Diss. ETH No. 21970

Strengthening the Security of Key Exchange Protocols

A thesis submitted to attain the degree of
Doctor of Sciences of ETH Zurich
(Dr. sc. ETH Zurich)

presented by
Michèle Feltz
MSc in Mathematics, University of Fribourg
born on August 4, 1986
citizen of Luxembourg

accepted on the recommendation of
Prof. Dr. David Basin, examiner
Prof. Dr. Cas Cremers, co-examiner
Prof. Dr. Marc Fischlin, co-examiner

2014

Abstract

Authenticated key exchange (AKE) protocols are central building blocks of security protocols such as TLS, IPsec, and SSH, that are used in modern distributed applications. The security of these protocols can however be affected by threats such as attacks on users' long-term secret keys, attacks based on malicious key registration, and attacks on the random number generator used by the protocol. The goal of this thesis is to model advanced security threats against authenticated key exchange protocols and to develop methods that strengthen the security of these protocols and make them secure against the considered threats.

In the first part of this thesis we extend existing security models to capture relevant attacks that lie outside the scope of these models. We advance the state-of-the-art by integrating perfect forward secrecy into a model that captures key compromise impersonation attacks and the leakage of session-specific randomness. We provide a generic security-strengthening transformation to achieve perfect forward secrecy, and show that two-message AKE protocols can achieve security in our model by applying our transformation to protocols that are secure in weaker models. Most security models for authenticated key exchange do not explicitly model the associated certification system, which includes the certification authority and its behaviour. We launch the first systematic analysis of AKE incorporating certification systems. To this end, we develop a framework that allows for explicit modelling of the public key certification process; this framework enables us to capture attacks based on dynamic adversarial registration of arbitrary public keys. We are the first to provide a generic approach to achieve strong security guarantees against adversaries who can register arbitrary public keys with a certification authority that does not perform any verification on combinations of identities and public keys.

In the second part of this thesis we explore the limits of AKE security with regard to different protocol classes. We derive a hierarchy of strong security models from impossibility results on the security of each protocol class. Our impossibility results show the impossibility of achieving certain security guarantees in specific protocol classes. In particular, we analyze the security of protocols in the presence of adversaries who can perform attacks based on choosing the randomness used in protocol sessions. We construct novel variants of the NAXOS protocol, which achieve security against such attacks.

Résumé

Les protocoles d'échange de clé sont des éléments centraux de protocoles de sécurité tels que TLS, IPsec, et SSH, qui sont utilisés dans de nombreuses applications distribuées de commerce électronique. La sécurité de ces protocoles peut cependant être affectée par des menaces comme, par exemple, des attaques sur les clés secrètes à long terme, des attaques basées sur l'exploitation de certificats frauduleux émis par des autorités de certification, et des attaques sur le générateur de nombres aléatoires utilisé par le protocole. L'objectif de cette thèse consiste à modéliser des menaces avancées contre les protocoles d'échange de clé et de développer des méthodes qui renforcent la sécurité de ces protocoles et les immunisent contre les attaques considérées.

Dans la première partie de cette thèse, nous étendons des modèles de sécurité existants à des attaques qui ne peuvent actuellement pas être prises en compte par ces modèles. Nous intégrons la notion de "perfect forward secrecy" dans un modèle qui prend en compte des attaques sur la clé secrète à long terme de l'auteur de la session sous attaque et la fuite de valeurs aléatoires spécifiques à des sessions. Nous proposons une transformation générique qui permet d'assurer la propriété de "perfect forward secrecy", et montrons que les protocoles d'échange de clé à deux messages peuvent garantir la sécurité dans notre modèle en appliquant notre transformation à des protocoles qui sont sûrs dans des modèles plus faibles. La plupart des modèles de sécurité ne modélisent pas explicitement le système de certification associé, qui comprend l'autorité de certification et son comportement. Nous procédons à une analyse systématique de l'échange de clé tenant compte du système de certification. Pour cela, nous développons un système de référence qui prend en compte la modélisation explicite du processus de certification; ce système nous permet de saisir des attaques basées sur l'enregistrement dynamique de clés publiques arbitraires. À notre connaissance, nous sommes les premiers à proposer une approche générique afin d'atteindre des garanties de sécurité élevées face à des adversaires qui peuvent obtenir des certificats pour des clés publiques arbitraires de la part d'une autorité de certification qui n'effectue aucune vérification de combinaisons d'identités et de clés publiques.

Dans la seconde partie de cette thèse, nous explorons les limites de la sécurité de l'échange de clé par rapport à différentes catégories de protocoles. Nous dérivons une hiérarchie de modèles de sécurité de résultats d'impossibilité sur la sécurité de chaque catégorie. Nos résultats d'impossibilité montrent l'impossibilité d'atteindre certaines garanties de sécurité dans des catégories de protocoles spécifiques. En particulier, nous analysons la sécurité de protocoles en présence d'adversaires qui effectuent des attaques basées sur le choix de valeurs aléatoires utilisées dans des sessions. Nous construisons de nouvelles variantes du protocole NAXOS qui assurent la sécurité contre de telles attaques.

Zusammenfassung

Schlüsselaustauschprotokolle sind zentrale Komponenten von Sicherheitsprotokollen wie TLS, IPsec, und SSH, die in modernen verteilten Applikationen benutzt werden. Die Sicherheit dieser Protokolle kann jedoch durch diverse Gefahren beeinträchtigt werden, wie Angriffe auf die langfristigen geheimen Schlüssel von Teilnehmern, Angriffe, die auf der Ausstellung von Zertifikaten für beliebige Schlüssel des Angreifers basieren, und Angriffe auf den Zufallsgenerator der vom Protokoll benutzt wird. Ziel dieser Dissertation ist es, Gefahren für Schlüsselaustauschprotokolle zu modellieren und Methoden zu entwickeln, welche die Sicherheit dieser Protokolle verstärkt.

In dem ersten Teil dieser Dissertation erweitern wir existierende Sicherheitsmodelle um Angriffe zu modellieren, welche ausser Reichweite dieser Modelle liegen. Wir integrieren "perfect forward secrecy" in ein Modell, welches Angriffe auf die langfristigen geheimen Schlüssel von gewissen Teilnehmern vor dem Ende einer Sitzung sowie den Verlust von sitzungsspezifischen Zufallswerten modelliert. Wir schlagen eine generische sicherheitstärkende Transformation vor, die uns erlaubt, "perfect forward secrecy" zu garantieren. Wir zeigen, dass Schlüsselaustauschprotokolle mit nur zwei Nachrichten schon Sicherheit in unserem Modell erreichen können, indem wir unsere Transformation auf schwächere Protokolle anwenden. Die meisten Sicherheitsmodelle modellieren nicht das dazugehörige Zertifizierungssystem, welches die Zertifizierungsinstanz und deren Verhalten einschliesst. Wir starten die erste systematische Untersuchung von Schlüsselaustauschprotokollen unter Berücksichtigung von Zertifizierungssystemen. Zu diesem Zweck entwickeln wir ein System das explizites Modellieren des Zertifizierungsprozesses von öffentlichen Schlüsseln berücksichtigt. Dieses System erlaubt es uns Angriffe zu modellieren, die auf der dynamischen Registrierung von beliebigen öffentlichen Schlüsseln basieren. Unter der Annahme, dass die Zertifizierungsinstanz die Informationen, die in den Anfragen enthalten sind, wie Identität oder öffentlichen Schlüssel, nicht überprüft, führen wir eine generische Methode ein um starke Garantien zu erzielen gegen Angreifer, die Zertifikate über beliebige öffentliche Schlüssel verwenden, welche von dieser Zertifizierungsinstanz ausgestellt wurden.

In dem zweiten Teil dieser Dissertation erforschen wir die Grenzen des sicheren Schlüsselaustausches bezüglich verschiedener Protokollklassen. Aus unseren Unmöglichkeitsresultaten leiten wir eine Hierarchie von starken Sicherheitsmodellen ab. Im Speziellen untersuchen wir die Sicherheit von Protokollen gegen Angreifer die sitzungsspezifische Zufallswerte manipulieren können. Wir präsentieren neue Varianten des NAXOS Protokolls und beweisen deren Sicherheit gegen solche Angriffe.

Acknowledgements

I would like to express my gratitude to my advisor David Basin for giving me the opportunity of pursuing research in the Institute of Information Security and for his support over the last years. I owe many thanks to my advisor Cas Cremers for his time, patience, and for the many helpful and productive discussions we had at ETH Zurich and at the University of Oxford. It was a great pleasure for me to work with Colin Boyd, Cas Cremers, Kenneth Paterson, Bertram Poettering, and Douglas Stebila on authenticated key exchange security incorporating certification systems. Thanks for the very efficient collaboration and for your support. Also, I would like to thank Marc Fischlin for his willingness to serve as a co-examiner.

During my PhD, I have had many constructive discussions with other researchers, including Bruno Blanchet, Marko Horvat, Jean Lancrenon, Arno Mittelbach, and Peter Ryan. I would like to thank Jannik Dreier, Matúš Harvan, Srdjan Marinovic, Simon Meier, Binh Thanh Nguyen, Saša Radomirović, Ralf Sasse, Benedikt Schmidt, Christoph Sprenger, Björn Tackmann, and Eugen Zălinescu for valuable feedback on my work, and my office mates Joel Reardon, Mario Frank, and Petar Tsankov for an excellent working atmosphere.

I am grateful to the organizers of the Summer School Marktoberdorf and the workshop on Formal and Computational Cryptographic Proofs for giving me the opportunity to participate in such inspiring and enriching events.

I would like to thank my friends Annick, Bob, Claude, Corinne, Danièle, Eric, France, Jeff, Jos, Manon, Marianne, Max, Mireille, and especially Matúš for the many enjoyable moments we spent together at lunch breaks, dinners, and cinema evenings.

Most importantly, I would like to thank my parents, Christiane and Fernand, and my brother Jean-Marc for their support during my studies in Zurich and for many encouraging words.

Contents

1. Introduction

1.1. Motivation

Recent years have seen the proliferation of various electronic services such as online banking, online shopping or social networking. These electronic services rely on security protocols, which are supposed to establish a secure connection between the customer and the website offering the service. In particular, the communicating parties desire confidentiality of sensitive information transmitted over the Internet, and guarantees on the identity of their communication partner.

Authenticated key exchange (AKE) protocols are central building blocks of security protocols such as TLS and IPsec that are used to secure electronic communication. They establish a shared symmetric *session key* between two parties. Subsequent communication between the parties is then protected with this session key. Some examples of standardized AKE protocols are MQV [70], SIGMA [63], and the ISO variant of signed Diffie-Hellman [47, 63].

The security of AKE protocols relies on certain keys being kept secret by the parties, on the quality of the random number generator, and on the security of the cryptographic primitives (e. g., digital signatures or hash functions) that are used. Various forms of compromise may however impact AKE security and hence endanger secure communication over the Internet: Companies, as well as individuals, may fall victim to corruption of secret keys, and random number generators may be flawed or deliberately weakened to produce weak, predictable values. It is thus important to design AKE protocols in such a way that they offer protection even against given damaging security threats.

Over the past twenty years, the design of AKE protocols was mainly driven by newly discovered attacks on existing protocols [11, 22, 64, 69], against which the protocols were intended to offer protection, and by the need for stronger security guarantees [63, 68, 70]. These approaches to design protocols often failed to yield protocols with the intended security properties due to the lack of formalization of the desired security goals and the absence or incompleteness of security proofs [64, 75, 81]. A more recent and systematic approach for constructing protocols consists in providing generic *security-strengthening transformations* (often called *compilers*) [32, 34, 61, 66], which are applied over a class of protocols to obtain refinements that are secure against stronger adversaries. To achieve stronger security guarantees, transformations usually rely on cryptographic primitives such as encryption, digital signatures, or message authentication codes. Several new protocols may thus be incrementally constructed from a library of security-strengthening transformations and secure base protocols.

The security of prominent protocols is proven in formal *security models* [17, 18, 34, 56, 64, 68] under certain cryptographic assumptions. These models specify the adversary's capabilities and the security requirements of a protocol. Security proofs

rely on reduction proofs. It is shown that if there is an efficient adversary breaking the security of the protocol, then there is an efficient adversary solving the underlying computational problem that is assumed to be hard (e. g., the Gap Diffie-Hellman assumption).

Despite intensive study over the past two decades, existing security models [17, 18, 34, 56, 64, 68] do not allow reasoning about certain practically relevant attacks. Furthermore, even though protocols resilient against some of these attacks exist, there are no security-strengthening transformations to facilitate their incremental construction.

In the first part of this thesis we model stronger security guarantees for AKE protocols than found in the state-of-the-art by incorporating the security features of *perfect forward secrecy* and *security against malicious key registration* into current security models. Our aim is to provide security-strengthening transformations to incrementally construct AKE protocols that achieve the respective stronger security guarantees. That is, starting from a protocol that is secure in a given base model, an enhanced protocol can be constructed by applying a security-strengthening transformation on the original protocol. We now motivate the two previously mentioned security properties in more detail.

Perfect forward secrecy guarantees secrecy of session keys even if the adversary *later* compromises all long-term secret keys. That is, a protocol satisfying perfect forward secrecy guarantees secrecy of the session key even if an adversary, who is active in the corresponding target session, i. e., the session under attack, later gains access to the long-term secret keys of the involved users. For example, if the user running the target session encrypts some confidential message such as his banking details using the established session key, then the adversary is not able to decrypt the user's message and, consequently, learn confidential information.

Most protocols that provide PFS (e. g., SIG-DH [34], or SIGMA [63]) are three-message protocols and are insecure when the adversary compromises session-specific random values. There exist however some two-message protocols that provide PFS, but all of these protocols are vulnerable to the compromise of either long-term [31] or session-specific [50] secrets. Although it was stated, but not proven, that PFS can be achieved in two messages via explicit message authentication (see, e. g., [28, 50]), nevertheless, this property is considered unachievable in two-message protocols in the presence of adversaries who can compromise certain user's long-term secret keys at any point in time or session-specific random values [31, 65, 68]. This motivates our first research question.

Research Question 1: Can two-message protocols achieve PFS even under compromise of session-specific values and user's long-term secret keys?

Security against malicious key registration. AKE protocols rely on a public key infrastructure (PKI) to support the authenticity of user's public keys. That is, from a user's point of view, participation in an AKE protocol encompasses three consecutive phases: First, users set up their individual key pairs; i. e., each user invokes a randomized algorithm that outputs a fresh secret/public key pair. Second, users contact a certification authority (CA) to get their public key certified: each user

provides the CA with its identifier and its public key, and obtains a certificate that binds the identifier to the key. After completing these setup steps, in the third phase, users can engage in interactive sessions with other users to establish shared session keys.

In formal security proofs [43, 105] and in various standards [7, 103] it is mandated that the CA performs proper checks on combinations of identities and public keys (e. g., that public keys belong to an approved key space or that the user self-signs the data that is to be certified). CAs in the real world have different verification procedures for checking claimed identities[1] and arithmetic properties of the public key to be registered. They may often only perform very few checks due to efficiency and economic reasons. Malicious parties might in some cases get arbitrary public keys certified against identifiers of their choice. The most egregious examples involve CAs who, either willingly, under coercion, or as a result of security compromises, have issued rogue certificates for keys and identifiers. For example, in June and July 2011, Dutch CA DigiNotar was hacked [46], with the intruder taking control of all 8 of the CA's signing servers; at least 531 rogue certificates were then issued.

These are real concerns that lead to attacks on AKE protocols, which cannot be captured in formal security models that omit CA and PKI aspects. For example, if the CA does not check the uniqueness of the public keys, then there is an *unknown key share* (UKS) attack on the KEA protocol [69, p. 380]. The attack exploits the adversary's ability to re-register some user's public key as his own public key. It works as follows. Suppose that some user $\hat{B}$ registered the public key $\mathsf{pk}_{\hat{B}}$ with the CA. The adversary $\hat{L}$ then re-registers the public key $\mathsf{pk}_{\hat{B}}$ as his own public key. Since the CA does not check whether the public key has already been registered before, the adversary is issued a certificate that binds his identity $\hat{L}$ to the public key $\mathsf{pk}_{\hat{B}}$. User $\hat{A}$ initiates a session of the protocol with user $\hat{B}$. The adversary then activates a session of $\hat{B}$ with peer $\hat{L}$ by forwarding $\hat{A}$'s message to $\hat{B}$. He then forwards $\hat{B}$'s message to $\hat{A}$. Both users $\hat{A}$ and $\hat{B}$ end up sharing the same session key, but they have a different view on whom they share the session key with; $\hat{A}$ correctly believes that she shares the session key with $\hat{B}$, while $\hat{B}$ mistakenly believes that he shares the session key with $\hat{L}$.

The previous discussion leads us to our second research question.

Research Question 2: How can we strengthen protocols to achieve security when adversaries can register arbitrary bitstrings and the CA does not perform any checks?

Prominent security models [34, 64, 68, 108] have been developed to capture additional security properties via extension of previous models. Many recent protocols [64, 68, 105] and transformations [34, 66, 108] were proven secure with respect to these stronger models. However, few results establish the exact limits of AKE security and, consequently, the strongest security guarantees achievable by AKE protocols. Current

[1] For example, issuance of Extended Validation (EV) certificates requires stronger identity-checking requirements than non-EV certificates, see `https://www.cabforum.org/certificates.html` for more details. The verification procedure of a CA issuing EV certificates includes, among others, the verification of the applicant's identity and the verification of his signature on the EV certificate request.

impossibility results are formulated as restrictions on the adversary's behavior with regard to the target session, and suggest that it is impossible to construct a protocol secure with respect to a less restricted adversary [65, 68, 108]. We argue that rigorous impossibility results for AKE protocol design are still missing in the context of ever stronger AKE security guarantees.

In the second part of this thesis we therefore derive strong security guarantees for certain classes of AKE protocols from rigorous impossibility results on the respective class. More precisely, we first specify attacks that are applicable against any protocol in a given class, and then define a security model in which the adversary is prevented from performing these attacks. In particular, we consider the class of protocols that do not modify memory shared among sessions, i. e., they only modify session-specific memory. This class includes all modern protocols [64, 68, 105] analyzed in advanced security models [64, 68]. It is shown that extending the protocol class enables the design of protocols that are secure against stronger adversaries who control the randomness used in protocol sessions. In particular, the construction of protocols withstanding such adversaries would enable us to weaken the security assumptions on the random number generator (RNG) used to produce session-specific randomness. This is particularly timely given the recently discovered security vulnerabilities that involve either flawed [1, 76, 86, 109] or weakened [88, 95, 100, 110] RNGs. In 2008, Bello discovered a randomness vulnerability in Debian's OpenSSL package; keys generated by the RNG of this package were predictable [1]. As a consequence, protocol sessions of the DHE variant of SSL, e. g., might have been compromised as attackers were able to predict the randomness used to establish the shared session key and hence to decrypt further communication between client and server [1, 109]. Furthermore, it seems that the security of certain cryptographic systems such as RNGs has been deliberately weakened to, e. g., eavesdrop on private communication [95]. To sum up, the second part of this thesis deals with the following two research questions.

Research Question 3: What are the exact limits of AKE security with respect to adversaries with certain capabilities?

Research Question 4: How to achieve security against adversaries who have the capability of controlling session-specific randomness?

1.2. Related work

In the section we discuss the state-of-the-art relative to the research questions raised in Section 1.1.

On perfect forward secrecy

The eCK model

The extended Canetti-Krawczyk (eCK) security model by LaMacchia et al. [68] is a game-based AKE security model in which the adversary is modelled as a probabilistic polynomial-time (PPT) Turing machine and controls all communication between the users. Security is defined in terms of a game played by the adversary against

a challenger implementing the users. The adversary interacts with the challenger through a set of *queries*. These queries specify the capabilities of the adversary.

The eCK model captures several relevant attacks such as key compromise impersonation (KCI) attacks [58], where the adversary compromises the long-term secret key of a user and then tries to impersonate other users to this user, and leakage of various combinations of long-term secret keys and session-specific random values [68]. In addition, security in the eCK model implies weak perfect forward secrecy [64], a weaker guarantee than perfect forward secrecy. Weak perfect forward secrecy guarantees secrecy of session keys when long-term secret keys are later compromised, but only for sessions in which the adversary did not actively interfere.

In comparison to earlier game-based security models [17, 34, 64], the eCK model (a) considers stronger adversaries in terms of their capabilities, and (b) captures several attacks in a single security model. As we illustrate in the next section, the eCK model was believed to be the strongest possible security model [36, 68, 72] for analyzing two-message protocols.

Negative results

Research question 1 has not been answered affirmatively in the literature. The majority of related works claim that perfect forward secrecy cannot be achieved in two-message protocols [30, 38, 65, 67, 68].

In [65, p. 15], Krawczyk described an attack that shows that the MQV protocol does not achieve perfect forward secrecy. This attack led him to the conclusion that "..., no 2-message protocol, and in particular HMQV, can provide full perfect forward secrecy." [65, p. 56]. To prove a slightly weaker notion of forward secrecy for the HMQV protocol, Krawczyk introduced the notion of *weak perfect forward secrecy* (weak-PFS) [65]. Weak perfect forward secrecy guarantees secrecy of session keys even when long-term secret keys are later compromised, but only for sessions in which the adversary did not actively interfere.

Based on Krawczyk's statements on PFS, the designers of the eCK model claimed that this property cannot be achieved by two-message AKE protocols: "As noted by Krawczyk [64], the PFS requirement is not relevant for 2-round AKE protocols since no 2-round protocol can achieve PFS" [68, p. 5]. Thus, Krawczyk's comments seem to have led to the (incorrect) belief that the best that can be achieved for two-message protocols is weak perfect forward secrecy (see, e. g., [68, pages 2 and 5], [38, p. 213]). As a result, even though the eCK security model [68] guarantees only weak perfect forward secrecy, it was described in the literature as the strongest possible security model for two-message protocols [36, 68, 72].

In [67], LaMacchia et al. define a variant of the eCK model for protocols with more than two messages that additionally guarantees perfect forward secrecy. However, this eCK variant cannot be met by any protocol from the class $\mathcal{DH}$-2 of two-message DH-type protocols that we consider in Chapter 3, because it uses the concept of matching session instead of origin-session. Boyd and González Nieto's replay attack [31, p. 458] serves as a generic counterexample: it shows that no two-message protocol in $\mathcal{DH}$-2 that does not provide message replay detection can achieve security in this eCK variant, assuming that the notion of matching sessions is defined as in Definition 42.

There exist however some two-message protocols that provide PFS, but are vulnerable to the compromise of either the long-term secret keys of the actor of the target session [31] or session-specific random values [50]. We now discuss some of these protocols in more detail.

The two-message modified-Okamoto-Tanaka (mOT) protocol by Gennaro et al. [50] provides perfect forward secrecy in the identity-based setting. Additionally, they sketch variants of the protocol for the PKI-based setting. As noted by the authors [50], the mOT protocol and its variants are not resilient against the compromise of session-specific random values, and they are therefore insecure in eCK-like models.

Jeong, Katz and Lee [56] introduce the one-round protocols TS2 and TS3 and show that these protocols achieve forward secrecy. The underlying security model with respect to which both protocols are proven secure is based on the Bellare-Rogaway model in [17] and captures forward secrecy by allowing the adversary to corrupt both actor and peer of the target session in case the adversary is passive during the execution of the target session (which corresponds to weak-PFS). Whereas protocol TS2 only achieves weak-PFS, it seems that protocol TS3 achieves PFS [10], although this is not stated or proven in [56]. However, protocol TS3 provides neither PFS under actor compromise nor security against the compromise of session-specific random values.

In parallel and independent work to ours, Boyd and González Nieto [31] suggest a transformation $\mathcal{C}$ based on adding MACs on the message exchange of an AKE protocol that satisfies weak perfect forward secrecy, to achieve perfect forward secrecy. However, the MAC transformation does not guarantee perfect forward secrecy under actor compromise and leakage of session-specific randomness, as shown in Section 3.2.4. It is important to note that our security model eCK-PFS, which we present in Chapter 3, prevents the attack scenario described in [31, p. 458] since we restrict the adversary from revealing the randomness of the *origin-session* for the test session (see Definition 16).

Generic transformations to achieve PFS

Some generic protocol transformations that aim towards achieving security in a stronger model capturing perfect forward secrecy have been proposed [31, 32, 34, 61].

Canetti and Krawczyk [34] specify transformations from the authenticated-links model (AM) to the unauthenticated-links adversarial model (UM). In the AM, the adversary is not allowed to actively interfere within the communication. In the UM, the adversary has basic attacker capabilities as well as the capability of corrupting agents, revealing session-keys and session-state. A protocol secure in the AM can be transformed into a protocol secure in the stronger model UM using an *authenticator*. Applying the signature-based authenticator from [13] to each of the flows of a two-message Diffie-Hellman protocol results in a three-message protocol that is secure in the UM [34, pp. 466-467]. Thus, in contrast to our SIG transformation from Chapter 3, the application of this signature-based authenticator increases the number of rounds of the protocol due to the use of the Diffie-Hellman exponentials as the challenges required by the authenticator.

In the context of authenticated group key exchange, Katz and Yung [61] propose a generic transformation which transforms any group KE protocol secure against a

passive adversary to an authenticated group KE protocol secure against an active adversary who fully controls the network. The transformed protocol has an additional protocol round and requires signatures on some broadcasted messages. Their security model captures perfect forward secrecy, but it does not capture KCI attacks.

Bresson et al. [32] propose a similar signature-based transformation than Katz and Yung [61]. Their transformation yields a protocol achieving authenticated key exchange security as well as mutual authentication when applied to any group key exchange protocol secure against a passive adversary. The passive adversary in [32] is slightly stronger than the one in [61] since in addition to eavesdropping on regular protocol executions, he may delay or delete messages, or change their delivery order. The security model $\mathsf{eCK}^{\mathsf{passive}}$ that we introduce in Section 3.2 considers an even stronger passive adversary that can also replay messages to sessions and is only restricted from injecting or modifying the messages received by the target session.

Even though the previous approaches allow to achieve security in a stronger security model capturing PFS, none of these stronger models captures key compromise impersonation attacks as well as leakage of session-specific randomness, and consequently perfect forward secrecy under these attack scenarios. In addition, all of the aforementioned transformations increase the round complexity of the protocol, where a protocol round consists of all the messages that can be transmitted simultaneously between two entities. In contrast, our SIG transformation transforms any one-round protocol secure in an eCK-like model into a one-round protocol that provides perfect forward secrecy under actor compromise and randomness reveal.

AKE security and certification systems

The original computational model for key exchange of Bellare and Rogaway [17] has a long-lived key generator, which is used to initialise all users' keys at the start of the game. This is a standard part of most computational models today. However, in common with several later models [34,56,66], the adversary cannot influence long-term keys: only honestly generated (and registered) keys are considered. Starting with the 1995 model of Bellare and Rogaway [18] it was recognized that the adversary may be able to choose long-term keys for certain users, whether public keys or symmetric keys. It is possible to identify four different methods that have been used to model such an adversary capability.

1. The adversary may register arbitrary valid public keys from the key space during the setup phase of the security experiment [68, p. 8].
2. The adversary can replace long-term keys by providing them as an input to a corrupt query. This was the method used originally by Bellare and Rogaway [18] and was subsequently used in the public key setting by others [21, 83].
3. The adversary is allowed to generate arbitrary keys for corrupted users at any time during the protocol run [64].
4. An additional query is added specifically to set up a user with a new key chosen by the adversary [35, 51, 105]. This query is typically called establishparty and takes as input the user name and its long-term public key.

All of these methods, except the first one, allow the models to capture Kaliski's UKS attack against the MQV protocol [59], which requires the adversary to register a new

public key after certain protocol messages have been obtained. Kaliski's attack works as follows. The adversary $\hat{L}$ intercepts the message sent by some user $\hat{A}$ who intends to execute a protocol session with user $\hat{B}$. He then registers a specific valid key, which depends on the intercepted message, as his own public key with the CA. Now, the adversary activates a session of $\hat{B}$ with peer $\hat{L}$ by sending a previously constructed message, which depends on $\hat{A}$'s message and on $\hat{A}$'s long-term public key, to $\hat{B}$, and then forwards $\hat{B}$'s message to $\hat{A}$. Both users $\hat{A}$ and $\hat{B}$ end up sharing the same session key, but they have a different view on whom the share the session key with. Clearly, the previous attack cannot be captured in a model that only allows the adversary to register keys with the CA in the setup phase of the security experiment, i. e., before executions of the protocol between users.

However, none of the previously mentioned methods has the generality of our ASICS framework in Chapter 4 and, in particular, all of them omit the following realistic features:

- registration of multiple public keys per user;
- flexible checking by certification authorities via a verification procedure;
- adversarial choice of public keys per session.

Special mention should also be made of the model of Shoup [99]. Unlike most popular AKE models today, it uses a simulatability definition of security comparing ideal and real world views. Security is defined to mean that for any real world adversary there is an ideal world adversary (benign by definition) such that the transcripts of the two are computationally indistinguishable. Real-world adversaries have the ability to assign users to public key certificates. Shoup's model has not been widely used and the examples in [99] are not fully worked through. Furthermore, the model cannot represent an adversary who obtains only ephemeral secret keys without knowing the long-term key of the same user and therefore cannot capture security properties common in more modern models.

Critically, all of the approaches mentioned above have only been used to establish results for a handful of specific protocols. The uncovering of new attacks on some of these protocols exploiting malicious key registration led to the recommendation of countermeasures to these attacks such as verifying that long-term public keys belong to a given key space [81]. However, the correctness of these recommendations has not been proven for a class of protocols. We establish in Chapter 4 generic results that facilitate the design and verification of AKE protocols under malicious key registration, and that can be applied over a class of protocols.

Impossibility results and randomness failures in the context of AKE

Impossibility results. Current impossibility results on the security of AKE protocols resulted in restrictions on the adversary's behaviour with respect to the target session, i. e., the session under attack. However, some of these results are either flawed [68] or only hold under unstated assumptions on the protocol class [108]. For example, Krawczyk's [64] perfect forward secrecy attack, has led to the statements that (a) no two-message protocol can provide PFS [65, p. 56], and (b) the eCK model capturing only weak perfect forward secrecy is the strongest possible model for analyzing two-

message AKE protocols [68, 72]. We show in Chapter 3 that two-message protocols can achieve PFS even in the presence of eCK-like adversaries.

Yang et al. state that no protocol can be secure against *reset-and-replay attacks* on the target session [108, p. 120]. In a reset-and-replay attack the adversary first sets the randomness of a session to the same randomness as used in a previous session of the same user and then replays messages to the session so that both sessions compute the same session key [108]. Based on their impossibility result, they design a security model in which reset attacks are disallowed on the target session. As we see in Chapter 5, their result only holds for a particular class of protocols, namely the class of stateless protocols, which do not modify memory that is shared among sessions. Since we consider a wider range of protocols, we arrive at the conclusion that there are protocols that are secure against reset-and-replay attacks on the target session. In particular, we construct variants of the NAXOS protocol [68] that are secure against such attacks.

Boyd and González Nieto [31] show that one-round AKE protocols that do not provide message replay detection cannot achieve PFS if the adversary can also reveal session-specific randomness of the target session's peer. Their argument is based on a variant of Krawczyk's PFS attack [64], which involves a replay attack together with leakage of session-specific randomness of the session the replayed message originates from. As stated by the authors, protocols that provide replay detection via timestamps or counters are not vulnerable to this attack. However, as we show in Chapter 5, there are one-round protocols that do not provide replay detection and are only vulnerable to Boyd and González Nieto's attack [31] if the target session is activated with a message replayed from the first session of its peer. In case the target session is activated with a message replayed from the n'th session of its peer, where $n > 1$, the adversary would need to additionally reveal the randomness of all the previous sessions of the peer to be able to compute the same session key as the target session. The protocol NXPR presented in Chapter 5 achieves security even under compromise of the randomness of the session from which the message received by the target session originated and the peer's long-term secret key as long as the randomness of at least one of the previous sessions of the peer has not been compromised. Similarly, the NXPR protocol is secure under compromise of the target session's randomness and the long-term secret key of the actor of the target session as long as the randomness of at least one of the previous sessions of that user has not been compromised.

