \vec{v}]_{\mathbb{G}} = [\vec{\Omega}^\intercal \cdot \vec{v}^*]_{\mathbb{G}}$
$C = \mathsf{Enc}(\mathsf{pk}, \vec{v}^*; r)$
$\pi = \mathsf{Prove}(\mathsf{crs}, (\mathsf{pars}, [x]_{\mathbb{G}}, C), (\mathsf{sk}, \vec{v}^*))$
return $([x]_{\mathbb{G}}, C, \pi)_{\mathbb{H}}$

$\mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{h})$
if $\neg \mathsf{Val}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{h})$ then
  return $\perp$
parse $\widehat{h} =: ([x]_{\mathbb{G}}, C, \pi)_{\mathbb{H}}$
$(v_1, \ldots, v_n)^\intercal =: \vec{v} = \mathsf{Dec}(\mathsf{sk}, C)$
return $(v_1 \cdot \alpha_1, \ldots, v_n \cdot \alpha_n)^\intercal$

$\mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{h})$
$u \leftarrow \{0,1\}^{\ell(\lambda)}$
return $\Lambda_{\mathsf{rerand}}(\widehat{h}, u)$

$C_{\mathsf{rerand}}[\mathsf{pars}, \mathsf{crs}, \mathsf{sk}](\widehat{h}; r_1, r_2)$
if $\neg \mathsf{Val}_{\mathbb{H}}((\mathsf{crs}, \mathsf{pars}), \widehat{h})$ then
  return $\perp$
parse $\widehat{h} =: ([x]_{\mathbb{G}}, C, \pi)_{\mathbb{H}}$
$\vec{v} := \mathsf{Dec}(\mathsf{sk}, C)$
$C_{\mathsf{out}} := \mathsf{FHE.Rerand}(\mathsf{pk}, C; r_1)$
$\pi_{\mathsf{out}} \leftarrow \mathsf{Prove}(\mathsf{crs},$
    $(\mathsf{pars}, [x]_{\mathbb{G}}, C_{\mathsf{out}}), (\mathsf{sk}, \vec{v}); r_2)$
return $\widehat{h_{\mathsf{out}}} := ([x]_{\mathbb{G}}, C_{\mathsf{out}}, \pi_{\mathsf{out}})_{\mathbb{H}}$
```

(b) Definition of the algorithms $\mathsf{PrivSam}_{\mathbb{H}}, \mathsf{PrivExt}_{\mathbb{H}}, \mathsf{Rerand}_{\mathbb{H}}$ and the circuit C_{rerand}.

Figure 7.2: Algorithms of our algebraic wrapper construction.

that the algorithm $\mathsf{Val}_{\mathbb{H}}$ which is evaluated inside C_{Add} and C_{rerand} only requires a certain part of the public parameters as input. In particular, $\mathsf{Val}_{\mathbb{H}}$ does not depend on Λ_{Add} and $\Lambda_{\mathsf{rerand}}$.

During "honest" use of our algebraic wrapper, encodings carry proofs produced for relation $\mathcal{R}_1$ or relation $\mathcal{R}_2$. Relation $\mathcal{R}_2$ enables sampling without knowledge of any trapdoors. Re-randomized encodings always carry proofs for relation $\mathcal{R}_1$. Relation $\mathcal{R}_3$ is a trapdoor branch enabling simulation. Note that during "honest" use of the algebraic wrapper $y \notin \mathtt{TD}$ and, hence, due to perfect soundness of Π, there exists no proof for relation $\mathcal{R}_3$.

Differences to [Alb+16; AH18; Far+18].

[Alb+16; Far+18] introduce similar constructions of a group scheme featuring a multilinear map and of a graded encoding scheme, respectively. More precisely, [Alb+16; Far+18] equip a base group with encodings carrying auxiliary information which can be used (in an obfuscated circuit) to "multiply in the exponent". We observe that these constructions already *wrap* a given base group in the sense that "unwrapping" encodings yields a group isomorphism to the base group.

Our construction builds upon these group schemes. In order to enable extractability with respect to a dynamically chosen basis[2], our group parameters must be generated depending on that basis.

This modification, however, comes at the cost of the multilinear map functionality. This is because any implementation of a multilinear map requires knowledge of discrete logarithms of each group element encoding to a fixed generator. This is undesirable for our purposes, since we want to be able to use sets of group elements as basis which we do not know discrete logarithms of (for instance group elements provided by a reduction). Thus, we have to give up the multiplication functionality.

Furthermore, looking ahead, we crucially require that the basis can be altered via computational game hops during proofs. We solve this problem by linearly perturbing the given basis $\left[\vec{b}\right]_{\mathbb{G}}$ (except for its first entry to enable meaningful public sampling). We refer to this perturbed basis as $\left[\vec{\Omega}\right]_{\mathbb{G}}$. Our group element encodings are defined to carry representation vectors with respect to $\left[\vec{\Omega}\right]_{\mathbb{G}}$. By construction of C_{Add}, these representation vectors are treated homomorphically by the group operation.

To preserve tightness of security reductions, we additionally introduce a statistical re-randomization mechanism.

As opposed to [Alb+16; Far+18], [AH18] uses a quite different approach. In [AH18], the group scheme is constructed from scratch, meaning there is no necessity for an underlying group. The consequences are twofold. On one hand, very strong decisional assumptions can be proven to hold in the resulting group scheme. On the other hand, however, the group scheme from [AH18] lacks a $\mathsf{GetID}_{\mathbb{H}}$ algorithm limiting its applicability.

Theorem 7.2.1. *Let*

(i) $\mathsf{Setup}_{\mathbb{G}}$ *be a group generator for a cyclic group* $\mathbb{G}$*,*

(ii) $\mathcal{TD}$ *be a family of hard subset membership problems,*

(iii) $\mathsf{FHE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval}, \mathsf{Rerand})$ *be a perfectly correct and perfectly re-randomizable fully homomorphic public-key encryption scheme,*

(iv) $\Pi := (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{HSetup}, \mathsf{Ext})$ *be a perfectly complete, perfectly sound and perfectly witness-indistinguishable dual-mode NIZK proof system for the language* $\mathcal{L}$*,*

[2]With basis we mean a set of group elements in the base group.

(v) piO *be a pIO scheme for the class of samplers* $\mathcal{S}^{X\text{-ind}}$ *and*

(vi) $\text{piO}^{\star}_{\ell}$ *be an* ℓ*-expanding pIO scheme for the class of samplers* $\mathcal{S}_{\ell}^{X\text{-}(\star)\text{-ind}}$.

Then, $\mathbb{H}$ *defined in Figures 7.2a and 7.2b is an algebraic wrapper.*

Here we provide a formal proof of the statistical re-randomization property and a high-level idea for the remaining properties.

Proof Sketch. Since piO is support respecting, the algorithms defined in Figure 7.2a equip the base group $\mathbb{G}$ with non-unique encodings but respect its group structure. Thus, the tuple $(\mathsf{Setup}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}, \mathsf{Val}_{\mathbb{H}}, \mathsf{Eq}_{\mathbb{H}}, \mathsf{Add}_{\mathbb{H}}, \mathsf{GetID}_{\mathbb{H}})$ forms a group scheme such that $\mathsf{Unwrap}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \cdot)$ defines a group isomorphism from $\mathbb{H}$ to $\mathbb{G}$. Therefore, $\mathbb{H}$ satisfies $\mathbb{G}$-wrapping. Extractability follows (more or less) directly by the soundness of the consistency proof and correctness of FHE. Correctness of extraction follows by construction and the correctness of FHE and the fact that piO and $\text{piO}^{\star}_{\ell}$ are support respecting. Correctness of sampling follows directly by correctness of FHE.

Since our construction builds upon techniques developed in [Alb+16], we also employ similar strategies to remove information about the secret decryption key from the public group parameters $\mathsf{pp}_{\mathbb{H}}$. To prove $\overline{k}$/switching, we next use the IND-CPA security of FHE to remove all information about the basis from the group element encodings. Finally, the only remaining information about the basis used to setup the group parameters resides in $\left[\overrightarrow{\Omega}\right]_{\mathbb{G}}$ which thus looks uniformly random to even an unbounded adversary.

A crucial technical difference to previous work [Alb+16; AH18; Far+18] is the ability to statistically re-randomize encodings. The key ingredient enabling this is our statistically correct pIO scheme due to Theorem 2.11.3. We prove this in the following.

Lemma 7.2.2. *The group scheme* $\mathbb{H}$ *defined in Figures 7.2a and 7.2b satisfies statistical re-randomizability.*

Proof. The circuit C_{rerand} takes inputs from $\{0,1\}^{p_{\mathsf{enc}}(\lambda)}$ and expects a randomness from $\{0,1\}^{m'(\lambda)} \times \{0,1\}^{m''(\lambda)}$. We recall that $\text{piO}^{\star}_{\ell}$ is an ℓ-expanding pIO scheme for $\ell(\lambda) = m'(\lambda) + m''(\lambda) + 2(\lambda+1) + 3$. Since for every distribution X_1 over $\{0,1\}^{p_{\mathsf{enc}}(\lambda)}$, $\widetilde{\mathrm{H}}_{\infty}(U_{\ell(\lambda)} \mid X_1) = \ell(\lambda) > m'(\lambda) + m''(\lambda) + 2(\lambda+1) + 2$, the statistical distance between

$$\begin{aligned} &\left\{\Lambda_{\mathsf{rerand}} \leftarrow \text{piO}^{\star}_{\ell}(C_{\mathsf{rerand}}) \colon (\Lambda_{\mathsf{rerand}}, \Lambda_{\mathsf{rerand}}(X_1, X_2))\right\} \\ \text{and } &\left\{\Lambda_{\mathsf{rerand}} \leftarrow \text{piO}^{\star}_{\ell}(C_{\mathsf{rerand}}) \colon (\Lambda_{\mathsf{rerand}}, C_{\mathsf{rerand}}(X_1; U_{m'(\lambda)+m''(\lambda)}))\right\} \end{aligned}$$

is at most $2^{-(\lambda+1)}$.

Let $\widehat{h_0} =: ([x_0]_{\mathbb{G}}, C_0, \pi_0)_{\mathbb{H}}, \widehat{h_1} =: ([x_1]_{\mathbb{G}}, C_1, \pi_1)_{\mathbb{H}} \in \mathbb{H}$ be the encodings chosen by the adversary A. Since A is a legitimate re-randomization adversary, $\mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{h_0}) = \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{h_1})$. Due to perfect correctness of FHE and since $\alpha_1, \ldots, \alpha_n \in \mathbb{Z}_p^{\times}$ are invertible, $\mathsf{Dec}(\mathsf{sk}, C_0) = \mathsf{Dec}(\mathsf{sk}, C_1)$. Due to perfect re-randomizability of FHE, the ciphertexts produced by $C_{\mathsf{rerand}}(\widehat{h_0})$ and $C_{\mathsf{rerand}}(\widehat{h_1})$ are identically distributed. Furthermore, since $C_{\mathsf{rerand}}(\widehat{h_b})$ produces the consistency proof using the witness $(\mathsf{sk}, \mathsf{Dec}(\mathsf{sk}, C_b))$, the distributions produced by $C_{\mathsf{rerand}}(\widehat{h_0})$ and $C_{\mathsf{rerand}}(\widehat{h_1})$ are identical. Therefore, $\mathrm{Adv}^{\mathtt{rerand}}_{\mathbb{H},\mathsf{A}} \lambda \leq 2 \cdot 2^{-(\lambda+1)} = 2^{-\lambda}$.

Note that since $\mathbb{G}$ has unique encodings, A is unable to extract auxiliary information from the encodings of $\mathsf{Unwrap}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{h})$. This is crucial since such auxiliary information may be used to distinguish whether $\widehat{h_0}$ or $\widehat{h_1}$ was used to derive $\widehat{h}$. $\square$

$\square$

7.3 How to Use Algebraic Wrappers – Implementing Proofs from the AGM

In the following, we demonstrate how proof techniques from the algebraic group model can be implemented with our algebraic wrapper. Mainly, we want to use the extracted representation provided by the algebraic wrapper in a similar way as in AGM proofs. We adapt the proofs of Diffie-Hellman assumptions from [FKL18] in Section 7.3.2 and Section 7.3.3as well as the proof for the EUF-CMA security of Schnorr signatures from [FPS20] in Section 7.3.4. Before we demonstrate how to use the algebraic wrapper, we sketch two modifications which will be necessary when we replace the AGM with the algebraic wrapper.

7.3.1 Common Techniques in the Algebraic Wrapper

The Symmetrization technique. Information-theoretically, the basis[3] the algebraic wrapper enables extraction for, as well as the representation vectors inside group element encodings are known to the adversary. However, several security reductions in [FKL18] employ case distinctions where different reduction algorithms embed their challenge in different group elements. For instance, in the CDH game, the discrete logarithm challenge Z can be embedded either in $[x]_{\mathbb{H}}$ or $[y]_{\mathbb{H}}$, leading to two different security reductions. Due to the ideal properties of the AGM, both reductions simulate identically distributed games.

However, transferring this strategy directly using algebraic wrappers fails, since the two reductions are information-theoretically distinguishable depending on the choice of basis. An unbounded adversary who knows which game he is playing could therefore influence the representation of his output in such a way that it always becomes impossible for the reduction to use the representation to compute the discrete logarithm. We call such a situation a *bad case*. It is necessary that the different reduction subroutines have mutually exclusive bad cases, so that extraction is always possible in at least one game type. Thus, we need find a way that even these representations (and the basis used to generate $\mathsf{pp}_{\mathbb{H}}$) are identically distributed.

We therefore introduce a proof technique which we call *symmetrization*. We extend the basis and group element representations in such a way that the games played by different reduction subroutines are identically distributed (as they would be in the AGM). This is done by choosing additional base elements to which the reduction knows the discrete logarithm (or partial logarithms), so that these additional base elements do not add any unknowns when solving for the discrete logarithm. With this technique, we achieve that the games defined by the different reduction algorithms are identically distributed but entail different mutually-exclusive bad cases. For the CDH reduction, this means that both challenge elements $[x]_{\mathbb{H}}$ and $[y]_{\mathbb{H}}$ are contained in the basis, so that it is not known to the adversary which one is the reduction's discrete logarithm challenge. This allows to adopt the proofs from AGM.

The Origin Element Trick. Applying the algebraic wrapper to AGM proofs where an oracle (e.g. a random oracle or a signing oracle) is present, entails the need to change the representation vectors of all oracle responses. One possibility to realize this is to apply Q-`switching`, where Q denotes a polynomial upper bound on the number of oracle queries. However, as the switching property only provides computational guarantees, this naive approach results in a non-tight reduction. Since we are interested in preserving the tightness of AGM proofs when applying the algebraic wrapper, we use so-called *origin elements* from which we construct the oracle responses using the group operation. This

[3]With *basis* we mean the set of group elements in the base group to which we can extract.

cdh	sqdh	lcdh
$x, y \leftarrow \mathbb{Z}_q$	$x \leftarrow \mathbb{Z}_q$	$x, y \leftarrow \mathbb{Z}_q$
$s \leftarrow \mathsf{A}([1]_{\mathbb{G}}, [x]_{\mathbb{G}}, [y]_{\mathbb{G}})$	$s \leftarrow \mathsf{A}([1]_{\mathbb{G}}, [x]_{\mathbb{G}})$	$u, v, w, s \leftarrow \mathsf{A}([1]_{\mathbb{G}}, [x]_{\mathbb{G}}, [y]_{\mathbb{G}})$
return $s = [xy]_{\mathbb{G}}$	**return** $s = \left[x^2\right]_{\mathbb{G}}$	**return** $s = \left[u \cdot x^2 + v \cdot xy + w \cdot y^2\right]_{\mathbb{G}}$

Figure 7.3: The different types of Diffie-Hellman games shown in [FKL18]

enables to use n-`switching` for a constant number n of origin elements instead of Q-`switching` for Q oracle responses.

Limitations of Our Techniques. While our algebraic wrapper provides an extraction property that is useful for many proofs in the AGM, it also has its limitations. Mainly, the base elements to which the PrivExt algorithm can extract need to be fixed at the time of group parameter generation. Therefore, we cannot mimic reductions to assumptions with a variable amount of challenge elements, where extraction needs to be possible with respect to all these challenge elements. For instance, q-type assumptions which are used in [FKL18] to prove CCA1-security of ElGamal and the knowledge-soundness of Groth's ZK-SNARK.

Furthermore, there are security proofs in the AGM that rely on the representation used by the reduction being information-theoretically hidden from the adversary. An example for this is the tight reduction for the BLS scheme from [FKL18]. As the reduction can forge a signature for any message, it relies on the representations provided by the adversary being different from what the reduction could have computed on its own. In the AGM, it is highly unlikely that the adversary computes the forged signature in the exact same way as the reduction simulates the signing oracle, because the reduction does not provide the adversary with an algebraic representation. However, since we need to be able to extract privately from group element encodings, the group elements output by the reduction information theoretically contain algebraic representations. Therefore, information/theoretically, an adversary sees how the reduction simulates hash responses and signatures, and thus could provide signatures with a representation that is useless to the reduction.

This problem is circumvented in the Schnorr proof in Section 7.3.4 due to the representation provided by the adversary already being fixed by the time it receives a challenge through the Random Oracle. We leave it as an open problem to transfer the BLS proof to the algebraic wrapper.

Another limitation is that due to the reduction being private, we cannot use the extraction in reductions between problems in the same group. That is, our wrapper does not allow for "multi-step" reductions as in the AGM.

7.3.2 Diffie-Hellman Assumptions

We show how to adapt the security reductions for Diffie-Hellman problems from [FKL18] to our algebraic wrapper (see Figure 7.3 for the definitions). The main proof idea, namely to use the representation of the adversary's output to compute the discrete logarithm, stays the same; however, due to the nature of our wrapper, we need to apply the symmetrization technique to achieve the same distributions as in the AGM.

Theorem 7.3.1. *Let $\mathbb{G}$ be a group where the discrete logarithm is hard. Then, the computational Diffie-Hellman assumption holds in an algebraic wrapper $\mathbb{H}$ for $\mathbb{G}$ of dimension ≥ 3.*