Randomness failures. The first models addressing the leakage of session-specific information include the eCK model [68] and the CK model [34]. The eCK model considers an information-leaking RNG that leaks values after they have been generated, which is modelled via the query ephemeralkey. Intermediate protocol computations are assumed to be outside of the adversary's control. In contrast, the CK model considers long-term keys stored in secure memory (e. g., in a hardware security module (HSM)), whereas protocol computations are (partly) done in less-protected memory. The adversary has read-only access to the less-protected memory through a query session-state. However, in the vast majority of proofs in the CK model, the less-protected memory has been defined to contain exactly the randomness, thereby effectively modeling an information-leaking RNG. Unlike our work, the models CK

and eCK consider neither chosen randomness nor repeated randomness. We design in Chapter 5 eCK-like models that additionally incorporate the adversarial ability of choosing session-specific randomness; security in these models implies security against attacks exploiting repeated randomness.

Ristenpart and Yilek [92] show that virtual machine (VM) snapshots might lead to VM reset attacks as these snapshots can be used to reset a system to a prior state. In the context of key exchange, VM resets lead to the use of the same randomness in more than one session. As a countermeasure to randomness failures, Ristenpart and Yilek propose a framework for *hedging* cryptographic operations based on preprocessing potentially bad RNG-supplied randomness together with additional inputs with HMAC to provide pseudorandomness for the cryptographic operation [92]; their framework uses hedging techniques for public-key encryption of Bellare et al. [12]. Hedging a cryptographic operation means designing it in such a way that, given good randomness, the operation provably achieves strong security goals, and, given bad randomness, the operation achieves weaker, but still meaningful, security goals [12]. In particular, Ristenpart and Yilek discuss hedging for key exchange and apply their approach to the signed Diffie-Hellman protocol. The main difference between the resulting hedged protocol and the CNX protocol, which we present in Chapter 5, is that the CNX protocol relies on a global counter, which is shared across different sessions of a user; this global counter is included, together with the user's long-term secret key and session-specific randomness, as input to the hash function H_1.

Yang et al. [108] analyze AKE security with respect to adversaries who can manipulate the randomness that is used in protocol sessions. They define two security models: Reset-1 and Reset-2. In the Reset-1 model the adversary controls the randomness of each session, with the restrictions that he is not allowed to (a) issue the query $\mathsf{corrupt}$ to either the actor or the peer of the target session and (b) reset the randomness of the target session or its partner session to the same randomness as used in another session of the corresponding user. Thus, the Reset-1 model captures neither weak perfect forward secrecy nor reset-and-replay attacks on the target session or its partner session. In contrast, the models that we develop in Chapter 5 capture reset attacks on the target session, reset attacks on its partner session, and weak perfect forward secrecy. The Reset-2 model captures repeated secret randomness in multiple sessions due to reset attacks, but no chosen-randomness attacks. Whereas the Reset-2 model captures weak perfect forward secrecy, it does not allow the adversary to perform reset attacks against either the target session or its partner session. In Chapter 5 we design a similar model to the Reset-2 model, which we call X_{AKE}. In the X_{AKE} model, the adversary is allowed to perform reset attacks even against the target session and its partner session. We show that security in a model that captures chosen-randomness attacks implies security in the X_{AKE} model. Yang et al. [108] provide a transformation that turns a protocol secure in the Reset-2 model into a protocol that is secure in both models, by replacing the randomness x used in the original protocol by $F_K(x)$, where F is a pseudorandom function family, and K is an additional long-term secret key.

1.3. Results

In this section we describe our approach to answering the questions raised in Section 1.1.

Perfect forward secrecy under actor compromise and randomness reveal

We provide an affirmative answer to Research Question 1. Namely, we show that it is possible to achieve perfect forward secrecy in two-message protocols even in the presence of very strong active adversaries who can reveal random values of sessions and compromise long-term secret keys of users.

We start by generalizing the eCK security model [68]. The resulting model, which we call eCK^w, specifies a slightly stronger variant of weak perfect forward secrecy than the eCK model (the w in eCK^w refers to the $w$eak PFS element). We then integrate perfect forward secrecy into the eCK^w model, which gives rise to the eCK-PFS model. The eCK-PFS model is stronger than eCK^w, and also provides more guarantees than independently considering eCK/eCK^w security and PFS. In particular, security in eCK-PFS implies perfect forward secrecy in the presence of an active attacker who can even learn the actor's long-term secret key before the start of the attacked session, or who can learn session-specific randomness.

We propose a generic security-strengthening transformation to achieve PFS in two-message protocols. Given a two-message Diffie-Hellman (DH) type protocol that is secure in eCK^w or in a weaker model, which we call $\mathsf{eCK}^{\mathsf{passive}}$, our transformation yields a two-message protocol that is secure in the eCK-PFS model. As our transformation is not based on a challenge-response mechanism and does not introduce additional protocol rounds, the result of applying our transformation to a one-round protocol, in which all outgoing messages can be computed before any message is received, is a one-round protocol.

We apply our transformation to two concrete protocols. First, we show that NAXOS [68], the first key exchange protocol proven secure in the eCK model, is secure in eCK^w and use our transformation to construct a protocol that is secure in eCK-PFS. Second, we show how a particular protocol that is insecure in eCK^w can be turned into a protocol that is secure in eCK-PFS by proving it secure in the weaker $\mathsf{eCK}^{\mathsf{passive}}$ model. Both examples illustrate how our transformation enables the modular design of key exchange protocols.

ASICS: AKE security incorporating certification systems

To answer Research Question 2, we first present a framework for reasoning about the security of AKE protocols with respect to various CA key registration procedures. Our framework, which we call ASICS, allows us to capture, i. e., to model, several attacks based on adversarial key registration, including UKS attacks, small-subgroup attacks, attacks that occur when the CA does not check if public keys are registered twice, and attacks that occur when multiple public keys can be registered per identifier. In particular, we capture a previously unreported attack against the KEA+ protocol based on an impersonation attack during key registration in an appropriate ASICS model.

We then provide a generic approach to achieve strong security guarantees against adversaries who can register arbitrary public keys for certain types of protocols. In particular, we show how to transform Diffie–Hellman type AKE protocols that are secure in a model where only honest key registration is allowed into protocols that are secure even when adversaries can register arbitrary valid or invalid public keys, and the CA does not perform any checks on combinations of identities and public keys. In such cases, security is still guaranteed for all sessions (that were considered *fresh* in the base model) except those in which the peer's public key is valid but registered by the adversary.

Finally, we demonstrate how our methodology can be used to establish strong security guarantees, even when the adversary can register arbitrary public keys, for concrete protocols such as CMQV [105], NAXOS [68], and UP [106], using a variant of CMQV as a running example.

On the limits of AKE security with an application to bad randomness

We address Research Question 3 by fixing a set of adversarial capabilities, and by distinguishing between various classes of AKE protocols. We establish impossibility results on the security of each of these protocol classes. That is, given an arbitrary protocol from a specific class, our results indicate attacks that are applicable to this protocol. We derive strong security models from our impossibility results and indicate protocols that are secure in these models. Each security model is associated to a protocol class, and reflects strong guarantees that can be achieved by protocols in the corresponding class.

Using the framework built up to formulate impossibility results, we analyze the security of AKE protocols in the presence of adversaries who can choose the randomness used in protocol sessions. While stateless protocols, which leave a user's memory invariant under protocol execution, fail to provide security against attacks based on chosen randomness, we construct stateful variants of the NAXOS protocol that achieve security against such attacks. Our security-strengthening methods, which transform a stateless protocol into a stateful protocol that is secure even when adversaries can control session-specific randomness, provide answers to Research Question 4.

1.4. Contributions

We summarize the four main contributions of our work as follows.

1. Unlike earlier works, we integrate perfect forward secrecy into a security model that captures key compromise impersonation attacks and leakage of session-specific values. We show that it is possible for two-message protocols to achieve security in the resulting strong model, which we call eCK-PFS, by first providing a security-strengthening transformation to achieve PFS, and then applying our transformation to concrete protocols. Our transformation introduces no new message dependencies and does not increase the round complexity of the protocol it is applied on. Our generic approach allows us to construct new efficient protocols secure in the new model eCK-PFS. Our results refute the claims made

in the literature that (a) no two-message protocol can provide PFS [65, p. 56], and (b) the eCK model capturing only weak perfect forward secrecy is the strongest possible model for analyzing two-message protocols [68, 72].

2. We launch the first systematic analysis of *AKE security incorporating certification systems* (ASICS). We develop a framework for reasoning about the security of protocols taking into account the certification authority and its behavior. Our framework enables us to capture several attacks from the literature as well as a newly discovered attack on the KEA+ protocol. We are the first to propose transformations that strengthen protocols to achieve security when adversaries can register arbitrary bitstrings as keys and the certification authority does not perform proper checks. Our results not only provide formal foundations for the importance of public-key validation but also lead us to recommendations for the design of protocols that are secure in our framework.
3. We perform the first systematic analysis of the limits of game-based AKE security. We identify several relevant protocol classes for which we provide formal impossibility results. Our impossibility results (a) clarify which security guarantees cannot be achieved by any protocol in the respective class, and (b) allow us to systematically develop strong security models for each class. Our exploration of the limits of game-based AKE security leads to a protocol hierarchy, which classifies certain protocols from each class according to their relative strength in our derived security models. Our hierarchy highlights the security guarantees that can be achieved by each protocol class, and provides a novel way of selecting an optimal trade-off between the type of protocol and the security it offers.
4. We consider the security of protocols in the presence of adversaries who can perform chosen-randomness attacks even against the target session, going far beyond attacks covered in previous security models. We demonstrate that security in our models capturing chosen randomness implies security under repeated randomness. We are the first to show that stateful protocols, which modify memory that is shared among sessions, can achieve security against attacks based on chosen randomness. Our novel stateful protocols allow us to weaken the assumptions made on the security of the RNG used to generate session-specific randomness.

1.5. Outline and publications

The remainder of this thesis is structured as follows. In Chapter 2 we provide background information on cryptographic primitives and computational hardness assumptions, which are used to prove the security of our protocols and transformations.

In Part 1, entitled Stronger Security in Extended Models, we provide stronger guarantees in extended security models. Part 1 consists of Chapter 3 and Chapter 4. In Chapter 3 we show how to generically achieve perfect forward secrecy under actor compromise and randomness reveal in two-message AKE protocols. In Chapter 4 we describe a framework, which we call ASICS, for the analysis of AKE protocols specifically designed to incorporate certification systems. We provide generic results

to achieve strong ASICS security even if the certification authority does not perform any checks and apply our approach to a variant of the CMQV protocol.

In Part 2, entitled Stronger Security via Impossibility Results, we explore the limits of game-based AKE security models. Part 2 consists of Chapter 5. In this chapter we derive strong security models from impossibility results on specific protocol classes. In addition, we show that security against bad session-specific randomness can be achieved when considering protocols from a broader protocol class.

In Chapter 6 we present additional related work that is relevant for the thesis. Finally, in Chapter 7, we present conclusions and future work.

Publications

The work on perfect forward secrecy under actor compromise and randomness reveal in Chapter 3 is joint work with Cas Cremers and has been published in:

- C. Cremers and M. Feltz. One-round Strongly Secure Key Exchange with Perfect Forward Secrecy and Deniability. Cryptology ePrint Archive, Report 2011/300, 2011, `http://eprint.iacr.org/`.
- C. Cremers and M. Feltz. Beyond eCK: Perfect Forward Secrecy under Actor Compromise and Ephemeral-Key Reveal. In *Proceedings of the 17th European Conference on Research in Computer Security, ESORICS 2012*, pages 734-751. Springer-Verlag, 2012. Full version available at `http://eprint.iacr.org/2012/416`.
- C. Cremers and M. Feltz. Beyond eCK: Perfect Forward Secrecy under Actor Compromise and Ephemeral-Key Reveal. In *Designs, Codes and Cryptography*, 2013.

The work on AKE security incorporating certification systems (ASICS) in Chapter 4 is joint work with Colin Boyd, Cas Cremers, Kenneth G. Paterson, Bertram Poettering, and Douglas Stebila, and has been published in:

- C. Boyd, C. Cremers, M. Feltz, K. G. Paterson, B. Poettering and D. Stebila. ASICS: Authenticated Key Exchange Security Incorporating Certification Systems. In *Proceedings of the 18th European Conference on Research in Computer Security, ESORICS 2013*, pages 381-399. Springer-Verlag, 2013. Full Version available at `http://eprint.iacr.org/2013/398`.

The work on the limits of AKE security with an application to bad randomness in Chapter 5 is joint work with Cas Cremers.

2. Background

In this chapter we recall some definitions and cryptographic assumptions that we need in order to prove the security of our protocols and transformations. We denote by $x \in_R \mathrm{S}$ the element x being chosen uniformly at random from the set S. We denote by $\|a\|$ the bit length of the positive integer a (i. e., $\|a\| = \lceil \log_2(a+1) \rceil$, where $\lceil . \rceil$ is the ceiling function). We say that a function f is *negligible* in the parameter k if, for all integers $c > 0$, there exists a real number $k_c > 0$ such that, for all $k > k_c$, it holds that $f(k) < k^{-c}$ (see, e. g., [60]).

2.1. Groups

In this section we provide some background material on groups, which can be found in any standard abstract algebra book, such as Artin's *Algebra* [5].

Definition 1 (Group)**.** *A* group *is a set G together with a law of composition $\cdot : G \times G \to G$ such that*

- *the law of composition is associative, i. e., $x \cdot (y \cdot z) = (x \cdot y) \cdot z$, for all $x, y, z \in G$,*
- *there exists an element $e \in G$, called the identity element, such that $e \cdot x = x \cdot e$, for all $x \in G$, and*
- *every element in G is invertible, i. e., for every $x \in G$, there exists $y \in G$ such that $x \cdot y = y \cdot x = e$.*

For ease of notation, we write xy instead of $x \cdot y$. The identity element will be denoted by 1 if the law of composition is multiplication, and 0 if it is addition. The *order* of a group G is the number of elements in G. The order may be finite or infinite.

Here are some examples of groups.

Example 1. The set $\mathbb{Z}$ of integers with addition $+$ as the law of composition is a group. The identity element is 0 and the inverse of $x \in \mathbb{Z}$ is $-x \in \mathbb{Z}$.

Example 2. The set $\mathbb{Z}_N^* = \{1 \leq a \leq N-1 \mid \gcd(a, N) = 1\}$ with multiplication modulo N is a group with identity element 1. Recall that $\gcd(a, N) = 1$ if and only if a is invertible modulo N.

Definition 2 (Subgroup)**.** *A subset H of a group G is said to be a* subgroup *of G if the following properties hold:*

- Closure: *If $x, y \in H$, then $xy \in H$,*
- Inverse: *If $x \in H$, then $x^{-1} \in H$, and*
- Identity: $1 \in H$.

Definition 3 (Order of a group element)**.** *Let G be a finite group. Let $x \in G$. The* order of the element *x is the smallest positive integer i such that $x^i = 1$.*

Let G be a finite group of order n. Let x be an arbitrary element of G of order $i \leq n$. The element x generates the set $\langle x \rangle := \{1, x, ..., x^{i-1}\}$. It is not difficult to verify that $\langle x \rangle$ is a subgroup of G. We call $\langle x \rangle$ the cyclic subgroup generated by x. If there is an element $g \in G$ that generates the entire group G, then the group is called cyclic and g is called a generator of G.

Proposition 1. *If G is a group of prime order, then G is cyclic. Furthermore, all elements of $G \setminus \{1\}$ are generators of G.*

In Chapter 4 we consider verification procedures on public keys and identifiers that a certification authority (CA) deploys. Given an arbitrary group of prime order, we establish generic results that facilitate the design of authenticated key exchange protocols under malicious key registration. In the examples below we indicate, for two specific types of groups that are used in cryptographic applications, some arithmetic checks on a candidate public key that a CA might perform (see also [9]).

Example 3. Let $G = \langle g \rangle$ denote the subgroup of prime order q of $\mathbb{Z}_p^*$, where p is a large prime, q divides $p - 1$ with multiplicity 1. Let y be a candidate public key. Public key validation may verify that $y^q = 1 \pmod{p}$.

Example 4. Let p be a large prime. Let $a, b \in \mathbb{Z}_p$ be constants with $4a^3 + 27b^2 \neq 0 \pmod{p}$. Let G be an additive prime-order group $E(\mathbb{Z}_p)$ of points over the elliptic curve E given by the equation

$$y^2 = x^3 + ax + b \pmod{p} \ . \tag{2.1}$$

The group G consists of all the points $(x, y) \in \mathbb{Z}_p \times \mathbb{Z}_p$, which satisfy Equation (2.1), together with the *point at infinity*. Let $Q = (x_Q, y_Q)$ be a candidate public key. Public key validation may verify that $x_Q \in \mathbb{Z}_p$, $y_Q \in \mathbb{Z}_p$, and $y_Q^2 = x_Q^3 + ax_Q + b \pmod{p}$.

2.2. Computational hardness assumptions

The Discrete Logarithm problem was first described in the seminal paper of Diffie and Hellman [45]. In a finite cyclic group G of order p with generator g the problem is defined as follows. Given $g, g^u \in G$, where $u \in \mathbb{Z}_p$, compute $u = \mathrm{DLog}_g(g^u)$.

Definition 4 (Discrete Logarithm Assumption)**.** *Let k be a security parameter. Let G be a finite cyclic group of order p (with $\|p\| = k$) and let g be a generator of G. The* Discrete Logarithm assumption *in G states that, given g^u, for $u \in_R \mathbb{Z}_p$, it is computationally infeasible to compute u. More precisely, we say that the Discrete Logarithm assumption holds in G, if for all probabilistic polynomial-time algorithms $\mathcal{A}$, there exists a negligible function negl such that*

$$P(\mathcal{A}(G, p, g, g^u) = \mathrm{DLog}_g(g^u) \mid u \in_R \mathbb{Z}_p) \leq negl(k) \ .$$

Define $\mathrm{CDH}(g^u, g^v) = g^{uv}$. The Computational Diffie-Hellman problem [45] in a cyclic group G of order p with generator g is defined as follows. Given $g, g^u, g^v \in G$, where $u, v \in \mathbb{Z}_p$, compute $\mathrm{CDH}(g^u, g^v)$.

Definition 5 (Computational Diffie-Hellman Assumption)**.** *Let k be a security parameter. Let G be a finite cyclic group of order p (with $\|p\| = k$) and let g be a generator of G. The* Computational Diffie-Hellman (CDH) assumption *in G states that, given g^u and g^v, for $u, v \in_R \mathbb{Z}_p$, it is computationally infeasible to compute* $\mathrm{CDH}(g^u, g^v)$*. More precisely, we say that the CDH assumption holds in G, if for all probabilistic polynomial-time algorithms $\mathcal{A}$, there exists a negligible function $negl$ such that*

$$P(\mathcal{A}(G, p, g, g^u, g^v) = \mathrm{CDH}(g^u, g^v) \mid u, v \in_R \mathbb{Z}_p) \leq negl(k) \ .$$

Let p be a large prime. Groups in which the Discrete Logarithm problem and the CDH problem are believed to be hard and that are used in cryptographic applications include the multiplicative group $\mathbb{Z}_p^*$ with $p-1$ not containing any small prime factors [25] and large prime-order subgroups of the group $E(\mathbb{Z}_p)$ of points over an elliptic curve [29, 101].

The Decisional Diffie-Hellman (DDH) problem [45] in a finite cyclic group G of order p with generator g is defined as follows. Given $g, g^u, g^v, g^w \in G$, where $u, v, w \in \mathbb{Z}_p$, decide whether or not $g^w = \mathrm{CDH}(g^u, g^v)$.

Whereas the CDH problem is believed to be hard in the group $\mathbb{Z}_p^*$, where p is a large prime, the DDH problem is not hard in this group (see, e. g., [25]). A variant of the CDH problem is the Gap Diffie-Hellman problem [87], which requires solving the CDH problem given access to an oracle that solves the DDH problem.

Definition 6 (Gap Diffie-Hellman Assumption)**.** *Let k be a security parameter. Let G be a finite cyclic group of order p (with $\|p\| = k$) and let g be a generator of G. The* Gap Diffie-Hellman assumption *in G states that, given g^u and g^v, for u, v chosen uniformly at random from $\mathbb{Z}_p$, it is computationally infeasible to compute g^{uv} with the help of a DDH oracle $\mathcal{O}^{\mathrm{DDH}}$, which, for any three elements $g^u, g^v, g^w \in G$, replies whether or not $g^w = \mathrm{CDH}(g^u, g^v)$. More precisely, we say that the Gap Diffie-Hellman assumption holds in G, if for all probabilistic polynomial-time algorithms $\mathcal{A}$, there exists a negligible function $negl$ such that*

$$P(\mathcal{A}(G, p, g, g^u, g^v, \mathcal{O}^{\mathrm{DDH}}) = \mathrm{CDH}(g^u, g^v) \mid u, v \in_R \mathbb{Z}_p) \leq negl(k) \ .$$

As shown in [26, 57], groups over certain supersingular elliptic curves are groups in which the CDH problem is believed to be hard, but the DDH problem is easy; the Weil pairing [85] can be used to construct a DDH oracle (see, e. g., [26]).

A variant of the Discrete Logarithm problem is the Gap Discrete Logarithm problem [77, 87], which requires solving the Discrete Logarithm problem given access to an oracle that solves the DDH problem.

Definition 7 (Gap Discrete Logarithm Assumption)**.** *Let k be a security parameter. Let G be a finite cyclic group of order p (with $\|p\| = k$) and let g be a generator of G. The* Gap Discrete Logarithm assumption *in G states that, given g^u, for u chosen uniformly at random from $\mathbb{Z}_p$, it is computationally infeasible to compute u with the help of a DDH oracle $\mathcal{O}^{\mathrm{DDH}}$, which, for any three elements $g^u, g^v, g^w \in G$, replies whether or not $g^w = \mathrm{CDH}(g^u, g^v)$. More precisely, we say that the Gap Discrete Logarithm assumption holds relative to $\mathcal{G}$, if for all probabilistic polynomial-time algorithms $\mathcal{A}$, there exists a negligible function $negl$ such that*

$$P(\mathcal{A}(G, p, g, g^u, \mathcal{O}^{\mathrm{DDH}}) = \mathrm{DLog}_g(g^u) \mid u \in_R \mathbb{Z}_p) \leq negl(k) \ .$$

2.3. Digital signature schemes

Digital signatures schemes [54] are usually employed in protocol design to ensure *message origin authentication*, i. e., that a recipient is able to verify that each message he receives originated from a certain entity.

Definition 8 (Signature Scheme [60])**.** *A* signature scheme Σ *is a tuple of three polynomial-time algorithms* $(Gen, Sign, Vrfy)$ *satisfying the following:*

1. *The probabilistic key-generation algorithm Gen takes as input a security parameter* k *and outputs a secret/public key pair* $(\mathsf{sk}, \mathsf{pk})$.
2. *The (possibly probabilistic) signing algorithm Sign takes as input a secret key* sk *and a message* $m \in \{0,1\}^*$. *It outputs a signature* σ.
3. *The deterministic verification algorithm Vrfy takes as input a public key* pk, *a message* m, *and a signature* σ. *It outputs* 1 *if* σ *is a valid signature on message* m *with respect to public key* pk, *and* 0 *otherwise.*

It is required that for every k, *every* $(\mathsf{sk}, \mathsf{pk})$ *output by* $Gen(1^k)$, *and every* $m \in \{0,1\}^*$, *it holds that*

$$Vrfy_{\mathsf{pk}}(m, Sign_{\mathsf{sk}}(m)) = 1 \ .$$

Most signature schemes are designed to achieve *existential unforgeability under an adaptive chosen-message attack (EU-CMA)* [54]. Informally, a signature scheme is EU-CMA if an adversary who obtains valid signatures on messages of his choice should not be able to produce a signature on a new message. To prove the security of our signature transformation from Chapter 3, we need a stronger assumption than EU-CMA, namely *strong existential unforgeability under an adaptive chosen-message attack (SUF-CMA)* [3, 27]. An adversary should not only be unable of forging a signature for a new message, but also unable of producing a new signature on a previously signed message.

Definition 9 (SUF-CMA [27])**.** *Let* $\mathcal{O}^{Sign}$ *be a signature oracle that returns a signature for any message of the adversary's choice. Let* $Adv_A^{Sig}(k)$ *denote the probability of successfully forging a valid signature* σ *on a message* m, *where* (m, σ) *is not among the pairs* (m_i, σ_i) $(i = 1, ..., q)$ *generated during the query phase to the oracle* $\mathcal{O}^{Sign}$. *A signature scheme* $\Sigma = (Gen, Sign, Vrfy)$ *is said to be* strongly existentially unforgeable under an adaptive chosen-message attack *if for all probabilistic polynomial-time adversaries A, there exists a negligible function negl such that* $Adv_A^{Sig}(k) \leq negl(k)$.

Examples of strongly existentially unforgeable signature schemes under an adaptive chosen-message attack are Boneh et al.'s scheme based on the CDH problem [27], Gennaro et al.'s scheme based on a variant of the RSA assumption [27, 49], and Boneh et al.'s GDH scheme based on the Gap Diffie-Hellman problem [26, 27].

2.4. The random oracle model

The random oracle model [16] assumes the existence of a public function H that behaves as a random function. The function H can be evaluated on some input $x \in \{0,1\}^*$ only by querying an oracle on x; the oracle returns $H(x) \in \{0,1\}^*$ given

query x. The oracle is public in the sense that users as well as the adversary can submit a query x and receive $H(x)$ from the oracle [16, 60]. These queries are however private so that if a user queries the oracle on some input x, then nobody else, and in particular not the adversary, learns x [60].

The oracle can be seen as implementing a random function H as follows. Suppose $H : \{0,1\}^n \to \{0,1\}^m$. When queried with $x \in \{0,1\}^n$, the oracle first checks whether x has been queried before. If the oracle has not received query x before, then it chooses $y \in_R \{0,1\}^m$, returns y, and stores the entry (x, y) in a table T (initially empty). Else, the oracle retrieves the entry (x, y) from table T and returns y [60].

Secure protocols can be designed using the following methodology [16, 60]:

1. First, the protocol is proven secure in the random oracle model.
2. Second, the random oracle is instantiated with a particular cryptographic hash function. Thus, instead of querying the oracle, each user evaluates the hash function on his own.

If the hash function used in the second step is "good" at emulating a random oracle, then the protocol should be secure in the real world as well. It is however unclear how to define the concept of a hash function being "good" at emulating a random oracle, and whether such functions can be constructed [60]. Even though there has been some debate about the random oracle model [52], it remains to be a very useful tool in provable security [62], which allows researchers to make formal statements about the security of their protocols.

Part I.

Stronger Security in Extended Models

3. Perfect Forward Secrecy under Actor Compromise and Randomness Reveal

In this chapter we show that it is possible to achieve perfect forward secrecy in two-message or one-round authenticated key exchange (AKE) protocols even in the presence of very strong active adversaries who can reveal random values of sessions and compromise long-term secret keys of users. In Section 3.1 we formalize PFS and weak-PFS, and introduce our security notions eCK^w and eCK-PFS. The eCK^w model is a slightly stronger variant of the eCK security model. The eCK-PFS model captures perfect forward secrecy in the presence of eCK^w adversaries. In Section 3.2 we provide a security-strengthening transformation that turns any two-message Diffie-Hellman type protocol secure in either eCK^w or $\mathsf{eCK}^{\mathsf{passive}}$, a weaker model than eCK^w, into a two-message protocol secure in eCK-PFS. We demonstrate how our transformation can be applied to concrete protocols in Section 3.3. In particular, our methodology allows us to develop new AKE protocols that achieve perfect forward secrecy under actor compromise and randomness reveal.

3.1. Defining new eCK-like security models

We propose two new eCK-like security models for the analysis of AKE protocols. The first model, called eCK^w, captures a slightly stronger form of weak-PFS than the original eCK model. The second model, called eCK-PFS, integrates perfect forward secrecy directly into eCK^w. We first describe a framework for defining security models in Section 3.1.1. Using this framework, we define our new security notions in Sections 3.1.2 and 3.1.3. We then formally compare them in Section 3.1.4.

3.1.1. Framework for security models

Terminology. An *AKE protocol* π consists of a set of domain parameters, a key generation algorithm KeyGen, and the protocol description that describes how key exchange protocol messages are generated and responded to as well as how the session key is derived. We say that an AKE protocol is a *two-message protocol* if the sum of the number of messages sent and received by a user during an execution of the protocol is exactly two. A more formal definition of AKE protocols is given in Chapter 5.

Let $\mathcal{P} = \left\{\hat{P}_1, \hat{P}_2, ..., \hat{P}_N\right\}$ be a finite set of N honest users represented by binary strings. Each honest user can execute multiple instances of an AKE protocol, called *sessions*, concurrently. We denote session i of honest user $\hat{P}$ as the tuple $(\hat{P}, i) \in \mathcal{P} \times \mathbb{N}$. We associate to each session $s \in \mathcal{P} \times \mathbb{N}$ a quintuple of variables $T_s = (s_{actor}, s_{peer}, s_{\text{role}}, s_{sent}, s_{recv}) \in \mathcal{P} \times \{0,1\}^* \times \{\mathcal{I}, \mathcal{R}\} \times \{0,1\}^* \times \{0,1\}^*$. The variables

s_{actor}, s_{peer} denote the identities of the actor and peer of session s, s_{role} denotes the role that the session is executing (either initiator or responder), and s_{sent}, s_{recv} denote the concatenation of timely ordered messages as sent/received by s_{actor} during session s. The values of the variables s_{peer} and s_{role} are set upon activation of session s and the values of the variables s_{sent} and s_{recv} are updated during protocol execution. A session can only be activated once. We say that a session is *completed* if it has accepted a key K as the session key.

Adversarial capabilities. As is standard for Bellare-Rogaway style security notions for AKE [17], we model the adversary as a probabilistic polynomial-time (PPT) Turing machine that controls all communications between users. Similar to the eCK model [68], we consider the following adversarial capabilities, so-called queries:

1. $\mathsf{send}(s, v)$. This query models the adversary sending message v to session s of honest user s_{actor}. The adversary is given the response generated by the session according to the protocol. The variables s_{sent} and s_{recv} are updated accordingly (by concatenation). Abusing notation, we allow the adversary to activate an initiator session with peer $\hat{Q}$, via a $\mathsf{send}(s, \hat{Q})$ query and a responder session by sending a message m to session s on behalf of $\hat{Q}$, via a $\mathsf{send}(s, \hat{Q}, m)$ query. In these cases, s_{peer} is set to $\hat{Q}$ and s_{role} is set to $\mathcal{I}$ and $\mathcal{R}$, respectively. The adversary is given the session's response according to the protocol and the variables s_{sent}, s_{recv} are initialized accordingly.
2. $\mathsf{corrupt}(\hat{P})$. If $\hat{P} \in \mathcal{P}$, then the query returns the long-term secret keys of user $\hat{P}$. Otherwise the query returns $\perp$.
3. $\mathsf{randomness}(s)$. This query returns the randomness used in session s.
4. $\mathsf{session\text{-}key}(s)$. This query returns the session key for a completed session s.
5. $\mathsf{test\text{-}session}(s)$. To respond to this query, a bit b is chosen uniformly at random. If $b = 1$, then the session key established in session s is returned. Otherwise, a random key is returned according to the probability distribution of keys generated by the protocol. This query can only be issued to a completed session.

Notions of Freshness. An adversary that can perform the above queries can simply reveal the session key of all sessions, breaking any protocol. The intuition underlying Bellare-Rogaway style AKE models is to put minimal restrictions on the adversary with respect to performing these queries, such that there still exist protocols that are secure in the presence of such an adversary. The restrictions on the queries made by the adversary are formalized via freshness predicates, which take a session of the protocol and a sequence of queries (including arguments and results). Examples of such predicates will be given in the following two sections.