```
$G_0$
$\mathsf{pp}_{\mathbb{G}} \leftarrow \mathsf{Setup}_{\mathbb{G}}(1^\lambda)$
$\beta_2, \beta_3 \leftarrow \mathbb{Z}_q$
$(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{Setup}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, ([1]_{\mathbb{G}}, [\beta_2]_{\mathbb{G}}, [\beta_3]_{\mathbb{G}})^\intercal)$
$x, y \leftarrow \mathbb{Z}_q$
$\widehat{1} = \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, 1))$
$\widehat{x} = \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, x))$
$\widehat{y} = \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, y))$
$s \leftarrow \mathsf{A}(\mathsf{pp}_{\mathbb{H}}, \widehat{1}, \widehat{x}, \widehat{y})$
return $\mathsf{Eq}_{\mathbb{H}}(\widehat{x}^y, s)$
```

```
$G_1$
$\mathsf{pp}_{\mathbb{G}} \leftarrow \mathsf{Setup}_{\mathbb{G}}(1^\lambda)$
$X \leftarrow \mathbb{G}$
$z \leftarrow \mathbb{Z}_q$
$(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{Setup}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, ([1]_{\mathbb{G}}, [x]_{\mathbb{G}}, [y]_{\mathbb{G}})^\intercal)$
$\widehat{1} = \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, 1))$
$\widehat{x} = \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, (0, 1, 0)^\intercal))$
$\widehat{y} = \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, (0, 0, 1)^\intercal))$
$s \leftarrow \mathsf{A}(\mathsf{pp}_{\mathbb{H}}, \widehat{1}, \widehat{x}, \widehat{y})$
return $\mathsf{Eq}_{\mathbb{H}}([xy]_{\mathbb{G}}, s)$
```

Figure 7.4: The CDH games used in the security proof. G_0 corresponds to the honest CDH-game. Games of type G_1 allow the reduction to embed its discrete logarithm challenge and extract a useful representation.

Proof. We show this as a series of games. The first game G_0 corresponds to the 'honest' CDH game in $\mathbb{H}$ where all group elements are represented in the first component. We then switch to a basis and group element representations that allow the reduction to embed its challenge and extract a useful representation. The reduction uses the extracted representation like in [FKL18] to compute the discrete logarithm. The games are shown in Figure 7.4.

Game hop from $G_0 \rightsquigarrow G_1$. The two games in Figure 7.4 are computationally indistinguishable due to re-randomizability and 2-`switching`. For the re-randomizability, we define four hybrid games H_0 to H_3 where H_0 is G_0. In H_1, we use $\mathsf{PrivSam}_{\mathbb{H}}$ for generation of $\widehat{1}$. In H_2, we additionally use $\mathsf{PrivSam}_{\mathbb{H}}$ for $\widehat{x}$ and in H_3 we additionally use $\mathsf{PrivSam}_{\mathbb{H}}$ for generation of $\widehat{y}$. A reduction between distinguishing the hybrid games and re-randomization embeds its challenge encoding in the ith group element contained in the challenge and thus simulates either H_i or H_{i+1} perfectly. As the representation vectors of the challenge group elements are the same, this reduction constitutes a legitimate adversary in the `rerand` game, and therefore $\mathrm{Adv}^{\texttt{rerand}}_{\mathsf{R},\mathbb{H}}\lambda \leq \frac{1}{2^\lambda}$. This results in

$$|\Pr[\mathrm{out}_{G_0} = 1] - \Pr[\mathrm{out}_{H_3} = 1]| \leq \frac{3}{2^\lambda}$$

We apply 2-`switching` to hop from H_3 to G_1. The reduction to 2-`switching` works as follows: Assume there is an adversary that can distinguish games G_0 and G_1. Then, a reduction chooses the two bases for the games, and the corresponding representation vectors of $\widehat{x}$ and $\widehat{y}$ as in the two games. On input of the group parameters and the vectors, it uses these elements as well as $\mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, 1)$ as a challenge to the distinguisher between the games and outputs whatever the distinguisher outputs.

Games $G_{1.0}$ and $G_{1.1}$. Here, the reduction applies the symmetrization technique to achieve identical distribution of the two games. To the CDH-adversary, the embedding of Z is information-theoretically hidden.

A reduction algorithm for the discrete logarithm simulates the games of type G_1 by replacing X with its discrete logarithm challenge. If the CDH adversary A wins the game, the reduction extracts $(\eta, \iota, \theta) = \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, s)$. For a valid solution s, the following holds in $G_{1.0}$:

$$x \cdot y = \eta + \theta \cdot x + \iota \cdot y \qquad \Leftrightarrow$$

```
$G_0$
$\mathsf{pp}_{\mathbb{G}} \leftarrow \mathsf{Setup}_{\mathbb{G}}(1^\lambda)$
$x \leftarrow \mathbb{Z}_q$
$\beta \leftarrow \mathbb{Z}_q$
$\mathsf{pp}_{\mathbb{H}} \leftarrow \mathsf{Setup}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, (1,\beta)^\intercal)$
$\widehat{1} = \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, 1))$
$\widehat{x} = \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, x))$
$s \leftarrow \mathsf{A}(\mathsf{pp}_{\mathbb{H}}, \widehat{1}, \widehat{x})$
**return** $\mathsf{Eq}_{\mathbb{H}}(s, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, (x^2, 0)^\intercal))$
```

```
$G_1$
$\mathsf{pp}_{\mathbb{G}} \leftarrow \mathsf{Setup}_{\mathbb{G}}(1^\lambda)$
$X \leftarrow \mathbb{G}$
$\mathsf{pp}_{\mathbb{H}} \leftarrow \mathsf{Setup}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, (1,X)^\intercal)$
$\widehat{1} = \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, (1,0)^\intercal))$
$\widehat{x} = \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, (0,1)^\intercal))$
$s \leftarrow \mathsf{A}(\mathsf{pp}_{\mathbb{H}}, \widehat{1}, \widehat{x})$
**return** $\mathsf{Eq}_{\mathbb{H}}(s, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, (x^2, 0)^\intercal))$
```

Figure 7.5: Square Diffie-Hellman games

$$x = \frac{\eta + \iota \cdot y}{y - \theta}$$

The bad case here is if $y = \theta$, in which case the reduction can not solve for x. However, if the challenge is embedded in $\widehat{y}$ (as in $G_{1.1}$), we can solve as follows:

$$y = \frac{n + \theta \cdot x}{x - \iota}$$

If both $\iota = x$ and $\theta = y$, then $\eta = -xy$, because

$$\begin{aligned} \eta + \theta \cdot x + \iota \cdot y &= xy && \Leftrightarrow \\ \eta + y \cdot x + x \cdot y &= xy && \Leftrightarrow \\ \eta &= -xy \end{aligned}$$

The reduction can check in either game type whether both bad cases appeared by checking if $[xy]_{\mathbb{G}} = \mathsf{Unwrap}_{\mathbb{H}}(s) = X^\theta = y^\iota$. In this case, the reduction can solve for x or y (depending on where the challenge was embedded) as $x = -\frac{\eta}{y}$ and $y = -\frac{\eta}{x}$.

As the games $G_{1.0}$ and $G_{1.1}$ are identically distributed, the probability that the discrete logarithm was embedded in such a way that it is possible to extract is at least $\frac{1}{2}$. Thus

$$\mathrm{Adv}^{\mathbf{DLOG}}_{\mathcal{R},\mathbb{G}}\lambda \geq \frac{\mathrm{Adv}^{\mathsf{CDH}}_{\mathsf{A},\mathbb{H}}\lambda - 2 \cdot \mathrm{Adv}^{\mathtt{2\text{-}switching}}_{\mathsf{A}',\mathbb{H}}\lambda - \frac{3}{2^\lambda}}{2}$$

which concludes the proof. □

7.3.3 More Diffie-Hellman Proofs

We further show that [FKL18]'s proof for the square Diffie-Hellman and linear combination Diffie-Hellman assumptions can be transferred to the algebraic wrapper.

Theorem 7.3.2. *Let $\mathbb{G}$ be a group where the discrete logarithm is hard. Then, the square Diffie-Hellman assumption holds in an algebraic wrapper $\mathbb{H}$ of dimension ≥ 2 for $\mathbb{G}$.*

Proof. Under 1-`switching` and re-randomizability, the games in Figure 7.5 are computationally indistinguishable. The game hop works the same as for CDH. A reduction can embed its discrete logarithm challenge as X. It can check the solution by solving for the discrete logarithm of X (if it is impossible to solve for x, it returns 0). The discrete logarithm solving works as follows.

For a successful adversary in G_1, the reduction can solve for x because $x^2 = \eta + \theta \cdot x$ for $(\eta, \theta) = \mathsf{PrivExt}_{\mathbb{H}}(s)$. For a correct square Diffie-Hellman solution this quadratic equation has at least one solution. Let x_1, x_2 the (possibly equal) solutions to the equation. The reduction can compute $[x_1]_{\mathbb{G}}$ and $[x_2]_{\mathbb{G}}$ to check which of the two possible solutions is the correct one. Thus,

$$\mathrm{Adv}^{\mathbf{DLOG}}_{\mathcal{R},\mathbb{G}}\lambda \geq \mathrm{Adv}^{\mathsf{SQ\text{-}DH}}_{\mathsf{A},\mathbb{H}}\lambda - \mathrm{Adv}^{\mathtt{1\text{-}switching}}_{\mathsf{A}',\mathbb{H}}\lambda - \frac{2}{2^\lambda}$$

□

Theorem 7.3.3. *Let $\mathbb{G}$ be a group where $\mathbf{DLOG}$ is hard and $\mathbb{H}$ be an algebraic wrapper of dimension ≥ 3 for $\mathbb{G}$. Then, the linear-combination Diffie-Hellman problem is hard in $\mathbb{H}$.*

Proof. Similar to the above theorems, we embed the DLOG-challenge as one of the base elements. The games are similar to Figure 7.4. When the adversary outputs $\mathbf{z}, u, v, w$, we extract η, θ, ι s.t. $\eta + x \cdot \theta + y \cdot \iota = z$. In the case that $u \neq 0$, we can solve the resulting quadratic equation for x (with probability $\frac{1}{2}$ this is where the DLOG was embedded). In the case that $w \neq 0$, we solve for y in a similar fashion. As the games are identically distributed, (even an unbounded adversary cannot decide where the DLOG challenge is embedded), we can solve for the DLOG with probability $\frac{1}{2}$ in these cases. In the case that $w = 0$ and $u = 0$, we can either solve for

$$x = \frac{-\eta - \iota \cdot y}{vy - \theta}$$

or for

$$y = \frac{-\eta - \theta \cdot x}{vx - \iota}.$$

This is analogous to the reduction for CDH. Thus the probability that a reduction $\mathcal{R}$ solves the DLOG problem is

$$\mathrm{Adv}^{\mathbf{DLOG}}_{\mathcal{R},\mathbb{G}}\lambda \geq \frac{\mathrm{Adv}^{\mathsf{LC\text{-}DH}}_{\mathsf{A},\mathbb{H}}\lambda - 2 \cdot \mathrm{Adv}^{\mathtt{2\text{-}switching}}_{\mathsf{A}',\mathbb{H}}\lambda - \frac{3}{2^\lambda}}{2}$$

□

7.3.4 Schnorr Signatures

We apply the algebraic wrapper to mimic the proof of tight EUF-CMA security of Schnorr Signatures from [FPS20].

Theorem 7.3.4. *Let* $\mathsf{Setup}_{\mathbb{G}}$ *be a group generator for a cyclic group $\mathbb{G}$ such that $\mathbf{DLOG}$ is hard relative to* $\mathsf{Setup}_{\mathbb{G}}$ *and let $\mathbb{H}$ be an algebraic wrapper of dimension ≥ 2 for $\mathbb{G}$. Then, the Schnorr signature scheme in $\mathbb{H}$ (Figure 7.6) is tightly EUF-CMA secure in the random oracle model.*

More precisely, for all PPT adversaries A*, there exists a PPT adversary* B *and a legitimate switching adversary* A'' *both running in time $T(\mathsf{B}) \approx T(\mathsf{A}) + (q_s + q_h) \cdot \mathsf{poly}(\lambda)$ and $T(\mathsf{A}'') \approx T(\mathsf{A}) + (q_s + q_h) \cdot \mathsf{poly}(\lambda)$ such that*

$$\mathrm{Adv}^{\mathsf{euf\text{-}cma}}_{\Sigma_{\mathtt{schnorr}},\mathsf{A}}\lambda \leq \mathrm{Adv}^{\mathbf{DLOG}}_{\mathsf{B},\mathbb{G}}\lambda + \mathrm{Adv}^{\mathtt{1\text{-}switching}}_{\mathsf{A}'',\mathbb{H}}\lambda + \frac{O(q_s(q_s + q_h))}{2^\lambda},$$

where q_h is a polynomial upper bound on the number of random oracle queries, q_s is a polynomial upper bound on the number of signing queries and poly *is a polynomial independent of q_s and q_h.*

```
KGen(pp_H)
    $x \leftarrow \mathbb{Z}_q$
    $\widehat{1} := \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, 1))$
    $\widehat{X} := \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, x))$
    $\mathsf{pk} := (\mathsf{pp}_{\mathbb{H}}, \widehat{1}, \widehat{X})$
    $\mathsf{sk} := (\mathsf{pk}, x)$
    return $(\mathsf{pk}, \mathsf{sk})$

Sign(sk, m)
    $r \leftarrow \mathbb{Z}_q$
    $\widehat{R} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, r))$
    $c := H(\widehat{R}, m)$
    $s := r + c \cdot x \mod q$
    return $\sigma := (\widehat{R}, s)$

Ver(pk = (pp_H, 1^, X^), m, σ = (R^, s))
    $c := H(\widehat{R}, m)$
    return $\mathsf{Eq}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, s), \widehat{R} \cdot \widehat{X}^c)$
```

Figure 7.6: The Schnorr signature scheme $\Sigma_{\texttt{schnorr}}$. Note that to compensate for the non-uniqueness of group element encodings, the (random oracle) hash value of a group element encoding is computed for the unique identifier produced by $\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \cdot)$.

```
$\mathrm{Exp}^{\mathsf{euf\text{-}cma}}_{\Sigma_{\texttt{schnorr}},\mathsf{A}}(\lambda)$
    $\mathsf{pp}_{\mathbb{G}} \leftarrow \mathsf{Setup}_{\mathbb{G}}(1^\lambda)$
    $(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{Setup}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, ([1]_{\mathbb{G}}, [\beta_2]_{\mathbb{G}})^\intercal)$
    $x \leftarrow \mathbb{Z}_p$
    $\xi_1 \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, 1))$
    $\xi_2 \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, x))$
    $\mathsf{pk} := (\mathsf{pp}_{\mathbb{H}}, \xi_1, \xi_2)$
    $Q := \varnothing, T := []$
    $(m^*, \widehat{R^*}, s^*) \leftarrow \mathsf{A}^{H,\mathsf{Sign}}(1^\lambda, \mathsf{pk})$
    if $m^* \in Q$ then return 0
    $c^* = H(\widehat{R^*}, m^*)$
    return $\mathsf{Eq}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, s^*), \widehat{R^*} \cdot \xi_2^{c^*})$

$H(\widehat{R}, m)$
    if $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), m)] = \perp$ then
        $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), m)] \leftarrow \mathbb{Z}_p$
    return $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), m)]$

Sign(m)
    $r \leftarrow \mathbb{Z}_p$
    $\widehat{R} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, r))$
    $c := H(\widehat{R}, m)$
    $s := r + cx$
    $Q := Q \cup \{m\}$
    return $(\widehat{R}, s)$
```

Figure 7.7: The EUF-CMA game for Schnorr signatures. Note that β_2 can be chosen arbitrarily.

```
Game 1
    $\mathsf{pp}_{\mathbb{G}} \leftarrow \mathsf{Setup}_{\mathbb{G}}(1^\lambda)$
    $(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{Setup}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, ([1]_{\mathbb{G}}, [\beta_2]_{\mathbb{G}})^\intercal)$
    $x \leftarrow \mathbb{Z}_p$
    $\xi_1 \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, 1))$
    $\xi_2 \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, x))$
    $\mathsf{pk} := (\mathsf{pp}_{\mathbb{H}}, \xi_1, \xi_2)$
    $Q := \varnothing, T := []$
    $(m^*, \widehat{R^*}, s^*) \leftarrow \mathsf{A}^{H,\mathsf{Sign}}(1^\lambda, \mathsf{pk})$
    if $m^* \in Q$ then return 0
    $c^* = H(\widehat{R^*}, m^*)$
    return $\mathsf{Eq}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, s), \widehat{R^*} \cdot \xi_2^{c^*})$

$H(\widehat{R}, m)$
    if $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), m)] = \perp$ then
        $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), m)] \leftarrow \mathbb{Z}_p$
    return $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), m)]$

Sign(m)
    $r \leftarrow \mathbb{Z}_p$
    $\widehat{R_1} \leftarrow \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, r)$
    $c := H(\widehat{R_1}, m)$
    $s := r + cx$
    $\widehat{R_2} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, s - cx))$
    $Q := Q \cup \{m\}$
    return $(\widehat{R_2}, s)$
```

Figure 7.8: The randomness for signatures is drawn using an x-component. Game 1 is identically distributed to $\mathrm{Exp}^{\mathsf{euf\text{-}cma}}_{\Sigma_{\texttt{schnorr}},\mathsf{A}}(\lambda)$.