Security Model. A game-based security model M is given by a set of queries $\{\mathsf{send}\} \subseteq Q \subseteq \{\mathsf{send}, \mathsf{corrupt}, \mathsf{randomness}, \mathsf{session\text{-}key}\}$ and a freshness predicate F.

Security experiment W in model $M = (Q, F)$. Security of a key-exchange protocol π is defined via a security experiment W (or attack game) played by an adversary E, modeled as a PPT algorithm, against a challenger.

Before the experiment starts properly, there is a setup phase, in which the challenger runs a key-generation algorithm specified by the protocol that takes as input a security parameter 1^k and outputs valid long-term secret/public key pair(s), for each user

$\hat{P} \in \mathcal{P}$. The adversary is then given all public data, including the public keys of all the honest users in $\mathcal{P}$. Then, the adversary can choose to register arbitrary valid public keys (even public keys of honest users) on behalf of a set of adversary-controlled users $\hat{L} \notin \mathcal{P}$.

After the above setup phase, the security experiment W can be described in four successive stages, as follows:

1. The adversary E can perform any sequence of queries from Q.
2. At some point in the experiment, E issues a test-session query to a completed session that satisfies F at the time the query is issued.
3. The adversary may continue with queries from Q, under the condition that the test session must continue to satisfy F.
4. Finally, E outputs a bit b' as his guess for b.

The adversary E wins the security experiment W if he correctly guesses the bit b chosen by the challenger during the test-session query (i. e., if $b = b'$ where b' denotes E's guess). Success of E in the experiment is expressed in terms of E's advantage in distinguishing whether he received the real or a random session key in response to the test-session query. The advantage of adversary E in the above security experiment against an AKE protocol π for security parameter k is defined as $Adv_E^{\pi}(k) = |2P(b = b') - 1|$.

The notion of *matching sessions* specifies when two sessions are supposed to be intended communication partners. Here we formalize the matching sessions definition from the eCK model [68] which is based on matching conversations.

Definition 10 (matching sessions)**.** *Two completed sessions s and s' are said to be* matching *if*

$$s_{actor} = s'_{peer} \wedge s_{peer} = s'_{actor} \wedge s_{sent} = s'_{recv} \wedge s_{recv} = s'_{sent} \wedge s_{role} \neq s'_{role}.$$

As in the eCK model, we require that matching sessions perform different roles. The consequences of such a choice are explored in detail in [41]. Two issues are important here. First, there is a strong connection between the information used in a matching definition and the information used to compute the session key. Second, some protocols like the two-message versions of MQV and HMQV allow sessions to compute the same key even if they perform the same role, whereas other protocols such as NAXOS and π_1-core from Section 3.3 require the sessions that compute the same key to perform different roles. In this paper we follow the eCK setup, which applies directly to protocols of the second type. Protocols of the first type can be dealt with by dropping the requirement of different roles from the matching definition.

Definition 11 (security)**.** *An AKE protocol π is said to be* secure in model M *if, for all PPT adversaries E, it holds that*

- *if two honest users successfully complete matching sessions, then they compute the same session key, and*
- *E has no more than a negligible advantage in winning security experiment W in model M, that is, there exists a negligible function negl in the security parameter k such that $Adv_E^{\pi}(k) \leq negl(k)$.*

3.1.2. eCK^w: strengthening weak-PFS

The eCK model captures weak perfect forward secrecy but not perfect forward secrecy, based on Krawczyk's PFS attack on MQV [64, 65]. We first formally define perfect forward secrecy, and then briefly recall the attack.

It is hard to find a formal definition of perfect forward secrecy, as it is common to argue informally about PFS. For example, the following informal definition is given in [84, p. 496]:

> "A protocol is said to have *perfect forward secrecy* if compromise of long-term keys does not compromise past session keys."

However, such a definition does not suffice when we want to formally prove that our models imply PFS or similar properties. To address this, we provide a formal definition of PFS in the form of a Bellare-Rogaway-style security definition. This allows us to make precise formal statements about the properties that our models achieve and the relations between them.

Definition 12 (PFS)**.** *The* PFS *model is defined by* (Q, F)*, where* $Q = \{\mathsf{send}, \mathsf{corrupt}\}$ *and* F *is defined as follows. A session* s *is said to satisfy* F *if, before the completion of session* s*, no keys have been registered on behalf of adversary-controlled users and no* corrupt *query has been issued. We say that a protocol satisfies perfect forward secrecy (PFS) if it is secure in the* PFS *model.*

We now return to Krawczyk's PFS attack. Consider a two-message protocol in which the agents exchange ephemeral Diffie-Hellman exponentials, i. e., g^x and g^y, where x and y are chosen uniformly at random from $\mathbb{Z}_p$ (for some large prime p). Then, Krawczyk's attack proceeds as follows. The adversary, impersonating user $\hat{A}$, generates a value x ($\in \mathbb{Z}_p$) and sends g^x to a responder session of user $\hat{B}$. $\hat{B}$ responds by sending g^y and computes the session key. The adversary chooses $\hat{B}$'s session as the test session, i. e., the session under attack, and reveals $\hat{A}$'s long-term secret key after $\hat{B}$'s session ends. Now the adversary can simply follow all protocol steps that an honest user $\hat{A}$ would have performed using x and $\hat{A}$'s long-term secret key. In particular, the adversary can compute the same session key as the test session, violating perfect forward secrecy.

In Chapter 5 we generalize Krawczyk's PFS attack on MQV to all one-round Diffie-Hellman type protocols; this class includes, e. g., the HMQV protocol [64], where the exchanged Diffie-Hellman exponentials do not involve the sender's long-term secret key, and the NAXOS protocol [68], where the exponent $z \in \mathbb{Z}_p$ in the Diffie-Hellman exponential is a hash of the concatenation of the sender's long-term secret key and a random value.

To still prove some form of forward secrecy for two-message Diffie-Hellman type protocols such as HMQV, Krawczyk introduced the notion of weak-PFS. In weak-PFS, the adversary is not allowed to actively interfere with the messages exchanged by the test session. This prevents the attack because the adversary is no longer allowed to insert his own DH exponential. Similarly, in the eCK model, this restriction on interfering with the test session is modeled by checking if a matching session exists [68, p. 5]. If this is the case, then the adversary must have been passive

and he is allowed to reveal the long-term secret keys of the actor and the intended communication partner of a session. If there is no matching session, the adversary is not allowed to reveal the long-term secret key of the intended communication partner.

We observe that Krawczyk's attack only depends on the adversary injecting or modifying the message *received* by the test session; he does not need to actively interfere with the message *sent* by the test session. However, eCK models passivity of the adversary in the test session by checking whether a matching session for the test session exists, which also prevents the adversary from modifying (or deleting) the message sent by the test session. In this sense, the restriction on the adversary in eCK is sufficient but not necessary for the prevention of Krawczyk's attack. We therefore relax the notion of matching sessions and introduce the concept of *origin-session*.

Definition 13 (origin-session)**.** *We say that a (possibly incomplete) session s' is an* origin-session *for a completed session s when $s'_{sent} = s_{recv}$.*

Note that if two completed sessions s, s' are matching, then s and s' are origin-sessions for each other. However, if session s is an origin-session for some session s', then it might not necessarily be a matching session for s' (e. g. in case the roles of the sessions are identical). Thus, a session being a matching session for some session is a stronger requirement than a session being an origin-session for some session.

Using this notion, we give the first formal definition of weak Perfect Forward Secrecy. In order to exclude Krawczyk's generic PFS attack, we disallow the adversary from injecting his own messages into the test session. However, whereas Krawczyk enforced this by requiring a matching session to exist, we merely require the messages received by the test session to have been sent by a so-called origin-session. In other words, if an origin-session s' for some session s exists, then the messages received by session s have not been modified or injected by the adversary.

Definition 14 (wPFS)**.** *The* wPFS *model is defined by (Q, F), where $Q = \{$*send, corrupt*$\}$ and F is defined as follows. A session s is said to satisfy F if all of the following conditions hold:*

1. *there exists an origin-session for session s, and*
2. *before the completion of session s, no keys have been registered on behalf of adversary-controlled users and no* corrupt *query has been issued.*

We say that a protocol satisfies weak perfect forward secrecy (weak-PFS) if it is secure in the wPFS *model.*

Summarizing, we capture weak-PFS by the compromise of long-term secret keys of users after the end of the test session under the condition that an origin-session for the test session exists. Thus, we model passivity of the adversary in the test session by the existence of an origin-session for the test session (and not by the existence of a matching session for the test session, as in [64, 68]).

Compared to the original eCK model, our definition of weak-PFS enables us to capture an additional capability of the adversary: revealing the long-term secret key of the intended communication partner (i. e. the peer) of the test session s in case an origin-session s' for s exists, even when no matching session exists for s. Thus, in contrast to the eCK model, the adversary may reveal the long-term key of the peer of the test session s in case an origin-session s' for session s exists and

- actively interfere with the message sent by the test session (e. g. by modifying it or injecting his own message), or
- replay a message from another session to the test session (as in [31]), or
- leave session s' incomplete (in case s' is an initiator session).

We call our strengthened variant of the eCK model the eCK^w model.

Definition 15 (eCK^w)**.** *The* eCK^w *model is defined by* (Q, F)*, where* $Q = \{\mathsf{send}, \mathsf{corrupt}, \mathsf{randomness}, \mathsf{session\text{-}key}\}$ *and* F *is defined as follows. A session* s *is said to satisfy* F *if all of the following conditions hold:*

1. s_{actor} *and* s_{peer} *are honest users, i. e.* $(s_{actor}, s_{peer}) \in \mathcal{P} \times \mathcal{P}$*,*
2. *no* $\mathsf{session\text{-}key}(s)$ *query has been issued,*
3. *for all sessions* s^* *such that* s^* *matches* s*, no* $\mathsf{session\text{-}key}(s^*)$ *query has been issued,*
4. *not both queries* $\mathsf{corrupt}(s_{actor})$ *and* $\mathsf{randomness}(s)$ *have been issued,*
5. *for all sessions* s' *such that* s' *is an origin-session for session* s*, not both queries* $\mathsf{corrupt}(s_{peer})$ *and* $\mathsf{randomness}(s')$ *have been issued, and*
6. *if there exists no origin-session for session* s*, then no* $\mathsf{corrupt}(s_{peer})$ *query has been issued.*

In Section 3.3.1 we will show that the NAXOS protocol is secure in the eCK^w model.

3.1.3. eCK-PFS: integrating perfect forward secrecy into eCK^w.

We next extend the eCK^w model by integrating perfect forward secrecy, which yields the stronger eCK-PFS model. Perfect forward secrecy is reflected in eCK-PFS by allowing the adversary to reveal the long-term secret keys of all the protocol participants *after* the end of the test session, as in the PFS model. These keys can be revealed irrespective of the existence of an origin-session (or a matching session). The PFS attack scenario is neither captured in eCK^w (nor in eCK) if the origin-session (matching session) does not exist for the test session. In contrast to the way in which the CK-NSR model from [31] incorporates PFS, eCK-PFS additionally captures leakage of various combinations of long-term secret keys and randomness as well as perfect forward secrecy under actor compromise.

Definition 16 (eCK-PFS)**.** *The* eCK-PFS *model is defined by* (Q, F)*, where* $Q = \{\mathsf{send}, \mathsf{corrupt}, \mathsf{randomness}, \mathsf{session\text{-}key}\}$ *and* F *is defined as follows. A session* s *is said to satisfy* F *if all of the following conditions hold:*

1. s_{actor} *and* s_{peer} *are honest users, i. e.* $(s_{actor}, s_{peer}) \in \mathcal{P} \times \mathcal{P}$*,*
2. *no* $\mathsf{session\text{-}key}(s)$ *query has been issued,*
3. *for all sessions* s^* *such that* s^* *matches* s*, no* $\mathsf{session\text{-}key}(s^*)$ *has been issued,*
4. *not both queries* $\mathsf{corrupt}(s_{actor})$ *and* $\mathsf{randomness}(s)$ *have been issued,*
5. *for all sessions* s' *such that* s' *is an origin-session for session* s*, not both queries* $\mathsf{corrupt}(s_{peer})$ *and* $\mathsf{randomness}(s')$ *have been issued, and*
6. *if there exists no origin-session for session* s*, then no* $\mathsf{corrupt}(s_{peer})$ *query has been issued before the completion of session* s*.*

3.1.4. Relations between the security models

We formalize the relative strengths of security between game-based security models investigated by Choo et al. [37] as follows. Let $secure(M, \pi)$ be a predicate that is true if and only if the protocol π is secure in security model M.

Definition 17. *Let* Π *be a class of AKE protocols. Let* M *and* M' *be two security models. We say that model* M' *is* at least as strong as *model* M *with respect to* Π*, denoted by* $M \leq_{Sec}^{\Pi} M'$*, if*

$$\forall\ \pi \in \Pi.\ secure(M', \pi) \rightarrow secure(M, \pi)\ . \tag{3.1}$$

We say that model M' *is* stronger *than model* M *with respect to protocol class* Π*, denoted by* $M <_{Sec}^{\Pi} M'$*, if* $M \leq_{Sec}^{\Pi} M'$ *and not* $M' \leq_{Sec}^{\Pi} M$*.*

The previous definition implies that protocols proven secure in model M' will be secure in model M, where $M \leq_{Sec}^{\Pi} M'$. To show that model M' is not at least as strong as model M, it suffices to find a protocol $\pi \in \Pi$ such that π is secure in model M' and insecure in model M, as in [37, 41].

Informally, the eCK-PFS model is at least as strong as the eCK^w model because the eCK-PFS model allows the adversary to corrupt all users *after* the test session is completed (regardless of whether an origin-session exists for the test session), capturing perfect forward secrecy. In contrast, in case the adversary is active in the message received by the test session, he is not allowed to reveal the long-term secret key of the peer of the test session in the eCK^w model.

Proposition 2. *Let* Π *be the class of two-message protocols. The* eCK-PFS *model is stronger than the* eCK^w *model with respect to* Π*.*

The first part of the proof of Proposition 2, namely that eCK-PFS is at least as strong as eCK^w, proceeds in a similar way as the reduction proofs in [37].

Proof. We first show that the eCK-PFS model is at least as strong as the eCK^w model with respect to Π. The first condition of Definition 11 is satisfied since matching is defined in the same way for both models eCK^w and eCK-PFS. Let $\pi \in \Pi$. To show that the second condition of Definition 11 holds, we construct an adversary E' attacking protocol π in model eCK-PFS using an adversary E attacking π in eCK^w. Adversary E' proceeds as follows. Whenever E issues a query send, corrupt, randomness, session-key or test-session, adversary E' issues the same query and forwards the answer received to E. At the end of E's execution, i. e. after it has output its guess bit b, E' outputs b as well. Note that if the freshness condition of eCK^w holds for the test session, then by definition the freshness condition of eCK-PFS also holds. In particular, if there is no origin-session, then the sixth condition of the freshness condition of eCK^w requires that there is no corrupt of the peer, which implies the sixth condition of the freshness condition of eCK-PFS. Hence, it holds that $Adv_E^{\pi}(k) \leq Adv_{E'}^{\pi}(k)$, where k denotes the security parameter. Since by assumption protocol π is secure in eCK-PFS, there is a negligible function g such that $Adv_{E'}^{\pi}(k) \leq g(k)$. It follows that protocol π is secure in eCK^w.

The model eCK-PFS is stronger than eCK^w since, e. g., the NAXOS protocol is secure in eCK^w, as we show in Section 3.3, but insecure in eCK-PFS due to the PFS attack described in Section 3.1.2. □

The following proposition states that PFS is a stronger property than weak-PFS.

Proposition 3. *Let* Π *be the class of two-message protocols. The* PFS *model is stronger than the* wPFS *model with respect to* Π.

Proof. The proof that the PFS model is at least as strong as the wPFS model is similar to the corresponding proof of Proposition 2. Note that the test session satisfying the freshness predicate of wPFS implies that it satisfies the freshness predicate of PFS since we have a further freshness requirement in the wPFS model. The PFS model is stronger than the wPFS model since, e. g., the NAXOS protocol achieves weak-PFS as can be easily deduced from the proof of Proposition 8, but does not satisfy PFS due to the generic PFS attack described in Section 3.1.2. □

The relations between these four models and the $\mathsf{eCK}^{\mathsf{passive}}$ model that we define in Section 3.2.3 are depicted in Figure 3.1. The notation $X \xrightarrow{<^{\Pi}_{Sec}} Y$ denotes that model Y is stronger than model X with respect to Π. The section in which the implication is proven is indicated in parentheses next to the arrow.

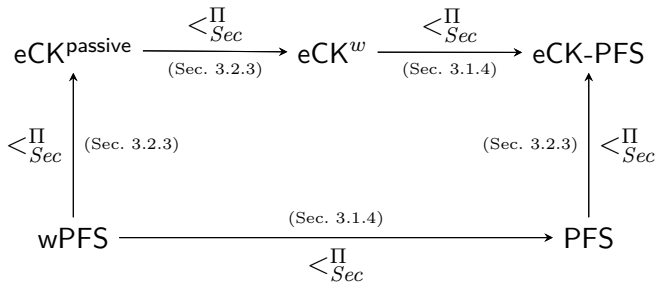

Figure 3.1.: Relations between the security models for the class of two-message protocols.

3.2. A security-strengthening transformation from eCK^w to eCK-PFS

3.2.1. Protocol class $\mathcal{DH}$-2

We define a class of two-message Diffie-Hellman type AKE protocols (similar to the class of protocols in [31]). Then, we present a security-strengthening transformation that can be applied to any such protocol. Finally we show that this transformation turns any protocol secure in eCK^w into a protocol secure in eCK-PFS.

Definition 18 (DH-type AKE protocol)**.** *Let* k *be a security parameter. Let* Ω *be static publicly known information such as users' identifiers (binary strings in* $\mathcal{P}$*), their*

long-term public keys or publicly known functions and parameters. Let S be a set of constants from which random values are chosen. The elements in brackets in the exchanged messages represent optional information. A DH-type AKE protocol *is an AKE protocol of the following form, specified by functions $f_{\mathcal{I}}, f_{\mathcal{R}}, F_{\mathcal{I}}, F_{\mathcal{R}}$:*

- *Domain parameters (G, g, p), where $G = \langle g \rangle$ is a group of prime order p generated by g with $\|p\| = k$.*
- $\mathsf{KeyGen}()$: *Choose $a \in_R [0, p-1]$. Set $A \leftarrow g^a$. Return secret key $\mathsf{sk} = a$ and public key $\mathsf{pk} = A$.*
- *Protocol description, illustrated in Figure 3.2:*
  1. *Upon activation of a new initiator session s with a $\mathsf{send}(s = (\hat{A}, i), \hat{B})$ query, $\hat{A}$ chooses randomness $r_{\hat{A}} \in_R S$, computes $x = f_{\mathcal{I}}(r_{\hat{A}}, a, \Omega)$ and the outgoing ephemeral public key $X = g^x$, and returns X as an outgoing message.*
  2. *Upon activation of responder session s' via the query $\mathsf{send}(s' = (\hat{B}, j), \hat{A}, X)$, $\hat{B}$ checks that $X \in G$, chooses randomness $r_{\hat{B}} \in_R S$, computes $y = f_{\mathcal{R}}(r_{\hat{B}}, b, \Omega)$ and the outgoing ephemeral public key $Y = g^y$, and returns $[X,] Y$ as an outgoing message. User $\hat{B}$ computes a session key $K_{\hat{B}} = F_{\mathcal{R}}(y, b, X, \Omega)$, and completes the session by accepting $K_{\hat{B}}$ as the session key.*
  3. *Upon receiving message Y in session s with the query $\mathsf{send}(s, Y)$, user $\hat{A}$ checks that $Y \in G$, computes a session key $K_{\hat{A}} = F_{\mathcal{I}}(x, a, Y, \Omega)$, and completes the session by accepting $K_{\hat{A}}$ as the session key.*

  The above description also applies to protocols with additional checks, which we omit for clarity. We assume that whenever a check in a session fails, all session-specific data is erased from memory and the session is aborted, i. e., it terminates without establishing a session key.

Initiator $\mathcal{I}$		**Responder $\mathcal{R}$**
$\hat{A}$: (a, A)		$\hat{B}$: (b, B)
$x \leftarrow f_{\mathcal{I}}(r_{\hat{A}}, a, \Omega)$		
$X \leftarrow g^x$	$\xrightarrow{X}$	
		$y \leftarrow f_{\mathcal{R}}(r_{\hat{B}}, b, \Omega)$
		$Y \leftarrow g^y$
	$\xleftarrow{[X,]Y}$	$K_{\hat{B}} \leftarrow F_{\mathcal{R}}(y, b, X, \Omega)$
$K_{\hat{A}} \leftarrow F_{\mathcal{I}}(x, a, Y, \Omega)$		

Figure 3.2.: Messages for generic DH-type AKE protocol

Definition 19 (Protocol class $\mathcal{DH}$-2)**.** *We define $\mathcal{DH}$-2 as the class of all DH-type AKE protocols that meet the following validity requirement:*

- *In the presence of an eavesdropping adversary, two honest users $\hat{A}$ and $\hat{B}$ can complete matching sessions (in the sense of Definition 10), in which case they hold the same session key.*

Note that, e. g., the protocols NAXOS [68], NAXOS+ [72], NETS [71] and CMQV [104] belong to the class $\mathcal{DH}$-2.

Remark 1. The protocol class $\mathcal{DH}$-2 contains the subclass of one-round DH-type protocols in which messages are generated independently from each other.

3.2.2. Protocol transformation SIG

Here we show how to transform any protocol $\pi \in \mathcal{DH}$-2 into a two-message protocol SIG(π), shown in Figure 3.3, by applying the signature transformation SIG . User $\hat{A}$ has two *independent* valid long-term secret/public key pairs, one pair (a, A) from protocol π and one pair $(sk_{\hat{A}}, pk_{\hat{A}})$ for use in a digital signature scheme Σ with security parameter k. Similarly, user $\hat{B}$'s long-term secret/public key pairs are (b, B) and $(sk_{\hat{B}}, pk_{\hat{B}})$. The transformed protocol SIG(π) in Figure 3.3 proceeds as protocol π except that each user needs to additionally sign a message using its secret signature key and check that the received signature on a message is valid with respect to the long-term public key of its peer. The fields between square brackets within the signature are optional. Note that if the objective is to obtain a one-round protocol, then X should not be included in the second message.

Initiator $\mathcal{I}$		**Responder** $\mathcal{R}$
$\hat{A}$: (a, A), $(sk_{\hat{A}}, pk_{\hat{A}})$		$\hat{B}$: (b, B), $(sk_{\hat{B}}, pk_{\hat{B}})$
$x \leftarrow f_{\mathcal{I}}(r_{\hat{A}}, a, \Omega)$ $X \leftarrow g^x$		
$\sigma_{\hat{A}} \leftarrow Sign_{sk_{\hat{A}}}(X \left[, \hat{B}\right])$	$\xrightarrow{X, \sigma_{\hat{A}}}$	
		$y \leftarrow f_{\mathcal{R}}(r_{\hat{B}}, b, \Omega)$ $Y \leftarrow g^y$ $\sigma_{\hat{B}} \leftarrow Sign_{sk_{\hat{B}}}(Y \left[, X, \hat{A}\right])$
	$\xleftarrow{Y, \sigma_{\hat{B}}}$	$K_{\hat{B}} \leftarrow F_{\mathcal{R}}(y, b, X, \Omega)$
$K_{\hat{A}} \leftarrow F_{\mathcal{I}}(x, a, Y, \Omega)$		

Figure 3.3.: A transformed generic protocol SIG(π)

Informally, a security-strengthening protocol transformation is a mapping between protocols such that the transformed protocol satisfies stronger security properties. We formally define a security-strengthening protocol transformation as follows.

Definition 20 (Security-strengthening protocol transformation)**.** *Let* Π_1 *and* Π_2 *be two classes of AKE protocols. We say that a function* $f : \Pi_1 \rightarrow \Pi_2$ *is a* security-strengthening protocol transformation *from the model* M *to the model* M' *if*

1. $M \leq_{Sec}^{\Pi_2} M'$*, and*
2. $\forall \pi \in \Pi_1.\ secure(M, \pi) \rightarrow secure(M', f(\pi))$.

The previous definition implies that if an AKE protocol $\pi \in \Pi_1$ is secure in model M, $M \leq_{Sec}^{\Pi_2} M'$, and f is a security-strengthening transformation from M to M', then protocol $f(\pi) \in \Pi_2$ is secure in model M'. Note that, by Definition 17, it follows that protocol $f(\pi)$ is secure in model M.

3.2.3. Security analysis of SIG

Our original intent was to show that SIG is security-strengthening from eCK^w to eCK-PFS, but we will in fact show a more general result: we show that SIG is a security-strengthening protocol transformation from a weaker version of the eCK^w model, which we call eCK$^{\text{passive}}$, to eCK-PFS.

***Weakening* eCK^w *to* eCK$^{\text{passive}}$.** Informally, the eCK$^{\text{passive}}$ model weakens eCK^w by capturing only passive attacks on the test session. The adversary can delay, forward, or replay messages to the test session. However, he is not allowed to inject a message to the test session or to modify the message that the test session receives. As in the eCK^w model, the adversary is allowed to reveal the randomness that is used in sessions and long-term secret keys of users, thus it also captures, e. g., weak perfect forward secrecy. Formally, the eCK$^{\text{passive}}$ model only differs from eCK^w in its definition of freshness. In eCK$^{\text{passive}}$, there must exist an origin-session for the test session.

Definition 21 (eCK$^{\text{passive}}$)**.** *The* eCK$^{\text{passive}}$ *model is defined by* (Q, F)*, where* $Q = \{\mathsf{send}, \mathsf{corrupt}, \mathsf{randomness}, \mathsf{session\text{-}key}\}$ *and* F *is defined as follows. A session* s *is said to satisfy* F *if all of the following conditions hold:*

1. s_{actor} *and* s_{peer} *are honest users, i. e.* $(s_{actor}, s_{peer}) \in \mathcal{P} \times \mathcal{P}$*,*
2. *there exists an origin-session* s' *for session* s*,*
3. *no* session-key(s) *query has been issued,*
4. *for all sessions* s^* *such that* s^* *matches* s*, no* session-key(s^*) *query has been issued,*
5. *not both queries* corrupt(s_{actor}) *and* randomness(s) *have been issued, and*
6. *for all sessions* s' *such that* s' *is an origin-session for session* s*, not both queries* corrupt(s_{peer}) *and* randomness(s') *have been issued.*

Proposition 4. *Let* Π *be the class of two-message protocols.*

- *The* eCK$^{\text{passive}}$ *model is stronger than the* wPFS *model with respect to* Π*.*
- *The* eCK-PFS *model is stronger than the* PFS *model with respect to* Π*.*

Proof. The proof that the eCK$^{\text{passive}}$ model is at least as strong as the wPFS model is similar to the corresponding proof of Proposition 2. Note that the test session satisfying the freshness predicate of model wPFS implies that it is also satisfies the freshness predicate of model eCK$^{\text{passive}}$ since in the wPFS model the adversary is not given access to the queries randomness and session-key. Also, in the wPFS model the adversary is not allowed to either register keys on behalf of adversary-controlled users or to issue a corrupt query before the completion of the test session. A similar argument applies to show that the eCK-PFS model is at least as strong as the PFS model.

The eCK$^{\text{passive}}$ model is stronger than the wPFS model. It can be easily shown that protocol TS2 achieves weak-PFS by adapting the proof of [56, Theorem 2]. However, protocol TS2 is insecure against an adversary who can reveal the long-term secret key of the actor of the test session and the randomness of the origin-session for the test session. Hence, TS2 is insecure in eCK$^{\text{passive}}$. The eCK-PFS model is stronger than the PFS model. By Theorem 1, it holds that SIG(TS2) is secure in the PFS model.

However, SIG(TS2) is insecure in eCK-PFS for a similar reason as TS2 is insecure in $\mathsf{eCK}^{\mathsf{passive}}$. □

Propositions 5 and 6 will be used in the proofs of Theorem 1 and Corollary 1.

Proposition 5. *Let* Π *be the class of two-message protocols. The* eCK^{w} *model is stronger than the* $\mathsf{eCK}^{\mathsf{passive}}$ *model with respect to* Π*.*

Proof. The proof that the eCK^{w} model is at least as strong as the $\mathsf{eCK}^{\mathsf{passive}}$ model is similar to the corresponding proof of Proposition 2. Note that if the test session satisfies the freshness predicate of $\mathsf{eCK}^{\mathsf{passive}}$, then it also satisfies the freshness predicate of eCK^{w}. This follows from the fact that compared to the eCK^{w} model, we have a further freshness condition on the test session in $\mathsf{eCK}^{\mathsf{passive}}$, namely that an origin-session exists for the test session.

The model eCK^{w} is stronger than the $\mathsf{eCK}^{\mathsf{passive}}$ model since, e. g., the protocol π_1-core is secure in $\mathsf{eCK}^{\mathsf{passive}}$, as we show in Section 3.3, but insecure in eCK^{w} as observed in [65]. □

Proposition 6. *Let* Π *be the class of two-message protocols. The* eCK-PFS *model is stronger than the* $\mathsf{eCK}^{\mathsf{passive}}$ *model with respect to* Π*.*

Proof. Since $\mathsf{eCK}^{\mathsf{passive}} <^{\Pi}_{Sec} \mathsf{eCK}^{w}$ and $\mathsf{eCK}^{w} <^{\Pi}_{Sec}$ eCK-PFS, it follows that $\mathsf{eCK}^{\mathsf{passive}} <^{\Pi}_{Sec}$ eCK-PFS by transitivity of Implication (3.1). The eCK-PFS model is stronger than $\mathsf{eCK}^{\mathsf{passive}}$ since, e. g., the protocol π_1-core is secure in $\mathsf{eCK}^{\mathsf{passive}}$ but insecure in eCK-PFS. □

How to provably achieve eCK-PFS ***security.*** We show in Theorem 1 below that the SIG transformation is a security-strengthening protocol transformation from $\mathsf{eCK}^{\mathsf{passive}}$ to eCK-PFS provided that the digital signature scheme is strongly existentially unforgeable under an adaptive chosen-message attack (SUF-CMA) as well as deterministic. An example of such a scheme is the GDH signature scheme from [26]. We require a deterministic signature scheme so that we do not have to consider additional randomness from the signature generation procedure when reasoning about randomness queries. For certain randomized signature schemes, an efficient adversary can compute the secret (signature) key given the corresponding public key, a signature on any message using the secret key, and the randomness involved in the signature generation learned through a randomness query (as noted in [68]). The following lemma is used in the proof of Theorem 1.

Lemma 1 (Difference Lemma [98])**.** *Let* A, B, F *be events defined on some probability space. Suppose that event* $A \wedge F^c$ *occurs if and only if event* $B \wedge F^c$ *occurs (where* F^c *denotes the complement of event* F*). Then*

$$|P(A) - P(B)| \leq P(F).$$

Theorem 1. *Let* Π *denote the class of two-message protocols. Under the assumption that the signature scheme is deterministic and SUF-CMA, the transformation* SIG : $\mathcal{DH}$*-2* $\rightarrow$ Π *is a security-strengthening protocol transformation from* $\mathsf{eCK}^{\mathsf{passive}}$ *to* eCK-PFS *according to Definition 20.*

Proof. The first condition of Definition 20 is satisfied by Proposition 6. We next verify whether the second condition of Definition 20 holds. Let $\pi \in \mathcal{DH}$-2 be secure in model eCK$^{\text{passive}}$. It is straightforward to verify the first condition of Definition 11, i. e., that matching sessions of protocol SIG(π) compute the same key (since matching sessions of protocol π compute the same key). We show next that the second condition of Definition 11 holds, i. e., an adversary against SIG(π) in eCK-PFS has no more than a negligible advantage in distinguishing the session key from a random key. We present a security proof structured as a sequence of games, a proof technique introduced in [98]. Let S_i denote the event that the adversary correctly guesses the bit chosen by the challenger to answer the test-session query in Game i and let $\alpha_i = |2P(S_i) - 1|$ denote the advantage of the adversary in Game i. Let N, q_s be upper bounds on the number of users and activated sessions, respectively.