```
Game 2
$\mathsf{pp}_{\mathbb{G}} \leftarrow \mathsf{Setup}_{\mathbb{G}}(1^\lambda)$
$(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{Setup}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, ([1]_{\mathbb{G}}, [\beta_2]_{\mathbb{G}})^\intercal)$
$x \leftarrow \mathbb{Z}_p$
$\xi_1 \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, 1))$
$\xi_2 \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, x))$
$\mathsf{pk} := (\mathsf{pp}_{\mathbb{H}}, \xi_1, \xi_2)$
$Q := \varnothing, T := []$
$(m^*, \widehat{R^*}, s^*) \leftarrow \mathsf{A}^{H,\mathsf{Sign}}(1^\lambda, \mathsf{pk})$
if $m^* \in Q$ then return 0
$c^* = H(\widehat{R^*}, m^*)$
return $\mathsf{Eq}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, [s^*]_{\mathbb{H}}, \widehat{R^*} \cdot \xi_2^{c^*})$

$H(\widehat{R}, m)$
if $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), m)] = \bot$ then
    $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), m)] \leftarrow \mathbb{Z}_p$
return $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), m)]$

$\mathsf{Sign}(m)$
$r \leftarrow \mathbb{Z}_p$
$\widehat{R_1} \leftarrow \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, r)$
$c := H(\widehat{R_1}, m)$
$s := r + cx$
$\widehat{R_2} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \xi_1^s \cdot \xi_2^{-c})$
$Q := Q \cup \{m\}$
return $(\widehat{R_2}, s)$
```

Figure 7.9: We construct the randomness from two origin elements. This is statistically close to Game 1 due to the re-randomizability.

Proof. We use the origin element trick to avoid using q_s-`switching` (see Definition 7.1.1) which would compromise tightness of the reduction. Figure 7.7 shows the EUF-CMA game with Schnorr signatures instantiated with the algebraic wrapper. We note that for groups with non-unique encodings, the hash function hashes the unique identifier returned by $\mathsf{GetID}_{\mathbb{H}}$, hence, encodings corresponding to the same group element are mapped to the same hash value. The reduction uses a table T to keep track of previously made hash queries and their responses, as well as a set Q to keep track of the messages the adversary has requested signatures for. We show Game 1 in Figure 7.8.

Game hop from $\mathrm{Exp}^{\mathsf{euf\text{-}cma}}_{\Sigma_{\mathsf{schnorr}},\mathsf{A}}(\lambda) \rightsquigarrow$**Game 1.** Since $r = s - cx \bmod q$ and hence $\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R_1}) = \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R_2})$, these two games are identically distributed.

Game hop Game 1$\rightsquigarrow$Game 2. In Game 2(see Figure 7.9), we construct $\widehat{R_2}$ from origin elements through the group operation instead of sampling. This game hop is justified by the re-randomizability of the algebraic wrapper. A reduction to this property works as a series of $q_s + 1$ hybrids where H_0 is Game 1, where q_s denotes a polynomial upper bound on the number of signing queries. In H_i, the first i signature queries are answered as in Game 2 and the $i+1$-th to q_s-th signature queries are answered as in Game 1. In the last hybrid, the public key is also changed to private sampling. If there is an (unbounded) adversary that distinguishes H_i and H_{i+1}, the reduction A′ uses this adversary to attack the re-randomizability as follows. On input of base group parameters $\mathsf{pp}_{\mathbb{G}}$, A′ picks a basis $([1]_{\mathbb{G}}, [\beta_2]_{\mathbb{G}})$ and gives it to the `rerand` challenger. It receives public parameters and the trapdoor. Then, it simulates H_i to the adversary for the first i signature queries, i.e. it samples $\widehat{R_{2,j}} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \xi_1^{s_j} \cdot \xi_2^{-c_j})$ for $j < i$. For the $i+1$-th signature query, A′ sends the two elements $\widehat{h_0} = \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, s_{i+1} - c_{i+1} \cdot x)$ and $\widehat{h_1} = \xi_1^{s_{i+1}} \cdot \xi_2^{-c_{i+1}}$ to the challenger and receives a challenge $\widehat{C}$. It uses this challenge $\widehat{C}$ as $\widehat{R_{2,i+1}}$ to answer the $i+1$-th hash query and responds to the remaining queries as in H_{i+1}, i.e. it samples $\widehat{R_j} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, s_j - c_j \cdot x))$ for $j > i+1$. Depending on the challenge encoding $\widehat{C}$, A′ either simulates H_i or H_{i+1} perfectly and outputs the output of the corresponding game.

In hybrid H_{q_s}, all signature queries are answered as in game Game 2. The last step to game H_{q_s+1} =Game 2 changes how ξ_2 (which is part of the public key) is sampled. An adversary distinguishing H_{q_s} and H_{q_s+1} can be used to build an adversary A′ in `rerand` similarly as above. More precisely, A′

```
Game 3
$\mathsf{pp}_{\mathbb{G}} \leftarrow \mathsf{Setup}_{\mathbb{G}}(1^\lambda)$
$x \leftarrow \mathbb{Z}_p$
$(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{Setup}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, ([1]_{\mathbb{G}}, [x]_{\mathbb{G}})^{\intercal})$
$Q := \varnothing, T := []$
$\xi_1 \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, 1))$
$\xi_2 = \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, (0,1)^{\intercal}))$
$\mathsf{pk} := (\mathsf{pp}_{\mathbb{H}}, \xi_1, \xi_2)$
$(m^*, \widehat{R^*}, s^*) \leftarrow \mathsf{A}^{H,\mathsf{Sign}}(1^\lambda, \mathsf{pk})$
if $m^* \in Q$ then return 0
$c^* = H(\widehat{R^*}, m^*)$
return $\mathsf{Eq}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, [s^*]_{\mathbb{H}}, \widehat{R^*} \cdot \xi_2^{c^*})$

$H(\widehat{R}, m)$
if $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), m)] = \perp$ then
    $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), m)] \leftarrow \mathbb{Z}_p$
return $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), m)]$

$\mathsf{Sign}(m)$
$r \leftarrow \mathbb{Z}_p$
$\widehat{R_1} \leftarrow \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, r)$
$c := H(\widehat{R_1}, m)$
$s := r + cx$
$\widehat{R_2} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \xi_1^s \cdot \xi_2^{-c})$
$Q := Q \cup \{m\}$
return $(\widehat{R_2}, s)$
```

Figure 7.10: We switch the basis and the representation of ξ_2.

outputs the encodings $\widehat{h_0} \leftarrow \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, x)$ and $\widehat{h_1} \leftarrow \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, x)$ (note that $\tau_{\mathbb{H}}$ is known during the `rerand` game) and uses the challenge encoding from the `rerand` challenger as ξ_2. We note that this last game hop paves the way to apply `1-switching`.

Due to correctness of sampling and correctness of extraction, the representation vectors of the elements used in the `rerand` game are identical and hence A′ is a legitimate adversary in the `rerand` game and its advantage is upper bounded by $\frac{1}{2^\lambda}$. Therefore,

$$|\Pr[\mathrm{out}_1 = 1] - \Pr[\mathrm{out}_2 = 1]| \leq \frac{q_s + 1}{2^\lambda}.$$

Game hop Game 2⇝Game 3. In game Game 3 (see Figure 7.10) we switch the basis and the representation of the origin element ξ_2. This game hop is justified by `1-switching`. Let A be an adversary distinguishing Game 2 and Game 3. We construct an adversary A″ on `1-switching` as follows. Initially, A″ on input of $\mathsf{pp}_{\mathbb{G}}$, outputs $\left[\overrightarrow{b}\right]_{\mathbb{G}}^{(Game2)} = [(1, \beta_2)^{\intercal}]_{\mathbb{G}}$ and $\left[\overrightarrow{b}\right]_{\mathbb{G}}^{(Game3)} = [(1, x)^{\intercal}]_{\mathbb{G}}$ and the representation vectors $\overrightarrow{v^{(Game2)}} := (x, 0)^{\intercal}$ and $\overrightarrow{v^{(Game3)}} := (0, 1)^{\intercal}$. In return, A″ receives public parameters $\mathsf{pp}_{\mathbb{H}}$ and an encoding $\widehat{C}$ and samples $\xi_2 \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{C})$. The trapdoor $\tau_{\mathbb{H}}$ is not necessary to simulate Game 2 and Game 3 (except for sampling ξ_2). Hence, A″ perfectly simulates Game 2 or Game 3 for A depending on the challenge provided by the `1-switching` challenger. Thus, $|\Pr[\mathrm{out}_3 = 1] - \Pr[\mathrm{out}_2 = 1]| \leq \mathrm{Adv}_{\mathbb{H},\mathsf{A}''}^{\texttt{1-switching}}\lambda$. Note that A″ is a legitimate switching adversary since $[(1, \beta_2)]_{\mathbb{G}} \cdot (x, 0)^{\intercal} = [x]_{\mathbb{G}} = [(1, x)]_{\mathbb{G}} \cdot (0, 1)^{\intercal}$ and hence $\mathrm{Adv}_{\mathbb{H},\mathsf{A}''}^{\texttt{1-switching}}\lambda$ is negligible.

Game hop Game 3⇝Game 4. In Game 4 (see Figure 7.11), we introduce a list U to keep track of the representations of group elements used in Random Oracle queries. The games Game 3 and Game 4 differ in the fact that Game 4 extracts the representation vectors contained in the encoding of a group element when this group element message tuple is queried for the first time and stores this representation in a list. Furthermore, Game 4 introduces an abort condition which is triggered if the representation of $\widehat{R^*}$ originally used to query the random oracle on $(\widehat{R^*}, m^*)$ already contained the response in the second component ζ^*. This corresponds to the game hop from G_0 to G_1 in [FPS20]. The game only aborts if the hash $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R^*}), m^*)]$ is the same as the second component ζ^* of the representation

```
Game 4
$\mathsf{pp}_{\mathbb{G}} \leftarrow \mathsf{Setup}_{\mathbb{G}}(1^\lambda)$
$x \leftarrow \mathbb{Z}_p$
$(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{Setup}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, ([1]_{\mathbb{G}}, [x]_{\mathbb{G}})^\intercal)$
$Q := \varnothing, T := [], U := []$
$\xi_1 = \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, 1))$
$\xi_2 = \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, (0,1)^\intercal))$
$\mathsf{pk} := (\mathsf{pp}_{\mathbb{H}}, \xi_1, \xi_2)$
$(m^*, \widehat{R^*}, s^*) \leftarrow \mathsf{A}^{H,\mathsf{Sign}}(1^\lambda, \mathsf{pk})$
if $m^* \in Q$ then return $0$
if $U[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R^*}), m^*)] \neq \bot$ then
    $(\gamma^*, \zeta^*) := U[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R^*}), m^*)]$
    if $\zeta^* = -T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R^*}), m^*)]$ then return $0$
$c^* = H(\widehat{R^*}, m^*)$
return $\mathsf{Eq}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, [s^*]_{\mathbb{H}}, \widehat{R^*} \cdot \xi_2^{c^*})$
```

```
$H(\widehat{R}, m)$
if $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), m)] = \bot$ then
    $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), m)] \leftarrow \mathbb{Z}_p$
    $U[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), m)] = \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{R})$
return $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), m)]$

$\mathsf{Sign}(m)$
$r \leftarrow \mathbb{Z}_p$
$\widehat{R_1} \leftarrow \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, r)$
$c := H(\widehat{R_1}, m)$
$s := r + cx$
$\widehat{R_2} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \xi_1^s \cdot \xi_2^{-c})$
$Q := Q \cup \{m\}$
return $(\widehat{R_2}, s)$
```

Figure 7.11: Game 4 corresponds to G_1 in [FPS20].

extracted from $\widehat{R^*}$. Since the hash $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R^*}), m^*)]$ is chosen uniformly at random *after* the representation (γ^*, ζ^*) is fixed, the probability that an unbounded adversary can find such an $(\widehat{R^*}, m^*)$ is upper bounded by $\frac{q_h}{q} \leq \frac{q_h}{2^\lambda}$, where q_h denotes a polynomial upper bound on the number of random oracle queries. Hence, $|\Pr[\mathrm{out}_4 = 1] - \Pr[\mathrm{out}_3 = 1]| \leq \frac{q_h}{2^\lambda}$.

Game hop Game 4⇝Game 5. In game Game 5 (see Figure 7.12), we change how signature queries are answered such that it is not necessary anymore to know the discrete logarithm of the public key. This game hop corresponds to the hop from G_1 to G_2 in [FPS20]. On one hand, since $\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R_1}) = \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R_2})$, replacing $\widehat{R_1}$ with $\widehat{R_2}$ does not change the distribution. On the other hand, as we are only able to answer a signing query if we can program the random oracle at $(\widehat{R_2}, m)$ (for randomly chosen $\widehat{R_2}$), the signing oracle has to abort in case the hash was already queried before. Since $\widehat{R_2}$ is a independently sampled uniformly random group element, this happens only with probability $\frac{1}{q} \leq \frac{1}{2^\lambda}$. Hence, by a union bound, this abort occurs at most with probability $\frac{q_s(q_s+q_h)}{2^\lambda}$ cases, where q_s denotes a polynomial upper bound on the number of signing queries and q_h denotes a polynomial upper bound on the number of random oracle queries. Conditioned on the event that no abort occurs, Game 4 and Game 5 are distributed identically. Hence, by the Difference Lemma due to Shoup [Sho04], we have $|\Pr[\mathrm{out}_5 = 1] - \Pr[\mathrm{out}_4 = 1]| \leq \frac{q_s(q_s+q_h)}{2^\lambda}$. As in [FPS20], on extraction of the initial representation (γ^*, ζ^*) of $\widehat{R^*}$ from a valid signature $(\widehat{R^*}, s^*)$ output by the adversary, the reduction can use that $\widehat{R^*} = [\gamma^*]_{\mathbb{H}} \cdot [\zeta^* \cdot z]_{\mathbb{H}} = [s^* - c^* \cdot z]_{\mathbb{H}}$. Therefore,

$$z = \frac{s^* - \gamma^*}{\zeta^* - c^*}.$$

Due to the added check in Game 4, an adversary can only win Game 4 or Game 5 when $\zeta^* - c^* \neq 0$ and therefore the overall advantage of an adversary B on $\mathbf{DLOG}$ in $\mathbb{G}$ is

$$\mathrm{Adv}^{\mathbf{DLOG}}_{\mathsf{B},\mathbb{G}} \lambda$$

```
Game 5
$\mathsf{pp}_{\mathbb{G}} \leftarrow \mathsf{Setup}_{\mathbb{G}}(1^\lambda)$
$Z \leftarrow \mathbb{G}$
$(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{Setup}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, ([1]_{\mathbb{G}}, Z)^\intercal)$
$Q := \varnothing, T := [], U := []$
$\xi_1 \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, 1))$
$\xi_2 = \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, (0,1)^\intercal))$
$\mathsf{pk} := (\mathsf{pp}_{\mathbb{H}}, \xi_1, \xi_2)$
$(m^*, \widehat{R^*}, s^*) \leftarrow \mathsf{A}^{H,\mathsf{Sign}}(1^\lambda, \mathsf{pk})$
if $m^* \in Q$ then return 0
if $U[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R^*}), m^*)] \neq \perp$ then
    $(\gamma^*, \zeta^*) := U[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R^*}), m^*)]$
    if $\zeta^* = -T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R^*}), m^*)]$ then return
0
$c^* = H(\widehat{R^*}, m^*)$
return $\mathsf{Eq}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, [s^*]_{\mathbb{H}}, \widehat{R^*} \cdot \xi_2^{c^*})$

$H(\widehat{R}, m)$
if $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), m)] = \perp$ then
    $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), m)] \leftarrow \mathbb{Z}_p$
    $U[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), m)] = \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{R})$
return $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), m)]$

$\mathsf{Sign}(m)$
$c, s \leftarrow \mathbb{Z}_p$
$\widehat{R_2} = \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \xi_1^s \cdot \xi_2^{-c})$
if $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R_2}), m)] = \perp$ then
    $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R_2}), m)] := c$
else
    abort
$Q := Q \cup \{m\}$
return $(\widehat{R_2}, s)$
```

Figure 7.12: Game 5 corresponds to G_2 in [FPS20].

$$
\begin{aligned}
&\geq \Pr[\mathrm{out}_5 = 1]\\
&\geq \Pr[\mathrm{out}_4 = 1] - \frac{q_s(q_s+q_h)}{2^\lambda}\\
&\geq \Pr[\mathrm{out}_3 = 1] - \frac{q_h}{2^\lambda} - \frac{q_s(q_s+q_h)}{2^\lambda}\\
&\geq \Pr[\mathrm{out}_2 = 1] - \mathrm{Adv}^{\texttt{1-switching}}_{\mathsf{A}'',\mathbb{H}}\lambda - \frac{q_h}{2^\lambda} - \frac{q_s(q_s+q_h)}{2^\lambda}\\
&\geq \Pr[\mathrm{out}_1 = 1] - \frac{q_s+1}{2^\lambda} - \mathrm{Adv}^{\texttt{1-switching}}_{\mathsf{A}'',\mathbb{H}}\lambda - \frac{q_h}{2^\lambda} - \frac{q_s(q_s+q_h)}{2^\lambda}\\
&\geq \Pr[\mathrm{Exp}^{\mathsf{euf\text{-}cma}}_{\Sigma_{\mathsf{schnorr}},\mathsf{A}}(\lambda) = 1] - \frac{q_s+1+q_h+q_s(q_s+q_h)}{2^\lambda} - \mathrm{Adv}^{\texttt{1-switching}}_{\mathsf{A}'',\mathbb{H}}\lambda\\
&\geq \Pr[\mathrm{Exp}^{\mathsf{euf\text{-}cma}}_{\Sigma_{\mathsf{schnorr}},\mathsf{A}}(\lambda) = 1] - \frac{O(q_s(q_s+q_h))}{2^\lambda} - \mathrm{Adv}^{\texttt{1-switching}}_{\mathsf{A}'',\mathbb{H}}\lambda
\end{aligned}
$$

which concludes the proof. □

7.3.5 Signed ElGamal

In the hashed ElGamal key-encapsulation mechanism (KEM), a public key is a group element Y, the corresponding secret key is $y = \mathrm{dlog}_g(Y)$. For encryption, one picks a random exponent $x \leftarrow \mathbb{Z}_p$ to compute a key $H(Y^x)$ accompanied by an encapsulation $X := g^x$. Given the encapsulation and the secret key y, the receiver can recover that key $K = H(X^y)$. [FPS20] showed that Schnorr-signed ElGamal, a variant of hashed ElGamal, is tightly IND-CCA2 secure under the $\mathbf{DLOG}$ assumption in the AGM and the random oracle model. Schnorr-signed ElGamal (see Figure 7.13) works similarly as hashed ElGamal but every encapsulation is accompanied by a Schnorr signature for message X under public key X. Decryption works as before with the difference that decryption aborts if the provided Schnorr