Game 0. This game reflects the security experiment W in model eCK-PFS, as defined in Section 3.1.1, played by a PPT adversary E against the protocol SIG(π).

Game 1. [Transition based on a small failure event] Let $Coll_{\text{SIG}(\pi)}$ be the small failure event that a collision for protocol SIG(π) occurs (e. g., in session-specific randomness). As soon as event $Coll_{\text{SIG}(\pi)}$ occurs, the attack game stops.

Analysis of game 1. Game 0 is identical to Game 1 up to the point in the experiment where event $Coll_{\text{SIG}(\pi)}$ occurs for the first time. The Difference Lemma yields that $|P(S_0) - P(S_1)| \leq P(Coll_{\text{SIG}(\pi)})$. Hence,

$$\begin{aligned}\alpha_0 &= |2P(S_0) - 1| = 2|P(S_0) - P(S_1) + P(S_1) - 1/2| \\ &\leq 2(|P(S_0) - P(S_1)| + |P(S_1) - 1/2|) \\ &\leq 2P(Coll_{\text{SIG}(\pi)}) + \alpha_1.\end{aligned}$$

Game 2. [Transition based on a large failure event (see [30, 44])] Before the adversary E starts the attack game, the challenger chooses a random value $m \in_R \{1, 2, ..., q_s\}$. The m-th session activated by E, denoted by s^*, is the session on which the challenger wants the adversary to be tested. Let T be the event that the test session is not session s^*. If event T occurs, then the attack game halts and the adversary outputs a random bit.

Analysis of game 2. Event T is non-negligible, the environment can efficiently detect it and T is independent of the output in Game 1 (i. e. $P(S_1|T) = P(S_1)$). If T does not occur, then the attacker E will output the same bit in Game 2 as it did in Game 1 (so that $P(S_2|T^c) = P(S_1|T^c) = P(S_1)$). If event T occurs in Game 2, then the attack game halts and the adversary E outputs a random bit (so that $P(S_2|T) = 1/2$). We have,

$$\begin{aligned}P(S_2) &= P(S_2|T)P(T) + P(S_2|T^c)P(T^c) = \frac{1}{2}P(T) + P(S_1)P(T^c) \\ &= P(T^c)(P(S_1) - \frac{1}{2}) + \frac{1}{2}.\end{aligned}$$

Hence we get, $\alpha_2 = |2P(S_2) - 1| = P(T^c)|2P(S_1) - 1| = \frac{1}{q_s}\alpha_1$.

Suppose w. l. o. g. that $s^*_{role} = \mathcal{I}$ and that protocol π does not include optional public information in the sent messages. Let F be a forgery event with respect to the long-term public key $pk_{\hat{P}}$ of user $\hat{P}$, that is, adversary E issues a $\mathsf{send}(s^*, V, \sigma)$ query to session s^* being incomplete such that

- σ is a valid signature on message $m = (V[, W, s^*_{actor}])$ with respect to the public key of $\hat{P}$, where W is the Diffie-Hellman exponential contained in message s^*_{sent}, and
- (V, σ) has never been output by user $\hat{P}$ in response to a send query.

Game 3. [Transition based on a small failure event] This game is the same as the previous one except that when a forgery event F with respect to the long-term public key of some user $\hat{P} \in \mathcal{P}$ occurs, the experiment halts and E outputs a random bit.

Analysis of game 3. The analysis of Game 3 proceeds in several steps. Consider first the following three cases.

1. If E issues a $\mathsf{corrupt}(\hat{P})$ query before the completion of session s^* and no origin-session exists for s^*, then session s^* does not satisfy the freshness predicate of the eCK-PFS model. This would have caused Game 2 to abort since session s^* would not be the test session. Recall that the test-session query can only be issued to a session that satisfies the freshness predicate of eCK-PFS at the time the query is issued. Hence this case can be excluded.
2. If $\hat{P}$ were adversary-controlled (i. e. $\hat{P} \notin \mathcal{P}$), then session s^* would not satisfy the freshness predicate of eCK-PFS and Game 2 would have aborted. Hence this case can be excluded as well.
3. If E does not issue a $\mathsf{corrupt}(\hat{P})$ query before the completion of session s^*, then he can only impersonate user $\hat{P}$ to session s^* by forging a signature on a message with respect to the long-term public key of $\hat{P}$.

Claim 1. We have $|P(S_2) - P(S_3)| \leq P(F)$.

Proof. If event F does not occur, then Game 2 and 3 proceed identically (i. e. $S_2 \wedge F^c \Leftrightarrow S_3 \wedge F^c$). The Difference Lemma yields that $|P(S_2) - P(S_3)| \leq P(F)$. □

Claim 2. If the deterministic signature scheme is SUF-CMA, then $P(F)$ is negligible. More precisely, $P(F) \leq N Adv_M^{Sign}(k)$, where $Adv_M^{Sign}(k)$ denotes the probability of a successful forgery.

Proof. Consider the following algorithm M using adversary E as a subroutine. M is given a public signature key pk and access to the corresponding signature oracle $\mathcal{O}^{Sign}$. It selects at random one of the N users and sets its public key to pk. We denote this user by $\hat{P}$ and its signature key pair by $(sk_{\hat{P}}, pk_{\hat{P}})$. Further, the algorithm M chooses signature key pairs (sk_i, pk_i) for all honest users $\hat{P}_i \in \mathcal{P}$ with $\hat{P}_i \neq \hat{P}$ and stores the associated secret keys. It also chooses key pairs (c_i, C_i) for all honest users $\hat{P}_i \in \mathcal{P}$ as needed for protocol π and stores the associated secret keys. Algorithm M proceeds as follows.

1. Run E on input 1^k and the public keys for all of the N users.
2. If E issues a $\mathsf{send}(z, \hat{Q})$ query to activate session z with peer $\hat{Q} \in \mathcal{P}$, then answer it as follows.

- If $z_{actor} \neq \hat{P}$, then choose $x \in_R \mathbb{Z}_p$ to get $X = g^x$, compute the signature σ on message $m = (X[, \hat{Q}])$ on behalf of z_{actor} and return the message (X, σ) to E.
- If $z_{actor} = \hat{P}$, then choose $x \in_R \mathbb{Z}_p$ to get $X = g^x$ and query the signature oracle on message $m = (X[, \hat{Q}])$ which returns the signature σ on message m. Store the pair (m, σ) in a table L, initially empty, and return the message (X, σ) to E.

3. If E issues a $\mathsf{send}(z, \hat{Q}, m)$ query to activate session z, then answer it as follows. First check whether message m is of the form (X, σ) for some $X \in G$ and σ a valid signature on message $(X[, z_{actor}])$ with respect to the public key of $\hat{Q}$. If the checks succeed, then:
   - If $z_{actor} \neq \hat{P}$, then choose $y \in_R \mathbb{Z}_p$ to get $Y = g^y$, compute the signature σ on message $m = (Y[, X, \hat{Q}])$ on behalf of z_{actor} and return the message (Y, σ) to E.
   - If $z_{actor} = \hat{P}$, then choose $y \in_R \mathbb{Z}_p$ to get $Y = g^y$ and query the signature oracle on message $m = (Y[, X, \hat{Q}])$ which returns the signature σ on message m. Store the pair (m, σ) in table L (initially empty) and return the message (Y, σ) to E.

   If one of the checks does not succeed, then abort session z.
4. If E issues a $\mathsf{send}(z, m)$ query to session z in role $\mathcal{I}$, then check whether message m is of the form (Y, σ) for some $Y \in G$ and σ a valid signature on message $(Y[, X, z_{actor}])$ with respect to the public key of z_{peer} (where $X \in G$ is contained in message z_{sent}). If the check fails, then abort session z.
5. If E makes a $\mathsf{send}(s^*, V, \sigma)$ query, where σ is a valid signature with respect to the public key $pk_{\hat{P}}$ of user $\hat{P}$ on message $m = (V[, W, s^*_{actor}])$ (where $W \in G$ is contained in s^*_{sent}), before the completion of the test session s^* and $(m, \sigma) \notin L$, then stop E and output (m, σ) as a forgery.
6. The queries $\mathsf{session\text{-}key}$ and $\mathsf{randomness}$ are answered in the appropriate way since M has chosen the randomness for all the sessions and the long-term secret keys for use in protocol π for all the users.
7. The queries $\mathsf{corrupt}(\hat{Q}_i)$, where $\hat{Q}_i \in \mathcal{P} \setminus \{\hat{P}\}$, are answered in the appropriate way since M knows the secret key pairs of the honest users in the set $\mathcal{P} \setminus \{\hat{P}\}$. In case $\hat{Q}_i \notin \mathcal{P}$, M returns $\perp$. In case $\hat{Q}_i = \hat{P}$, M aborts with failure.
8. If E issues the query $\mathsf{test\text{-}session}(s^*)$, then abort with failure.

Under event F, algorithm M is successful as described in Step 5 and the abortions as in Step 7 and 8 do not occur. The probability that E succeeds in forging a signature with respect to the public key of $\hat{P}$ is bounded above by the probability that M outputs a forgery multiplied by the number of honest users, that is, $P(F) \leq N Adv^{Sign}_M(k)$. □

Claim 3. Let $Adv^{\mathrm{SIG}(\pi),\mathrm{Game}\ 3,O}_E(k) := |2P(S_3|O) - 1|$, where O denotes the event that there is an origin-session for the test session. It holds that $Adv^{\mathrm{SIG}(\pi),\mathrm{Game}\ 3}_E(k)$ $= max(0, Adv^{\mathrm{SIG}(\pi),\mathrm{Game}\ 3,O}_E(k))$.

Proof. Note that $|2P(S_3|F) - 1| = |2\frac{1}{2} - 1| = 0$ (since, when event F occurs in Game 3, E outputs a random bit) and that if event F does not occur, then there exists an

origin-session for the test session. □

We next establish an upper bound for $Adv_E^{\mathrm{SIG}(\pi),\text{Game 3},O}(k)$ in terms of the security of protocol π.

Claim 4. Assume that in Game 3 there exists a unique[1] origin-session s for the test session s^* with $s_{actor} = s^*_{peer}$. If there is an efficient adversary E in eCK-PFS succeeding in Game 3 against protocol SIG(π) with non-negligible advantage, then we can construct an efficient adversary E' in $\mathsf{eCK}^{\mathsf{passive}}$ succeeding in Game 3 against protocol π with non-negligible advantage using adversary E as a subroutine. Moreover, it holds that $Adv_E^{\mathrm{SIG}(\pi),\text{Game 3},O}(k) \leq Adv_{E'}^{\pi,\text{Game 3}}(k)$.

Proof. Fix an efficient adversary E in eCK-PFS succeeding in Game 3 against protocol SIG(π) with non-negligible advantage. Let us construct an adversary E' in $\mathsf{eCK}^{\mathsf{passive}}$ succeeding in Game 3 against protocol π with non-negligible advantage using adversary E as a subroutine.
Algorithm E' chooses secret/public signature key pairs for all the users and stores the associated secret signature keys. It is given all public knowledge, such as identities and long-term public (non-signature) keys for all the users. Algorithm E' proceeds as follows.

1. Run E against SIG(π) on input 1^k and the public key pairs for all of the N users.
2. When E issues a corrupt($\hat{P}$) query to some user $\hat{P}$, E' issues that query to user $\hat{P}$ and returns the answer to that query together with the secret signature key of $\hat{P}$ (that E' has chosen) to E.
3. When E issues a randomness or a session-key query to some session z, E' issues that query to session z and returns the answer to E.
4. send queries are answered in the following way.
   - If E issues a send($z, \hat{Q}$) query to activate session z with peer $\hat{Q}$, then E' issues the same query to session z. The response is a message $W(\in G)$. Since E' knows the secret signature key of z_{actor}, it can sign the message $m = (W[, \hat{Q}])$ on its behalf and then return the message (W, σ) to E, where σ denotes the signature on m with respect to the public key of z_{actor}.
   - If E issues a send($z, \hat{Q}, m$) query to activate session z, where message m is of the form (W, σ), then E' first checks whether $W \in G$ and second whether σ is a valid signature on message $(W[, z_{actor}])$ with respect to the public key of $\hat{Q}$. If the checks succeed, then E' issues the query send(z, W) to session z. The response is a message $V \in G$. Since E' knows the secret signature key of z_{actor}, it can sign the message $m = (V[, W, \hat{Q}])$ on its behalf and then return the message (V, σ) to E, where σ denotes the signature on m with respect to the public key of z_{actor}.
   - If E issues a send(z, m) query, where message m is of the form (V, σ), then E' first checks whether $V \in G$ and second whether σ is a valid signature on message $(V[, W, z_{actor}])$ with respect to the public key of z_{peer}, where W

[1] No collision in the randomness occurs for SIG(π) (where $\pi \in \mathcal{DH}$-2) since otherwise Game 1 would have caused the game to abort.

is the Diffie-Hellman exponential contained in z_{sent}. If the checks succeed, then E' issues the query $\mathsf{send}(z, V)$ to session z.

If one of the checks fails, then session z is aborted (i. e. E' aborts session z).

5. In case E issues the test-session query to session s^*, E' issues the test-session query to session s^* and returns the answer to E.
6. At the end of E's execution (after it has output its guess b'), output b' as well.

Since by assumption there exists a unique origin-session for the test session, the test session satisfying the freshness predicate of eCK-PFS also satisfies the freshness predicate of eCK$^{\mathsf{passive}}$. Thus, it holds that

$$Adv_E^{\mathrm{SIG}(\pi),\text{Game 3},O}(k) \leq Adv_{E'}^{\pi,\text{Game 3}}(k).$$

Finally,

$$\begin{aligned} Adv_E^{\mathrm{SIG}(\pi)}(k) &\leq 2P(Coll_{\mathrm{SIG}(\pi)}) + 2q_s N Adv_M^{Sign}(k) + q_s Adv_E^{\mathrm{SIG}(\pi),\text{Game 3},O}(k) \\ &\leq 2P(Coll_{\mathrm{SIG}(\pi)}) + 2q_s N Adv_M^{Sign}(k) + q_s Adv_{E'}^{\pi,\text{Game 3}}(k) \end{aligned}$$

Since by assumption protocol π is secure in eCK$^{\mathsf{passive}}$, there is a negligible function g such that $Adv_{E'}^{\pi,\text{Game 3}}(k) \leq g(k)$, which completes the proof. □

□

We obtain the following corollary as a consequence of Theorem 1.

Corollary 1. *Let* Π *denote the class of two-message protocols. Under the assumption that the signature scheme is deterministic and SUF-CMA, the transformation* SIG : $\mathcal{DH}$*-2* $\rightarrow \Pi$ *is a security-strengthening protocol transformation from* eCK^w *to* eCK-PFS *according to Definition 20.*

Proof. The first condition of Definition 20 is satisfied by Proposition 2. We next verify the second requirement. Let $\pi \in \mathcal{DH}$-2 secure in eCK^w. Since by Proposition 5 we have eCK$^{\mathsf{passive}} \leq_{Sec}^{\Pi}$ eCK^w, it follows that protocol π is secure in eCK$^{\mathsf{passive}}$. By Theorem 1, SIG is a security strengthening protocol transformation from eCK$^{\mathsf{passive}}$ to eCK-PFS. Therefore, the transformed protocol SIG(π) is secure in eCK-PFS. □

Remark 2. Let eCK-NEK$^{\mathsf{passive}}$ and eCK-NEK-PFS be the security models obtained from eCK$^{\mathsf{passive}}$ and eCK-PFS (respectively) by removing the randomness query from the adversary's capabilities and related restrictions in the freshness definitions. Then it can be shown in a similar way as above that for any protocol $\pi \in \mathcal{DH}$-2 secure in eCK-NEK$^{\mathsf{passive}}$, the transformed protocol SIG(π) is secure in eCK-NEK-PFS using either a deterministic or a randomized SUF-CMA signature scheme. The same statement holds when replacing eCK-NEK$^{\mathsf{passive}}$ by wPFS and eCK-NEK-PFS by PFS.

Remark 3. Blake-Wilson and Menezes [22, p. 160] introduced the duplicate-signature key selection (DSKS) attack on signature schemes: after observing a user's signature σ on a message m, the adversary E is able to compute a signature key pair (sk_E, pk_E) (or sometimes just a verification key pk_E) such that σ is also E's signature on the message m. Now, the adversary in our setting can only register public keys at the

onset of the experiment W described in Section 3.1, i. e. before interacting with the users through queries. Thus, DSKS attacks, which exploit the adversary's ability to register a public key after observing signed messages, are not captured in our models. Note however that UKS attacks based on public-key re-registration (such as the ones on STS-MAC and STS-ENC [22, p. 159] as well as on KEA [69, p. 380]) are captured in our models $\mathsf{eCK}^{\mathsf{passive}}$, eCK^{w}, and eCK-PFS. Such UKS attacks can e. g. be prevented by making the session key derivation depend on the identifiers of actor and peer of the session.

3.2.4. Comparison of SIG to MAC

The $\mathcal{C}$ transformation by Boyd and González Nieto [31] transforms any two-message protocol that satisfies weak perfect forward secrecy into a two-message protocol that achieves perfect forward secrecy. In the transformed protocol, the exchanged messages are authenticated via message authentication codes (MACs) using a static Diffie-Hellman key. However, an adversary capable of actor compromise can compute the static Diffie-Hellman key used in the test session. Hence, in this setting, the MAC does not provide any message origin authentication. More precisely, an attacker can impersonate the peer of the test session by first revealing the long-term secret keys of the actor (which allows him to create valid MACs on messages of his choice), and, after the completion of the test session, revealing the long-term secret keys of the peer. Thus, the attacker can effectively perform a variant of Krawczyk's PFS attack. We next detail a concrete instance of this attack, showing that $\mathcal{C}$(NAXOS) [31] is insecure in eCK-PFS. We denote by $S = g^{a'b'}$ the shared static DH key between the users $\hat{A}$ and $\hat{B}$.

1. The adversary E first reveals the long-term secret keys of user $\hat{A}$ by issuing the query $\mathsf{corrupt}(\hat{A})$.
2. He then activates an initiator session s via the query $\mathsf{send}(s = (\hat{A}, i), \hat{B})$ and receives as a response the message $m = X, \mathrm{MAC}_S(\hat{A}, \hat{B}, X)$, where $X = g^{H_1(r_{\hat{A}}, a)}$ with $r_{\hat{A}}$ chosen uniformly at random from $\{0,1\}^k$ in session s.
3. E chooses $z \in_R \mathbb{Z}_p$, computes an ephemeral public key $Z = g^z$, and sends message $\tilde{m} = Z, \mathrm{MAC}_S(\hat{B}, \hat{A}, Z)$ to session s.
4. Upon receiving message $\tilde{m}$ in session s, $\hat{A}$ computes the session key $K_{\hat{A}} = H_2(Z^a, B^{H_1(r_{\hat{A}}, a)}, Z^{H_1(r_{\hat{A}}, a)}, \hat{A}, \hat{B})$ and accepts $K_{\hat{A}}$ as the session key.
5. Now, E chooses the completed session s as the test session, and reveals the long-term secret keys of user $\hat{B}$ via the query $\mathsf{corrupt}(\hat{B})$. This enables him to compute the session key of the test session as $K_E = H_2(A^z, X^b, X^z, \hat{A}, \hat{B})$.

Hence, in contrast to the SIG transformation, the transformation $\mathcal{C}|_{\mathcal{DH}\text{-}2} : \mathcal{DH}\text{-}2 \to \Pi$ is not a security-strengthening protocol transformation from eCK^{w} to eCK-PFS according to Definition 20, where Π denotes the class of two-message protocols.

3.3. Application of SIG to concrete protocols

In Section 3.3.1 we demonstrate that the NAXOS protocol is secure in eCK^{w} and construct a protocol secure in eCK-PFS using our SIG transformation. In Section 3.3.2

we show how to prove the security of the protocol π_1 in eCK-PFS by proving the much weaker protocol π_1-core secure in $\mathsf{eCK}^{\mathsf{passive}}$ and applying the SIG transformation to π_1-core.

3.3.1. NAXOS revisited

The NAXOS protocol [68], shown in Figure 3.4, provides an example of a protocol belonging to the class $\mathcal{DH}$-2, where $H_1 : \{0,1\}^* \to \mathbb{Z}_p$ and $H_2 : \{0,1\}^* \to \{0,1\}^k$ denote two hash functions and $r_{\hat{A}}, r_{\hat{B}} \in_R \{0,1\}^k$. In analogy to Figure 3.2, note that $f_{\mathcal{I}}(r_{\hat{A}}, a, \Omega) = H_1(r_{\hat{A}}, a)$, $f_{\mathcal{R}}(r_{\hat{B}}, b, \Omega) = H_1(r_{\hat{B}}, b)$, $F_{\mathcal{I}}(f_{\mathcal{I}}(r_{\hat{A}}, a, \Omega), a, Y, \Omega) = H_2(Y^a, B^{H_1(r_{\hat{A}},a)}, Y^{H_1(r_{\hat{A}},a)}, \hat{A}, \hat{B})$, and $F_{\mathcal{R}}(f_{\mathcal{R}}(r_{\hat{B}}, b, \Omega), b, X, \Omega) = H_2(A^{H_1(r_{\hat{B}},b)}, X^b, X^{H_1(r_{\hat{B}},b)}, \hat{A}, \hat{B})$.

Initiator $\mathcal{I}$		**Responder $\mathcal{R}$**
$\hat{A}$: (a, A)		$\hat{B}$: (b, B)
$x \leftarrow H_1(r_{\hat{A}}, a)$		
$X \leftarrow g^x$	$\xrightarrow{X}$	
		$y \leftarrow H_1(r_{\hat{B}}, b)$
		$Y \leftarrow g^y$
	$\xleftarrow{Y}$	$K_{\hat{B}} \leftarrow H_2(A^y, X^b, X^y, \hat{A}, \hat{B})$
$K_{\hat{A}} \leftarrow H_2(Y^a, B^x, Y^x, \hat{A}, \hat{B})$		

Figure 3.4.: NAXOS protocol [68]

The following proposition states that the NAXOS protocol is secure in eCK^w.

Proposition 7. *Under the Gap Diffie-Hellman assumption in the cyclic group G of prime order p,* NAXOS *is secure in the* eCK^w *model, when H_1 and H_2 are modeled as independent random oracles.*

In contrast to the proof of NAXOS in the eCK model [68], the proof of Proposition 7 distinguishes between the cases whether or not an origin-session (instead of a matching session) exists for the test session.

Proof. Here we show that NAXOS is secure in eCK^w. We use the structure of the security proof of the CMQV protocol in [104] as it is more detailed than the proof of NAXOS in [68].

Let the test session s^* be given by $T_{s^*} = (\hat{A}, \hat{B}, \mathcal{I}, X, Y)$. We first consider event K^c where the adversary M wins the security experiment against NAXOS (with non-negligible advantage) and does not query H_2 with $(\sigma_1, \sigma_2, \sigma_3, \hat{A}, \hat{B})$, where $\sigma_1 = \mathrm{CDH}(Y, A), \sigma_2 = \mathrm{CDH}(B, X)$ and $\sigma_3 = \mathrm{CDH}(X, Y)$.

Event K^c
If event K^c occurs, then the adversary M must have issued a session-key query to some session s such that $K_s = K_{s^*}$ (where K_s and K_{s^*} denote the session keys computed in sessions s and s^*, respectively) and s does not match s^*. We consider the following four events:

1. A_1 : there exist two sessions s_1, s_2 such that $r_{s_1} = r_{s_2}$ (where r_{s_1} and r_{s_2} denote the randomness used in sessions s_1 and s_2, respectively).
2. A_2 : there exists a session s such that $H_1(r_s, sk_{s_{actor}}) = H_1(r_{s^*}, sk_{s^*_{actor}})$ and $r_s \neq r_{s^*}$.
3. A_3 : there exists a session s' such that $H_2(\text{input}_{s'}) = H_2(\text{input}_{s^*})$ with $\text{input}_{s'} \neq \text{input}_{s^*}$.
4. A_4 : there exists an adversarial query input_M to the oracle H_2 such that $H_2(\text{input}_M) = H_2(\text{input}_{s^*})$ with $\text{input}_M \neq \text{input}_{s^*}$.

Analysis of event K^c

We denote by q_s an upper bound on the number of activated sessions by the adversary and by q_{ro2} an upper bound on the number of queries to the random oracle H_2. We have that

$$\begin{aligned} P(K^c) &\leq P(A_1 \vee A_2 \vee A_3 \vee A_4) \leq P(A_1) + P(A_2) + P(A_3) + P(A_4) \\ &\leq \frac{q_s^2}{2}\frac{1}{2^k} + \frac{q_s}{p} + \frac{q_s + q_{\text{ro2}}}{2^k}, \end{aligned}$$

which is a negligible function of the security parameter k.

In the subsequent events (and their analyses) we assume that no collisions in the queries to the oracle H_1 occur and that none of the events $A_1, ..., A_4$ occurs. Similar to [68, 104], we next consider the following three events:

1. $DL \wedge K$,
2. $T_O \wedge DL^c \wedge K$, and
3. $(T_O)^c \wedge DL^c \wedge K$, where

T_O denotes the event that there exists an origin-session for the test session, DL denotes the event where there exists a user $\hat{C} \in \mathcal{P}$ such that the adversary M, during its execution, queries H_1 with $(*, c)$ before issuing a $\mathsf{corrupt}(\hat{C})$ query and K denotes the event that M wins the security experiment against NAXOS by querying H_2 with $(\sigma_1, \sigma_2, \sigma_3, \hat{A}, \hat{B})$, where $\sigma_1 = \text{CDH}(Y, A), \sigma_2 = \text{CDH}(B, X)$ and $\sigma_3 = \text{CDH}(X, Y)$.

Note that we analyze the security of the NAXOS protocol in case the messages only contain the Diffie-Hellman exponentials.

Event $DL \wedge K$

This event is independent of the event that there exists an origin-session for the test session.

Let the input to the GDL challenge be C. Suppose that event $DL \wedge K$ occurs with non-negligible probability. In this case, the simulator S chooses one user $\hat{C} \in \mathcal{P}$ at random and sets its long-term public key to C. S chooses long-term secret/public key pairs for the remaining honest users and stores the associated long-term secret keys. Additionally S chooses a random value $m \in_R \{1, 2, ..., q_s\}$. We denote the m'th activated session by adversary M by s^*. Suppose further that $s^*_{actor} = \hat{A}, s^*_{peer} = \hat{B}$ and $s^*_{role} = \mathcal{I}$, w. l. o. g.. The simulation of M's environment proceeds as follows:

1. send queries are answered in the usual way. In case a session s is activated via a send query, S stores an entry of the form $(s, r_s, sk_{s_{actor}}, \kappa) \in (\mathcal{P} \times \mathbb{N}) \times \{0,1\}^k \times (\mathbb{Z}_p \cup \{*\}) \times \mathbb{Z}_p$ in a table Q, initially empty, (unless ephemeral public key validation on the received element fails in which case the session is aborted).

When computing the (outgoing) Diffie-Hellman exponential of session s, S does the following:

- S chooses $r_s \in_R \{0,1\}^k$ (i. e. the randomness of session s),
- S chooses $\kappa \in_R \mathbb{Z}_p$,
- if $s_{actor} \neq \hat{C}$, then S stores the entry $(s, r_s, sk_{s_{actor}}, \kappa)$ in Q, else S stores the entry $(s, r_s, *, \kappa)$ in Q,[2] and
- S returns the Diffie-Hellman exponential g^κ to M.

2. S stores entries of the form $\left(\hat{Q}_i, \hat{Q}_j, r, U, V, \lambda\right) \in \mathcal{P} \times \{0,1\}^* \times \{\mathcal{I}, \mathcal{R}\} \times G \times G \times \{0,1\}^k$ in a table T, initially empty. Upon completion of session s with $T_s = \left(\hat{Q}_i, \hat{Q}_j, \mathcal{I}, U, V\right)$, S does the following:
   - If there exists an entry $\left(\hat{Q}_j, \hat{Q}_i, \mathcal{R}, V, U, \lambda\right)$ in table T, then S stores $\left(\hat{Q}_i, \hat{Q}_j, \mathcal{I}, U, V, \lambda\right)$ in table T.
   - Else if there exists an entry $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j, \lambda\right)$ in table L, for some $\lambda \in \{0,1\}^k$, such that $\mathrm{DDH}(V, U, \sigma_3) = 1$, $\mathrm{DDH}(U, Q_j, \sigma_2) = 1$ and
     - $V^{sk_{\hat{Q}_i}} = \sigma_1$ (in case $\hat{Q}_i \neq \hat{C}$) or $\mathrm{DDH}(V, Q_i, \sigma_1) = 1$ (in case $\hat{Q}_i = \hat{C}$),

     then S stores $\left(\hat{Q}_i, \hat{Q}_j, \mathcal{I}, U, V, \lambda\right)$ in table T.
   - Else, S chooses $\mu \in_R \{0,1\}^k$ and stores the entry $\left(\hat{Q}_i, \hat{Q}_j, \mathcal{I}, U, V, \mu\right)$ in T.

   The session key of a completed session s with $T_s = \left(\hat{Q}_j, \hat{Q}_i, \mathcal{R}, V, U\right)$ is determined and stored similarly.
3. $\mathsf{randomness}(s)$: S answers this query in the appropriate way.
4. $\mathsf{session\text{-}key}(s)$: S answers this query by look-up in table T.
5. $\mathsf{test\text{-}session}(s)$: If $s \neq s^*$, then S aborts; otherwise S answers the query in the appropriate way.
6. $\mathsf{corrupt}(\hat{P})$: S answers this query in the appropriate way, except if $\hat{P} = \hat{C}$ in which case S aborts with failure.
7. S stores entries of the form $(r, h, \kappa) \in \{0,1\}^k \times \mathbb{Z}_p \times \mathbb{Z}_p$ in a table J, initially empty. When M makes a query of the form (r, h) to the random oracle for H_1, answer it as follows:
   - If $C = g^h$, then S aborts M and is successful by outputting $\mathrm{DLog}_g(C) = h$.
   - Else if $(r, h, \kappa) \in J$ for some $\kappa \in \mathbb{Z}_p$, then S returns κ to M.
   - Else if there exists an entry $(s, r_s, sk_{s_{actor}}, \kappa)$ in Q, for some $s \in \mathcal{P} \times \mathbb{N}, r_s \in \{0,1\}^k, sk_{s_{actor}} \in \mathbb{Z}_p$ and $\kappa \in \mathbb{Z}_p$, such that $r_s = r$ and $sk_{s_{actor}} = h$, then S returns κ to M and stores the entry (r, h, κ) in table J.
   - Else, S chooses $\kappa \in_R \mathbb{Z}_p$, returns it to M and stores the entry (r, h, κ) in J.
8. S stores entries of the form $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j, \lambda\right) \in G \times G \times G \times \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^k$ in a table L, initially empty. When M makes a query of the form $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j\right)$ to the random oracle for H_2, answer it as follows:
   - If $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j, \lambda\right) \in L$ for some $\lambda \in \{0,1\}^k$, then S returns λ to M.

[2] We do not need to keep consistency with H_1 queries via lookup in table J since the probability that the adversary guesses the random data of a session is negligible.

- Else if there exist entries $\left(\hat{Q}_i, \hat{Q}_j, \mathcal{I}, U, V, \lambda\right)$ or $\left(\hat{Q}_j, \hat{Q}_i, \mathcal{R}, V, U, \lambda\right)$ in table T, for some $\lambda \in \{0,1\}^k$ and $U, V \in G$, such that $\mathrm{DDH}(V, U, \sigma_3) = 1$, $\mathrm{DDH}(V, Q_i, \sigma_1) = 1$ and $\mathrm{DDH}(U, Q_j, \sigma_2) = 1$, then S returns λ to M and stores the entry $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j, \lambda\right)$ in table L.
- Else, S chooses $\mu \in_R \{0,1\}^k$, returns it to M and stores the entry $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j, \mu\right)$ in L.