```
$\underline{\mathsf{KGen}(\mathsf{pp}_{\mathbb{H}})}$
$y \leftarrow \mathbb{Z}_q$
$\widehat{G} := \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, 1))$
$\widehat{Y} := \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, y))$
$\mathsf{pk} := (\mathsf{pp}_{\mathbb{H}}, \widehat{G}, \widehat{Y})$
$\mathsf{sk} := (\mathsf{pk}, y)$
return $(\mathsf{pk}, \mathsf{sk})$
```

```
$\underline{\mathsf{Enc}(\mathsf{pk} = (\mathsf{pp}_{\mathbb{H}}, \widehat{G}, \widehat{Y}))}$
$x, r \leftarrow \mathbb{Z}_q$
$\widehat{R} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, r))$
$\widehat{X} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, x))$
$k := H'(\widehat{Y}^x)$
$s := r + H(\widehat{R}, \widehat{X}) \cdot x \mod q$
return $(k, (\widehat{X}, \widehat{R}, s))$
$\underline{\mathsf{Dec}(\mathsf{sk}, (\widehat{X}, \widehat{R}, s))}$
if $[s]_{\mathbb{H}} \neq_{\mathbb{H}} \widehat{X}^{H(\widehat{R},\widehat{X})} \cdot \widehat{R}$ then
    return $\perp$
return $k := H'(\widehat{X}^y)$
```

Figure 7.13: The Schnorr-signed ElGamal encryption scheme $\mathsf{PKE}_{\mathsf{sElG}}$. Note that to compensate for the non-uniqueness of group element encodings, the (random oracle) hash value of a group element encoding is computed for the unique identifier produced by $\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \cdot)$. The hash function H maps tuples of group elements from $\mathbb{H}$ to elements in $\mathcal{K}$ and the hash function H' maps group elements from $\mathbb{H}$ to $\mathbb{Z}_p$ elements.

signature is invalid.

In this section, we demonstrate that our algebraic wrapper can be applied to mimic the proof of tight IND-CCA2 security of Schnorr-signed ElGamal $\mathsf{PKE}_{\mathsf{sElG}}$ from [FPS20]. In contrast to our tight reduction for Schnorr signatures from Section 7.3.4, the tightness for Schnorr-signed ElGamal does not require the "origin element trick" since it is not necessary to apply switching to oracle responses.

Theorem 7.3.5. *Let* $\mathsf{Setup}_{\mathbb{G}}$ *be a group generator for a cyclic group* $\mathbb{G}$ *such that* $\mathbf{DLOG}$ *is hard relative to* $\mathsf{Setup}_{\mathbb{G}}$ *and let* $\mathbb{H}$ *be an algebraic wrapper of dimension* ≥ 2 *for* $\mathbb{G}$*. Then,* $\mathsf{PKE}_{\mathsf{sElG}}$ *in* $\mathbb{H}$ *is tightly IND-CCA2 secure in the random oracle model.*

More precisely, for all PPT adversaries A*, there exists a PPT adversary* B *and a legitimate switching adversary* A' *both running in time* $T(\mathsf{B}) \approx T(\mathsf{A}) + (q_d + q_h) \cdot \mathsf{poly}(\lambda)$ *and* $T(\mathsf{A}') \approx T(\mathsf{A}) + (q_d + q_h) \cdot \mathsf{poly}(\lambda)$ *such that*

$$\mathrm{Adv}^{\mathsf{ind\text{-}cca2}}_{\mathsf{PKE}_{\mathsf{sElG}},\mathsf{A}}\lambda \leq \mathrm{Adv}^{\mathbf{DLOG}}_{\mathsf{B},\mathbb{G}}\lambda + \mathrm{Adv}^{\mathtt{2\text{-}switching}}_{\mathsf{A}',\mathbb{H}}\lambda + \frac{O(q_d + q_h)}{2^\lambda},$$

where q_h *is a polynomial upper bound on the number of random oracle queries,* q_d *is a polynomial upper bound on the number of decryption queries and* poly *is a polynomial independent of* q_d *and* q_h*.*

Proof. The proof strategy follows (up to some preparations) the outline of [FPS20]. The hybrid Game 0 is identical to $\mathrm{Exp}^{\mathsf{ind\text{-}cca2}}_{\mathsf{PKE}_{\mathsf{sElG}},\mathsf{A}}(\lambda)$. The initial game transitions until hybrid Game 3 are preparation steps due to the algebraic wrapper. The following hybrids Game 4, Game 5, Game 6 correspond exactly to the hybrids G_1, G_2, G_3 from [FPS20], respectively. The preparation steps set up the randomness for the challenge ciphertext as $x^* := z \cdot y$. Further, the randomness for the signature in the challenge ciphertext is chosen using an x^*-component similar to the proof of Schnorr signatures Section 7.3.4. Subsequently, re-randomizability and switching are applied such that the public key $\widehat{Y}$ uses the representation vector $(0, 1)^\intercal$ and the randomness for the challenge ciphertext $\widehat{X^*}$ uses the representation vector $(0, z)^\intercal$. The remaining proof proceeds as in [FPS20].

For simplicity, we introduce the notation $\widehat{A} =_{\mathbb{H}} \widehat{B}$ for $\mathsf{Eq}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{A}, \widehat{B})$. We proceed over a series of games starting from the IND-CCA2 game in the random oracle model, see Figure 7.14. The hash functions $\widetilde{H}\colon \mathbb{H} \times \mathbb{H} \to \mathbb{Z}_p$ and $\widetilde{H}'\colon \mathbb{H} \to \mathcal{K}$ behave exactly as there counterparts H and H', respectively,

```
Game 0
$\mathsf{pp}_{\mathbb{G}} \leftarrow \mathsf{Setup}_{\mathbb{G}}(1^\lambda)$
$(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{Setup}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, ([1]_{\mathbb{G}}, [\beta_2]_{\mathbb{G}})^\intercal)$
$y \leftarrow \mathbb{Z}_p$
$\widehat{G} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, 1))$
$\widehat{Y} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, y))$
$\mathsf{pk} := (\mathsf{pp}_{\mathbb{H}}, \widehat{G}, \widehat{Y})$
$T, T' = []$
$x^*, r^* \leftarrow \mathbb{Z}_p$
$\widehat{X^*} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, x^*))$
$\widehat{R^*} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, r^*))$
$c^* := \widetilde{H}(\widehat{R^*}, \widehat{X^*})$
$s^* := r^* + c^* \cdot x^* \bmod p$
$k_0 := \widetilde{H}'(\widehat{Y}^{x^*}), k_1 \leftarrow \mathcal{K}$
$b' \leftarrow \mathsf{A}^{H,H',\mathsf{Dec}}(\mathsf{pk}, k_b, (\widehat{R^*}, \widehat{X^*}, s^*))$
return $b = b'$

$H(\widehat{R}, \widehat{X})$
if $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X}))] = \perp$ then
    $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X}))] \leftarrow \mathbb{Z}_p$
return $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X}))]$

$H'(\widehat{K})$
if $T'[\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{K})] = \perp$ then
    $T'[\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{K})] \leftarrow \mathcal{K}$
return $T'[\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{K})]$

$\mathsf{Dec}(\widehat{R}, \widehat{X}, s)$
if $\widehat{R} =_{\mathbb{H}} \widehat{R^*} \wedge \widehat{X} =_{\mathbb{H}} \widehat{X^*} \wedge s = s^*$ then
    return $\perp$
$c := \widetilde{H}(\widehat{R}, \widehat{X})$
if $[s]_{\mathbb{H}} \neq_{\mathbb{H}} \widehat{R} \cdot \widehat{X}^c$ then
    return $\perp$
$k := \widetilde{H}'(\widehat{X}^y)$
return $k$
```

Figure 7.14: The description of Game 0. Game 0is identical to $\mathrm{Exp}^{\mathsf{ind\text{-}cca2}}_{\mathsf{PKE}_{\mathsf{sEIG}},\mathsf{A}}(\lambda)$.

and act solely as helper functions. The adversary A only has access to the oracles H and H' (and Dec). Throughout the proof, the behavior of $\widetilde{H}$ and $\widetilde{H}'$ will not be altered.

Game hop Game 0⇝Game 1. Similarly to the security proof of Schnorr signatures, we first change how the signature in the challenge ciphertext is generated. Particularly, the randomness used for the signature is chosen using a y-component, see Figure 7.15. Because x^* is in both games uniformly distributed and $r^* = s^* - c^* \cdot x^* \bmod p$ and thus $\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R_1^*}) = \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R_2^*})$, Game 0 and Game 1 are distributed identically.

Game hop Game 1⇝Game 2. In Game 2 (see Figure 7.16), the encodings $\widehat{Y}$, $\widehat{X^*}$ and $\widehat{R_2^*}$ are produced using private sampling or the group operation instead of public sampling. Since these encodings are re-randomized, this game hop is justified by the re-randomizability of the algebraic wrapper $\mathbb{H}$. More precisely, we successively replace $\mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, y))$ by $\mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, y))$, $\mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, x^*))$ by $\mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, x^*))$ and, finally, $\mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, s^* - c^* \cdot x^*))$ by $\mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{G}^{s^*} \cdot (\widehat{X^*})^{-c^*})$. Due to correctness of sampling, we have $\mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, y)) = y \cdot \overrightarrow{e_1} = \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, y))$ and $\mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, x^*)) = x^* \cdot \overrightarrow{e_1} = \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, x^*))$. Further, due to correctness of sampling and correctness of extraction, we have $\mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, s^* - c^* \cdot x^*)) = (s^* - c^* \cdot x^*) \cdot \overrightarrow{e_1} = \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{G}^{s^*} \cdot (\widehat{X^*})^{-c^*})$. Hence, due to statistical re-randomizability, $|\Pr[\mathrm{out}_2 = 1] - \Pr[\mathrm{out}_1 = 1]| \leq \frac{3}{2^\lambda}$.

Game hop Game 2⇝Game 3. Towards removing the necessity to know y for the simulation, we change the basis to be $\left[\overrightarrow{b}\right]_{\mathbb{G}} := ([1]_{\mathbb{G}}, [y]_{\mathbb{G}})^\intercal$ and adapt the representation vectors used for private sampling of $\widehat{Y}$ and $\widehat{X^*}$ accordingly, see Figure 7.17. This game hop is justified by 2-`switching`. We construct an adversary A′ on 2-`switching` as follows. Initially, on input of $\mathsf{pp}_{\mathbb{G}}$, A′ outputs two basis vectors $\left[\overrightarrow{b}\right]_{\mathbb{G}}^{(G2)} := ([1]_{\mathbb{G}}, [\beta_2]_{\mathbb{G}})^\intercal$ and $\left[\overrightarrow{b}\right]_{\mathbb{G}}^{(G3)} := ([1]_{\mathbb{G}}, [y]_{\mathbb{G}})^\intercal$ and representation vectors

```
Game 1
$\mathsf{pp}_{\mathbb{G}} \leftarrow \mathsf{Setup}_{\mathbb{G}}(1^\lambda)$
$(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{Setup}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, ([1]_{\mathbb{G}}, [\beta_2]_{\mathbb{G}})^\intercal)$
$y \leftarrow \mathbb{Z}_p$
$\widehat{G} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, 1))$
$\widehat{Y} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, y))$
$\mathsf{pk} := (\mathsf{pp}_{\mathbb{H}}, \widehat{G}, \widehat{Y})$
$T, T' = []$
$z, r^* \leftarrow \mathbb{Z}_p, x^* := z \cdot y$
$\widehat{X^*} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, z \cdot y))$
$\widehat{R_1^*} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, r^*))$
$c^* := \widetilde{H}(\widehat{R^*}, \widehat{X^*})$
$s^* := r^* + c^* \cdot z \cdot y \bmod p$
$\widehat{R_2^*} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, s^* - c^* \cdot x^*))$
$k_0 := \widetilde{H}'(\widehat{Y}^{z \cdot y}), k_1 \leftarrow \mathcal{K}$
$b' \leftarrow \mathsf{A}^{H, H', \mathsf{Dec}}(\mathsf{pk}, k_b, (\widehat{R_2^*}, \widehat{X^*}, s^*))$
return $b = b'$

$H(\widehat{R}, \widehat{X})$
if $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X}))] = \perp$ then
    $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X}))] \leftarrow \mathbb{Z}_p$
return $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X}))]$

$H'(\widehat{K})$
if $T'[\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{K})] = \perp$ then
    $T'[\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{K})] \leftarrow \mathcal{K}$
return $T'[\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{K})]$

$\mathsf{Dec}(\widehat{R}, \widehat{X}, s)$
if $\widehat{R} =_{\mathbb{H}} \widehat{R_2^*} \wedge \widehat{X} =_{\mathbb{H}} \widehat{X^*} \wedge s = s^*$ then
    return $\perp$
$c := \widetilde{H}(\widehat{R}, \widehat{X})$
if $[s]_{\mathbb{H}} \neq_{\mathbb{H}} \widehat{R} \cdot \widehat{X}^c$ then
    return $\perp$
$k := \widetilde{H}'(\widehat{X}^y)$
return $k$
```

Figure 7.15: The description of the hybrid Game 1.

```
Game 2
$\mathsf{pp}_{\mathbb{G}} \leftarrow \mathsf{Setup}_{\mathbb{G}}(1^\lambda)$
$(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{Setup}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, ([1]_{\mathbb{G}}, [\beta_2]_{\mathbb{G}})^\intercal)$
$y \leftarrow \mathbb{Z}_p$
$\widehat{G} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, 1))$
$\widehat{Y} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, y))$
$\mathsf{pk} := (\mathsf{pp}_{\mathbb{H}}, \widehat{G}, \widehat{Y})$
$T, T' = []$
$z, r^* \leftarrow \mathbb{Z}_p, x^* := z \cdot y$
$\widehat{X^*} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, z \cdot y))$
$\widehat{R_1^*} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, r^*))$
$c^* := \widetilde{H}(\widehat{R^*}, \widehat{X^*})$
$s^* := r^* + c^* \cdot z \cdot y \bmod p$
$\widehat{R_2^*} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{G}^{s^*} \cdot (\widehat{X^*})^{-c^*})$
$k_0 := \widetilde{H}'(\widehat{Y}^{z \cdot y}), k_1 \leftarrow \mathcal{K}$
$b' \leftarrow \mathsf{A}^{H, H', \mathsf{Dec}}(\mathsf{pk}, k_b, (\widehat{R_2^*}, \widehat{X^*}, s^*))$
return $b = b'$

$H(\widehat{R}, \widehat{X})$
if $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X}))] = \perp$ then
    $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X}))] \leftarrow \mathbb{Z}_p$
return $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X}))]$

$H'(\widehat{K})$
if $T'[\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{K})] = \perp$ then
    $T'[\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{K})] \leftarrow \mathcal{K}$
return $T'[\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{K})]$

$\mathsf{Dec}(\widehat{R}, \widehat{X}, s)$
if $\widehat{R} =_{\mathbb{H}} \widehat{R_2^*} \wedge \widehat{X} =_{\mathbb{H}} \widehat{X^*} \wedge s = s^*$ then
    return $\perp$
$c := \widetilde{H}(\widehat{R}, \widehat{X})$
if $[s]_{\mathbb{H}} \neq_{\mathbb{H}} \widehat{R} \cdot \widehat{X}^c$ then
    return $\perp$
$k := \widetilde{H}'(\widehat{X}^y)$
return $k$
```

Figure 7.16: The description of the hybrid Game 2.