9. M outputs a guess: S aborts with failure.

Analysis of event $DL \wedge K$
S's simulation of M's environment is perfect except with negligible probability. The probability that M selects s^* as the test session is at least $\frac{1}{q_s}$. Assuming that this is indeed the case, S does not abort in Step 5. With probability at least $\frac{1}{N}$, S assigns the public key C to a user $\hat{C}$ for whom M queries H_1 with $(*, h)$ such that $C = g^h$ before issuing a $\mathsf{corrupt}(\hat{C})$ query. In this case, S is successful as described in Step 7 and does not abort in Steps 6 and 9. Hence, if event $DL \wedge K$ occurs, then the success probability of S is given by $P(S) \geq \frac{1}{Nq_s} P(DL \wedge K)$.

Event $T_O \wedge DL^c \wedge K$
Let s^* and s' denote the test session and the origin-session for the test session, respectively. We split event $\mathrm{Ev} := T_O \wedge DL^c \wedge K$ into the following events $B_1, ..., B_3$ so that $\mathrm{Ev} = B_1 \vee B_2 \vee B_3$:

1. B_1 : Ev occurs and $s^*_{peer} = s'_{actor}$.
2. B_2 : Ev occurs and $s^*_{peer} \neq s'_{actor}$ and M does not issue a $\mathsf{randomness}(s')$ query to the origin-session s' of s^*, but may issue a $\mathsf{corrupt}(s^*_{peer})$ query.
3. B_3 : Ev occurs and $s^*_{peer} \neq s'_{actor}$ and M does not issue a $\mathsf{corrupt}(s^*_{peer})$ query, but may issue a $\mathsf{randomness}(s')$ query to the origin-session s' of s^*.

Event B_1
Let the input to the GDH challenge be (X_0, Y_0). Suppose that event B_1 occurs with non-negligible probability. In this case S chooses long-term secret/public key pairs for all the honest users and stores the associated long-term secret keys. Additionally S chooses two random values $m, n \in_R \{1, 2, ..., q_s\}$. The m'th activated session by adversary M will be called s^* and the n'th activated session will be called s'. The randomness of session s^* is denoted by $\tilde{x}_0$ and the randomness of session s' is denoted by $\tilde{y}_0$. Suppose further that $s^*_{actor} = \hat{A}, s^*_{peer} = \hat{B}$ and $s^*_{role} = \mathcal{I}$, w. l. o. g.. The simulation of M's environment proceeds as follows:

1. $\mathsf{send}(s^*, \hat{B})$: S sets the ephemeral public key X to X_0 and answers the query with message X_0.
2. $\mathsf{send}(s^*, Y_0)$: S proceeds with Step 7.
3. $\mathsf{send}(s', \hat{P})$: S sets the ephemeral public key Y to Y_0 and answers the query with message Y_0.
4. $\mathsf{send}(s', \hat{P}, Z)$: S checks whether $Z \in G$, sets the ephemeral public key Y to Y_0, answers the query with message Y_0 and proceeds with Step 7. If the check fails, session s' is aborted.
5. $\mathsf{send}(s', Z)$: S proceeds with Step 7.

6. Other send queries are answered in the usual way.[3]
7. S stores entries of the form $\left(\hat{Q}_i, \hat{Q}_j, r, U, V, \lambda\right) \in \mathcal{P} \times \{0,1\}^* \times \{\mathcal{I}, \mathcal{R}\} \times G \times G \times \{0,1\}^k$ in a table T, initially empty. Upon completion of session s with $T_s = \left(\hat{Q}_i, \hat{Q}_j, \mathcal{I}, U, V\right)$, S does the following:
   - If there exists an entry $\left(\hat{Q}_j, \hat{Q}_i, \mathcal{R}, V, U, \lambda\right)$ in table T, then S stores $\left(\hat{Q}_i, \hat{Q}_j, \mathcal{I}, U, V, \lambda\right)$ in table T.
   - Else if there exists an entry $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j, \lambda\right)$ in table L, for some $\lambda \in \{0,1\}^k$, such that $V^{sk_{\hat{Q}_i}} = \sigma_1$, $\mathrm{DDH}(U, Q_j, \sigma_2) = 1$ and $\mathrm{DDH}(V, U, \sigma_3) = 1$, then S stores $\left(\hat{Q}_i, \hat{Q}_j, \mathcal{I}, U, V, \lambda\right)$ in table T.
   - Else, S chooses $\mu \in_R \{0,1\}^k$, and stores the entry $\left(\hat{Q}_i, \hat{Q}_j, \mathcal{I}, U, V, \mu\right)$ in T.

   The session key of a completed session s with $T_s = \left(\hat{Q}_j, \hat{Q}_i, \mathcal{R}, V, U\right)$ is determined and stored similarly.
8. randomness(s): S answers this query in the appropriate way.
9. session-key(s): S answers this query by look-up in table T.
10. test-session(s): If $s \neq s^*$ or if s' is not the origin-session for session s^*, then S aborts; otherwise S answers the query in the appropriate way.
11. $H_1(r_{\hat{C}}, c)$: S simulates a random oracle in the usual way except if $\hat{C} = \hat{A}$ (i.e. $c = a$) and $r_{\hat{C}} = \tilde{x}_0$ or if $\hat{C} = \hat{B}$ (i.e. $c = b$) and $r_{\hat{C}} = \tilde{y}_0$, in which case S aborts with failure.
12. corrupt($\hat{P}$): S answers this query in the appropriate way.
13. S stores entries of the form $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j, \lambda\right) \in G \times G \times G \times \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^k$ in a table L, initially empty. When M makes a query of the form $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j\right)$ to the random oracle for H_2, answer it as follows:
   - If $\left\{\hat{Q}_i, \hat{Q}_j\right\} = \left\{\hat{A}, \hat{B}\right\}$, $\sigma_1 = Y_0^a$, $\sigma_2 = X_0^b$ and $\mathrm{DDH}(X_0, Y_0, \sigma_3) = 1$, then S aborts M and is successful by outputting $\mathrm{CDH}(X_0, Y_0) = \sigma_3$.
   - Else if $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j, \lambda\right) \in L$ for some $\lambda \in \{0,1\}^k$, then S returns λ to M.
   - Else if there exist entries $\left(\hat{Q}_i, \hat{Q}_j, \mathcal{I}, U, V, \lambda\right)$ or $\left(\hat{Q}_j, \hat{Q}_i, \mathcal{R}, V, U, \lambda\right)$, for some $\lambda \in \{0,1\}^k$ and $U, V \in G$, such that $\mathrm{DDH}(V, Q_i, \sigma_1) = 1$, $\mathrm{DDH}(U, Q_j, \sigma_2) = 1$ and $\mathrm{DDH}(V, U, \sigma_3) = 1$ in table T, then S returns λ to M and stores the entry $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j, \lambda\right)$ in table L.
   - Else, S chooses $\mu \in_R \{0,1\}^k$, returns it to M and stores the entry $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j, \mu\right)$ in L.
14. M outputs a guess: S aborts with failure.

Analysis of event B_1
S's simulation of M's environment is perfect except with negligible probability. The probability that M selects s^* as the test session and s' as the origin-session for the

[3]Note that, if the group membership test fails, then the session is aborted.

test session is at least $\frac{1}{q_s^2}$. Assuming that this is indeed the case, S does not abort in Step 10. Recall that $T_{s^*} = (\hat{A}, \hat{B}, \mathcal{I}, X_0, Y_0)$. Since $\tilde{x}_0$ is used only in the test session, M can only obtain it via a $\mathsf{randomness}(s^*)$ query before making an H_1 query that includes $\tilde{x}_0$. Similarly, M can only obtain $\tilde{y}_0$ via a $\mathsf{randomness}(s')$ query on the origin-session s' before making an H_1 query that includes $\tilde{y}_0$. Under event DL^c, the adversary first issues a $\mathsf{corrupt}(\hat{P})$ query to user $\hat{P}$ before making an H_1 query that involves the long-term secret key of user $\hat{P}$. Freshness of the test session guarantees that the adversary can reveal at most one value in each of the pairs $(\tilde{x}_0, a)$ and $(\tilde{y}_0, b)$; hence S does not abort in Step 11. Under event K, except with negligible probability of guessing $\mathrm{CDH}(X_0, Y_0)$, S is successful as described in the first case of Step 13 and does not abort as in Step 14. Hence, if event B_1 occurs, then the success probability of S is given by $P(S) \geq \frac{1}{q_s^2} P(B_1)$.

Event B_2
Let the input to the GDH challenge be (X_0, Y_0). Suppose that event B_2 occurs with non-negligible probability. The simulation of S proceeds in a similar way as for event B_1. Steps 8 and 11 need to be replaced by the following:

- $\mathsf{randomness}(s)$: S answers this query in the appropriate way, except if $s = s'$ in which case S aborts with failure.
- $H_1(r_{\hat{C}}, c)$: S simulates a random oracle in the usual way except if $\hat{C} = \hat{A}$ (i.e. $c = a$) and $r_{\hat{C}} = \tilde{x}_0$, in which case S aborts with failure.

Analysis of event B_2
S's simulation of M's environment is perfect except with negligible probability. The probability that M selects s^* as the test session and s' as the origin-session for the test session is $\frac{1}{q_s^2}$. Recall that $T_{s^*} = (\hat{A}, \hat{B}, \mathcal{I}, X_0, Y_0)$. Since $\tilde{x}_0$ is used only in the test session, M can only obtain it via a $\mathsf{randomness}(s^*)$ query before making an H_1 query that includes $\tilde{x}_0$. Under event DL^c, the adversary first issues a $\mathsf{corrupt}(\hat{P})$ query to user $\hat{P}$ before making an H_1 query that involves the long-term secret key of user $\hat{P}$. Freshness of the test session guarantees that the adversary can reveal at most one value of the pair $(\tilde{x}_0, a)$. Under event B_2 the simulation does not fail as in Step 8. Under event K, except with negligible probability of guessing $\mathrm{CDH}(X_0, Y_0)$, S is successful as described in the first case of Step 13 and does not abort as in Step 14. Hence, if event B_2 occurs, then the success probability of S is given by $P(S) \geq \frac{1}{q_s^2} P(B_2)$.

Event B_3
Let the input to the GDH challenge be (X_0, B). Suppose that event B_3 occurs with non-negligible probability. In this case, S chooses one user $\hat{B} \in \mathcal{P}$ at random and sets its long-term public key to B. S chooses long-term secret/public key pairs for the remaining users in $\mathcal{P}$ and stores the associated long-term secret keys. Additionally S chooses two random values $m, n \in_R \{1, 2, ..., q_s\}$. We denote the m'th activated session by adversary M by s^* and the n'th activated session by s'. The randomness of session s^* is denoted by $\tilde{x}_0$. Suppose further that $s^*_{actor} = \hat{A}, s^*_{peer} = \hat{B}$ and $s^*_{role} = \mathcal{I}$, w. l. o. g.. The simulation of M's environment proceeds as follows:

1. $\mathsf{send}(s^*, \hat{B})$: S sets the ephemeral public key X to X_0 and answers the query with message X_0.

2. $\mathsf{send}(s^*, Z)$: S proceeds with Step 4.
3. Other send queries are answered as for event $DL \wedge K$.
4. S stores entries of the form $\left(\hat{Q}_i, \hat{Q}_j, r, U, V, \lambda\right) \in \mathcal{P} \times \{0,1\}^* \times \{\mathcal{I}, \mathcal{R}\} \times G \times G \times \{0,1\}^k$ in a table T, initially empty. Upon completion of session s with $T_s = \left(\hat{Q}_i, \hat{Q}_j, \mathcal{I}, U, V\right)$, S proceeds as for event $DL \wedge K$ (see above).
5. $\mathsf{randomness}(s)$: S answers this query in the appropriate way.
6. $\mathsf{session\text{-}key}(s)$: S answers this query by look-up in table T.
7. $\mathsf{test\text{-}session}(s)$: If $s \neq s^*$ or if s' is not the origin-session for session s^*, then S aborts; otherwise S answers the query in the appropriate way.
8. $H_1(r_{\hat{C}}, c)$: S simulates a random oracle in the usual way except if $\hat{C} = \hat{A}$ (i.e. $c = a$) and $r_{\hat{C}} = \tilde{x}_0$, in which case S aborts with failure.
9. $\mathsf{corrupt}(\hat{P})$: S answers this query in the appropriate way, except if $\hat{P} = \hat{B}$ in which case S aborts with failure.
10. S stores entries of the form $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j, \lambda\right) \in G \times G \times G \times \mathcal{P} \times \mathcal{P} \times \{0,1\}^k$ in a table L, initially empty. When M makes a query of the form $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j\right)$ to the random oracle for H_2, answer it as follows:
    - If $\left\{\hat{Q}_i, \hat{Q}_j\right\} = \left\{\hat{A}, \hat{B}\right\}$, $\sigma_1 = A^{H_1(r_{s'}, sk_{s'_{actor}})}$, $\mathrm{DDH}(X_0, B, \sigma_2) = 1$, and $\sigma_3 = X_0^{H_1(r_{s'}, sk_{s'_{actor}})}$, then S aborts M and is successful by outputting $\mathrm{CDH}(X_0, B) = \sigma_2$.
    - Else if $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j, \lambda\right) \in L$ for some $\lambda \in \{0,1\}^k$, then S returns λ to M.
    - Else if there exist entries $\left(\hat{Q}_i, \hat{Q}_j, \mathcal{I}, U, V, \lambda\right)$ or $\left(\hat{Q}_j, \hat{Q}_i, \mathcal{R}, V, U, \lambda\right)$ in table T, for some $\lambda \in \{0,1\}^k$ and $U, V \in G$, such that $\mathrm{DDH}(V, U, \sigma_3) = 1$, $\mathrm{DDH}(V, Q_i, \sigma_1) = 1$ and $\mathrm{DDH}(U, Q_j, \sigma_2) = 1$, then S returns λ to M and stores the entry $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j, \lambda\right)$ in table L.
    - Else, S chooses $\mu \in_R \{0,1\}^k$, returns it to M and stores the entry $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j, \mu\right)$ in L.
11. M outputs a guess: S aborts with failure.

Analysis of event B_3
S's simulation of M's environment is perfect except with negligible probability. The probability that M selects s^* as the test session and s' as its origin-session is at least $\frac{1}{q_s^2}$. Assuming that this is indeed the case, S does not abort in Step 7. With probability $\frac{1}{N}$, S assigns the public key B to the peer of the test session $\hat{B}$. Under event B_3, M does not issue a $\mathsf{corrupt}(\hat{B})$ query, and so S does not abort in Step 9. Similarly, S does not abort in Step 11 and is successful as described in Step 10. Hence, if event B_3 occurs, then the success probability of S is given by $P(S) \geq \frac{1}{Nq_s^2} P(B_3)$.

Event $(T_O)^c \wedge DL^c \wedge K$
If there is no origin-session for the test session, then there is also no matching session for the test session. Hence $((T_O)^c \wedge DL^c \wedge K) \subseteq ((T_M)^c \wedge DL^c \wedge K)$ (where T_M denotes the event that there exists a matching session for the test session) which implies that

event $(T_O)^c \wedge DL^c \wedge K$ is covered in the analysis of event $(T_M)^c \wedge DL^c \wedge K$ for which we refer the reader to [67, 68]. Note that, similar to the simulation related to Event B_3,

- S checks whether there is a query $(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j)$ by M to H_2 such that $\left\{\hat{Q}_i, \hat{Q}_j\right\} = \left\{\hat{A}, \hat{B}\right\}$, $\mathrm{DDH}(A, Y, \sigma_1) = 1, \mathrm{DDH}(X_0, B, \sigma_2) = 1$, and $\mathrm{DDH}(X_0, Y, \sigma_3) = 1$ (assuming that the test session s^* is given by $T_{s^*} = (\hat{A}, \hat{B}, \mathcal{I}, X_0, Y)$ to solve the GDH instance (X_0, B), and
- S keeps consistency between session-key and H_2 queries as well as between send and H_1 queries.

□

Combining Proposition 7 with Theorem 1, we obtain the following result.

Corollary 2. *Under the Gap Diffie-Hellman assumption in the cyclic group G of prime order p, using a deterministic SUF-CMA signature scheme, the* SIG(NAXOS) *protocol shown in Figure 3.5 is secure in the* eCK-PFS *model, when H_1, H_2 are modeled as independent random oracles.*

Initiator $\mathcal{I}$		**Responder $\mathcal{R}$**
$\hat{A}$: $(a, A), (sk_{\hat{A}}, pk_{\hat{A}})$		$\hat{B}$: $(b, B), (sk_{\hat{B}}, pk_{\hat{B}})$
$x \leftarrow H_1(r_{\hat{A}}, a)$ $X \leftarrow g^x$		
$\sigma_{\hat{A}} \leftarrow Sign_{sk_{\hat{A}}}(X \left[, \hat{B}\right])$	$\xrightarrow{X, \sigma_{\hat{A}}}$	
		$y \leftarrow H_1(r_{\hat{B}}, b)$ $Y \leftarrow g^y$ $\sigma_{\hat{B}} \leftarrow Sign_{sk_{\hat{B}}}(Y \left[, X, \hat{A}\right])$
	$\xleftarrow{Y, \sigma_{\hat{B}}}$	$K_{\hat{B}} \leftarrow H_2(A^y, X^b, X^y, \hat{A}, \hat{B})$
$K_{\hat{A}} \leftarrow H_2(Y^a, B^x, Y^x, \hat{A}, \hat{B})$		

Figure 3.5.: SIG(NAXOS) protocol

3.3.2. Proving π_1 secure in eCK-PFS via π_1-core

Figure 3.6 shows the protocol π_1-core $\in \mathcal{DH}$-2, where $H : \{0,1\}^* \to \{0,1\}^k$ denotes a hash function and $x, y \in_R \mathbb{Z}_p$. As observed in [65], the π_1-core protocol is insecure with respect to an active adversary. The adversary can impersonate $\hat{B}$ to $\hat{A}$ by simply sending the message $B^{-1}Z$ (where $Z = g^i$ for some $i \in \mathbb{Z}_p$) to $\hat{A}$. $\hat{A}$ computes the secret value that is used to derive the session key as $(BB^{-1}Z)^{x+a} = Z^{x+a}$. The latter value can be easily computed by the adversary. This attack shows that π_1-core is insecure in eCK^w.

However, even though π_1-core is insecure in eCK^w, it can be proven secure in the weaker $\mathsf{eCK}^{\mathsf{passive}}$ model, as the following proposition shows.

Initiator $\mathcal{I}$		**Responder** $\mathcal{R}$
$\hat{A}$: (a, A)		$\hat{B}$: (b, B)
$x \leftarrow r_{\hat{A}}$		
$X \leftarrow g^x$	$\xrightarrow{X}$	
		$y \leftarrow r_{\hat{B}}$
		$Y \leftarrow g^y$
	$\xleftarrow{Y}$	$K_{\hat{B}} \leftarrow H(\hat{A}, \hat{B}, (XA)^{y+b}, X)$
$K_{\hat{A}} \leftarrow H(\hat{A}, \hat{B}, (YB)^{x+a}, X)$		

Figure 3.6.: Protocol π_1-core

Proposition 8. *Under the Gap Diffie-Hellman assumption in the cyclic group G of prime order p, the protocol π_1-core is secure in the* $\mathsf{eCK}^{\mathsf{passive}}$ *model, when H is modeled as a random oracle.*

Proof. Let the test session s^* be given by $T_{s^*} = (\hat{A}, \hat{B}, \mathcal{I}, X, Y)$. We first consider event K^c where the adversary M wins the security experiment against π_1-core (with non-negligible advantage) and does not query H with $(\hat{A}, \hat{B}, \sigma, X)$, where $\sigma = \mathrm{CDH}(YB, XA)$.

Event K^c
If event K^c occurs, then the adversary M must have issued a session-key query to some session s such that $K_s = K_{s^*}$ (where K_s and K_{s^*} denote the session keys computed in sessions s and s^*, respectively) and s does not match s^*. We consider the following three events:

1. A_1 : there exist two sessions s_1, s_2 such that $r_{s_1} = r_{s_2}$ (where r_{s_1} and r_{s_2} denote the randomness used in sessions s_1 and s_2, respectively). Note that A_1 includes the event where there exists a session s with $T_s = T_{s^*}$ as well as the event where two sessions use the same randomness (possibly leading to randomness queries).
2. A_2 : there exists a session s such that $H(\mathrm{input}_s) = H(\mathrm{input}_{s^*})$ with $\mathrm{input}_s \neq \mathrm{input}_{s^*}$.
3. A_3 : there exists an adversarial query input_M to the oracle H such that $H(\mathrm{input}_M) = H(\mathrm{input}_{s^*})$ with $\mathrm{input}_M \neq \mathrm{input}_{s^*}$.

Analysis of event K^c
We denote by q_s an upper bound on the number of activated sessions by the adversary and by q_{ro} an upper bound on the number of queries to the random oracle H. We have that

$$\begin{aligned} P(K^c) &\leq P(A_1 \vee A_2 \vee A_3) \leq P(A_1) + P(A_2) + P(A_3) \\ &\leq \frac{q_s^2}{2p} + \frac{q_s + q_{\mathrm{ro}}}{2^k}, \end{aligned}$$

which is a negligible function of the security parameter k.

In the subsequent events (and their analyses) we assume that none of the events

$A_1, ..., A_3$ occurs. We consider the following event:

$$T_O \wedge K, \text{ where}$$

T_O denotes the event that there exists an origin-session for the test session, and K denotes the event that M wins the security experiment against π_1-core by querying H with $(\hat{A}, \hat{B}, \sigma, X)$, where $\sigma = \text{CDH}(YB, XA)$. Recall that in case there is no origin-session for the test session, the test session does not satisfy the freshness predicate of $\mathsf{eCK}^{\mathsf{passive}}$.

Event $T_O \wedge K$
Let s^* and s' denote the test session and the origin-session for the test session, respectively. We split event Ev $:= T_O \wedge K$ into the following events $B_1, ..., B_4$ so that Ev $= B_1 \vee B_2 \vee B_3 \vee B_4$:

1. B_1 : Ev occurs and the adversary does issue neither $\mathsf{randomness}(s')$ nor $\mathsf{randomness}(s^*)$, but may issue the queries $\mathsf{corrupt}(s^*_{actor})$ and $\mathsf{corrupt}(s^*_{peer})$.
2. B_2 : Ev occurs and the adversary does issue neither $\mathsf{randomness}(s^*)$ nor $\mathsf{corrupt}(s^*_{peer})$, but may issue the queries $\mathsf{corrupt}(s^*_{actor})$ and $\mathsf{randomness}(s')$.
3. B_3 : Ev occurs and the adversary does issue neither $\mathsf{randomness}(s')$ nor $\mathsf{corrupt}(s^*_{actor})$, but may issue the queries $\mathsf{corrupt}(s^*_{peer})$ and $\mathsf{randomness}(s^*)$.
4. B_4 : Ev occurs and the adversary does issue neither $\mathsf{corrupt}(s^*_{actor})$ nor $\mathsf{corrupt}(s^*_{peer})$, but may issue the queries $\mathsf{randomness}(s')$ and $\mathsf{randomness}(s^*)$.

Event B_1
We denote by X, Y the ephemeral public keys sent, received during the test session s^*. Revealing the long-term secret keys of both s^*_{actor} and s^*_{peer}, the adversary E could distinguish the session key of the test session from a random key by computing $\text{CDH}(X, Y) = g^{xy}$ (where $X = g^x$ and $Y = g^y$) since

$$g^{xy} = (YB)^{x+a} Y^{-a} X^{-b} B^{-a}.$$

We solve the Gap Diffie-Hellman problem with probability $\frac{1}{(q_s)^2} P(Q)$ where $P(Q)$ must be negligible since the Gap Diffie-Hellman problem is hard in G.

Consider the following algorithm C which uses adversary E as a subroutine. Algorithm C is given a pair (X, Y) of elements from G as an instance of the Gap Diffie-Hellman problem. The algorithm randomly selects a session number n from $\{1, ..., q_s\}$ which reflects the guess that the n-th activated session, say session s', is the origin-session for session s^*. C chooses long-term public keys for all users and stores the associated secret keys. Algorithm C proceeds as follows.

1. Run E on input 1^k and the public keys for all of the N users.
2. $\mathsf{send}(s^*, \hat{B})$: C sets the ephemeral public key to X and answers the query with the message X.
3. $\mathsf{send}(s', \hat{P})$ or $\mathsf{send}(s', \hat{P}, Z)$: C sets the ephemeral public key to Y and answers the query with the message Y.
4. Other send queries are answered in the usual way (note that, if the group check fails, the session is aborted).
5. $\mathsf{randomness}(s)$: C answers in the appropriate way, except if $s = s'$ or $s = s^*$ in which cases C aborts with failure.

6. corrupt($\hat{P}$): C answers in the appropriate way.
7. test-session(s): If $s \neq s^*$ or if s' is not the origin-session for session s^*, then C aborts; otherwise C answers the query in the appropriate way.
8. Store entries of the form $\left(\hat{Q}_i, \hat{Q}_j, Z, U, \lambda\right) \in \{0,1\}^* \times \{0,1\}^* \times G \times G \times \{0,1\}^k$ in a table L, initially empty. When E makes a query of the form $\left(\hat{Q}_i, \hat{Q}_j, Z, U\right)$ to the random oracle for H, answer it as follows:
   - If $\left\{\hat{Q}_i, \hat{Q}_j\right\} = \left\{\hat{A}, \hat{B}\right\}$, $U = X$ and $\mathrm{DDH}(XA, YB, Z) = 1$, then C aborts E and is successful by outputting $\mathrm{CDH}(X, Y) = ZY^{-a}X^{-b}B^{-a}$.
   - Else if $\left(\hat{Q}_i, \hat{Q}_j, Z, U, \lambda\right) \in L$ for some $\lambda \in \{0,1\}^k$, then C returns λ to E.
   - Else if there exist entries $\left(\hat{Q}_i, \hat{Q}_j, \mathcal{I}, U, V, \lambda\right)$ or $\left(\hat{Q}_j, \hat{Q}_i, \mathcal{R}, V, U, \lambda\right)$, for some $\lambda \in \{0,1\}^k$ and $V \in G$, such that $\mathrm{DDH}(VP_j, UP_i, Z) = 1$ in table T, then C returns λ to E and stores the entry $\left(\hat{Q}_i, \hat{Q}_j, Z, U, \lambda\right)$ in table L.
   - Else, C chooses $\mu \in_R \{0,1\}^k$, returns it to E and stores the entry $\left(\hat{Q}_i, \hat{Q}_j, Z, U, \mu\right)$ in L.
9. Store entries of the form $\left(\hat{Q}_i, \hat{Q}_j, r, U, V, \lambda\right) \in \mathcal{P} \times \{0,1\}^* \times \{\mathcal{I}, \mathcal{R}\} \times G \times G \times \{0,1\}^k$ in a table T, initially empty. Upon completion of session s with $T_s = \left(\hat{Q}_i, \hat{Q}_j, \mathcal{I}, U, V\right)$, C proceeds as follows:
   - If there exists an entry $\left(\hat{Q}_j, \hat{Q}_i, \mathcal{R}, V, U, \lambda\right)$ in table T, then C stores $\left(\hat{Q}_i, \hat{Q}_j, \mathcal{I}, U, V, \lambda\right)$ in table T.
   - Else if there exists an entry $\left(\hat{Q}_i, \hat{Q}_j, Z, U, \lambda\right)$ in table L, for some $\lambda \in \{0,1\}^k$, such that $\mathrm{DDH}(UP_i, VP_j, Z) = 1$, then C stores $\left(\hat{Q}_i, \hat{Q}_j, \mathcal{I}, U, V, \lambda\right)$ in table T.
   - Else, C chooses $\mu \in_R \{0,1\}^k$ and stores the entry $\left(\hat{Q}_i, \hat{Q}_j, \mathcal{I}, U, V, \mu\right)$ in T.

   The session key of a completed session s with $T_s = \left(\hat{Q}_j, \hat{Q}_i, \mathcal{R}, V, U\right)$ is determined and stored similarly.
10. session-key(s): C answers this query by look-up in table T.
11. E outputs a guess: C aborts with failure.

Analysis of event B_1
The probability that E selects s^* as the test session and s' as the origin-session for the test session is at least $\frac{1}{(q_s)^2}$. Assume that this is indeed the case. Then C does not abort as in Step 7. Under event B_1 the simulation does not fail as in Step 5. Under event Q, C is successful as described in the first case of Step 8 and does not abort as in Step 11. C correctly computes the Gap Diffie-Hellman instance with probability at least $\frac{1}{(q_s)^2}P(Q)$ which implies that $P(Q) \leq (q_s)^2 Adv_C^{\text{Gap Diffie-Hellman}}(k)$.

Event B_2
We denote by $X = g^x, Y = g^y$ the ephemeral public keys sent, received during the test session s^*. Revealing the long-term secret key of the actor $\hat{A}$ of the test session and the ephemeral key of the origin-session s' for session s^*, the adversary E could distinguish the session key of the test session from a random key by computing $DH_g(X, B) = g^{xb}$

where $B = g^b$ denotes the public key of $s^*_{peer} = \hat{B}$, since

$$g^{xb} = (YB)^{x+a} X^{-y} Y^{-a} B^{-a}.$$

We solve the Gap Diffie-Hellman problem with probability $\frac{1}{q_s N} P(Q)$ where $P(Q)$ must be negligible since Gap Diffie-Hellman problem is hard in G.

Consider the following algorithm C' which uses adversary E as a subroutine. Algorithm C' is given a pair (X, B) of elements from G as an instance of the Gap Diffie-Hellman problem. C' selects one user $\hat{B}$ (uniformly at random from the set $\mathcal{P}$) and sets its long-term public key to B. C' chooses long-term public keys for the remaining users and stores the associated secret keys. Let us denote the ephemeral public key sent by the origin-session (and received by the test session) by Y. Algorithm C' proceeds as follows.

1. Run E on input 1^k and the public keys for all of the N users.
2. $\mathsf{send}(s^*, \hat{P})$: If $\hat{P} \neq \hat{B}$, then C' aborts; otherwise C' sets the ephemeral public key to X and answers the query with the message X.
3. Other send queries are answered in the usual way, e. g. if E issues a $\mathsf{send}(s, \hat{P}, V)$ query to session s, then check whether $V \in G$. If yes, choose $w \in_R \mathbb{Z}_p$, compute $W = g^w$ $(\in G)$ and return W to E. If no, then abort session s.
4. $\mathsf{randomness}(s)$: C' answers in the appropriate way, except if $s = s^*$ in which case C' aborts with failure.
5. $\mathsf{corrupt}(\hat{P})$: C' answers in the appropriate way, except if $\hat{P} = \hat{B}$ in which case C' aborts with failure.
6. $\mathsf{test\text{-}session}(s)$: If $s \neq s^*$, then C' aborts; otherwise C' answers the query appropriately.
7. Store entries of the form $\left(\hat{Q}_i, \hat{Q}_j, Z, U, \lambda\right) \in \{0,1\}^* \times \{0,1\}^* \times G \times G \times \{0,1\}^k$ in a table L, initially empty. When E makes a query of the form $\left(\hat{Q}_i, \hat{Q}_j, Z, U\right)$ to the random oracle for H, answer it as follows:
   - If $\left\{\hat{Q}_i, \hat{Q}_j\right\} = \left\{\hat{A}, \hat{B}\right\}$, $U = X$ and $\mathrm{DDH}(XA, YB, Z) = 1$, then C' aborts E and is successful by outputting $\mathrm{CDH}(X, B) = ZY^{-a}X^{-y}B^{-a}$ (this computation requires the knowledge of a, therefore we must require that $\hat{A} \neq \hat{B}$).
   - Else, proceed as in Step 8 of the simulation related to event B_1.
8. Store entries of the form $\left(\hat{Q}_i, \hat{Q}_j, r, U, V, \lambda\right) \in \mathcal{P} \times \{0,1\}^* \times \{\mathcal{I}, \mathcal{R}\} \times G \times G \times \{0,1\}^k$ in a table T, initially empty, as in the previous simulation related to event B_1.
9. $\mathsf{session\text{-}key}(s)$: C' answers this query by look-up in table T.
10. E outputs a guess: C' aborts with failure.