```
Game 3
    $\mathsf{pp}_{\mathbb{G}} \leftarrow \mathsf{Setup}_{\mathbb{G}}(1^\lambda)$
    $y \leftarrow \mathbb{Z}_p$
    $(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{Setup}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, ([1]_{\mathbb{G}}, [y]_{\mathbb{G}})^{\mathsf{T}})$
    $\widehat{G} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, 1))$
    $\widehat{Y} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, (0,1)^{\mathsf{T}}))$
    $\mathsf{pk} := (\mathsf{pp}_{\mathbb{H}}, \widehat{G}, \widehat{Y})$
    $T, T' = []$
    $z, r^* \leftarrow \mathbb{Z}_p, x^* := z \cdot y$
    $\widehat{X^*} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, (0,z)^{\mathsf{T}}))$
    $\widehat{R_1^*} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, r^*))$
    $c^* := \widetilde{H}(\widehat{R^*}, \widehat{X^*})$
    $s^* := r^* + c^* \cdot z \cdot y \bmod p$
    $\widehat{R_2^*} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{G}^{s^*} \cdot (\widehat{X^*})^{-c^*})$
    $k_0 := \widetilde{H}'(\widehat{Y}^{z \cdot y}), k_1 \leftarrow \mathcal{K}$
    $b' \leftarrow \mathsf{A}^{H,H',\mathsf{Dec}}(\mathsf{pk}, k_b, (\widehat{R_2^*}, \widehat{X^*}, s^*))$
    return $b = b'$

$H(\widehat{R}, \widehat{X})$
    if $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X}))] = \perp$ then
        $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X}))] \leftarrow \mathbb{Z}_p$
    return $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X}))]$

$H'(\widehat{K})$
    if $T'[\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{K})] = \perp$ then
        $T'[\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{K})] \leftarrow \mathcal{K}$
    return $T'[\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{K})]$

$\mathsf{Dec}(\widehat{R}, \widehat{X}, s)$
    if $\widehat{R} =_{\mathbb{H}} \widehat{R_2^*} \wedge \widehat{X} =_{\mathbb{H}} \widehat{X^*} \wedge s = s^*$ then
        return $\perp$
    $c := \widetilde{H}(\widehat{R}, \widehat{X})$
    if $[s]_{\mathbb{H}} \neq_{\mathbb{H}} \widehat{R} \cdot \widehat{X}^c$ then
        return $\perp$
    $k := \widetilde{H}'(\widehat{X}^y)$
    return $k$
```

Figure 7.17: The description of the hybrid Game 3.

$\overrightarrow{v^{(1),(G2)}} := (y, 0)^{\mathsf{T}}$, $\overrightarrow{v^{(2),(G2)}} := (z \cdot y, 0)^{\mathsf{T}}$ and $\overrightarrow{v^{(1),(G3)}} := (0, 1)^{\mathsf{T}}$, $\overrightarrow{v^{(2),(G3)}} := (0, z)^{\mathsf{T}}$. In return, A' receives public parameters $\mathsf{pp}_{\mathbb{H}}$ and two encodings $\widehat{C^{(1)}}$ and $\widehat{C^{(2)}}$. A' computes $\widehat{Y} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{C^{(1)}})$ and $\widehat{X^*} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{C^{(2)}})$ and simulates the remaining game as in Game 2. Note that this is possible since $\tau_{\mathbb{H}}$ is not necessary. A' simulates either Game 2 or Game 3 for A depending on the challenge provided by the 2-switching-game. Hence, $|\Pr[\mathrm{out}_3 = 1] - \Pr[\mathrm{out}_2 = 1]| \leq \mathrm{Adv}^{\texttt{2-switching}}_{\mathbb{H},\mathsf{A}'} \lambda$. Since A' is a legitimate 2-switching adversary, $\mathrm{Adv}^{\texttt{2-switching}}_{\mathbb{H},\mathsf{A}'} \lambda$ is negligible.

Game hop Game 3⇝Game 4. From this point on, we are able to closely follow the lines of [FPS20]. In Game 4 (see Figure 7.18), the oracle H stores the private extractions of the encodings used to call H in a list U. Furthermore, the decryption oracle obtains representation vectors corresponding to the supplied encodings $\widehat{R}$ and $\widehat{X}$ by first looking for a matching entry in U and, if no such entry is present, by applying private extraction. Let (ν, μ) and (ν', μ') be the thus obtained representation vectors of $\widehat{R}$ and $\widehat{X}$, respectively. Game 4 additionally introduces an abort condition. If $\mu + \mu' \cdot c = 0$ and $\mu' \neq 0$, Game 4 aborts and outputs a random bit. The games Game 3 and Game 4 only differ if Game 4aborts.

Note that all values in the table T are set in an adversarial call to either H or Dec, except for $c^* = T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R_2^*}), \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X^*}))]$ which is set using $\widetilde{H}$ in the game. If $\mathsf{Dec}(\widehat{R}, \widehat{X}, s) \neq \perp$, then $(\widehat{R}, \widehat{X}) \neq (\widehat{R_2^*}, \widehat{X^*})$ since otherwise $s = s^*$. Hence, the value $c = T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X}))]$ is independent of (ν, μ, ν', μ'). Therefore, the probability that Game 4 aborts is upper bounded by the probability that c is chosen as $c = -\frac{\mu}{\mu'} \bmod p$ which can be upper bounded by $\frac{1}{p} \leq 2^{-\lambda}$. By a union bound, the probability that Game 4 aborts is upper bounded by $\frac{q_d}{2^\lambda}$. Since Game 3 and Game 4 behave identical unless Game 4 aborts, we have $|\Pr[\mathrm{out}_4 = 1] - \Pr[\mathrm{out}_3 = 1]| \leq \frac{q_d}{2^\lambda}$.

Game hop Game 4⇝Game 5. Figure 7.19 shows the description of Game 5. Instead of sampling r^* and querying the $\widetilde{H}$ for c^* to obtain $s^* = r^* + c^* \cdot x^*$, Game 5samples s^* and c^* independently and computes $\widehat{R_2^*} = \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{G}^{s^*} \cdot (\widehat{X^*})^{-c^*})$ as in Game 4. This behavior is identical to Game 4 except

```
Game 4 (corresponds to $G_1$ from [FPS20])
$\mathsf{pp}_{\mathbb{G}} \leftarrow \mathsf{Setup}_{\mathbb{G}}(1^\lambda)$
$y \leftarrow \mathbb{Z}_p$
$(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{Setup}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, ([1]_{\mathbb{G}}, [y]_{\mathbb{G}})^{\intercal})$
$\widehat{G} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, 1))$
$\widehat{Y} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, (0,1)^{\intercal}))$
$\mathsf{pk} := (\mathsf{pp}_{\mathbb{H}}, \widehat{G}, \widehat{Y})$
$T, T' = [], U := []$
$z, r^* \leftarrow \mathbb{Z}_p, x^* := z \cdot y$
$\widehat{X^*} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, (0,z)^{\intercal}))$
$\widehat{R^*_1} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, r^*))$
$c^* := \widetilde{H}(\widehat{R^*}, \widehat{X^*})$
$s^* := r^* + c^* \cdot z \cdot y \bmod p$
$\widehat{R^*_2} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{G}^{s^*} \cdot (\widehat{X^*})^{-c^*})$
$k_0 := \widetilde{H}'(\widehat{Y}^{z \cdot y}), k_1 \leftarrow \mathcal{K}$
$b' \leftarrow \mathsf{A}^{H, H', \mathsf{Dec}}(\mathsf{pk}, k_b, (\widehat{R^*_2}, \widehat{X^*}, s^*))$
return $b = b'$

$H'(\widehat{K})$
if $T'[\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{K})] = \perp$ then
    $T'[\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{K})] \leftarrow \mathcal{K}$
return $T'[\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{K})]$
```

```
$H(\widehat{R}, \widehat{X})$
if $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X}))] = \perp$ then
    $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X}))] \leftarrow \mathbb{Z}_p$
    $U[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X}))] :=$
        $(\mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{R}), \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{X}))$
return $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X}))]$

$\mathsf{Dec}(\widehat{R}, \widehat{X}, s)$
if $\widehat{R} =_{\mathbb{H}} \widehat{R^*_2} \wedge \widehat{X} =_{\mathbb{H}} \widehat{X^*} \wedge s = s^*$ then
    return $\perp$
$c := \widetilde{H}(\widehat{R}, \widehat{X})$
if $[s]_{\mathbb{H}} \neq_{\mathbb{H}} \widehat{R} \cdot \widehat{X}^c$ then
    return $\perp$
$(\nu, \mu) \leftarrow \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{R})$
$(\nu', \mu') \leftarrow \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{X})$
if $U[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X}))] \neq \perp$ then
    $(\nu, \mu, \nu', \mu') := U[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X}))]$
if $(\mu + \mu' \cdot c = 0) \wedge (\mu' \neq 0)$ then
    abort game and output random bit
$k := \widetilde{H}'(\widehat{X}^y)$
return $k$
```

Figure 7.18: The description of the hybrid Game 4.

for the event that the tuple $(\widehat{R^*_2}, \widehat{X^*})$ already has an entry in T. If this event occurs, Game 5 aborts. Since $\widehat{R^*_2}$ and $\widehat{X^*}$ are uniformly random and T contains at most $q_d + q_h$ many entries after at most q_d Dec-queries and q_h H-queries, the probability that Game 5 aborts but Game 4 does not can be upper bounded by $\frac{q_d+q_h}{2^{2\lambda}}$. Hence, $|\Pr[\mathrm{out}_5 = 1] - \Pr[\mathrm{out}_4 = 1]| \leq \frac{q_d+q_h}{2^{2\lambda}}$.

Game hop Game 5$\rightsquigarrow$Game 6. Game 6 (see Figure 7.20) introduces two further abort conditions $(\star)$ and $(\star\star)$. As in [FPS20], we show that if Game 6 differs from Game 5, then we can solve discrete logarithms.

We construct an adversary B on the discrete logarithm problem. Given $(\mathsf{pp}_{\mathbb{G}}, [1]_{\mathbb{G}}, [y]_{\mathbb{G}})$, B produces $(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{Setup}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, ([1]_{\mathbb{G}}, [y]_{\mathbb{G}})^{\intercal})$ and simulates Game 6 for A. Note that $\widehat{Y}$ and $\widehat{X^*}$ can be sampled without knowing y (and x^*).

- B simulates queries to H' as follows. When A queries H' for $\widehat{K}$, B computes $(\nu'', \mu'') \leftarrow \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{K})$. Hence,

  $$\mathsf{Unwrap}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{K}) = (\nu'', \mu'') \cdot ([1]_{\mathbb{G}}, [y]_{\mathbb{G}})^{\intercal}.$$

  To test whether $\widehat{K} = (\widehat{X^*})^y$ which in turn (implicitly) equals $\widehat{G}^{z \cdot y^2}$, B solves the equation

  $$z \cdot y^2 - \mu'' \cdot y - \nu'' = 0 \bmod p$$

  for y. If one solution is the discrete logarithm of the given DLOG challenge game Game 6 aborts and B outputs y. (Note that due to $(\star)$, if the game does not abort, A's view is independent if it receives k_b or k_1.)

```
Game 5 (corresponds to $G_2$ from [FPS20])
$\mathsf{pp}_{\mathbb{G}} \leftarrow \mathsf{Setup}_{\mathbb{G}}(1^\lambda)$
$y \leftarrow \mathbb{Z}_p$
$(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{Setup}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, ([1]_{\mathbb{G}}, [y]_{\mathbb{G}})^\intercal)$
$\widehat{G} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, 1))$
$\widehat{Y} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, (0,1)^\intercal))$
$\mathsf{pk} := (\mathsf{pp}_{\mathbb{H}}, \widehat{G}, \widehat{Y})$
$T, T' = [], U := []$
$z, c^*, s^* \leftarrow \mathbb{Z}_p, x^* := z \cdot y$
$\widehat{X^*} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, (0,z)^\intercal))$
$\widehat{R_2^*} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{G}^{s^*} \cdot (\widehat{X^*})^{-c^*})$
if $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R_2^*}), \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X^*}))] = \perp$ then
    $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R_2^*}), \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X^*}))] := c^*$
else
    abort game and output random bit
$k_0 := \widetilde{H}'(\widehat{Y}^{z \cdot y}), k_1 \leftarrow \mathcal{K}$
$b' \leftarrow \mathsf{A}^{H, H', \mathsf{Dec}}(\mathsf{pk}, k_1, (\widehat{R_2^*}, \widehat{X^*}, s^*))$
return $b = b'$

$H'(\widehat{K})$
if $T'[\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{K})] = \perp$ then
    $T'[\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{K})] \leftarrow \mathcal{K}$
return $T'[\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{K})]$

$H(\widehat{R}, \widehat{X})$
if $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X}))] = \perp$ then
    $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X}))] \leftarrow \mathbb{Z}_p$
    $U[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X}))] :=$
        $(\mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{R}), \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{X}))$
return $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X}))]$

$\mathsf{Dec}(\widehat{R}, \widehat{X}, s)$
if $\widehat{R} =_{\mathbb{H}} \widehat{R_2^*} \wedge \widehat{X} =_{\mathbb{H}} \widehat{X^*} \wedge s = s^*$ then
    return $\perp$
$c := \widetilde{H}(\widehat{R}, \widehat{X})$
if $[s]_{\mathbb{H}} \neq_{\mathbb{H}} \widehat{R} \cdot \widehat{X}^c$ then
    return $\perp$
$(\nu, \mu) \leftarrow \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{R})$
$(\nu', \mu') \leftarrow \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{X})$
if $U[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X}))] \neq \perp$ then
    $(\nu, \mu, \nu', \mu') := U[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X}))]$
if $(\mu + \mu' \cdot c = 0) \wedge (\mu' \neq 0)$ then
    abort game and output random bit
$k := \widetilde{H}'(\widehat{X}^y)$
return $k$
```

Figure 7.19: The description of the hybrid Game 5.

```
Game 6 (corresponds to $G_3$ from [FPS20])
$\mathsf{pp}_{\mathbb{G}} \leftarrow \mathsf{Setup}_{\mathbb{G}}(1^\lambda)$
$y \leftarrow \mathbb{Z}_p$
$(\mathsf{pp}_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{Setup}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{G}}, ([1]_{\mathbb{G}}, [y]_{\mathbb{G}})^\intercal)$
$\widehat{G} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, 1))$
$\widehat{Y} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, (0,1)^\intercal))$
$\mathsf{pk} := (\mathsf{pp}_{\mathbb{H}}, \widehat{G}, \widehat{Y})$
$T, T' = [], U := []$
$z, c^*, s^* \leftarrow \mathbb{Z}_p$
$\widehat{X^*} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, (0,z)^\intercal))$
$\widehat{R_2^*} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{G}^{s^*} \cdot (\widehat{X^*})^{-s^*})$
if $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R_2^*}), \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X^*}))] = \bot$ then
    $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R_2^*}), \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X^*}))] := c^*$
else
    abort game and output random bit
$k_0 := \widetilde{H}'(\widehat{Y}^{z \cdot y}), k_1 \leftarrow \mathcal{K}$
$b' \leftarrow \mathsf{A}^{H, H', \mathsf{Dec}}(\mathsf{pk}, k_1, (\widehat{R_2^*}, \widehat{X^*}, s^*))$
return $b = b'$

$H'(\widehat{K})$
if $\widehat{K} = \widehat{X^*}^{y}$ then
    abort game and output random bit                    $(\star)$
if $T'[\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{K})] = \bot$ then
    $T'[\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{K})] \leftarrow \mathcal{K}$
return $T'[\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{K})]$

$H(\widehat{R}, \widehat{X})$
if $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X}))] = \bot$ then
    $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X}))] \leftarrow \mathbb{Z}_p$
    $U[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X}))] :=$
        $(\mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{R}), \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{X}))$
return $T[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X}))]$