Analysis of event B_2

The probability that E selects s^* as the test session and $\hat{B}$ as the peer for the test session is at least $\frac{1}{q_s N}$. Assume that this is indeed the case. Then C' does not abort as in Step 2 or Step 6. Under event B_2 the simulation does not fail as in steps $4, 5$. Under event Q, C' is successful as described in the first case of Step 7 and does not

abort as in Step 10. C' correctly computes the Gap Diffie-Hellman instance with probability at least $\frac{1}{q_s N} P(Q)$ which implies that $P(Q) \leq q_s N Adv_{C'}^{\text{Gap Diffie-Hellman}}(k)$.
The analyses of events B_3 and B_4 are similar to the previous analyses. □

Applying the SIG transformation to the π_1-core protocol yields the π_1 protocol, depicted in Figure 3.7.

Initiator $\mathcal{I}$		**Responder** $\mathcal{R}$
$\hat{A}$: $(a, A), (sk_{\hat{A}}, pk_{\hat{A}})$		$\hat{B}$: $(b, B), (sk_{\hat{B}}, pk_{\hat{B}})$
$x \leftarrow r_{\hat{A}}$ $X \leftarrow g^x$ $\sigma_{\hat{A}} \leftarrow Sign_{sk_{\hat{A}}}(X \left[, \hat{B}\right])$	$\xrightarrow{X, \sigma_{\hat{A}}}$	
		$y \leftarrow r_{\hat{B}}$ $Y \leftarrow g^y$ $\sigma_{\hat{B}} \leftarrow Sign_{sk_{\hat{B}}}(Y \left[, X, \hat{A}\right])$
	$\xleftarrow{Y, \sigma_{\hat{B}}}$	$K_{\hat{B}} \leftarrow H(\hat{A}, \hat{B}, (XA)^{y+b}, X)$
$K_{\hat{A}} \leftarrow H(\hat{A}, \hat{B}, (YB)^{x+a}, X)$		

Figure 3.7.: π_1 protocol

Combining Proposition 8 with Theorem 1, we immediately obtain the following result.

Corollary 3. *Under the Gap Diffie-Hellman assumption in the cyclic group G of prime order p, using a deterministic SUF-CMA signature scheme, the protocol π_1 is secure in the* eCK-PFS *model, when H is modeled as a random oracle.*

3.4. Summary

In this chapter we provided a generic approach to achieve perfect forward secrecy in two-message or one-round protocols even in the presence of eCK-like adversaries.

We defined new security models for analyzing the security of AKE protocols. The eCK^w model slightly strengthens eCK by including a more precise modeling of weak-PFS. The stronger eCK-PFS notion guarantees PFS, even in the presence of eCK-like adversaries. Separately proving eCK^w (or eCK) security and PFS does *not* imply security in eCK-PFS. For example, eCK-PFS additionally considers PFS under actor compromise. In the process of formalizing these models, we also provided formal definitions of PFS and, for the first time, weak-PFS. We formally related our models and their intended properties.

Existing two-message protocols such as CMQV, HMQV, or $\mathcal{C}$(NAXOS) fail to achieve security in eCK-PFS. We specified a security-strengthening transformation, called SIG, that transforms any two-message DH-type protocol secure in eCK^w or $\mathsf{eCK}^{\mathsf{passive}}$, a weaker model than eCK^w, into a two-message protocol secure in eCK-PFS. Thus, the SIG transformation can be applied to protocols such as NAXOS, that are

secure in eCK^w. Additionally, it can be applied to protocols that fail to achieve security in eCK^w, but that can instead be proven secure in the weaker $\mathsf{eCK}^{\mathsf{passive}}$ model. As a concrete example we have proven that the π_1-core protocol is secure in $\mathsf{eCK}^{\mathsf{passive}}$. Applying the SIG transformation to π_1-core, we obtained the protocol π_1 that is secure in $\mathsf{eCK\text{-}PFS}$. These examples illustrate the use of SIG in the modular design of AKE protocols.

4. Authenticated Key Exchange Security Incorporating Certification Systems

Most security models for authenticated key exchange (AKE) do not explicitly model the associated *certification system*, which includes the certification authority (CA) and its behaviour. However, there are several well-known and realistic attacks on AKE protocols which exploit various forms of malicious key registration and which therefore lie outside the scope of these models:

- Kaliski's *unknown key share* (UKS) attack [59] on early versions of MQV exploits the ability of the adversary to dynamically register a public key (which is valid and for which the adversary *does* know the secret key).
- The UKS attack on KEA described by Lauter and Mityagin [69, p. 380] exploits the adversary's ability to re-register some party's static public key as his own public key.
- Blake-Wilson and Menezes [22] introduced the *duplicate-signature key selection* (DSKS) attack on signature schemes: after observing a user's signature σ on a message m, the adversary E is able to compute a signature key pair $(\mathsf{sk}_E, \mathsf{vk}_E)$ (or sometimes just a verification key vk_E) such that σ is also E's signature on the message m. Now, for example, if the Station-to-Station (STS) protocol is implemented using a signature scheme that is vulnerable to DSKS attacks, and the adversary can register arbitrary public keys with the CA, then the protocol is vulnerable to an online UKS attack [22].
- In Lim and Lee small subgroup attacks [73], the adversary extracts partial (or even complete) information about a party's long-term secret key. Some of these attacks require registering invalid public keys with the CA before engaging in protocol runs with honest participants. Of particular note are the Lim–Lee-style attacks of Menezes and Ustaoglu [81] on the HMQV protocol [64].

In this chapter we initiate the first systematic analysis of *AKE security incorporating certification systems* (ASICS). In Section 4.1 we define a family of security models that, in addition to allowing different sets of standard AKE adversary queries, also permit the adversary to register arbitrary bitstrings as keys. In Section 4.2 we show how several attacks based on adversarial key registration can be captured in ASICS security models. In Section 4.3 we establish generic results that enable the design and verification of protocols that achieve security even if some keys have been produced maliciously. In Section 4.4 we apply our powerful generic approach to a variant of the CMQV protocol in an eCK-like ASICS model. In Section 4.5 we reveal insights gained from our results with regard to the design of protocols satisfying security in our ASICS models.

4.1. ASICS model family

In this section we define a parameterized AKE security model that allows for explicit modelling of the certification of public keys. Prominent AKE security frameworks can be instantiated in this family of models, as well as extensions that allow dynamic adversarial registration of arbitrary bitstrings as public keys.

Generally speaking, from a user's point of view, participation in key exchange encompasses three consecutive phases: First, users set up their individual key pairs; more precisely, each user invokes a randomized algorithm KeyGen that outputs a fresh secret-key/public-key pair $(\mathsf{sk}, \mathsf{pk})$. Second, users contact a certification authority (CA) to get their keys certified: each user provides the CA with its identifier $\hat{P}$ and its public key pk, and obtains a certificate C that binds the identifier to the key. After completing these setup steps, in the third phase, users can engage in interactive sessions with other users to establish shared keys. To do so, they usually require knowledge of their own key pair $(\mathsf{sk}, \mathsf{pk})$, their identifier $\hat{P}$, and the corresponding certificate C. In addition to that, protocols may require a priori knowledge of (a subset of) the peer's public key pk', peer's identifier $\hat{Q}$, and peer's certificate C'. As we will see, our execution model is general enough to cover all these settings. To ease notation, we assume that public key pk and identifier $\hat{P}$ can be readily derived from any certificate C; we use notation $C.\mathsf{pk} = \mathsf{pk}$ and $C.\mathsf{id} = \hat{P}$ correspondingly. This assumption holds for all practical PKIs we are aware of, and in particular for X.509 certificates (as used in SSL/TLS); the latter carry public key and identifier in a canonically formatted data structure. In potential other cases, where keys and identifiers cannot be extracted from certificates but instead are auxiliary input to the verification routine, certificates can still be expected to be valid for only a single key/identifier pair, i.e., the mapping from C to compatible pk and $\hat{P}$ is again unambiguous. Proposed notations $C.\mathsf{pk}$ and $C.\mathsf{id}$ are hence meaningful also in such cases.

Here, we focus on the interaction between users and the CA that occurs in the certification process. We enable the modeling of different degrees of rigour in the checks of consistency and ownership of public keys pk presented to the CA. On the one hand, CAs could be pedantic with such verifications (e.g., require a proof of knowledge[1] of the secret key corresponding to pk); on the other hand, CAs could also just accept any given bitstring pk as valid and issue a certificate on it. The ability to precisely assess the security of key establishment in the face of different CA behaviours is a key contribution of our new model family.

Definition 22. *An* ASICS protocol Π *consists of a set of domain parameters, a key generation algorithm* KeyGen*, a public key verification procedure* VP*, and the protocol description* π *that describes how key exchange protocol messages are generated and responded to as well as how the session key is derived.*

[1] Although interactive or non-interactive zero-knowledge proof systems seem to yield ideal solutions in this context, standards like X.509, OpenPGP, and PKCS#10 content themselves with the purely heuristic (and inferior) approach of demanding a so-called *proof-of-possession* (PoP), mostly implemented via self-signed certificates or self-signed certificate requests (cf. [2, 93]). The security of such constructions seems to be difficult to formally assess [91].

We denote by VP the specific *verification procedure* on public keys and identifiers that a considered CA deploys. As different checks on pk and $\hat{P}$ might require different levels of interaction between the registering user and the CA, we model it as a procedure, as opposed to a function. VP takes as input a public key and an identifier. To enable the specification of realistic VP behaviour, we additionally allow the definition of VP to access elements of the game state, as we will see in Example 5. We require that VP is efficient and has binary output. Furthermore, we require that the CA issues the requested certificate only if VP outputs value 1; all certification requests where VP outputs value 0 are rejected. Note that, for simplicity, we only consider non-interactive verification procedures (i.e., two-message registration protocols) between the user and the CA. A more general treatment covering interactive verification procedures as well would introduce additional complexities to our framework.

Specific key exchange protocols might be insecure for one (liberal) instantiation of VP, and be secure for another (stricter) one. Note that CAs that do not perform any check on pk and $\hat{P}$ are modelled by a verification procedure VP that always outputs 1. A verification procedure that performs few checks may output 1 for at least all $\mathsf{pk} \in \mathbf{PK}$, where $\mathbf{PK}$ denotes the set of possible public keys output by KeyGen. Precisely, if the inputs of algorithm KeyGen are security parameter 1^k and randomness $r \in_R \{0,1\}^k$, then we define

$$\mathbf{PK} = \left\{ \mathsf{pk} \mid \text{there exists } r \in \{0,1\}^k \text{ such that } \mathsf{KeyGen}(1^k; r) = (\,\cdot\,, \mathsf{pk}) \right\} \ .$$

A verification procedure with high assurance may require a zero-knowledge argument that the requester knows the secret key corresponding to the public key, and even that the key was generated verifiably at random. Note that we allow VP to keep an internal state between invocations; our model hence covers possible implementations of CAs that reject certification requests with public keys that have already been registered (e.g., for a different identifier).

4.1.1. Security model

At a high level, our model stipulates users that generate one or more keys, obtain certificates for these keys from a CA, and use keys and certificates to run (potentially concurrent) sessions of the key agreement protocol. Similar to other security models [17, 34], the adversary controls all communication in these sessions, corrupts users at will to obtain their secret keys, and arbitrarily reveals established session keys. Innovative is the adversary's additional ability to steer the registration process with the CA: it can obtain from the CA valid certificates for public keys and identifiers of its choosing (as long as VP evaluates to 1), and provides users with such certificates.

To keep our model simple and comprehensible, we abstract away any forgeability issues of certificates and assume the following ideal functionality: no certificate will be considered valid unless it has been issued by the CA. We model this by letting the challenger keep a list $\mathcal{C}$ of all CA-issued certificates and by equipping users with a *certificate verification oracle* $\mathcal{O}_{CV}$ that checks membership in that list; concretely, we assume that $\mathcal{O}_{CV}(C) = 1 \Leftrightarrow C \in \mathcal{C}$. Of course, in concrete implementations, this oracle is replaced by an explicit local verification routine; for instance, if certification

acert	certificate of the actor (the user running this session)
pcert	certificate of this session's peer
role	taken role; either $\mathcal{I}$ (initiator) or $\mathcal{R}$ (responder)
sent	concatenation of all messages sent in this session
rcvd	concatenation of all messages received in this session
status	session status; either `active`, `accepted`, or `rejected`
key	key in $\{0,1\}^k$ established in this session
rand	randomness used in this session
data	any additional protocol-specific data

Table 4.1.: Elements of session state

$\mathcal{Q}_N$	$= \{\mathsf{kgen}, \mathsf{hregister}, \mathsf{create}, \mathsf{send}\}$	(Normal protocol behaviour)
$\mathcal{Q}_S$	$= \{\mathsf{corrupt}, \mathsf{randomness}, \mathsf{session\text{-}key}\}$	(corruption of Secrets)
$\mathcal{Q}_R$	$= \{\mathsf{pkregister}, \mathsf{npkregister}\}$	(adversarial key Registration)

Table 4.2.: Overview of query sets. Additionally, there is a test-session query.

is implemented via a signature scheme, this will include its verification procedure (besides further checking of validity, chain-validity, revocation state, etc.).

Sessions and session state.

Users, once they have created their keys and obtained corresponding certificates, can execute protocol sessions. Within a user, each such session is uniquely identified by a pair $s = (C, i)$, where C denotes the certificate used by the user (by himself) in that session, and i is a counter. The user maintains session-specific variables as indicated in Table 4.1. Some session variables are fixed upon session creation, whereas others can be assigned or updated during protocol execution. Some, such as pcert, status, and key, are considered to be outputs of the key establishment and might be used in higher-level protocols or applications. A session s has *accepted* if $s_{\text{status}} = \texttt{accepted}$.

Adversarial queries.

The adversary interacts with users by issuing queries. The adversary can direct users to establish long-term key pairs and certificates (kgen, hregister), to initiate protocol sessions (create), and to respond to protocol messages (send). The adversary may be able to learn long-term keys (corrupt), session-specific randomness (randomness), or session keys (session-key) from users. The adversary can also maliciously obtain certificates from the CA (pkregister, npkregister).

The queries in set $\mathcal{Q}_N = \{\mathsf{kgen}, \mathsf{hregister}, \mathsf{create}, \mathsf{send}\}$, defined as follows, model normal operation of the protocol; they are required in any security model. Initially, the auxiliary variables $\mathcal{HK}$, $\mathcal{C}$, $\mathcal{C}_h$, $\mathcal{C}_{\mathsf{pk}}$, and $\mathcal{C}_{\mathsf{npk}}$ are set to $\emptyset$.

- kgen () By running algorithm KeyGen, a fresh key pair (sk, pk) is generated. Public key pk is returned to the adversary; secret key sk is stored for processing

potential later queries corresponding to pk. The public key is added to the set of honestly generated keys: $\mathcal{HK} \leftarrow \mathcal{HK} \cup \{\mathsf{pk}\}$.

- $\mathsf{hregister}(\mathsf{pk}, \hat{P})$ The query requires that $\mathsf{pk} \in \mathcal{HK}$ and that VP outputs 1 on input pk[2] and $\hat{P}$; otherwise, it returns $\perp$. The public key pk is registered at the CA for the identifier $\hat{P}$. The resulting certificate C is added to the global set of certificates and to the set of honestly generated certificates: $\mathcal{C} \leftarrow \mathcal{C} \cup \{C\}$ and $\mathcal{C}_h \leftarrow \mathcal{C}_h \cup \{C\}$. The query returns C.
- $\mathsf{create}\,(s = (C, i)\,, r, [C'])$ The query requires that $C \in \mathcal{C}_h$, that a session with counter i for certificate C does not already exist, and that $r \in \{\mathcal{I}, \mathcal{R}\}$; otherwise, it returns $\perp$. A new session s is created for the user with public key $C.\mathsf{pk}$ and identifier $C.\mathsf{id}$. Session variables are initialized as

$$(s_{\text{acert}}, s_{\text{pcert}}, s_{\text{role}}, s_{\text{sent}}, s_{\text{rcvd}}, s_{\text{status}}, s_{\text{key}}) \leftarrow (C, \perp, r, \epsilon, \epsilon, \texttt{active}, \perp)\ .$$

  If the optional certificate C' is provided, we set $s_{\text{pcert}} \leftarrow C'$. In addition, a string in $\{0,1\}^k$ is sampled uniformly at random and assigned to s_{rand}; we assume that all randomness required during the execution of session s is deterministically derived from s_{rand}. The user also runs the initialization procedure for the key exchange protocol, which may further initialize its own (internal) state variable s_{data} and optionally generate a message M. If M was generated, set $s_{\text{sent}} \leftarrow M$, and return M. Otherwise, return $\perp$.
- $\mathsf{send}\,(s, M)$ The query requires that session s exists and that $s_{\text{status}} = \texttt{active}$; otherwise, it returns $\perp$. The user continues the protocol execution for this session with incoming message M, which may optionally generate a response message M'. Next, s_{rcvd} is set to $(s_{\text{rcvd}} \parallel M)$ and, if M' is output, then s_{sent} is set to $(s_{\text{sent}} \parallel M')$. The protocol execution may (re-)assign values to s_{status} and s_{key}, and to the session's internal state variable s_{data}. Also, if the value s_{pcert} was not provided to the create query, then the protocol execution may assign a value to s_{pcert}. (For example, in TLS the client does not learn the server's certificate until it receives the `ServerCertificate` message.) If M' was generated, return M'; otherwise return $\perp$.

Remark 4 (Multiple CAs). Our model stipulates a PKI with a single CA. This could be considered a limitation for the analysis of many practically relevant scenarios. For example, the PKI currently in place to secure web traffic in the Internet consists of a large number of independently acting CAs. However, our single-CA setting can readily emulate a multi-CA setting: If $\mathsf{VP}_1, \ldots, \mathsf{VP}_n$ denote the verification procedures of n independent CAs, then composed verification procedure $\mathsf{VP}_{1\ldots n}$ that outputs 1 if there exists at least one i, $1 \leq i \leq n$, such that VP_i outputs 1, effectively folds the n CAs into a single one, adequately modeling the aforementioned multi-CA setting.

Remark 5 (One key for many identifiers). A frequently observed setting in the web PKI is that users hold one public key that they bind to a set of identifiers (with the corresponding number of certificates). This happens, for instance, if globally acting companies register domain names in several different countries, but process HTTPS requests on a single centralized facility. Observe that our model covers such cases of

[2] Reasonable implementations of VP output 1 on all keys $\mathsf{pk} \in \mathcal{HK}$, because $\mathcal{HK} \subseteq \mathbf{PK}$.

multiple registration of a key: hregister may be queried many times for the same pk. Moreover, through being identified by pairs (C, i) (as opposed to (pk, i)), sessions associated with one pk are aware of the identifier $C.\mathsf{id}$ in use.

The queries in set $\mathcal{Q}_S = \{\mathsf{corrupt}, \mathsf{randomness}, \mathsf{session\text{-}key}\}$ model the corruption of a user's secrets. Similar queries are found in other standard AKE models [17, 34].

- corrupt (pk) The query requires $\mathsf{pk} \in \mathcal{HK}$; otherwise, it returns $\perp$. This query returns the secret key sk corresponding to public key pk.
- randomness (s) The query requires that session s exists; otherwise, it returns $\perp$. The query returns the randomness s_{rand}.
- session-key (s) The query requires that session s exists and that $s_{\text{status}} =$ `accepted`; otherwise, it returns $\perp$. The query returns the session key s_{key}.

Remark 6 (Semantics of randomness query). The randomness query gives the adversary access to the randomness s_{rand} that is used in a particular session. Formally, one may think of the session as an instance of a probabilistic Turing machine: the query gives the adversary the values that are read from the random tape. Practically, this models a randomness generator whose values are revealed after they are produced, for example, by a side-channel attack. This is similar to the ephemeral key reveal query from [68].

The hregister query introduced above only allows registration of keys $\mathsf{pk} \in \mathcal{HK}$, i.e., keys held by honest users. In contrast, the adversary can obtain certificates on *arbitrary* (valid) public keys using the following pkregister query. Going even further, the npkregister query allows registration of objects that are not even public keys (always assuming that VP outputs 1 on the candidate object). These queries will allow modelling Kaliski's attack on MQV [59] and small subgroup attacks [73], amongst others. We emphasize that the queries in set $\mathcal{Q}_R = \{\mathsf{pkregister}, \mathsf{npkregister}\}$ have no counterparts in standard definitions of key exchange security.

- pkregister$(\mathsf{pk}, \hat{P})$ The query requires that $\mathsf{pk} \in \mathbf{PK}$ and that VP outputs 1 on input pk and $\hat{P}$; otherwise, it returns $\perp$. The public key pk is registered at the CA for identifier $\hat{P}$. The resulting certificate C is added to the global set of certificates and to the set of certificates generated through pkregister query: $\mathcal{C} \leftarrow \mathcal{C} \cup \{C\}$ and $\mathcal{C}_{\mathsf{pk}} \leftarrow \mathcal{C}_{\mathsf{pk}} \cup \{C\}$. The query returns C.
- npkregister$(\mathsf{pk}, \hat{P})$ The query requires that $\mathsf{pk} \notin \mathbf{PK}$ and that VP outputs 1 on input pk and $\hat{P}$; otherwise, it returns $\perp$. The public key pk is registered at the CA for the identifier $\hat{P}$. The resulting certificate C is added to the global set of certificates and to the set of certificates generated through npkregister query: $\mathcal{C} \leftarrow \mathcal{C} \cup \{C\}$ and $\mathcal{C}_{\mathsf{npk}} \leftarrow \mathcal{C}_{\mathsf{npk}} \cup \{C\}$. The query returns C.

Remark 7 (Efficiency of pkregister and npkregister processing). Observe that pkregister and npkregister queries require a membership test for set **PK**. In some settings, such tests might correspond to computationally hard problems like quadratic residuosity or DDH[3]. Although we stress that the named queries and the ensuing security experiments are well-defined nevertheless, we discuss two possible problems that might

[3] For instance, consider an RSA modulus N and the (cyclic) group QR_N of quadratic residues modulo N. As the CDH problem in QR_N is provably as hard as factoring N [78, 97], it is conceivable to instantiate a DL-based key agreement protocol (like UM, HMQV, etc.) with that

arise when analyzing key exchange protocols where **PK**-membership is difficult to assess: (1) If the queries, by whatever means, correctly decide membership in **PK**, then the adversary implicitly gets access to a decision oracle for that set; this has to be taken into account in corresponding security reductions, e.g., when selecting appropriate hardness assumptions. (2) If an analyzed ASICS protocol Π is used as a component in a higher-level construction Π', then any security argument that reduces the security of Π' to the security of Π will inevitably have to specify how $\mathsf{pkregister}$ and $\mathsf{npkregister}$ oracles are to be simulated. As mentioned before, in some settings this seems to be infeasible.

Example 5. Here we formalize some verification procedures that reflect practically relevant checks. Note that the algorithms can access the current state of game state variables, such as $\mathcal{C}, \mathcal{C}_h, \mathcal{C}_{\mathsf{pk}}, \mathcal{C}_{\mathsf{npk}}$. We use q to denote the query that invoked VP.
(a) **Identity validation.**

$$\mathsf{VP}(\mathsf{pk}, \hat{P}) = \begin{cases} 1 & \text{if } (q = \mathsf{hregister} \ \wedge \ \neg\exists C \in \mathcal{C}_{\mathsf{pk}} \cup \mathcal{C}_{\mathsf{npk}} \ : \ C.\mathsf{id} = \hat{P}) \ , \\ 1 & \text{if } (q \in \mathcal{Q}_R \ \wedge \ \neg\exists C \in \mathcal{C}_h \ : \ C.\mathsf{id} = \hat{P}) \ , \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

The verification procedure prevents the adversary from registering a key for identifier $\hat{P}$ via a query in $\mathcal{Q}_R$ if there already exists a certificate returned as response to a query $\mathsf{hregister}$ with identifier $\hat{P}$. Similarly, VP prevents the adversary from registering a key for identifier $\hat{P}$ via the query $\mathsf{hregister}$ if there already exists a certificate returned as response to a query in $\mathcal{Q}_R$ with identifier $\hat{P}$. This splits the identifiers into two disjoint sets and reflects strong identity validation where impersonation of honest users is impossible.
(b) **Uniqueness of the public key.**

$$\mathsf{VP}(\mathsf{pk}, \hat{P}) = \begin{cases} 1 & \text{if } \neg\exists C \in \mathcal{C} \ : \ C.\mathsf{pk} = \mathsf{pk} \ , \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

The verification procedure takes as input $\mathsf{pk}, \hat{P}$ and the global set of certificates $\mathcal{C}$. If there is no certificate in $\mathcal{C}$ which contains public key pk for which a certificate is being requested, then it returns 1. Otherwise it returns 0. It is unclear how uniqueness of the public key can be enforced if there are multiple independent CAs that do not communicate.
(c) **No two public keys for one identifier.**

$$\mathsf{VP}(\mathsf{pk}, \hat{P}) = \begin{cases} 1 & \text{if } \neg\exists C \in \mathcal{C} \ : \ C.\mathsf{id} = \hat{P} \ , \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

The verification procedure takes as input $\mathsf{pk}, \hat{P}$ and the global set of certificates $\mathcal{C}$. If there is no certificate in $\mathcal{C}$ which contains identifier $\hat{P}$, then it returns 1. Otherwise it returns 0.

group. This would result in $\mathbf{PK} = QR_N$, and $\mathsf{pkregister}$ and $\mathsf{npkregister}$ queries would have to perform membership tests for this set. However, the latter is assumed to be a hard problem, by the quadratic residuosity assumption [53].

(d) **No checks at all.**

$$\mathsf{VP}(\mathsf{pk}, \hat{P}) = 1 \quad \text{(for all } \mathsf{pk} \text{ and all } \hat{P}\text{)}.$$

Remark 8 (On certificate signing requests (CSRs)). Some CAs may require users to self-sign the data that is to be certified and to submit the resulting certificate signing request. To model CSRs and related checks, one needs to introduce an additional parameter ϕ in the definitions of the queries hregister, pkregister, and npkregister. Then the verification procedure of a CA checking CSRs takes as input $\mathsf{pk}, \hat{P}$ and a signature ϕ. If ϕ is a valid signature on message $m = \hat{P} \parallel \mathsf{pk}$, then it returns 1. Otherwise it returns 0.

4.1.2. Security experiment

Using the above queries, we define a parameterized family of AKE security models. As is common in BR-style AKE models, we must restrict query usage so that the adversary cannot trivially win the security experiment. The conditions under which queries are disallowed are expressed by a *freshness* condition, which typically uses a *matching* condition to formalize intended partner sessions.

Definition 23 (Matching, freshness, ASICS model)**.** *Let* Π *be an ASICS protocol. A* matching condition M *for* Π *is a binary relation on the set of sessions of* Π*. Let* Q *be a set of queries such that* $\mathcal{Q}_N \subseteq Q \subseteq \mathcal{Q}_N \cup \mathcal{Q}_S \cup \mathcal{Q}_R$*. A* freshness condition F *for* (Π, Q) *is a predicate (usually depending on a matching condition* M*) that takes a session of* Π *and a sequence of queries (including arguments and results) of a security experiment over queries in* Q*. We call* $X = (M, Q, F)$ *an* ASICS model *for* Π*.*

Definition 24 gives two possible matching conditions. We will later give examples of freshness conditions, in Example 6 on page 64 and in Section 4.4.1.

The intricacies of matching definitions in AKE protocols are explored in detail by Cremers [41]. Two issues are important here. First, there is a strong connection between the information used in a matching definition and the information used to compute the session key. Second, some protocols like the two-message versions of MQV and HMQV allow sessions to compute the same key even if they perform the same role, whereas other protocols such as NAXOS require the sessions that compute the same key to perform different roles. In the remainder of the paper we will use one of the definitions below, depending on the type of protocol.

Definition 24 (M1-matching, M2-matching)**.** *Let* s *and* s' *denote two sessions of an ASICS protocol. We say that session* s' M1-matches *(or is* M1-matching*) session* s *if* $s_{\text{status}} = s'_{\text{status}} = \texttt{accepted}$ *and*

$$\begin{aligned}&(s_{\text{acert}}.\mathsf{pk}, s_{\text{acert}}.\mathsf{id}, s_{\text{pcert}}.\mathsf{pk}, s_{\text{pcert}}.\mathsf{id}, s_{\text{sent}}, s_{\text{rcvd}})\\ &= (s'_{\text{pcert}}.\mathsf{pk}, s'_{\text{pcert}}.\mathsf{id}, s'_{\text{acert}}.\mathsf{pk}, s'_{\text{acert}}.\mathsf{id}, s'_{\text{rcvd}}, s'_{\text{sent}})\end{aligned}$$

Similarly, we say that session s' M2-matches *(or is* M2-matching*) session* s *if* s' M1*-matches session* s *and* $s_{\text{role}} \neq s'_{\text{role}}$*.*

Thus, we will use M1 for protocols such as the two-message versions of MQV and HMQV, and we will use M2 for protocols such as NAXOS [41].

Remark 9 (Comparison to matching definition if only one key per identifier is allowed). Note that when users are allowed to have multiple public keys, matching sessions (that is, intended communication partners) of some AKE protocols will not always compute the same session key if the definition of "matching" does not require agreement on the public keys used in the respective sessions.

The goal of the adversary is to distinguish the session key of a fresh session from a completely random string. This is modelled through an additional query:

- test-session (s) This query requires that session s exists and that $s_{\text{status}} =$ `accepted`; otherwise, it returns $\perp$. A bit b is chosen at random. If $b = 1$, then s_{key} is returned. If $b = 0$, a random element of $\{0,1\}^k$ is returned.

Definition 25 (ASICS_X experiment). *Let Π be an ASICS protocol and $X = (M, Q, F)$ be an ASICS model. We define experiment ASICS_X, between an adversary E and a challenger who implements all users and the CA, as follows:*

1. *The experiment is initialized with domain parameters for security parameter k.*
2. *The adversary E can perform any sequence of queries from Q.*
3. *At some point in the experiment, E issues a* test-session *query for a session s that has accepted and satisfies F at the time the query is issued.*
4. *The adversary may continue with queries from Q, under the condition that the test session must continue to satisfy F.*
5. *Finally, E outputs a bit b' as E's guess for b.*

Definition 26 (ASICS_X advantage). *The adversary E wins the security experiment if it correctly outputs the bit b chosen in the* test-session *query. The ASICS_X-*advantage *of E is defined as $\mathrm{Adv}^{\mathsf{ASICS}_X}_{\Pi,E}(k) = |2\Pr(b = b') - 1|$.*

Definition 27 (ASICS security). *Let Π be an ASICS protocol and $X = (M, Q, F)$ be an ASICS model. Π is said to be* secure in ASICS model *X if, for all PPT adversaries E, it holds that*

1. *if two users successfully accept in M-matching sessions, then they both compute the same session key, and*
2. *E has no more than a negligible advantage in winning the ASICS_X experiment; that is, there exists a negligible function* negl *in the security parameter k such that $\mathrm{Adv}^{\mathsf{ASICS}_X}_{\Pi,E}(k) \leq \mathrm{negl}(k)$.*

Remark 10 (Comparison to other models, including BR, CK and eCK). By choosing the appropriate query subset, freshness, and matching definitions, our parameterised model can be instantiated to produce many existing AKE models. The vast majority of existing models assume honestly-generated key pairs that are securely distributed to users. They therefore do not have queries to model pkregister and npkregister. Our model family includes the models eCK [68] and eCK^w from Chapter 3, which capture the leakage of session specific ephemeral data, as well as the eCK-PFS model from Chapter 3 incorporating perfect forward secrecy. The Bellare-Rogaway model [17] can be captured in our approach by including timestamps as part of the messages, to

accurately model the matching definition from [17]. Unlike the CK model [34], we do not assume unique session identifiers to be available to the protocol up-front. We do not consider this a drawback as real-world protocols may not have such identifiers either. Instead, our family includes the basic $\mathrm{CK}_{\mathrm{HMQV}}$ model [64] in which the HMQV protocol was analysed. For the CK and $\mathrm{CK}_{\mathrm{HMQV}}$ models we have chosen to interpret the session-state as the generated randomness, in the same fashion as the models have been used by their authors [34, 64].