$\mathsf{Dec}(\widehat{R}, \widehat{X}, s)$
if $\widehat{R} =_{\mathbb{H}} \widehat{R_2^*} \wedge \widehat{X} =_{\mathbb{H}} \widehat{X^*} \wedge s = s^*$ then
    return $\bot$
$c := \widetilde{H}(\widehat{R}, \widehat{X})$
if $[s]_{\mathbb{H}} \neq_{\mathbb{H}} \widehat{R} \cdot \widehat{X}^c$ then
    return $\bot$
$(\nu, \mu) \leftarrow \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{R})$
$(\nu', \mu') \leftarrow \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{X})$
if $\mu + \mu' \cdot c \neq 0$ then
    abort game and output random bit                    $(\star\star)$
if $U[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X}))] \neq \bot$ then
    $(\nu, \mu, \nu', \mu') := U[(\mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X}))]$
if $(\mu + \mu' \cdot c = 0) \wedge (\mu' \neq 0)$ then
    abort game and output random bit
$k := \widetilde{H}'(\widehat{X}^y)$
return $k$
```

Figure 7.20: The description of the hybrid Game 6. The view of A is independent of b.

- B simulates queries to Dec as follows. As argued above, if $\mathsf{Dec}(\widehat{R}, \widehat{X}, s)$ does not return $\bot$, then $(\widehat{R}, \widehat{X}) \neq (\widehat{R^*}, \widehat{X^*})$. We have that

$$\mathsf{Unwrap}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{R}) = (\nu, \mu) \cdot ([1]_{\mathbb{G}}, [y]_{\mathbb{G}})^\intercal \bmod p$$
$$\mathsf{Unwrap}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X}) = (\nu', \mu') \cdot ([1]_{\mathbb{G}}, [y]_{\mathbb{G}})^\intercal \bmod p$$

If Dec does not return $\bot$, we have $s = r + c \cdot x \bmod p$ and hence

$$y \cdot (\mu + \mu' \cdot c) = s - \nu - \nu' \cdot c \bmod p. \tag{7.1}$$

If $\mu + \mu' \cdot c \neq 0 \bmod p$, Game 6 aborts and B solves Eq. (7.1) for y. If $\mu + \mu' \cdot c = 0 \bmod p$ and $\mu' \neq 0$ then both Game 5 and Game 6 abort. If $\mu + \mu' \cdot c = 0 \bmod p$ and $\mu' = 0$ then $\mu = 0$ and $\mathrm{dlog}_{[1]_{\mathbb{G}}}(\mathsf{Unwrap}_{\mathbb{H}}(\mathsf{pp}_{\mathbb{H}}, \widehat{X})) = x = \nu'$ allowing the reduction to simulate Dec response as $k := \widetilde{H}'(\widehat{Y}^x)$.

Therefore, $|\Pr[\mathrm{out}_6 = 1] - \Pr[\mathrm{out}_5 = 1]| \leq \mathrm{Adv}^{\mathbf{DLOG}}_{\mathbb{G},\mathsf{B}} \lambda$.

□

Chapter 8

Conclusion

In this thesis, we looked into several ways in which rewinding as well as algebraic group model related techniques can be used to prove security of cryptographic primitives.

In particular, in Chapter 3, we used a novel rewinding technique to prove adaptive security of two schemes with an underlying tree structure, namely the Goldreich-Goldwasser-Micali PRF and the Logical Key Hierarchy. The first result circumvents a lower bound for straight-line reductions.

In Chapter 4, we closed a gap in the proof of one-more unforgeabilty of the Abe-Okamoto partially blind signature scheme. This proof also relies heavily on rewinding, however the core of the analysis is the computation of probability of extracting the correct witness. While overall we employed a forking-lemma style argument, estimating the sizes of the sets to fork over takes some careful counting.

Staying in the realm of blind signatures, we then revisited Abe's blind signature scheme. We showed that with a minor modification, it can be turned partially blind, thus extending the realm of potential applications, and then proved one-more unforgeability in the AGM + ROM assuming the hardness of the discrete logarithm problem.

As a minor result, we showed that blind Schnorr signatures are sequentially secure in the AGM assuming the one-more discrete logarithm problem is hard. We further proved that, even in the AGM, a reduction needs to make as many OMDL queries as the user closes signing sessions, otherwise the reduction breaks OMDL already.

In our last chapter, we showed that, in some cases, the AGM can be replaced by a weaker construct called the algebraic wrapper. The algebraic wrapper 'wraps' around a cryptographic group and provides reductions with a limited extraction of algebraic explanations from adversaries. We showed how this construction can be used to implement proof strategies from the AGM, namely for some constant-size Diffie-Hellman assumptions, Schnorr signatures, and Signed ElGamal.

8.1 Open Questions and Future Research Directions

In the following, we discuss some open questions and potential follow-up work. Since the publication of the results we presented in this thesis, there have also been independent works that answer some of the questions left open at the time.

8.1.1 Undirected Rewinding and Adaptive Security

In the context of *undirected rewinding*, it could be interesting to find a general framework for the application of our rewinding technique as the two applications we presented are still rather involved. While we provide a somewhat modular framework where we formulated the two rewinding conditions as functions, the two proofs still have some differences, e.g. the proof for adaptive security of LKH contains hybrids without a bound on the number of rewinding loops as a method to change the stopping conditions of the rewinding. Such a game hop strategy could be generalized more using some abstract lemmata. Similarly one could formulate the underlying security assumptions, i.e. PRG security and IND-CPA security as generic indistinguishability notions. All of these generalizations might allow for a generic framework that allows to prove security of further schemes in a very compact way, i.e. just proving that certain properties hold both for the underlying assumption as well as for the hybrids.

It seems that protocols and primitives with an underlying tree structure are particularly suited to the application of 'undirected rewinding', and thus another research direction could be to investigate the applicability of this technique to other protocols with such a structure, either with or without the generalization described above.

On the other hand, in the current variant, the 'undirected rewinding' strategy rewinds the adversary to many points where no meaningful change can happen, in particular in the setting of the Goldreich-Goldwasser-Micali PC-PRF. To reduce the loss, one could look into how to save on these extra rewinding cycles while still allowing for the probabilistic analysis presented in the main body.

8.1.2 Blind Signatures

In the realm of *blind signatures*, a key open question is to find efficient, concurrently secure blind or partially blind signature schemes. Many current candidate schemes, such as Abe's scheme [Abe01], our partially blind variant thereof, the schemes by Tessaro and Zhu [TZ22], as well as the threshold scheme from [Cri+23] all come with a proof in an idealized group model. It remains an open question whether there exists a pairing-free scheme that can be proven concurrently secure without relying on idealized group models such as the AGM or GGM(s).

As all of the schemes mentioned above use an interactive version of the Fiat-Shamir transform, it is a natural question to ask whether it is possible to prove security using a forking-lemma/rewinding technique. The rewinding technique we used for the Abe-Okamoto scheme incurs a fairly large loss in the number of concurrent signing sessions. This is unsurprising and inherent for the Abe-Okamoto scheme as it is vulnerable to the ROS attack [Ben+21]. However, for some other schemes that rely on the witness-indistinguishability of an OR-proof, this ROS attack does not apply. The question is therefore two-fold: On the one hand, it is possible that there is an inherent lower bound for the 'naive' rewinding strategy for two-witness blind signature schemes. It would be interesting to see whether the proof strategy and modelling of 'naive' rewinding of Baldimtsi and Lysyanskaya [BL13b] can be extended to two-witness schemes. In their proof, they assume that the reduction programs the random oracle in a naive way, in particular independent of the inputs that the adversary makes. This allows a meta-reduction to predict the reduction's random oracle responses and use them to generate signature. When the reduction resets the adversary, it can use (most) forking runs to actually obtain the unique secret key. The forking runs it cannot use for extracting the secret key correspond to the notions of 'partners' in our proof, i.e. forking runs that yield the same query transcript from the adversary. As the general rewinding strategy in our setting also relies on this technique, in particular on the resampling of the random oracle outputs independent of the inputs, a meta-reduction could also use predictions of the random oracle responses as its leverage for forging signatures. However, the key difference is that now, forking runs would reveal only the witness used by the reduction. As we prove using the counting argument however, any real adversary is forced to at least sometimes give the reduction the other witness. Thus, a more elaborate strategy would be needed to actually construct a (potentially unbounded) adversary that actually gives the reduction the undesired witness most of the time, i.e. in all the cases except the cases proven by our reduction.

The other angle from which one could look at this problem is to construct a reduction that could circumvent such issues. For example, in the original proof of security in [Abe01], in the case of the main theorem, only one session is simulated using the other witness. This is possible when one has already proved the restrictive blinding lemma which shows that all signatures have a hidden link to a session. All other sessions use the $\mathbf{z}$-side witness, one session uses either the $\mathbf{z}$-side witness or the $\mathbf{y}$-side witness. As this setting already assumes that the adversary links all signatures to a signing session, the reduction can embed a discrete logarithm challenge in one of the session keys and hope it obtains two signatures linked to that session from which it can then extract. Such a strategy of individual session keys with alternate keys could be one approach for constructing efficient, provably concurrently secure schemes without idealized group models. One problem one would need to overcome in this context would be to prove that it is infeasible for the adversary to generate a signature that is not linked to any signing session - this corresponds to the 'restrictive blinding' lemma in [Abe01] whose forking-based proof in the ROM contains a flaw.

As there are many two-witness OR-proof based schemes that build on the Abe-Okamoto scheme and its proof, such as Anonymous Credentials Light [BL13a], BlindOR [AHJ21], and CSI-Otter [Kat+23], another interesting question could be how to formulate a general framework for this type of schemes, for example building on the Linear Function Family framework by Hauck, Kiltz and Loss [HKL19]. Especially for those schemes that are not vulnerable to the ROS attack, such as ACL [1] or Abe, a generic proof strategy could yield variants of the schemes from other assumptions using different building blocks such as lattices, pairings, or even isogenies[2]

8.1.3 The AGM, the Algebraic Wrapper, and Their Relationships to Other Models

Since the introduction of the AGM in 2018, researchers have investigated its relationships to other models. While we provided a partial instantiation in [AHK20], this left open at the time whether the AGM could be instantiated fully. Zhandry [Zha22] showed that no group can 'force' an adversary to behave algebraically by providing a security game that is hard to win assuming the discrete logarithm problem in the AGM, but easy to win in the standard model. The key strategy in the proof is that the game 'feeds' the adversary the encoding of a group element, but split into its bits such that it is 'independent' of any group element encoding (a requirement made by [FKL18]). This does not contradict the algebraic wrapper, as a group element of the algebraic wrapper would always contain an encrypted algebraic representation. Thus, if the adversary was to simply output a group element it got bit-by-bit from an oracle, this group element would contain whichever algebraic explanation the oracle encrypted.

On the other hand, our work on the algebraic wrapper only investigated the relationship of the AGM to the standard model, not to other models such as the generic group models. When introducing the

[1]Benhamouda, Lepoint, Loss, Orrù and Raykova [Ben+21] claimed that ACL was vulnerable to their attack. Later on, we showed in [KLR23a] that this is not true. The application of the ROS attack on ACL contained a typo in the verification equations that when fixed, rendered the attack inefficient. We proved concurrent security in the AGM+ROM.

[2]While the plain ROS attack is not applicable to the isogeny-based CSI-Otter scheme [Kat+23], other ROS-like attacks have since been discovered [KLR23b; DHP23].

AGM as a model, Fuchsbauer, Kiltz and Loss [FKL18] argued that it was a natural weakening of 'the generic group model', i.e. any algorithm that is generic is also algebraic in the sense that it outputs an algebraic representation. It turned out that this is true for Maurer's variant of the GGM [Zha22], it does not hold for Shoups variant [ZZK22].

More closely related to the algebraic wrapper, we left open the question of where it can be applied outside of the reductions we showed how to implement. In fact, the while the algebraic wrapper already relies on matching components of the algebraic explanation to a basis, we only introduced 'reduced representations' as a concept in our work on pairing-free blind signatures [KLX22a]. It seems that reduction strategies using such reduced representations would be a good match for the algebraic wrapper, as the reduced representation, especially to a constant-sized number of group elements, could be easily mapped to a basis of the wrapper. However, many of these proofs, such as for example the proofs of the LRSW assumption [Lys+99] and the BLS signature scheme [BLS01] by Fuchsbauer, Kiltz and Loss [FKL18], or also our proof of Abe's scheme, rely on two reduction strategies that are perfectly indistinguishable in the AGM, but at best computationally indistinguishable in the algebraic wrapper. It remains an open question whether such a reduction strategy can be implemented in the algebraic wrapper or a similar construct.

Another limitation of the wrapper is to deal with reduced representations or bases that are larger than constant size, such as for example when using a q-type assumption. As the base size influences the size of the group description as well as the size of group element encodings in the wrapper, one cannot guess the number of queries the adversary will make and use a matching assumption internally, as the number of queries of the adversary can depend on its input size, and thus on the size of the group description. This leads to a cyclic dependency. It therefore also remains an open question whether it is possible to find a partial instantiation of the AGM where reductions relying on q-type assumptions can be implemented.

There have also been several follow-up works to the introduction of the AGM that introduced variants of what it means to be algebraic. For example, Rotem and Segev [RS20] introduced a variant of the AGM where adversaries are additionally required to explain decision bits, and Katz, Loss and Xu [KLX20] introduced a variant of the AGM where adversaries further need to explain the order in which they performed group operations. While both models have been compared to generic group models, there are no attempts at (partial) instantiation and their relationship to partial instantiations such as the algebraic wrapper remains an open question.

Another extension of the AGM [LPS23] covers the sampling of group elements without knowing their discrete logarithm. In many real-world groups, methods such as hashing into elliptic curves allow for such sampling. The algebraic wrapper offers no such method as one always has to provide an algebraic explanation when sampling a group element. Therefore, it remains an open question how to model such explanation-oblivious methods of sampling in the algebraic wrapper.

Looking towards the GGM(s) again, a recent work by Bauer, Farshim, Harasser and O'Neill [Bau+22] introduced the concept of pseudo-generic groups as a new notion for what it means for a group to behave 'like a generic group'. This yields the natural open question what is the relationship between such a pseudo-generic group and the AGM, as well as whether techniques like those used for the algebraic wrapper can be used to construct a candidate pseudo-generic group.

Lastly, we note that there is still a gap between the AGM and the algebraic wrapper itself, and further research is needed to find where exactly the standard model ends and the uninstantiability of the AGM begins.

Bibliography

[Abe01] M. Abe. 'A Secure Three-Move Blind Signature Scheme for Polynomially Many Signatures'. In: *EUROCRYPT 2001*. Ed. by B. Pfitzmann. Vol. 2045. LNCS. Springer, Heidelberg, May 2001, pp. 136–151. DOI: 10.1007/3-540-44987-6_9 (cit. on pp. 11, 15, 101, 102, 104, 106, 107, 111, 127, 174, 175).

[AF96] M. Abe and E. Fujisaki. 'How to Date Blind Signatures'. In: *ASIACRYPT'96*. Ed. by K. Kim and T. Matsumoto. Vol. 1163. LNCS. Springer, Heidelberg, Nov. 1996, pp. 244–251. DOI: 10.1007/BFb0034851 (cit. on p. 11).

[Agr21] T. Agrikola. 'On Foundations of Protecting Computations'. PhD thesis. Karlsruher Institut für Technologie (KIT), 2021. 281 pp. DOI: 10.5445/IR/1000133798 (cit. on p. xi).

[Agr+22] S. Agrawal, E. Kirshanova, D. Stehlé and A. Yadav. 'Practical, Round-Optimal Lattice-Based Blind Signatures'. In: *ACM CCS 2022*. Ed. by H. Yin, A. Stavrou, C. Cremers and E. Shi. ACM Press, Nov. 2022, pp. 39–53. DOI: 10.1145/3548606.3560650 (cit. on p. 11).

[AH18] T. Agrikola and D. Hofheinz. 'Interactively Secure Groups from Obfuscation'. In: *PKC 2018, Part II*. Ed. by M. Abdalla and R. Dahab. Vol. 10770. LNCS. Springer, Heidelberg, Mar. 2018, pp. 341–370. DOI: 10.1007/978-3-319-76581-5_12 (cit. on pp. 4, 148, 149, 151, 153, 154).

[AHJ21] N. Alkeilani Alkadri, P. Harasser and C. Janson. 'BlindOR: an Efficient Lattice-Based Blind Signature Scheme from OR-Proofs'. In: *CANS 21*. Ed. by M. Conti, M. Stevens and S. Krenn. Vol. 13099. LNCS. Springer, Heidelberg, Dec. 2021, pp. 95–115. DOI: 10.1007/978-3-030-92548-2_6 (cit. on pp. 11, 175).

[AHK20] T. Agrikola, D. Hofheinz and J. Kastner. 'On Instantiating the Algebraic Group Model from Falsifiable Assumptions'. In: *EUROCRYPT 2020, Part II*. Ed. by A. Canteaut and Y. Ishai. Vol. 12106. LNCS. Springer, Heidelberg, May 2020, pp. 96–126. DOI: 10.1007/978-3-030-45724-2_4 (cit. on pp. xi, 4, 19, 22, 147, 175).

[Alb+16] M. R. Albrecht, P. Farshim, D. Hofheinz, E. Larraia and K. G. Paterson. 'Multilinear Maps from Obfuscation'. In: *TCC 2016-A, Part I*. Ed. by E. Kushilevitz and T. Malkin. Vol. 9562. LNCS. Springer, Heidelberg, Jan. 2016, pp. 446–473. DOI: 10.1007/978-3-662-49096-9_19 (cit. on pp. 33, 148, 149, 151, 153, 154).

[AO00] M. Abe and T. Okamoto. 'Provably Secure Partially Blind Signatures'. In: *CRYPTO 2000*. Ed. by M. Bellare. Vol. 1880. LNCS. Springer, Heidelberg, Aug. 2000, pp. 271–286. DOI: 10.1007/3-540-44598-6_17 (cit. on pp. 7, 11, 13–15, 27, 67, 86, 98, 101).

[Bau+22] B. Bauer, P. Farshim, P. Harasser and A. O'Neill. 'Beyond Uber: Instantiating Generic Groups via PGGs'. In: *TCC 2022, Part III*. Ed. by E. Kiltz and V. Vaikuntanathan. Vol. 13749. LNCS. Springer, Heidelberg, Nov. 2022, pp. 212–242. DOI: 10.1007/978-3-031-22368-6_8 (cit. on pp. 4, 176).

[BBG05] D. Boneh, X. Boyen and E.-J. Goh. 'Hierarchical Identity Based Encryption with Constant Size Ciphertext'. In: *EUROCRYPT 2005*. Ed. by R. Cramer. Vol. 3494. LNCS. Springer, Heidelberg, May 2005, pp. 440–456. DOI: 10.1007/11426639_26 (cit. on pp. 4, 18).

[BBR18] K. Bhargavan, R. Barnes and E. Rescorla. *TreeKEM: Asynchronous Decentralized Key Management for Large Dynamic Groups A protocol proposal for Messaging Layer Security (MLS)*. Research Report. Inria Paris, May 2018. URL: https://hal.inria.fr/hal-02425247 (cit. on p. 48).

[BDT22] A. Bienstock, Y. Dodis and Y. Tang. 'Multicast Key Agreement, Revisited'. In: *CT-RSA 2022*. Ed. by S. D. Galbraith. Vol. 13161. LNCS. Springer, Heidelberg, Mar. 2022, pp. 1–25. DOI: 10.1007/978-3-030-95312-6_1 (cit. on p. 48).

[Bel+97] M. Bellare, A. Desai, E. Jokipii and P. Rogaway. 'A Concrete Security Treatment of Symmetric Encryption'. In: *38th FOCS*. IEEE Computer Society Press, Oct. 1997, pp. 394–403. DOI: 10.1109/SFCS.1997.646128 (cit. on p. 33).

[Ben+21] F. Benhamouda, T. Lepoint, J. Loss, M. Orrù and M. Raykova. 'On the (in)security of ROS'. In: *EUROCRYPT 2021, Part I*. Ed. by A. Canteaut and F.-X. Standaert. Vol. 12696. LNCS. Springer, Heidelberg, Oct. 2021, pp. 33–53. DOI: 10.1007/978-3-030-77870-5_2 (cit. on pp. 11, 133, 138, 174, 175).

[Beu+23] W. Beullens, V. Lyubashevsky, N. K. Nguyen and G. Seiler. 'Lattice-Based Blind Signatures: Short, Efficient, and Round-Optimal'. In: *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. CCS '23. Copenhagen, Denmark: Association for Computing Machinery, 2023, pp. 16–29. ISBN: 9798400700507. DOI: 10.1145/3576915.3616613. URL: https://doi.org/10.1145/3576915.3616613 (cit. on p. 11).

[BFL20] B. Bauer, G. Fuchsbauer and J. Loss. 'A Classification of Computational Assumptions in the Algebraic Group Model'. In: *CRYPTO 2020, Part II*. Ed. by D. Micciancio and T. Ristenpart. Vol. 12171. LNCS. Springer, Heidelberg, Aug. 2020, pp. 121–151. DOI: 10.1007/978-3-030-56880-1_5 (cit. on pp. 3, 18, 21, 142).

[BFM14] C. Brzuska, P. Farshim and A. Mittelbach. 'Indistinguishability Obfuscation and UCEs: The Case of Computationally Unpredictable Sources'. In: *CRYPTO 2014, Part I*. Ed. by J. A. Garay and R. Gennaro. Vol. 8616. LNCS. Springer, Heidelberg, Aug. 2014, pp. 188–205. DOI: 10.1007/978-3-662-44371-2_11 (cit. on p. 4).

[BGI14] E. Boyle, S. Goldwasser and I. Ivan. 'Functional Signatures and Pseudorandom Functions'. In: *PKC 2014*. Ed. by H. Krawczyk. Vol. 8383. LNCS. Springer, Heidelberg, Mar. 2014, pp. 501–519. DOI: 10.1007/978-3-642-54631-0_29 (cit. on pp. 5, 40).

[BHK13] M. Bellare, V. T. Hoang and S. Keelveedhi. 'Instantiating Random Oracles via UCEs'. In: *CRYPTO 2013, Part II*. Ed. by R. Canetti and J. A. Garay. Vol. 8043. LNCS. Springer, Heidelberg, Aug. 2013, pp. 398–415. DOI: 10.1007/978-3-642-40084-1_23 (cit. on p. 4).

[Bit+14] N. Bitansky, R. Canetti, O. Paneth and A. Rosen. 'On the existence of extractable one-way functions'. In: *46th ACM STOC*. Ed. by D. B. Shmoys. ACM Press, 2014, pp. 505–514. DOI: 10.1145/2591796.2591859 (cit. on p. 148).

[BL13a] F. Baldimtsi and A. Lysyanskaya. 'Anonymous credentials light'. In: *ACM CCS 2013*. Ed. by A.-R. Sadeghi, V. D. Gligor and M. Yung. ACM Press, Nov. 2013, pp. 1087–1098. DOI: 10.1145/2508859.2516687 (cit. on pp. 104, 175).

[BL13b] F. Baldimtsi and A. Lysyanskaya. 'On the Security of One-Witness Blind Signature Schemes'. In: *ASIACRYPT 2013, Part II*. Ed. by K. Sako and P. Sarkar. Vol. 8270. LNCS. Springer, Heidelberg, Dec. 2013, pp. 82–99. DOI: 10.1007/978-3-642-42045-0_5 (cit. on pp. 11, 174).

[BLS01] D. Boneh, B. Lynn and H. Shacham. 'Short Signatures from the Weil Pairing'. In: *ASIACRYPT 2001*. Ed. by C. Boyd. Vol. 2248. LNCS. Springer, Heidelberg, Dec. 2001, pp. 514–532. DOI: 10.1007/3-540-45682-1_30 (cit. on p. 176).

[BM14] C. Brzuska and A. Mittelbach. 'Using Indistinguishability Obfuscation via UCEs'. In: *ASIACRYPT 2014, Part II*. Ed. by P. Sarkar and T. Iwata. Vol. 8874. LNCS. Springer, Heidelberg, Dec. 2014, pp. 122–141. DOI: 10.1007/978-3-662-45608-8_7 (cit. on p. 4).

[Boy08] X. Boyen. 'The Uber-Assumption Family (Invited Talk)'. In: *PAIRING 2008*. Ed. by S. D. Galbraith and K. G. Paterson. Vol. 5209. LNCS. Springer, Heidelberg, Sept. 2008, pp. 39–56. DOI: 10.1007/978-3-540-85538-5_3 (cit. on pp. 4, 18).

[BR04] M. Bellare and P. Rogaway. *Code-Based Game-Playing Proofs and the Security of Triple Encryption*. Cryptology ePrint Archive, Report 2004/331. https://eprint.iacr.org/2004/331. 2004 (cit. on pp. 5, 24).

[BR93] M. Bellare and P. Rogaway. 'Random Oracles are Practical: A Paradigm for Designing Efficient Protocols'. In: *ACM CCS 93*. Ed. by D. E. Denning, R. Pyle, R. Ganesan, R. S. Sandhu and V. Ashby. ACM Press, Nov. 1993, pp. 62–73. DOI: 10.1145/168588.168596 (cit. on pp. 1, 2).

[Bra+22] N. Brandt, D. Hofheinz, J. Kastner and A. Ünal. 'The Price of Verifiability: Lower Bounds for Verifiable Random Functions'. In: *TCC 2022, Part II*. Ed. by E. Kiltz and V. Vaikuntanathan. Vol. 13748. LNCS. Springer, Heidelberg, Nov. 2022, pp. 747–776. DOI: 10.1007/978-3-031-22365-5_26 (cit. on p. xii).

[Bra94] S. Brands. 'Untraceable Off-line Cash in Wallets with Observers (Extended Abstract)'. In: *CRYPTO'93*. Ed. by D. R. Stinson. Vol. 773. LNCS. Springer, Heidelberg, Aug. 1994, pp. 302–318. DOI: 10.1007/3-540-48329-2_26 (cit. on p. 11).

[Bus+23] M. Buser, R. Dowsley, M. Esgin, C. Gritti, S. Kasra Kermanshahi, V. Kuchta, J. Legrow, J. Liu, R. Phan, A. Sakzad, R. Steinfeld and J. Yu. 'A Survey on Exotic Signatures for Post-Quantum Blockchain: Challenges and Research Directions'. In: *ACM Comput. Surv.* 55.12 (2023). ISSN: 0360-0300. DOI: 10.1145/3572771. URL: https://doi.org/10.1145/3572771 (cit. on p. 11).

[BV98] D. Boneh and R. Venkatesan. 'Breaking RSA May Not Be Equivalent to Factoring'. In: *EUROCRYPT'98*. Ed. by K. Nyberg. Vol. 1403. LNCS. Springer, Heidelberg, 1998, pp. 59–71. DOI: 10.1007/BFb0054117 (cit. on p. 3).

[BW13] D. Boneh and B. Waters. 'Constrained Pseudorandom Functions and Their Applications'. In: *ASIACRYPT 2013, Part II*. Ed. by K. Sako and P. Sarkar. Vol. 8270. LNCS. Springer, Heidelberg, Dec. 2013, pp. 280–300. DOI: 10.1007/978-3-642-42045-0_15 (cit. on pp. 5, 40).

[Can+15] R. Canetti, H. Lin, S. Tessaro and V. Vaikuntanathan. 'Obfuscation of Probabilistic Circuits and Applications'. In: *TCC 2015, Part II*. Ed. by Y. Dodis and J. B. Nielsen. Vol. 9015. LNCS. Springer, Heidelberg, Mar. 2015, pp. 468–497. DOI: 10.1007/978-3-662-46497-7_19 (cit. on pp. 34, 36).

[Can+99] R. Canetti, J. A. Garay, G. Itkis, D. Micciancio, M. Naor and B. Pinkas. 'Multicast Security: A Taxonomy and Some Efficient Constructions'. In: *IEEE INFOCOM'99*. New York, NY, USA, 1999, pp. 708–716 (cit. on pp. 9, 48).

[CDS94] R. Cramer, I. Damgård and B. Schoenmakers. 'Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols'. In: *CRYPTO'94*. Ed. by Y. Desmedt. Vol. 839. LNCS. Springer, Heidelberg, Aug. 1994, pp. 174–187. DOI: 10.1007/3-540-48658-5_19 (cit. on p. 11).

[CFN90] D. Chaum, A. Fiat and M. Naor. 'Untraceable Electronic Cash'. In: *CRYPTO'88*. Ed. by S. Goldwasser. Vol. 403. LNCS. Springer, Heidelberg, Aug. 1990, pp. 319–327. DOI: 10.1007/0-387-34799-2_25 (cit. on p. 11).

[CGH98] R. Canetti, O. Goldreich and S. Halevi. 'The Random Oracle Methodology, Revisited (Preliminary Version)'. In: *30th ACM STOC*. ACM Press, May 1998, pp. 209–218. DOI: 10.1145/276698.276741 (cit. on p. 3).

[Cha82] D. Chaum. 'Blind Signatures for Untraceable Payments'. In: *CRYPTO'82*. Ed. by D. Chaum, R. L. Rivest and A. T. Sherman. Plenum Press, New York, USA, 1982, pp. 199–203 (cit. on pp. 10, 11).

[Cha88] D. Chaum. 'Elections with Unconditionally-Secret Ballots and Disruption Equivalent to Breaking RSA'. In: *EUROCRYPT'88*. Ed. by C. G. Günther. Vol. 330. LNCS. Springer, Heidelberg, May 1988, pp. 177–182. DOI: 10.1007/3-540-45961-8_15 (cit. on p. 11).

[CKM23] E. C. Crites, C. Komlo and M. Maller. 'Fully Adaptive Schnorr Threshold Signatures'. In: *CRYPTO 2023, Part I*. Ed. by H. Handschuh and A. Lysyanskaya. Vol. 14081. LNCS. Springer, Heidelberg, Aug. 2023, pp. 678–709. DOI: 10.1007/978-3-031-38557-5_22 (cit. on p. 3).

[CL01] J. Camenisch and A. Lysyanskaya. 'An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation'. In: *EUROCRYPT 2001*. Ed. by B. Pfitzmann. Vol. 2045. LNCS. Springer, Heidelberg, May 2001, pp. 93–118. DOI: 10.1007/3-540-44987-6_7 (cit. on p. 11).

[Cor02] J.-S. Coron. 'Optimal Security Proofs for PSS and Other Signature Schemes'. In: *EUROCRYPT 2002*. Ed. by L. R. Knudsen. Vol. 2332. LNCS. Springer, Heidelberg, 2002, pp. 272–287. DOI: 10.1007/3-540-46035-7_18 (cit. on p. 142).

[Cra97] R. Cramer. 'Modular Design of Secure yet Practical Cryptographic Protocols'. PhD thesis. University of Amsterdam, Jan. 1997 (cit. on pp. 2, 7).

[Cri+23] E. C. Crites, C. Komlo, M. Maller, S. Tessaro and C. Zhu. 'Snowblind: A Threshold Blind Signature in Pairing-Free Groups'. In: *CRYPTO 2023, Part I*. Ed. by H. Handschuh and A. Lysyanskaya. Vol. 14081. LNCS. Springer, Heidelberg, Aug. 2023, pp. 710–742. DOI: 10.1007/978-3-031-38557-5_23 (cit. on pp. 3, 11, 18, 174).

[CS02] R. Cramer and V. Shoup. 'Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption'. In: *EUROCRYPT 2002*. Ed. by L. R. Knudsen. Vol. 2332. LNCS. Springer, Heidelberg, 2002, pp. 45–64. DOI: 10.1007/3-540-46035-7_4 (cit. on p. 25).

[Den02] A. W. Dent. 'Adapting the Weaknesses of the Random Oracle Model to the Generic Group Model'. In: *ASIACRYPT 2002*. Ed. by Y. Zheng. Vol. 2501. LNCS. Springer, Heidelberg, Dec. 2002, pp. 100–109. DOI: 10.1007/3-540-36178-2_6 (cit. on p. 4).

[DH76] W. Diffie and M. E. Hellman. 'New Directions in Cryptography'. In: *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654. DOI: 10.1109/TIT.1976.1055638 (cit. on pp. 1, 4).

[DHP23] K. Do, L. Hanzlik and E. Paracucchi. *M&M'S: Mix and Match Attacks on Schnorr-type Blind Signatures with Repetition*. Cryptology ePrint Archive, Paper 2023/1588. https://eprint.iacr.org/2023/1588. 2023. URL: https://eprint.iacr.org/2023/1588 (cit. on p. 175).

[dK22] R. del Pino and S. Katsumata. 'A New Framework for More Efficient Round-Optimal Lattice-Based (Partially) Blind Signature via Trapdoor Sampling'. In: *CRYPTO 2022, Part II*. Ed. by Y. Dodis and T. Shrimpton. Vol. 13508. LNCS. Springer, Heidelberg, Aug. 2022, pp. 306–336. DOI: 10.1007/978-3-031-15979-4_11 (cit. on p. 11).

[DL78] R. A. Demillo and R. J. Lipton. 'A probabilistic remark on algebraic program testing'. In: *Information Processing Letters* 7.4 (1978), pp. 193–195. ISSN: 0020-0190. DOI: https://doi.org/10.1016/0020-0190(78)90067-4. URL: https://www.sciencedirect.com/science/article/pii/0020019078900674 (cit. on pp. 3, 31).

[Far+18] P. Farshim, J. Hesse, D. Hofheinz and E. Larraia. 'Graded Encoding Schemes from Obfuscation'. In: *PKC 2018, Part II*. Ed. by M. Abdalla and R. Dahab. Vol. 10770. LNCS. Springer, Heidelberg, Mar. 2018, pp. 371–400. DOI: 10.1007/978-3-319-76581-5_13 (cit. on pp. 148, 149, 151, 153, 154).

[FJS19] N. Fleischhacker, T. Jager and D. Schröder. 'On Tight Security Proofs for Schnorr Signatures'. In: *Journal of Cryptology* 32.2 (Apr. 2019), pp. 566–599. DOI: 10.1007/s00145-019-09311-5 (cit. on p. 25).

[FKL18] G. Fuchsbauer, E. Kiltz and J. Loss. 'The Algebraic Group Model and its Applications'. In: *CRYPTO 2018, Part II*. Ed. by H. Shacham and A. Boldyreva. Vol. 10992. LNCS. Springer, Heidelberg, Aug. 2018, pp. 33–62. DOI: 10.1007/978-3-319-96881-0_2 (cit. on pp. 3, 18, 20, 21, 149, 155–158, 175, 176).