Remark 11 (Freshness will depend on the set of allowed queries and on the matching definition). Note that the exact definition of freshness F used in a security model X will likely depend heavily on the set of queries Q available to the adversary in that model, as well as the specific matching definition. For example, when Q includes pkregister and/or npkregister queries, then we will require an additional restriction on the registration status of the specific public key of the peer used in the test session.

Example 6. Let us consider the following ASICS model as a concrete example. Let $X = (\mathsf{M1}, Q, F)$ be the ASICS model given by $Q = \mathcal{Q}_N \cup \{\mathsf{session\text{-}key}\} \cup \mathcal{Q}_R$ and F defined as follows. Given a sequence of queries and a session s, F holds if:

- no $\mathsf{session\text{-}key}(s)$ query has been issued, and
- for all sessions s' such that s' M1-matches s, no query $\mathsf{session\text{-}key}(s')$ has been issued, and
- no query $\mathsf{pkregister}(s_{\mathrm{pcert}}.\mathsf{pk}, s_{\mathrm{pcert}}.\mathsf{id})$ has been issued.

The model X is an extension of a BR-like model with a CA that allows registration of arbitrary keys. If a protocol is secure in X, then it is secure even if the adversary can register arbitrary bitstrings as public keys, as long as the specific peer key used in the test session is not an adversary-generated valid public key.

4.2. Capturing attacks

We illustrate how several attacks exploiting the adversary's ability to register valid or invalid public keys can be captured in ASICS models. We revisit in Section 4.4.2 some of the attacks described in this section to illustrate that the modular approach that we develop in Section 4.3 cannot be applied to the corresponding protocols.

4.2.1. Existing attacks from the literature

Kaliski's online UKS attack against MQV [59]. Kaliski's attack against MQV can be captured in an ASICS model where the adversary can register a specific valid public key with his own identifier via a pkregister query. As the adversary knows the secret key corresponding to the registered public key, the attack cannot be prevented by VP requiring a proof-of-possession of the secret key.

UKS attack against KEA based on public-key re-registration [69, p. 380]. Suppose that public key pk has been honestly registered at the CA for some user with identifier $\hat{P}$ via the query $\mathsf{hregister}(\mathsf{pk}, \hat{P})$. In this UKS attack on the KEA protocol, the adversary re-registers the public key pk under his own identifier $\hat{L} \neq \hat{P}$ by issuing the query $\mathsf{pkregister}(\mathsf{pk}, \hat{L})$. The attack is prevented if VP checks for uniqueness of the public

key and outputs 0 when the public key was certified before (as observed in [69, p. 381]). Note that the UKS attack can also be prevented by making the session key derivation depend on users' identifiers.

Online UKS attack on STS-MAC based on duplicate-signature key selection (DSKS) [22]. Suppose that the signature scheme employed in the STS-MAC protocol is vulnerable to DSKS attacks. Note that resistance to DSKS attacks is not covered by standard security notions for signatures such as EUF-CMA or SUF-CMA security, which concern only single keys. The UKS attack on STS-MAC [22, p. 160] exploits the ability of the adversary to register a valid public key pk under his own identifier during the run of the protocol. More precisely, the adversary first intercepts a user's message containing a signature σ on message m. He then issues a query $\mathsf{pkregister}(\mathsf{pk}, \hat{L})$ such that σ is also a valid signature on m under pk. The query associates pk with the adversary's identifier $\hat{L}$. Since the adversary knows the secret key corresponding to pk, he obtains a certificate from the CA even if VP requires a proof-of-possession. Countermeasures to such UKS attacks via modification of the protocol are available [22].

Lim–Lee style attack against HMQV with special domain parameters, without validation of ephemeral public keys [81]. Suppose that the underlying group G is a subgroup of the group of units $U(R)$ of a ring R containing an element $T \neq 0$ such that $T^2 = 0$. Then, the HMQV protocol is vulnerable to the attack described in [81, p. 137] with respect to an ASICS model in which the adversary is only given access to standard protocol execution queries in the set $\mathcal{Q}_N$. However, as mentioned in [81, p. 134], this attack on HMQV can only be launched in certain exotic groups that have never been considered for practical use. The attack can be eliminated by restricting the group used for analyzing HMQV in some way or by adding group membership tests on ephemeral public keys to the protocol.

Lim–Lee style attack against HMQV with DSA domain parameters, without validation of ephemeral public keys [80]. Let $G = \langle g \rangle$ denote a q-order subgroup of $\mathbb{Z}_p^*$, where q and p are prime and $(p-1)/q$ is smooth (i. e., it contains no large prime factor). The attack on two-pass HMQV [80, p. 5] can be captured in an ASICS model where the adversary is given access to the queries in the set $Q = \mathcal{Q}_N \cup (\mathcal{Q}_S \setminus \{\mathsf{corrupt}\}) \cup (\mathcal{Q}_R \setminus \{\mathsf{pkregister}\})$. In particular, the adversary can register invalid public keys via the $\mathsf{npkregister}$ query. This attack can be prevented by countermeasures such as requiring VP to include a group membership test on the public key submitted for certification, or by including group membership tests on both ephemeral and long-term public keys during protocol execution. Small-subgroup attacks may also exist in other settings, for instance in groups over elliptic curves.

4.2.2. New attack against KEA+ based on impersonation attack during key registration

Lauter and Mityagin [69] produced the KEA+ protocol from the KEA protocol and Protocol 4 in [19] by incorporating the identifiers of the user and its peer in the session key computation to prevent UKS attacks; however, a similar but previously unreported UKS attack still works on the KEA+ protocol. This UKS attack involves a type of impersonation attack [102, p. 3]: it requires the adversary to successfully impersonate

a user to the CA who then issues a certificate containing the user's identifier, but the adversary's valid public key. We stress that the attack does not arise when only one public key per identifier can be registered.

Our UKS attack against KEA+ is captured in the ASICS model $Z = (\mathsf{M2}, Q, F)$ given by $Q = \mathcal{Q}_N \cup \{\mathsf{session\text{-}key}\} \cup \{\mathsf{pkregister}\}$ and F defined as follows. We say that a session s satisfies F if it holds that

- no $\mathsf{session\text{-}key}(s)$ query has been issued, and
- for all sessions s' such that s' M2-matches s, no query $\mathsf{session\text{-}key}(s')$ has been issued, and
- no query $\mathsf{pkregister}(s_{\mathrm{pcert}}.\mathsf{pk}, s_{\mathrm{pcert}}.\mathsf{id})$ has been issued.

We next show that KEA+ is insecure in ASICS model Z. The adversary creates an initiator session via the query $\mathsf{create}(s = (C, i), \mathcal{I}, C')$, where the public keys $C.\mathsf{pk} = A = g^a$ and $C'.\mathsf{pk} = B = g^b$ were honestly registered via the query $\mathsf{hregister}$. Let $C.\mathsf{id} = \hat{A}$. Now the adversary registers a second public key $A' = A^r$, for some r in $\mathbb{Z}_p$, with identifier $\hat{A}$, by issuing the query $\mathsf{pkregister}(A', \hat{A})$ returning certificate C''. He then creates a responder session via the query $\mathsf{create}(s' = (C', j), \mathcal{R}, C'')$ and activates session s' by sending the message $X = g^x$ sent by session s to session s'. The adversary intercepts the message $Y = g^y$ sent by session s', modifies it by setting $Y' = Y^r$, and then sends message Y' to session s. Session s accepts the key $s_{\mathrm{key}} = H(Y'^a, B^x, \hat{A}, \hat{B})$ as the session key, while session s' accepts as its key $s'_{\mathrm{key}} = H(A'^y, X^b, \hat{A}, \hat{B})$. The completed session s is chosen as the test session. Now a $\mathsf{session\text{-}key}$ query to session s' reveals the session key of the test session. The sessions s and s' are not M2-matching (since $s_{\mathrm{rcvd}} \neq s'_{\mathrm{sent}}$), but compute the same session key. This leads to our UKS attack. Adding either the long-term public keys of both actor and peer or the exchanged messages (in addition to the identifiers) in the session key computation would prevent the attack, since then the session keys computed in both sessions would be different with overwhelming probability (assuming that H is modelled as a random oracle). Note that if KCI attacks are permitted in the model, the UKS attack cannot be prevented by VP requiring a proof-of-possession of the secret key corresponding to the public key A'.

4.3. Achieving ASICS security

We provide a modular approach to obtain provable ASICS security for certain types of protocols. We first show in Section 4.3.1 how a result from Kudla and Paterson [66, Theorem 2] can be adapted to incorporate adversarial registration of valid public keys. Then, in Section 4.3.2, we indicate how to transform protocols to achieve security in the presence of adversaries that can register arbitrary invalid public keys.

4.3.1. Security against adversarial registration of valid keys

To simplify the proof process of protocols which are not built using the approach of [13, 34], Kudla and Paterson [66] present a modular technique that relies on gap assumptions. In particular, Kudla and Paterson [66] show that, under certain conditions, security of a protocol in a model that permits the adversary to reveal

session keys is implied by the security of a variant of the protocol in a reduced model that does not incorporate session key reveal queries. Their result holds in a setting where public keys are honestly generated and registered. We show in this section how their result can be extended on a particular class of protocols to incorporate adversarial registration of valid public keys.

We start by defining an adapted version of *strong partnering* [66].

Definition 28 (Strong partnering)**.** *Let* Π *be an ASICS protocol, and let* $X = (M, Q, F)$ *be an ASICS model. We say that* Π *has* strong partnering in the ASICS_X experiment *if no PPT adversary, when attacking* Π *in the* ASICS_X *experiment, can establish two sessions* s *and* s' *of protocol* Π *holding the same session key without being* M*-matching, with more than negligible probability in the security parameter* k*.*

Given an ASICS model $X = (M, Q, F)$, we denote by cNR-X (using terminology from [66], where cNR stands for "computational No-Reveals", referring to the absence of session-key reveals) the reduced *computational* ASICS_X experiment which is similar to the ASICS_X experiment except that the adversary (a) is not allowed to issue session-key and test-session queries, (b) must pick a session that has accepted and satisfies F at the end of its execution, and (c) output the session key for this session. See Kudla and Paterson [66] for a more detailed description of reduced games.

Definition 29 (cNR-X security)**.** *Let* Π *be an ASICS protocol and* $X = (M, Q, F)$ *be an ASICS model.* Π *is said to be* cNR-X-secure *if, for all PPT adversaries* E*, it holds that*

1. *if two users successfully accept in* M*-matching sessions, then they both compute the same session key, and*
2. E *has no more than a negligible advantage in winning the* cNR-X *experiment; that is, there exists a negligible function* negl *in the security parameter* k *such that* $\mathrm{Adv}^{\mathsf{cNR}\text{-}X}_{\Pi,E}(k) \leq \mathrm{negl}(k)$*, where* $\mathrm{Adv}^{\mathsf{cNR}\text{-}X}_{\Pi,E}(k)$ *is defined as the probability that* E *outputs* (s, s_{key}) *for a session* s *that has accepted and satisfies* F*.*

Theorem 2 deals with the security of DH-type ASICS protocols, which are a generalization of DH-type AKE protocols from Chapter 3 to include certificates and to explicitly identify session strings. This class of protocols includes the most prominent modern two-message AKE protocols.

Definition 30 (DH-type ASICS protocol)**.** *A* DH-type ASICS protocol *is an ASICS protocol of the following form, specified by functions* $f_{\mathcal{I}}, f_{\mathcal{R}}, F_{\mathcal{I}}, F_{\mathcal{R}}, H$*:*

- *Domain parameters* (G, g, q)*, where* $G = \langle g \rangle$ *is a group of prime order* q *generated by* g*.*
- $\mathsf{KeyGen}()$*: Choose* $a \in_R [0, q-1]$*. Set* $A \leftarrow g^a$*. Return secret key* $\mathsf{sk} = a$ *and public key* $\mathsf{pk} = A$*.*
- $\mathsf{VP}(x, \hat{P}) = 1$ *for all* x *and all* $\hat{P}$ *(i.e., the CAs do not perform any checks).*
- *When the initiator is activated to create a new session* s *with a* $\mathsf{create}(s = (C, i), r = \mathcal{I}, C')$ *query, where the secret key corresponding to* $C.\mathsf{pk}$ *is* a*, the initiator computes an outgoing ephemeral public key* $X \leftarrow g^{f_{\mathcal{I}}(s_{\mathrm{rand}}, a, C, C')}$ *and returns* X *as an outgoing message.*

- *The responder is activated to create a new session s' with a* $\mathsf{create}(s = (C', j), r = \mathcal{R}, C)$ *query, where the secret key corresponding to* $C'.\mathsf{pk}$ *is* b.
- *When the responder is activated in a session s' with a* $\mathsf{send}(s', M = X)$ *query, the responder computes an outgoing ephemeral public key* $Y \leftarrow g^{f_{\mathcal{R}}(s'_{\text{rand}}, b, C', C)}$ *and returns Y as an outgoing message. The responder computes a* session string *(an intermediate value to which we give a special name)* $ss' \leftarrow F_{\mathcal{R}}(f_{\mathcal{R}}(s'_{\text{rand}}, b, C', C), b, X, C', C)$, *a session key* $s'_{\text{key}} \leftarrow H(ss')$, *and accepts:* $s'_{\text{status}} \leftarrow \texttt{accepted}$.
- *When the initiator is activated in a session s with a* $\mathsf{send}(s, M = Y)$ *query, the initiator computes a session string* $ss \leftarrow F_{\mathcal{I}}(f_{\mathcal{I}}(s_{\text{rand}}, a, C, C'), a, Y, C, C')$, *a session key* $s_{\text{key}} \leftarrow H(ss)$, *and accepts:* $s_{\text{status}} \leftarrow \texttt{accepted}$.

Initiator $\mathcal{I}$		**Responder $\mathcal{R}$**
$s_{\text{acert}} = C$, $s_{\text{pcert}} = C'$ $\mathsf{sk} = a$, $C.\mathsf{pk} = g^a$, $C'.\mathsf{pk} = B$		$s'_{\text{acert}} = C'$, $s'_{\text{pcert}} = C$ $\mathsf{sk} = b$, $C'.\mathsf{pk} = g^b$, $C.\mathsf{pk} = A$
$x \leftarrow f_{\mathcal{I}}(s_{\text{rand}}, a, C, C')$		
$X \leftarrow g^x$	$\xrightarrow{X}$	$y \leftarrow f_{\mathcal{R}}(s'_{\text{rand}}, b, C', C)$
	$\xleftarrow{Y}$	$Y \leftarrow g^y$
$ss \leftarrow F_{\mathcal{I}}(x, a, Y, C, C')$		$ss' \leftarrow F_{\mathcal{R}}(y, b, X, C', C)$
if $ss = \perp$ then		if $ss' = \perp$ then
$s_{\text{status}} \leftarrow \texttt{rejected}$		$s'_{\text{status}} \leftarrow \texttt{rejected}$
else		else
$s_{\text{key}} \leftarrow H(ss)$		$s'_{\text{key}} \leftarrow H(ss')$
$s_{\text{status}} \leftarrow \texttt{accepted}$		$s'_{\text{status}} \leftarrow \texttt{accepted}$

Figure 4.1.: Messages for generic DH-type ASICS protocol

Theorem 2 below states that for any DH-type protocol Π, under certain conditions, it holds that security of the related protocol π in a reduced model (in which public keys can only be honestly registered) implies security of Π in the stronger non-reduced model that additionally captures adversarial registration of valid public keys. In the cNR-X experiment of Theorem 2 the queries session-key and pkregister are not allowed, whereas in ASICS_Y both queries are allowed.

Theorem 2. *Let $X = (M, Q, F)$ be an ASICS model with $\mathcal{Q}_N \subseteq Q \subseteq \mathcal{Q}_N \cup \mathcal{Q}_S$. Let $Y = (M, Q', F')$ be the ASICS model where $Q' = Q \cup \{\mathsf{pkregister}\}$ and F' is defined as follows. A session s is said to satisfy F' if it satisfies F and no* $\mathsf{pkregister}(s_{\text{pcert}}.\mathsf{pk}, s_{\text{pcert}}.\mathsf{id})$ *query has been issued. Let* Π *be a DH-type ASICS protocol. Suppose that*

- Π *has strong partnering in the* ASICS_Y *experiment,*
- cNR-X *security of the related protocol* π *(defined in the same way as* Π *except that the session key generated in* π *is the session string of* Π *(i.e.,* $s^{\pi}_{\text{key}} = ss^{\Pi}$*)) is probabilistic polynomial-time reducible to the hardness of the computational problem of some relation* ϕ,

- *the session string decisional problem in the* ASICS_Y *experiment for* Π *is polynomial-time reducible to the decisional problem of* ϕ*, and*
- *there is a polynomial-time algorithm that decides whether an arbitrary bitstring is an element of* G*,*

then the security of Π *in ASICS model* Y *is probabilistic polynomial-time reducible to the hardness of the gap problem of* ϕ*, if* H *is modelled as a random oracle.*

Proof. The proof structure is similar to the proof of Theorem 2 in [66]. We denote by Λ the session key space associated to protocol Π. Since the cNR-X security of protocol π is probabilistic polynomial-time reducible to the hardness of the computational problem of some relation ϕ, there exists an algorithm A that on input of a problem instance of the computational problem of ϕ and interacting with an adversary E which has non-negligible probability η of winning the cNR-X game for π in time τ is able to solve the computational problem of ϕ with non-negligible probability $h(\eta)$ and in time $v(\tau)$, for some polynomial functions h and v.

By assumption, the session string decisional problem in the ASICS_Y experiment for Π is polynomial-time reducible to the decisional problem of ϕ. Hence there is an algorithm W which solves the session string decisional problem for Π in polynomial-time τ'' given access to a decisional oracle for ϕ.

Let D be an adversary winning the ASICS_Y experiment against protocol Π with non-negligible probability η' in time τ'. Let K denote the event that D does not query H with the session string ss^* of the test session s^*. Since Π has strong partnering in the ASICS_Y experiment, it holds that, with overwhelming probability, if two sessions compute the same session key, then they must be M-matching. Thus, if event K occurs, then D can only win the experiment with negligible probability $u(k) + 1/|\Lambda|$, where $u(k)$ denotes the probability that D issues a $\mathsf{session\text{-}key}$ query to a session s that is not M-matching s^* and $s_{\text{key}} = s^*_{\text{key}}$.

We next define an algorithm B which solves the gap problem of ϕ with non-negligible probability $h'(\eta')$ and in time $v'(\tau')$, for some polynomial functions h' and v', using adversary D as a subroutine. B will also run algorithm A on the problem instance of the computational problem of ϕ, and an algorithm that decides, in polynomial-time τ''', whether an arbitrary bitstring pk submitted for certification is an element of G. We now define B's responses to D's queries for the pre-specified peer setting; the post-specified peer case proceeds similarly. Algorithm B maintains sets of certificates $\mathcal{C}_h$ and $\mathcal{C}_{\mathsf{pk}}$ as well as lists H-List and G-List, all of which are initially empty.

1. $q \in Q \cap \{\mathsf{kgen}, \mathsf{randomness}, \mathsf{corrupt}\}$: B forwards the query to A and passes A's response back to D.
2. $\mathsf{hregister}(\mathsf{pk}, \hat{P})$: B forwards the query to A and passes A's response back to D. In case A returns a certificate C, B adds C to the set $\mathcal{C}_h$, i.e., $\mathcal{C}_h \leftarrow \mathcal{C}_h \cup \{C\}$.
3. $\mathsf{pkregister}(\mathsf{pk}, \hat{P})$: B checks whether $\mathsf{pk} \in G$ and VP outputs 1 on input pk and $\hat{P}$. If all the checks succeed, then B adds a certificate C to the set $\mathcal{C}_{\mathsf{pk}}$, i.e., $\mathcal{C}_{\mathsf{pk}} \leftarrow \mathcal{C}_{\mathsf{pk}} \cup \{C\}$, and returns C. Else, B returns $\perp$.
4. $\mathsf{create}\,(s = (C, i), r, C')$: B checks whether $C \in \mathcal{C}_h$, a session s with counter i has not yet been created, $r \in \{\mathcal{I}, \mathcal{R}\}$, and $C' \in \mathcal{C}_h \cup \mathcal{C}_{\mathsf{pk}}$. If one of the checks fails, then B returns $\perp$. Else if $C' \in \mathcal{C}_{\mathsf{pk}}$, then B answers D's query by simulating the protocol execution itself. Else, B forwards the query to A and passes A's

response (if any) to D.

5. $\mathsf{send}(s, M)$: If session s does not exist or if $s_{\text{status}} \neq \texttt{active}$, then B returns $\perp$. Else if $s_{\text{pcert}} \in \mathcal{C}_{\mathsf{pk}}$, then B responds to the query by simulating the protocol execution itself. Else B forwards the query to A and passes A's response (if any) to D.
6. H query: To answer D's queries to the random oracle for H, B stores entries of the form (x_i, λ_i) with $\lambda_i \in \Lambda$ in the H-List. When D makes a query x to the random oracle for H, B determines the return value for D as follows:
   - If there exists an entry (x_i, λ_i) in the H-List with $x_i = x$, then return λ_i.
   - Else if there exists an entry $(s_{\text{acert}}.\mathsf{id}, s_{\text{acert}}.\mathsf{pk}, s_{\text{pcert}}.\mathsf{id}, s_{\text{pcert}}.\mathsf{pk}, s_{\text{role}}, s_{\text{sent}}, s_{\text{rcvd}}, \lambda_i)$ in the G-List, for some session s that has accepted and $\lambda_i \in \Lambda$, such that x is the session string of session s (i.e., $x = ss$) using algorithm W, then store the entry (x, λ_i) in the H-List and return λ_i.
   - Else, B chooses $\lambda \in_R \Lambda$, stores the entry (x, λ) in the H-List and return λ.
7. $\mathsf{session\text{-}key}(s)$: To answer D's $\mathsf{session\text{-}key}$ queries, B stores entries of the form $(s_{\text{acert}}.\mathsf{id}, s_{\text{acert}}.\mathsf{pk}, s_{\text{pcert}}.\mathsf{id}, s_{\text{pcert}}.\mathsf{pk}, s_{\text{role}}, s_{\text{sent}}, s_{\text{rcvd}}, \lambda_i)$ with $\lambda_i \in \Lambda$ in the G-List. When D makes a $\mathsf{session\text{-}key}(s)$ query to an initiator session s that has accepted, B determines the return value for D as follows:
   - If there exists an entry $(s_{\text{acert}}.\mathsf{id}, s_{\text{acert}}.\mathsf{pk}, s_{\text{pcert}}.\mathsf{id}, s_{\text{pcert}}.\mathsf{pk}, \mathcal{I}, s_{\text{sent}}, s_{\text{rcvd}}, \lambda_i)$ in the G-List, for some $\lambda_i \in \Lambda$, then return λ_i.
   - Else if there exists an entry $(s_{\text{pcert}}.\mathsf{id}, s_{\text{pcert}}.\mathsf{pk}, s_{\text{acert}}.\mathsf{id}, s_{\text{acert}}.\mathsf{pk}, \mathcal{R}, s_{\text{rcvd}}, s_{\text{sent}}, \lambda_i)$ in the G-List, then B stores the entry $(s_{\text{acert}}.\mathsf{id}, s_{\text{acert}}.\mathsf{pk}, s_{\text{pcert}}.\mathsf{id}, s_{\text{pcert}}.\mathsf{pk}, \mathcal{I}, s_{\text{sent}}, s_{\text{rcvd}}, \lambda_i)$ in the G-List and returns λ_i.
   - Else if there exists an entry of the form (x_i, λ_i) in the H-List, where $x_i = ss$ using algorithm W, then B stores the entry $(s_{\text{acert}}.\mathsf{id}, s_{\text{acert}}.\mathsf{pk}, s_{\text{pcert}}.\mathsf{id}, s_{\text{pcert}}.\mathsf{pk}, \mathcal{I}, s_{\text{sent}}, s_{\text{rcvd}}, \lambda_i)$ in the G-List and returns λ_i.
   - Else, B chooses $\lambda \in_R \Lambda$, stores the entry $(s_{\text{acert}}.\mathsf{id}, s_{\text{acert}}.\mathsf{pk}, s_{\text{pcert}}.\mathsf{id}, s_{\text{pcert}}.\mathsf{pk}, \mathcal{I}, s_{\text{sent}}, s_{\text{rcvd}}, \lambda)$ in the G-List, and returns λ.

   A $\mathsf{session\text{-}key}$ query to a responder session that has accepted is answered similarly.
8. $\mathsf{test\text{-}session}(s^*)$: B selects $\mu \in_R \Lambda$ and returns μ to D.
9. D outputs a guess: B aborts with failure.

B can detect the complementary event K^c by checking which of the entries (x_i, λ_i) in the H-List has $x_i = ss^*$ using algorithm W. B then passes x_i to A. Since the test session s^* must be fresh, no $\mathsf{pkregister}(s^*_{\text{pcert}}.\mathsf{pk}, s^*_{\text{pcert}}.\mathsf{id})$ occurred in the ASICS_Y experiment and hence the certificate s^*_{pcert} has been output through an $\mathsf{hregister}(s^*_{\text{pcert}}.\mathsf{pk}, s^*_{\text{pcert}}.\mathsf{id})$ query. A solves the computational problem of ϕ with non-negligible probability $h(\eta)$, where $\eta = \eta'(1 - u(k) - 1/|\Lambda|)$. B is successful by outputting A's solution to the instance of the computational problem of ϕ and solves the gap problem of ϕ with non-negligible probability $h(\eta)$ and in time $v(\tau)$, where $\tau = \tau' + \tau'' n_H(n_{\mathsf{session\text{-}key}} + 1) + \tau''' n_{\mathsf{pkregister}}$ with $n_H, n_{\mathsf{session\text{-}key}}$ and $n_{\mathsf{pkregister}}$ denoting the number of $H, \mathsf{session\text{-}key}$ and $\mathsf{pkregister}$ queries issued by D, respectively. □

Remark 12. We cannot show that Theorem 2 holds for more complex protocols Π such as UM or HMQV-C in arbitrary ASICS base models as the simulation of non-test sessions s of Π with s_{pcert} being the result of a $\mathsf{pkregister}$ query cannot be performed

in the appropriate way without the knowledge of long-term secret keys and without violating the freshness condition.

4.3.2. Security against adversarial registration of invalid keys

The following theorem, which is applicable to a wider range of protocols than Theorem 2 (e.g., to three-message protocols such as UM [83] or HMQV-C [65]), allows us to achieve security against adversaries that can obtain certificates from the CA for invalid public keys by transforming the protocol to include a group membership test on the peer's public key. In contrast to Theorem 2, no additional requirement is imposed on the freshness condition of model Y.

Theorem 3. *Let $X = (M, Q, F)$ be an ASICS model with $\mathcal{Q}_N \subseteq Q \subseteq \mathcal{Q}_N \cup \mathcal{Q}_S \cup (\mathcal{Q}_R \setminus \{\mathsf{npkregister}\})$.*

Let π be the class of all ASICS protocols where the domain parameters (G, g, q), the key generation algorithm KeyGen *and the verification procedure* VP *are as in Definition 30.*

Let $f : \pi \to \pi$ be the protocol transformation that to any protocol $\Pi \in \pi$ assigns the protocol $f(\Pi)$ derived from Π by adding the following protocol step for each role of the protocol. Upon creation with (or, via send*, receipt of) the certificate C' to be used for the peer of session s, the user running session s checks whether the public key $C'.\mathsf{pk}$ belongs to the group G before continuing the execution of the protocol. In case the check fails, the protocol execution is aborted and s_{status} is set to* `rejected`*.*

Suppose that there is a polynomial-time algorithm that decides whether an arbitrary bitstring is an element of G. Then f is a security-strengthening protocol transformation from ASICS model X to ASICS model $Y = (M, Q \cup \{\mathsf{npkregister}\}, F)$ according to Definition 20.

Proof. The first part of the proof, namely that ASICS model Y is at least as strong as ASICS model X, is straightforward and proceeds in the same way as the reduction proofs in Section 3.1.4.

We show next that the second condition of Definition 20 is satisfied. Let Π be an ASICS protocol secure in model X. It is straightforward to verify the first condition of Definition 27, that is, that M-matching sessions of protocol $f(\Pi)$ compute the same session key. This follows from the fact that M-matching sessions of protocol Π compute the same key as protocol Π is secure in ASICS model X. We next verify that the second condition of Definition 27 holds.

Claim. If there is a PPT adversary E succeeding in the ASICS_Y experiment against protocol $f(\Pi)$ with non-negligible advantage in time τ', then we can construct a PPT adversary E' succeeding in the ASICS_X experiment against protocol Π with non-negligible advantage in time $v(\tau)$ (for some polynomial function v) using adversary E as a subroutine. Let L be an algorithm that decides, in polynomial-time τ'', whether an arbitrary bitstring pk submitted for certification is an element of G.

Proof. Fix a PPT adversary E succeeding in the ASICS_Y experiment against protocol $f(\Pi)$ with non-negligible advantage. We define an algorithm E' which succeeds in the ASICS_X experiment against protocol Π with non-negligible advantage using E as a subroutine. Algorithm E' maintains sets of certificates $\mathcal{C}_h, \mathcal{C}_{\mathsf{pk}}$ and $\mathcal{C}_{\mathsf{npk}}$, all of

which are initially empty, and answers E's queries in the pre-specified peer setting as follows.

1. $q \in Q \cap \{\mathsf{kgen}, \mathsf{randomness}, \mathsf{corrupt}, \mathsf{session\text{-}key}\}$: E' issues the same query and returns the answer to E.
2. $\mathsf{hregister}(\mathsf{pk}, \hat{P})$: When E issues an $\mathsf{hregister}(\mathsf{pk}, \hat{P})$ query, E' issues the same query and returns the answer to E. In case a certificate C is returned, E' adds C to the set $\mathcal{C}_h$, i.e., $\mathcal{C}_h \leftarrow \mathcal{C}_h \cup \{C\}$.
3. $\mathsf{pkregister}(\mathsf{pk}, \hat{P})$: When E issues a $\mathsf{pkregister}(\mathsf{pk}, \hat{P})$ query, E' issues the same query and returns the answer to E. In case a certificate C is returned, E' adds C to the set $\mathcal{C}_{\mathsf{pk}}$, i.e., $\mathcal{C}_{\mathsf{pk}} \leftarrow \mathcal{C}_{\mathsf{pk}} \cup \{C\}$.
4. $\mathsf{npkregister}(\mathsf{pk}, \hat{P})$: E' checks whether $\mathsf{pk} \notin G$ (using algorithm L) and VP outputs 1 on input pk and $\hat{P}$. If the checks succeed (i.e., $\mathsf{pk} \notin G$ and $\mathsf{VP}(\mathsf{pk}, \hat{P}) = 1$), then E' returns a certificate C to E and adds C to the set $\mathcal{C}_{\mathsf{npk}}$, i.e., $\mathcal{C}_{\mathsf{npk}} \leftarrow \mathcal{C}_{\mathsf{npk}} \cup \{C\}$. Otherwise, E' returns $\perp$.
5. $\mathsf{create}\,(s = (C, i), r, C')$: E' checks whether $C \in \mathcal{C}_h$, a session s with counter i has not yet been created, $r \in \{\mathcal{I}, \mathcal{R}\}$, and $C' \in \mathcal{C}_h \cup \mathcal{C}_{\mathsf{pk}} \cup \mathcal{C}_{\mathsf{npk}}$. If one of the checks fails, then E' returns $\perp$. Else if $C' \in \mathcal{C}_h \cup \mathcal{C}_{\mathsf{pk}}$, then E' issues the same query and returns the answer (if any) to E. Else, E' rejects the session creation and sets s_{status} to `rejected`.
6. $\mathsf{send}\,(s, M)$: If session s does not exist or if $s_{\text{status}} \neq \texttt{active}$, then E' returns $\perp$. Else E' issues the same query and returns the response (if any) to E.
7. $\mathsf{test\text{-}session}(s)$: When E issues a $\mathsf{test\text{-}session}(s)$ query to a session s that has accepted, E' issues the same $\mathsf{test\text{-}session}$ query and returns the answer to E.
8. At the end of E's execution, that is, after it has output its guess b', E' outputs b' as well.