[FOO93] A. Fujioka, T. Okamoto and K. Ohta. 'A Practical Secret Voting Scheme for Large Scale Elections'. In: *AUSCRYPT'92*. Ed. by J. Seberry and Y. Zheng. Vol. 718. LNCS. Springer, Heidelberg, Dec. 1993, pp. 244–251. DOI: 10.1007/3-540-57220-1_66 (cit. on p. 11).

[FPS20] G. Fuchsbauer, A. Plouviez and Y. Seurin. 'Blind Schnorr Signatures and Signed ElGamal Encryption in the Algebraic Group Model'. In: *EUROCRYPT 2020, Part II*. Ed. by A. Canteaut and Y. Ishai. Vol. 12106. LNCS. Springer, Heidelberg, May 2020, pp. 63–95. DOI: 10.1007/978-3-030-45724-2_3 (cit. on pp. 3, 11, 18, 20, 21, 135, 137, 147, 155, 159, 162–165, 168–171).

[FS87] A. Fiat and A. Shamir. 'How to Prove Yourself: Practical Solutions to Identification and Signature Problems'. In: *CRYPTO'86*. Ed. by A. M. Odlyzko. Vol. 263. LNCS. Springer, Heidelberg, Aug. 1987, pp. 186–194. DOI: 10.1007/3-540-47721-7_12 (cit. on pp. 2, 7).

[FS90] U. Feige and A. Shamir. 'Witness Indistinguishable and Witness Hiding Protocols'. In: *22nd ACM STOC*. ACM Press, May 1990, pp. 416–426. DOI: 10.1145/100216.100272 (cit. on p. 33).

[GGH13] S. Garg, C. Gentry and S. Halevi. 'Candidate Multilinear Maps from Ideal Lattices'. In: *EUROCRYPT 2013*. Ed. by T. Johansson and P. Q. Nguyen. Vol. 7881. LNCS. Springer, Heidelberg, May 2013, pp. 1–17. DOI: 10.1007/978-3-642-38348-9_1 (cit. on pp. 147, 148).

[GGM84a] O. Goldreich, S. Goldwasser and S. Micali. 'How to Construct Random Functions (Extended Abstract)'. In: *25th FOCS*. IEEE Computer Society Press, Oct. 1984, pp. 464–479. DOI: 10.1109/SFCS.1984.715949 (cit. on p. 5).

[GGM84b] O. Goldreich, S. Goldwasser and S. Micali. 'On the Cryptographic Applications of Random Functions'. In: *CRYPTO'84*. Ed. by G. R. Blakley and D. Chaum. Vol. 196. LNCS. Springer, Heidelberg, Aug. 1984, pp. 276–288 (cit. on p. 40).

[Goo] *VPN by Google One, explained*. URL: https://one.google.com/about/vpn/howitworks (visited on 08/12/2023) (cit. on p. 11).

[GS08] J. Groth and A. Sahai. 'Efficient Non-interactive Proof Systems for Bilinear Groups'. In: *EUROCRYPT 2008*. Ed. by N. P. Smart. Vol. 4965. LNCS. Springer, Heidelberg, Apr. 2008, pp. 415–432. DOI: 10.1007/978-3-540-78967-3_24 (cit. on pp. 33, 34).

[HKK23] D. Hofheinz, J. Kastner and K. Klein. 'The Power of Undirected Rewindings for Adaptive Security'. In: *CRYPTO 2023, Part II*. Ed. by H. Handschuh and A. Lysyanskaya. Vol. 14082. LNCS. Springer, Heidelberg, Aug. 2023, pp. 725–758. DOI: 10.1007/978-3-031-38545-2_24 (cit. on pp. xi, 22, 30, 37).

[HKL19] E. Hauck, E. Kiltz and J. Loss. 'A Modular Treatment of Blind Signatures from Identification Schemes'. In: *EUROCRYPT 2019, Part III*. Ed. by Y. Ishai and V. Rijmen. Vol. 11478. LNCS. Springer, Heidelberg, May 2019, pp. 345–375. DOI: 10.1007/978-3-030-17659-4_12 (cit. on pp. 11, 175).

[HLW23] L. Hanzlik, J. Loss and B. Wagner. 'Rai-Choo! Evolving Blind Signatures to the Next Level'. In: *EUROCRYPT 2023, Part V*. Ed. by C. Hazay and M. Stam. Vol. 14008. LNCS. Springer, Heidelberg, Apr. 2023, pp. 753–783. DOI: 10.1007/978-3-031-30589-4_26 (cit. on p. 11).

[Hof+23] D. Hofheinz, J. Kastner, A. Ünal and B. Ursu. *Decoding LTFs in the Generic Group Model*. Cryptology ePrint Archive, Paper 2023/866. https://eprint.iacr.org/2023/866. 2023. URL: https://eprint.iacr.org/2023/866 (cit. on p. xii).

[Hof+24] D. Hofheinz, K. Hostáková, J. Kastner, K. Klein and A. Ünal. *Compact Lossy Trapdoor Functions and Selective Opening Security From LWE*. PKC 2024. To appear. 2024. URL: https://eprint.iacr.org/2023/864 (cit. on p. xii).

[HU19] D. Hofheinz and B. Ursu. 'Dual-Mode NIZKs from Obfuscation'. In: *ASIACRYPT 2019, Part I*. Ed. by S. D. Galbraith and S. Moriai. Vol. 11921. LNCS. Springer, Heidelberg, Dec. 2019, pp. 311–341. DOI: 10.1007/978-3-030-34578-5_12 (cit. on p. 34).

[Jaf+17] Z. Jafargholi, C. Kamath, K. Klein, I. Komargodski, K. Pietrzak and D. Wichs. 'Be Adaptive, Avoid Overcommitting'. In: *CRYPTO 2017, Part I*. Ed. by J. Katz and H. Shacham. Vol. 10401. LNCS. Springer, Heidelberg, Aug. 2017, pp. 133–163. DOI: 10.1007/978-3-319-63688-7_5 (cit. on pp. 10, 49, 50).

[JM18] D. Jost and U. Maurer. 'Security Definitions for Hash Functions: Combining UCE and Indifferentiability'. In: *SCN 18*. Ed. by D. Catalano and R. De Prisco. Vol. 11035. LNCS. Springer, Heidelberg, Sept. 2018, pp. 83–101. DOI: 10.1007/978-3-319-98113-0_5 (cit. on p. 4).

[Kam+21] C. Kamath, K. Klein, K. Pietrzak and M. Walter. 'The Cost of Adaptivity in Security Games on Graphs'. In: *TCC 2021, Part II*. Ed. by K. Nissim and B. Waters. Vol. 13043. LNCS. Springer, Heidelberg, Nov. 2021, pp. 550–581. DOI: 10.1007/978-3-030-90453-1_19 (cit. on p. 6).

[Kat+23] S. Katsumata, Y.-F. Lai, J. T. LeGrow and L. Qin. 'CSI -Otter: Isogeny-Based (Partially) Blind Signatures from the Class Group Action with a Twist'. In: *CRYPTO 2023, Part III*. Ed. by H. Handschuh and A. Lysyanskaya. Vol. 14083. LNCS. Springer, Heidelberg, Aug. 2023, pp. 729–761. DOI: 10.1007/978-3-031-38548-3_24 (cit. on pp. 11, 175).

[Ker83] A. Kerckhoffs. 'La cryptographie militaire'. In: *Journal des sciences militaires* IX (1883) (cit. on p. 1).

[Kia+13] A. Kiayias, S. Papadopoulos, N. Triandopoulos and T. Zacharias. 'Delegatable pseudorandom functions and applications'. In: *ACM CCS 2013*. Ed. by A.-R. Sadeghi, V. D. Gligor and M. Yung. ACM Press, Nov. 2013, pp. 669–684. DOI: 10.1145/2508859.2516668 (cit. on pp. 5, 40).

[Kle+21] K. Klein, G. Pascual-Perez, M. Walter, C. Kamath, M. Capretto, M. Cueto, I. Markov, M. Yeo, J. Alwen and K. Pietrzak. 'Keep the Dirt: Tainted TreeKEM, Adaptively and Actively Secure Continuous Group Key Agreement'. In: *2021 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2021, pp. 268–284. DOI: 10.1109/SP40001.2021.00035 (cit. on p. 48).

[KLR23a] J. Kastner, J. Loss and O. Renawi. 'Concurrent Security of Anonymous Credentials Light, Revisited'. In: *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. CCS '23. Copenhagen, Denmark: Association for Computing Machinery, 2023, pp. 45–59. ISBN: 9798400700507. DOI: 10.1145/3576915.3623184. URL: https://doi.org/10.1145/3576915.3623184 (cit. on pp. xi, 16, 22, 101, 109, 175).

[KLR23b] S. Katsumata, Y.-F. Lai and M. Reichle. *Breaking Parallel ROS: Implication for Isogeny and Lattice-based Blind Signatures*. Cryptology ePrint Archive, Paper 2023/1603. https://eprint.iacr.org/2023/1603. 2023. URL: https://eprint.iacr.org/2023/1603 (cit. on p. 175).

[KLX20] J. Katz, J. Loss and J. Xu. 'On the Security of Time-Lock Puzzles and Timed Commitments'. In: *TCC 2020, Part III*. Ed. by R. Pass and K. Pietrzak. Vol. 12552. LNCS. Springer, Heidelberg, Nov. 2020, pp. 390–413. DOI: 10.1007/978-3-030-64381-2_14 (cit. on pp. 3, 18, 176).

[KLX22a] J. Kastner, J. Loss and J. Xu. 'On Pairing-Free Blind Signature Schemes in the Algebraic Group Model'. In: *PKC 2022, Part II*. Ed. by G. Hanaoka, J. Shikata and Y. Watanabe. Vol. 13178. LNCS. Springer, Heidelberg, Mar. 2022, pp. 468–497. DOI: 10.1007/978-3-030-97131-1_16 (cit. on pp. xi, 3, 11, 16, 18, 21, 22, 27, 101, 112, 116, 135, 176).

[KLX22b] J. Kastner, J. Loss and J. Xu. *The Abe-Okamoto Partially Blind Signature Scheme Revisited*. Cryptology ePrint Archive, Report 2022/1232. https://eprint.iacr.org/2022/1232. 2022 (cit. on p. 127).

[KLX22c] J. Kastner, J. Loss and J. Xu. 'The Abe-Okamoto Partially Blind Signature Scheme Revisited'. In: *ASIACRYPT 2022, Part IV*. Ed. by S. Agrawal and D. Lin. Vol. 13794. LNCS. Springer, Heidelberg, Dec. 2022, pp. 279–309. DOI: 10.1007/978-3-031-22972-5_10 (cit. on pp. xi, 22, 27, 30, 67, 127).

[KNR23] J. Kastner, K. Nguyen and M. Reichle. *Pairing-Free Blind Signatures from Standard Assumptions in the ROM*. In Submission. 2023 (cit. on p. xii).

[KP19] J. Kastner and J. Pan. *Towards Instantiating the Algebraic Group Model*. Cryptology ePrint Archive, Report 2019/1018. https://eprint.iacr.org/2019/1018. 2019 (cit. on pp. 4, 147, 148).

[Los23] J. Loss. personal communication. 2023 (cit. on p. 3).

[LPS23] H. Lipmaa, R. Parisella and J. Siim. 'Algebraic Group Model with Oblivious Sampling'. In: *Theory of Cryptography*. Ed. by G. Rothblum and H. Wee. Cham: Springer Nature Switzerland, 2023, pp. 363–392. ISBN: 978-3-031-48624-1 (cit. on pp. 18, 176).

[Lys+99] A. Lysyanskaya, R. L. Rivest, A. Sahai and S. Wolf. 'Pseudonym Systems'. In: *SAC 1999*. Ed. by H. M. Heys and C. M. Adams. Vol. 1758. LNCS. Springer, Heidelberg, Aug. 1999, pp. 184–199. DOI: 10.1007/3-540-46513-8_14 (cit. on p. 176).

[Mau05] U. M. Maurer. 'Abstract Models of Computation in Cryptography (Invited Paper)'. In: *10th IMA International Conference on Cryptography and Coding*. Ed. by N. P. Smart. Vol. 3796. LNCS. Springer, Heidelberg, Dec. 2005, pp. 1–12 (cit. on pp. 3, 18).

[Mil82] F. Miller. *Telegraphic Code to Insure Privacy and Secrecy in the Transmission of Telegrams*. C.M. Cornwell, 1882. URL: https://books.google.ch/books?id=tT9WAAAAYAAJ (cit. on p. 1).

[Nec94] V. I. Nechaev. 'Complexity of a Determinate Algorithm for the Discrete Logarithm'. In: *Mathematical Notes* 55.2 (1994), pp. 165–172 (cit. on p. 2).

[OA03] M. Ohkubo and M. Abe. *Security of Some Three-move Blind Signature Schemes Reconsidered*. The 2003 Symposium on Cryptography and Information Security. Hamamatsu,Japan, 2003 (cit. on pp. 15, 101).

[Oka93] T. Okamoto. 'Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes'. In: *CRYPTO'92*. Ed. by E. F. Brickell. Vol. 740. LNCS. Springer, Heidelberg, Aug. 1993, pp. 31–53. DOI: 10.1007/3-540-48071-4_3 (cit. on p. 2).

[OO92] T. Okamoto and K. Ohta. 'Universal Electronic Cash'. In: *CRYPTO'91*. Ed. by J. Feigenbaum. Vol. 576. LNCS. Springer, Heidelberg, Aug. 1992, pp. 324–337. DOI: `10.1007/3-540-46766-1_27` (cit. on p. 11).

[Pan07] S. Panjwani. 'Tackling Adaptive Corruptions in Multicast Encryption Protocols'. In: *TCC 2007*. Ed. by S. P. Vadhan. Vol. 4392. LNCS. Springer, Heidelberg, Feb. 2007, pp. 21–40. DOI: `10.1007/978-3-540-70936-7_2` (cit. on pp. 9, 48).

[PS00] D. Pointcheval and J. Stern. 'Security Arguments for Digital Signatures and Blind Signatures'. In: *Journal of Cryptology* 13.3 (June 2000), pp. 361–396. DOI: `10.1007/s001450010003` (cit. on pp. 2, 7, 10, 30).

[PS19] C. Peikert and S. Shiehian. 'Noninteractive Zero Knowledge for NP from (Plain) Learning with Errors'. In: *CRYPTO 2019, Part I*. Ed. by A. Boldyreva and D. Micciancio. Vol. 11692. LNCS. Springer, Heidelberg, Aug. 2019, pp. 89–114. DOI: `10.1007/978-3-030-26948-7_4` (cit. on p. 34).

[PS96] D. Pointcheval and J. Stern. 'Security Proofs for Signature Schemes'. In: *EUROCRYPT'96*. Ed. by U. M. Maurer. Vol. 1070. LNCS. Springer, Heidelberg, May 1996, pp. 387–398. DOI: `10.1007/3-540-68339-9_33` (cit. on pp. 7, 30).

[PV05] P. Paillier and D. Vergnaud. 'Discrete-Log-Based Signatures May Not Be Equivalent to Discrete Log'. In: *ASIACRYPT 2005*. Ed. by B. K. Roy. Vol. 3788. LNCS. Springer, Heidelberg, Dec. 2005, pp. 1–20. DOI: `10.1007/11593447_1` (cit. on p. 3).

[RS20] L. Rotem and G. Segev. 'Algebraic Distinguishers: From Discrete Logarithms to Decisional Uber Assumptions'. In: *TCC 2020, Part III*. Ed. by R. Pass and K. Pietrzak. Vol. 12552. LNCS. Springer, Heidelberg, Nov. 2020, pp. 366–389. DOI: `10.1007/978-3-030-64381-2_13` (cit. on pp. 3, 18, 176).

[RSA78] R. L. Rivest, A. Shamir and L. M. Adleman. 'A Method for Obtaining Digital Signatures and Public-Key Cryptosystems'. In: *Communications of the Association for Computing Machinery* 21.2 (Feb. 1978), pp. 120–126. DOI: `10.1145/359340.359342` (cit. on p. 1).

[Rüc10] M. Rückert. 'Lattice-Based Blind Signatures'. In: *ASIACRYPT 2010*. Ed. by M. Abe. Vol. 6477. LNCS. Springer, Heidelberg, Dec. 2010, pp. 413–430. DOI: `10.1007/978-3-642-17373-8_24` (cit. on p. 11).

[Sch01] C.-P. Schnorr. 'Security of Blind Discrete Log Signatures against Interactive Attacks'. In: *ICICS 01*. Ed. by S. Qing, T. Okamoto and J. Zhou. Vol. 2229. LNCS. Springer, Heidelberg, Nov. 2001, pp. 1–12 (cit. on p. 11).

[Sch80] J. T. Schwartz. 'Fast Probabilistic Algorithms for Verification of Polynomial Identities'. In: *J. ACM* 27.4 (1980), 701?717. ISSN: 0004-5411. DOI: `10.1145/322217.322225`. URL: `https://doi.org/10.1145/322217.322225` (cit. on pp. 3, 31).

[Sch90] C.-P. Schnorr. 'Efficient Identification and Signatures for Smart Cards'. In: *CRYPTO'89*. Ed. by G. Brassard. Vol. 435. LNCS. Springer, Heidelberg, Aug. 1990, pp. 239–252. DOI: `10.1007/0-387-34805-0_22` (cit. on pp. 2, 7, 11).

[Sha49] C. E. Shannon. 'Communication theory of secrecy systems'. In: *Bell Systems Technical Journal* 28.4 (1949), pp. 656–715 (cit. on p. 1).

[Sho04] V. Shoup. *Sequences of games: a tool for taming complexity in security proofs*. Cryptology ePrint Archive, Report 2004/332. `https://eprint.iacr.org/2004/332`. 2004 (cit. on pp. 5, 24, 163).

[Sho97] V. Shoup. 'Lower Bounds for Discrete Logarithms and Related Problems'. In: *EUROCRYPT'97*. Ed. by W. Fumy. Vol. 1233. LNCS. Springer, Heidelberg, May 1997, pp. 256–266. DOI: 10.1007/3-540-69053-0_18 (cit. on pp. 2, 18).

[Sue21] Suetonius. *De Vita Caesarum*. 121. URL: http://thelatinlibrary.com/suetonius/suet.caesar.html#56 (cit. on p. 1).

[TZ22] S. Tessaro and C. Zhu. 'Short Pairing-Free Blind Signatures with Exponential Security'. In: *EUROCRYPT 2022, Part II*. Ed. by O. Dunkelman and S. Dziembowski. Vol. 13276. LNCS. Springer, Heidelberg, 2022, pp. 782–811. DOI: 10.1007/978-3-031-07085-3_27 (cit. on pp. 3, 11, 18, 174).

[TZ23] S. Tessaro and C. Zhu. 'Revisiting BBS Signatures'. In: *EUROCRYPT 2023, Part V*. Ed. by C. Hazay and M. Stam. Vol. 14008. LNCS. Springer, Heidelberg, Apr. 2023, pp. 691–721. DOI: 10.1007/978-3-031-30589-4_24 (cit. on pp. 3, 18).

[Ver26] G. S. Vernam. 'Cipher Printing Telegraph Systems For Secret Wire and Radio Telegraphic Communications'. In: *Transactions of the American Institute of Electrical Engineers* XLV (1926), pp. 295–301. DOI: 10.1109/T-AIEE.1926.5061224 (cit. on p. 1).

[Wag02] D. Wagner. 'A Generalized Birthday Problem'. In: *CRYPTO 2002*. Ed. by M. Yung. Vol. 2442. LNCS. Springer, Heidelberg, Aug. 2002, pp. 288–303. DOI: 10.1007/3-540-45708-9_19 (cit. on p. 11).

[WGL00] C. K. Wong, M. G. Gouda and S. S. Lam. 'Secure group communications using key graphs'. In: *IEEE/ACM Trans. Netw.* 8.1 (2000), pp. 16–30. DOI: 10.1109/90.836475. URL: https://doi.org/10.1109/90.836475 (cit. on pp. 9, 48).

[WHA98] D. M. Wallner, E. J. Harder and R. C. Agee. *Key Management for Multicast: Issues and Architectures*. Internet Draft. http://www.ietf.org/ID.html. Sept. 1998 (cit. on pp. 9, 48).

[YL19] X. Yi and K.-Y. Lam. 'A New Blind ECDSA Scheme for Bitcoin Transaction Anonymity'. In: *ASIACCS 19*. Ed. by S. D. Galbraith, G. Russello, W. Susilo, D. Gollmann, E. Kirda and Z. Liang. ACM Press, July 2019, pp. 613–620. DOI: 10.1145/3321705.3329816 (cit. on p. 11).

[Zha22] M. Zhandry. 'To Label, or Not To Label (in Generic Groups)'. In: *CRYPTO 2022, Part III*. Ed. by Y. Dodis and T. Shrimpton. Vol. 13509. LNCS. Springer, Heidelberg, Aug. 2022, pp. 66–96. DOI: 10.1007/978-3-031-15982-4_3 (cit. on pp. 3, 4, 18, 175, 176).

[Zip79] R. Zippel. 'Probabilistic algorithms for sparse polynomials'. In: *Symbolic and Algebraic Computation*. Ed. by E. W. Ng. Berlin, Heidelberg: Springer Berlin Heidelberg, 1979, pp. 216–226. ISBN: 978-3-540-35128-3 (cit. on pp. 3, 31).

[ZZK22] C. Zhang, H.-S. Zhou and J. Katz. 'An Analysis of the Algebraic Group Model'. In: *ASIACRYPT 2022, Part IV*. Ed. by S. Agrawal and D. Lin. Vol. 13794. LNCS. Springer, Heidelberg, Dec. 2022, pp. 310–322. DOI: 10.1007/978-3-031-22972-5_11 (cit. on pp. 4, 18, 176).

List of Figures