It follows that $\mathrm{Adv}^{\mathsf{ASICS}_Y}_{f(\Pi),E}(k) \leq \mathrm{Adv}^{\mathsf{ASICS}_X}_{\Pi,E'}(k)$, and adversary E' runs in time $v(\tau)$ with $\tau = \tau' + \tau'' n_{\mathsf{npkregister}}$, for some polynomial function v, where $n_{\mathsf{npkregister}}$ denotes the number of $\mathsf{npkregister}$ queries made by E. Since Π is secure in ASICS model X, $\mathrm{Adv}^{\mathsf{ASICS}_Y}_{f(\Pi),E}(k)$ is bounded above by a negligible function in the security parameter k. □

Definition 31. *Let π be a class of AKE protocols. We say that two security models X and Y are* equally strong *with respect to π, denoted by $X =^{\pi}_{Sec} Y$, if $X \leq^{\pi}_{Sec} Y$ and $Y \leq^{\pi}_{Sec} X$, where the relation $\leq^{\pi}_{Sec}$ is as in Definition 17.*

Remark 13. Let π be the class of ASICS protocols where the domain parameters (G, g, q), the key generation algorithm KeyGen are as in Definition 30 and the verification procedure $\mathsf{VP}(x, \hat{P})$ outputs 1 if $x \in G$ and 0 otherwise. Let $X = (M, Q, F)$ with $\mathcal{Q}_N \subseteq Q \subseteq \mathcal{Q}_N \cup \mathcal{Q}_S \cup (\mathcal{Q}_R \setminus \{\mathsf{npkregister}\})$ and $Y = (M, Q \cup \{\mathsf{npkregister}\}, F)$. It is easy to verify that the ASICS models X and Y are equally strong with respect to π according to Definition 31. Whenever the adversary issues a query $\mathsf{npkregister}$ in an ASICS_Y experiment, the symbol $\perp$ is returned. Thus, transforming an ASICS protocol from protocol class π as described in Theorem 3 does not yield a protocol that is secure in a stronger ASICS model. In contrast, the model Y is stronger than model X with respect to the protocol class considered in Theorem 3.

Combining both Theorem 2 and Theorem 3, we obtain the following result.

Corollary 4. *Let* Π *be a DH-type ASICS protocol. Let* $X = (M, Q, F)$ *and* $Y = (M, Q', F')$ *be defined as in Theorem 2, and let the conditions of Theorem 2 hold with respect to protocol* Π*. Let* $f(\Pi)$ *denote the protocol derived from* Π *as specified in Theorem 3. Then the transformed protocol* $f(\Pi)$ *is secure in ASICS model* $Z = (M, Q'', F')$*, where* $Q'' = Q' \cup \{\mathsf{npkregister}\}$*, if* H *is modelled as a random oracle.*

Applying Corollary 4 to a concrete DH-type ASICS protocol that satisfies all the preconditions, we obtain a protocol that is secure in an ASICS model in which (a) sessions (including the test session) may use a certificate for the peer that resulted from an npkregister query, and (b) the certificate of the test session's peer was not the result of a pkregister query.

4.4. Applications

To illustrate the power of our generic approach, we examine in Section 4.4.1 how to apply our technique to Ustaoglu's CMQV protocol [105]. CMQV is a modern DH-type protocol that is comparable in efficiency to HMQV, but enjoys a simpler security proof in the eCK model.

Our results allow us to analyze a slightly modified version of the CMQV protocol, which we call CMQV', in a model that does not include session-key, pkregister, and npkregister queries, which simplifies the overall proof. We verify that CMQV' meets the preconditions of Corollary 4, and conclude that a variant of CMQV' with group membership test on the peer's public key is ASICS-secure in an eCK-like model. Similarly, our generic approach can be applied to other DH-type candidates such as NAXOS [68] and UP [106]. We discuss in Section 4.4.2 where the preconditions of Corollary 4 are violated for protocols from Section 4.2 such as MQV or HMQV.

4.4.1. CMQV'

CMQV [105] was originally proven secure in the eCK model, where there is only one public key per identifier. In the ASICS setting, there is no such unique mapping between user identifiers and public keys. Hence, to be able to prove CMQV secure in the ASICS model, we include the public keys of the users in the session string to ensure that users have the same view of these public keys when deriving the session key. We call the resulting protocol CMQV'.

CMQV' as a DH-type ASICS protocol. Two-pass CMQV' can be stated as a DH-type ASICS protocol, by instantiating Definition 30 with the following functions. Let $\mathcal{H}_1 : \{0,1\}^k \times \mathbb{Z}_q^* \to \mathbb{Z}_q^*$, $\mathcal{H}_2 : \{0,1\}^* \to \mathbb{Z}_q$, and $H : \{0,1\}^* \to \{0,1\}^k$ be hash functions. We define $f_{\mathcal{I}}, f_{\mathcal{R}}, F_{\mathcal{I}}, F_{\mathcal{R}}$ as:

$$f_{\mathcal{I}}(r, a, C, C') = \mathcal{H}_1(r, a)$$

$$F_{\mathcal{I}}(x, a, Y, C, C') = \begin{cases} \perp & \text{, if } Y \notin G \setminus \{1\} \\ ((YB^e)^{x+da} \parallel g^x \parallel Y \parallel C.\mathsf{id} \parallel A \parallel C'.\mathsf{id} \parallel B) & \text{, if } Y \in G \setminus \{1\} \end{cases}$$

$$f_{\mathcal{R}}(r, b, C', C) = \mathcal{H}_1(r, b)$$

$$F_{\mathcal{R}}(y, b, X, C', C) = \begin{cases} \perp & \text{, if } X \notin G \setminus \{1\} \\ ((XA^d)^{y+eb} \parallel X \parallel g^y \parallel C.\mathsf{id} \parallel A \parallel C'.\mathsf{id} \parallel B) & \text{, if } X \in G \setminus \{1\}, \end{cases}$$

where $d = \mathcal{H}_2(X \parallel C.\mathsf{id} \parallel C'.\mathsf{id})$, $e = \mathcal{H}_2(Y \parallel C.\mathsf{id} \parallel C'.\mathsf{id})$, $A = C.\mathsf{pk}$, $B = C'.\mathsf{pk}$; $\parallel$ denotes tagged concatenation to avoid ambiguity with variable-length strings.

We now show, using Corollary 4, that the resulting DH-type CMQV' protocol is a secure ASICS protocol in an ASICS model with leakage queries corresponding to the eCK model.

ASICS model for eCK-like leakage. Define the ASICS model $\mathsf{eCK} = (\mathsf{M2}, Q, F)$ for eCK-like leakage [68] as follows. Let $Q = \mathcal{Q}_N \cup \mathcal{Q}_S$. Let F be the condition that a session s satisfies F if, for all sessions s' such that s' M2-matches s, none of the following conditions hold:

- a $\mathsf{session\text{-}key}(s)$ query has been issued;
- if s' exists:
  - a $\mathsf{session\text{-}key}(s')$ query has been issued;
  - both $\mathsf{corrupt}(s_{\text{acert}}.\mathsf{pk})$ and $\mathsf{randomness}(s)$ queries have been issued;
  - both $\mathsf{corrupt}(s'_{\text{acert}}.\mathsf{pk})$ and $\mathsf{randomness}(s')$ queries have been issued;
- if s' does not exist:
  - both $\mathsf{corrupt}(s_{\text{acert}}.\mathsf{pk})$ and $\mathsf{randomness}(s)$ queries have been issued;
  - a $\mathsf{corrupt}(s_{\text{pcert}}.\mathsf{pk})$ query has been issued.

Theorem 4. *Let f(CMQV') be the DH-type ASICS protocol derived from the* CMQV' *protocol defined above, as specified in Theorem 3. If $\mathcal{H}_1, \mathcal{H}_2$ and H are modelled as random oracles, G is a group where the gap Diffie–Hellman assumption holds and membership in G is decidable in polynomial time, then f(CMQV') is secure in ASICS model $Z = (\mathsf{M2}, \mathcal{Q}_N \cup \mathcal{Q}_S \cup \mathcal{Q}_R, F')$, where a session s is said to satisfy F' if it satisfies the freshness condition F from the* eCK *model and no* $\mathsf{pkregister}(s_{\text{pcert}}.\mathsf{pk}, s_{\text{pcert}}.\mathsf{id})$ *query has been issued.*

Sketch. We can readily show that CMQV' satisfies the preconditions of Corollary 4 under the above formulation of the eCK model as an ASICS model:

1. *Strong partnering.* It is straightforward to see that CMQV' has strong partnering in the $\mathsf{ASICS}_{\mathsf{eCK}'}$ game (where eCK' is derived from eCK as described in Theorem 2): since the session key in CMQV' is computed via a random oracle, the probability that two sessions derive the same session key without using the same session string input to the random oracle is negligible.

2. cNR-eCK-*security of the session string variant of CMQV'.* This can be shown by an adaptation of Ustaoglu's original proof of CMQV'. In large part, the main proof can be followed. However, a few simplifications can be made because the simulation need not answer session-key queries (so preventing key replication attacks and simulating sessions where the public key is a challenge value are easier).
3. *Hardness of the session string decision problem.* It can be easily seen that this is polynomial-time reducible to the decisional problem for Diffie–Hellman triples (U, V, W) by noting that the first component of the CMQV' session string σ is equal to $g^{(y+eb)(x+da)} = g^{xy} g^{ady} g^{bex} g^{abde}$; the DDH values (U, V) can be injected into either (X, Y), (A, Y), (B, X), or (A, B), with W inserted into the corresponding part of σ, yielding a polynomial-time reduction.

Detailed proofs of each of the above claims can be found in Appendix A. □

4.4.2. Discussion

In Section 4.4.1 we provided evidence that our modular approach can indeed be successfully applied to reason about the security of prominent key exchange protocols. Here we inspect which preconditions of Corollary 4 are violated for some commonly analyzed protocols.

Let $X_1 = (\mathsf{M1}, Q, F)$ be the ASICS model given by $Q = \mathcal{Q}_N \cup \{\text{session-key}\}$ and F defined as follows. We say that a session s satisfies F if it holds that neither a session-key(s) query nor a session-key(s') query for any session s' that is M1-matching s have been issued. Let $Y_1 = (\mathsf{M1}, Q', F')$ be derived from X_1 as specified in Theorem 2. The UKS attacks described in Section 4.2 against the DH-type protocols MQV and KEA caused two sessions that were not M1-matching to compute the same session key. Thus, the first precondition of Corollary 4 is violated for these protocols as they do not have strong partnering in the ASICS_{Y_1} experiment. The two-pass HMQV protocol without ephemeral public key validation is insecure in ASICS model X_1 when special domain parameters are used (see Section 4.2) due to the lack of ephemeral public key validation. Hence, it is easy to see that the related protocol, that we call HMQV', is not cNR-X_1-secure, which violates the second precondition of Corollary 4. Even though it is unclear whether there is an attack against two-pass HMQV with DSA domain parameters without ephemeral public key validation, the third precondition of Corollary 4 is violated for this version of the HMQV protocol. The lack of ephemeral public key validation hinders us from using a DDH oracle to solve the session string decisional problem for the protocol since there is no guarantee that all inputs to the DDH oracle belong to the group G. The latter observations on HQMV lead us to the conclusion that validation of ephemeral public keys is necessary to be able to apply Corollary 4 to the HMQV protocol, even when the adversary is not given access to the randomness query.

Let $X_2 = (\mathsf{M2}, Q, F)$ be the ASICS model given by $Q = \mathcal{Q}_N \cup \{\text{session-key}\}$ and F defined as follows. We say that a session s satisfies F if it holds that neither a session-key(s) query nor a session-key(s') query for any session s' that is M2-matching s have been issued. Let $Y_2 = (\mathsf{M2}, Q', F')$ be derived from X_2 as specified in Theorem 2. Note that model Y_2 corresponds to the model X defined in Section 4.2 capturing the

UKS attack against KEA+. The latter UKS attack described against the DH-type protocol KEA+ caused two sessions that were not M2-matching to compute the same session key. Thus, the first precondition of Corollary 4 is violated for KEA+ as this protocol does not have strong partnering in the ASICS_{Y_2} experiment.

4.5. Lessons learned and recommendations

As we started our systematic investigation we assumed that certification authorities would need to perform some minimal checks on public keys to obtain secure KE protocols. Perhaps surprisingly, nearly all of the effort can be shifted to the protocols; and modern protocols often perform sufficient checks. In particular, our results provide formal foundations for some of the protocol-specific observations of Menezes and Ustaoglu [81]: checking that short-term as well as long-term public keys are in the key space (i.e., in group G for DH-type protocols) is not superfluous.

Based on these observations, and given M public keys, N users may need to perform on the order of $M \times N$ such checks in total, even when caching the results. Reasoning purely about the overall amount of computation time used, one could consider moving the burden to the CAs. If the CAs only create certificates after a successful check, the CAs would only perform on the order of M checks in total. Depending on the deployment scenario, this might be a preferable alternative.

Similarly, CAs do not necessarily need to check uniqueness of public keys. As long as the key derivation involves the identifiers in an appropriate way, UKS attacks such as the one on KEA can be prevented.

In general, our results further justify using as much information as possible in the key derivation function (KDF). This helps with establishing formal security proofs and it is also a prudent engineering principle. In particular, we recommend that in settings where users may have multiple long-term public keys, the input to the KDF should not only include the identifiers and the message transcript, but also the specific public keys used in the session.

We hope our work can serve as a foundation for the development of a range of protocols specifically designed to incorporate certification systems, offering different tradeoffs between efficiency and trust assumptions of the involved parties.

4.6. Summary

In this chapter we developed the framework ASICS for analyzing AKE protocols while taking into account the certification authority and its behavior. In particular, we discovered a new attack against the KEA+ protocol based on an impersonation attack during key registration, and captured this attack in the appropriate ASICS model.

We provided generic results that show how protocols can be strengthened to achieve security when adversaries can register arbitrary bitstrings as keys, and the CA does not perform any checks. Our results provide evidence (a) that users can defend themselves against CAs that do not perform proper checks, and (b) for the importance of public key validation.

Finally, we applied our methodology to a variant of the CMQV protocol to obtain a protocol secure in the natural strengthening of the eCK model to the ASICS setting.

Part II.

Stronger Security via Impossibility Results

5. On the Limits of AKE Security with an Application to Bad Randomness

State-of-the-art authenticated key exchange protocols are proven secure in game-based security models. These models have considerably evolved in strength from the original Bellare-Rogaway model. However, so far only informal impossibility results, which suggest that no protocol can be secure against stronger adversaries, have been sketched. At the same time, there are many different security models being used, all of which aim to model the strongest possible adversary. In the first part of this thesis we provided stronger security guarantees than found in the state-of-the-art via extensions of current models. In this chapter we perform the first systematic analysis of the limits of game-based AKE security models.

We proceed as follows. In Section 5.1 we present a framework to reason about the security of AKE protocols in the presence of adversaries with diverse capabilities. In particular, this framework allows to express security models that permit the adversary to choose session-specific randomness. In Section 5.2 we define a series of relevant protocol classes. In Section 5.3 we derive strong security models from impossibility results on the security provided by each protocol class in the case where the adversary cannot choose the randomness used in protocol sessions. We then extend these models to additionally capture chosen-randomness attacks, in Section 5.4. In Section 5.5 we adapt our impossibility results to more complex protocol classes and derive strong models from these impossibility results. While a large class of protocols fails to achieve security in models that capture chosen-randomness attacks even on the target session and its partner session, we construct, in Section 5.6, variants of the NAXOS protocol that provide security against such attacks. In Section 5.7 we formally compare our security models and provide a protocol hierarchy for AKE security of the constructed protocols with respect to our derived security models.

5.1. AKE framework

In this section we define a framework to reason about the security of AKE protocols belonging to different classes against adversaries with diverse capabilities. This framework allows to express existing AKE security models such as the eCK model [68] and the models eCK^w and eCK-PFS from Chapter 3 as well as extensions of these models that permit the adversary to choose the randomness used in protocol sessions.

5.1.1. Security model

Sessions and session-specific memory. Let $\mathcal{P}$ be a finite set of N binary strings representing user identifiers. Each user can execute multiple instances of an AKE

actor	the session's actor (the user running the session)
peer	the session's peer (the intended communication partner)
role	taken role; either $\mathcal{I}$ (initiator) or $\mathcal{R}$ (responder)
sent, *recv*	concatenation of all messages sent, respectively received, in the session
status	session status; either `active`, `accepted`, or `rejected`
key	key established in the session
rand	randomness used in the session
data	any additional session-specific or protocol-specific data
step	protocol step to be executed (in the session)

Table 5.1.: Elements of session state

protocol, called sessions, concurrently. We can uniquely identify specific sessions of a user by referring to the order in which they are created. Thus, the i-th session of user $\hat{P}$ is denoted by the tuple $(\hat{P}, i) \in \mathcal{P} \times \mathbb{N}$. These tuples are not used by the protocol, but allow the adversary to identify the sessions he created. We model each user by a probabilistic Turing machine. For each user $\hat{P}$, the state of its Turing machine consists of the memory contents of the user, where we differentiate between session-specific memory and user memory, which is shared among different sessions. We take an abstract view on the session-specific memory and assume that it can be separated into distinct named fields, referred to as variables and listed in Table 5.1. Some of these variables are set upon session creation, whereas others are set or updated during execution of the protocol. The next step to be executed by the protocol is stored in the variable *step*. Alternatively, this value could be stored in the variable *data*. We choose to store it in a separate variable for clarity. We say that a session s has accepted (or is completed) if the value of its *status* variable taking values in the set $\{\texttt{active}, \texttt{accepted}, \texttt{rejected}\}$ is `accepted`. We denote by st_s the session-specific memory related to session s. The session-specific memory contains the session-specific variables of Table 5.1. Initially we assume that each session-specific variable is undefined, denoted by $\perp$.

User memory. The user memory of some user stores the user's long-term public/secret key pair, the public key of all other users $\hat{Q} \in \mathcal{P}$ as well as additional variables that might be required by the protocol. The information stored in the user memory is accessed and possibly updated by sessions of the user according to the protocol specification. In contrast to session-specific information, data stored in the user memory of some user $\hat{P}$ is shared among different sessions of the user $\hat{P}$. We denote by $st_{\hat{P}}$ the user memory of user $\hat{P} \in \mathcal{P}$.

Game state and game behaviour (see also [33]). The adversary, modeled as a probabilistic polynomial-time algorithm, interacts with the users in the set $\mathcal{P}$ within a game through queries in a set Q. The state of the game (or game state) contains session-specific state information st_s for all sessions s, user-specific information $st_{\hat{P}}$ for each user $\hat{P} \in \mathcal{P}$ as well as other information related to the game such as some bit that the adversary attempts to guess. The game behaviour, which we denote by Φ, describes how the game processes the queries in Q. More precisely, the game behaviour Φ is an algorithm taking as input the current state of the game GST, a

query $q \in Q$, a protocol π, and a security parameter 1^k, and returning a new state GST' as well as a response $\texttt{resp} \in \{0,1\}^* \cup \{\perp, \star\}$ to the adversary's query q.

Definition 32 (h-message protocol)**.** *Let 1^k be the security parameter. An h-*message protocol π*, where h is the sum of the number of messages sent and received during a protocol session, consists of*

- *a set of domain parameters,*
- *a probabilistic polynomial-time key generation algorithm* KeyGen*, which takes as input the security parameter and outputs a public/secret key pair, and*
- *a deterministic polynomial-time algorithm Ψ executed by a user in a session. This algorithm takes as input the security parameter 1^k, the session-specific memory st_s of a session s, the user memory $st_{\hat{P}}$ of the actor $\hat{P}$ of session s, and a message $m \in \{0,1\}^*$, and outputs a triple of elements $(m', st'_s, st'_{\hat{P}})$, where $m' \in \{0,1\}^* \cup \{\star\}$ is a message, st'_s is an updated internal session state, and $st'_{\hat{P}}$ is an updated state of the user memory of user $\hat{P}$.*

If h is even, then the number of messages $m' \neq \star$ output by Ψ during a protocol session is $\frac{h}{2}$ for both roles initiator and responder. If h is odd, then the number of messages $m' \neq \star$ output by Ψ during a protocol session is $\frac{h+1}{2}$ for the initiator role and $\frac{h-1}{2}$ for the responder role.

The output of the key exchange algorithm Ψ (see Definition 32) may include the value $\star$ to indicate that the session does not generate an outgoing message.

Remark 14 (Multiple key pairs). If a protocol uses $n > 1$ key pairs, then the key generation algorithm KeyGen represents the joint key generation algorithm $\mathsf{KeyGen} = (\mathsf{KeyGen}_1, ..., \mathsf{KeyGen}_n)$. The secret key $\mathsf{sk}_{\hat{P}}$ of the user with identifier $\hat{P}$ corresponds to the concatenation of n secret keys $\mathsf{sk}_{\hat{P}} = (\mathsf{sk}^{(1)}_{\hat{P}}, ..., \mathsf{sk}^{(n)}_{\hat{P}})$. Similarly, the public key $\mathsf{pk}_{\hat{P}}$ of the user with identifier $\hat{P}$ corresponds to the concatenation of n public keys $\mathsf{pk}_{\hat{P}} = (\mathsf{pk}^{(1)}_{\hat{P}}, ..., \mathsf{pk}^{(n)}_{\hat{P}})$.

Setup of the game. A setup algorithm *SetupG* is used to generate a set of a fixed number N of user identifiers, to set all session-specific variables to $\perp$, and to initialize the user memory of each user. The algorithm *SetupG* takes as input the protocol π and the security parameter 1^k, and outputs an initial game state GST_{init}. More precisely, the setup algorithm proceeds as follows:

1. generate a set $\mathcal{P} = \{\hat{P}_1, ..., \hat{P}_N\}$ of N distinct binary strings (representing user identifiers),
2. for all users $\hat{P} \in \mathcal{P}$: generate a long-term public/secret key pair $(\mathsf{pk}_{\hat{P}}, \mathsf{sk}_{\hat{P}})$ using algorithm KeyGen,
3. for all users $\hat{P} \in \mathcal{P}$: store the key pair $(\mathsf{pk}_{\hat{P}}, \mathsf{sk}_{\hat{P}})$ together with the set $\{(\hat{P}, \mathsf{pk}_{\hat{P}}) \mid \hat{P} \in \mathcal{P} \setminus \{\hat{P}\}\}$ in the user memory $st_{\hat{P}}$, and
4. initialize all other user-specific variables if such variables are used by the protocol.

Queries. The specification of some of the queries that we define below is similar to queries defined in the ASICS framework of Chapter 4. The $\mathsf{public\text{-}info}$ query, which was informally introduced in [33, p. 4], allows the adversary to obtain information

that was generated during the setup phase of the game such as the users' identifiers and their public keys.

- $\mathsf{public\text{-}info}()$. The query returns a set $\mathcal{L}$ of information which contains the set $\{(\hat{P}, \mathsf{pk}_{\hat{P}}) \mid \hat{P} \in \mathcal{P}\}$ as well as the initial values of all other variables stored in the user memory of each user, except for the users' long-term secret key, if such variables are used by the protocol.

The queries in the set $Q_{\mathsf{R}} = \{\mathsf{create}, \mathsf{send}\}$ model regular execution of the protocol.

- $\mathsf{create}(\hat{P}, r[, \hat{Q}])$. The query models the creation of a new session s for the user with identifier $\hat{P}$. It requires that $\hat{P} \in \mathcal{P}, \hat{Q} \in \mathcal{P}$, and that $r \in \{\mathcal{I}, \mathcal{R}\}$; otherwise, it returns $\perp$. Session variables are initialized as

$$(s_{actor}, s_{role}, s_{sent}, s_{recv}, s_{status}, s_{key}, s_{step}) \leftarrow (\hat{P}, r, \epsilon, \epsilon, \texttt{active}, \perp, 1) \ .$$

  A bit string in $\{0,1\}^k$ is sampled uniformly at random and assigned to s_{rand}; we assume that all randomness required during the execution of session s is deterministically derived from s_{rand}. If the optional peer identifier $\hat{Q}$ is provided, the variable s_{peer} is set to $\hat{Q}$.
  The key exchange algorithm Ψ is executed on input $(1^k, st_s, st_{\hat{P}}, \epsilon)$. The algorithm returns a triple of elements $(m', st'_s, st'_{\hat{P}})$. We set $st_s \leftarrow st'_s$ and $st_{\hat{P}} \leftarrow st'_{\hat{P}}$. The query returns m'.
- $\mathsf{send}(\hat{P}, i, m)$. The query models sending message m to the i'th session of user $\hat{P}$, which we denote by s. It requires that $s_{status} = \texttt{active}$; otherwise it returns $\perp$. The algorithm Ψ is run on input $(1^k, st_s, st_{\hat{P}}, m)$, and outputs a triple of elements $(m', st'_s, st'_{\hat{P}})$. We set $st_s \leftarrow st'_s$ and $st_{\hat{P}} \leftarrow st'_{\hat{P}}$. The query returns m'.

The queries in the set $Q_{\mathsf{C}} = \{\mathsf{session\text{-}key}, \mathsf{corrupt}, \mathsf{randomness}, \mathsf{cr\text{-}create}\}$ that we define next model the corruption of a user's secrets. The randomness query models the adversary's capability of learning the randomness s_{rand} of a particular session s. In contrast, the cr-create query models the adversary's capability of choosing the randomness used within a session. Note that we do not explicitly model repeated randomness, i. e. secret uniform bits that have been used in previous key exchange sessions. However, we show in Section 5.4.3 that security in the model that we present in Section 5.4.1 implies security in a similar model capturing repeated randomness. The significance of session-key queries is threefold. First, key exchange protocols are required to provide security against known-key attacks [84], which can be ensured through key independence among different sessions. Known-key attacks are captured in security models via session-key queries on non-matching sessions. Second, in an unknown-key share (UKS) attack, the adversary establishes two sessions which compute the same session key even if both sessions have different intended communication partners. The relevance of UKS attacks is discussed, e. g., in [22]. A formal definition of a UKS attack and how it is reflected in a security model via session-key queries is given in [42]. Third, session-key queries capture replay attacks combined with chosen-randomness attacks. In these replay attacks, the adversary causes two non-matching sessions to compute the same session key by setting the randomness of the second session to the same randomness as used in the first session and replaying the messages received by the first session to the second session [108].

In the definition of the queries session-key and randomness we denote the i'th session of user $\hat{P}$ by s.

- session-key$(\hat{P}, i)$. The query requires that $s_{status} = \texttt{accepted}$; otherwise, it returns $\perp$. The query returns the session key s_{key} of session s.
- corrupt$(\hat{P})$. If $\hat{P} \notin \mathcal{P}$, then S returns $\perp$. Otherwise the query returns the long-term secret key $\mathsf{sk}_{\hat{P}}$ of user $\hat{P}$.
- randomness$(\hat{P}, i)$. If $s_{status} \neq \perp$, then the randomness s_{rand} used in session s is returned. Otherwise, the query returns $\perp$.
- cr-create$(\hat{P}, r, rnd[, \hat{Q}])$. The query models the creation of a new session s, using randomness rnd chosen by the adversary, for the user $\hat{P}$. The query requires that $\hat{P} \in \mathcal{P}, \hat{Q} \in \mathcal{P}, rnd \in \{0,1\}^k$, and that $r \in \{\mathcal{I}, \mathcal{R}\}$; otherwise, it returns $\perp$. Session variables are initialized as

  $$(s_{actor}, s_{role}, s_{sent}, s_{recv}, s_{status}, s_{key}, s_{rand}, s_{step}) \leftarrow (\hat{P}, r, \epsilon, \epsilon, \texttt{active}, \perp, rnd, 1) \ .$$

  If the optional peer identifier $\hat{Q}$ is provided, the variable s_{peer} is set to $\hat{Q}$. The key exchange algorithm Ψ is executed on input $(1^k, st_s, st_{\hat{P}}, \epsilon)$. The algorithm returns a triple $(m', st'_s, st'_{\hat{P}})$. We set $st_s \leftarrow st'_s$ and $st_{\hat{P}} \leftarrow st'_{\hat{P}}$. The query returns m'.

The set $Q_{\mathsf{noCR}} = Q_{\mathsf{R}} \cup (Q_{\mathsf{C}} \setminus \{\text{cr-create}\})$ contains all execution and corruption queries, except the query cr-create.

The notion of matching sessions specifies when two sessions are supposed to be intended communication partners. It is formalized below via matching conversations as in Chapter 3.

Definition 33 (Matching sessions)**.** *Let π be an h-message protocol. We say that two sessions s and s' of π are* matching *if $s_{status} = s'_{status} = \texttt{accepted}$ and $s_{actor} = s'_{peer} \wedge s_{peer} = s'_{actor} \wedge s_{sent} = s'_{recv} \wedge s_{recv} = s'_{sent} \wedge s_{role} \neq s'_{role}$.*

We next define a parameterized family of AKE security models. The parameters for each model consist of a subset Q of the above adversary queries and a freshness predicate F, which restricts the adversary from performing certain combinations of queries.

Definition 34 (AKE security model)**.** *Let π be an h-message protocol. Let Q be a set of adversary queries such that $Q_{\mathsf{R}} \subseteq Q \subseteq Q_{\mathsf{R}} \cup Q_{\mathsf{C}}$. Let F be a freshness predicate, that is, a predicate that takes a session of protocol π and a sequence of queries (including arguments and results) in Q. We call (Q, F) an AKE security model.*

Remark 15. In this work we fix a particular definition for matching sessions (namely, Definition 33) and construct strong security models with respect to this definition. It is straightforward to adapt these models to other definitions of matching sessions that are suitable for analyzing protocols such as (H)MQV that allow two sessions performing the same role to compute the same session key.

5.1.2. Security experiment

We associate to each AKE security model $X = (Q, F)$ a security experiment $W(X)$, defined below, played by an adversary E against a challenger. To win the experiment,

the adversary aims to distinguish a real session key from a random key, modelled through the following query.

- test-session(s). This query requires that s_{status} = `accepted`; otherwise, it returns $\perp$. A bit b is chosen at random. If $b = 0$, then s_{key} is returned. If $b = 1$, then a random key is returned according to the probability distribution of keys generated by the protocol.

Definition 35 (Security experiment $W(X)$)**.** *Let π be an h-message protocol. Let $X = (Q, F)$ be an AKE security model. We define experiment $W(X)$, between an adversary E and a challenger who implements all the users, as follows:*

1. *The game is initialized with domain parameters for security parameter 1^k and the setup algorithm SetupG is executed.*
2. *The adversary E first issues the query* public-info*, and then performs any sequence of queries from the set Q.*
3. *At some point in the experiment, E issues a* test-session *query to a session s that has accepted and satisfies F at the time the query is issued.*
4. *The adversary may continue with queries from Q, under the condition that the test session must continue to satisfy F.*
5. *Finally, E outputs a bit b' as his guess for b.*

The adversary E wins the security experiment $W(X)$ if he correctly guesses the bit b chosen by the challenger during the test-session query (i. e., if $b = b'$, where b' is E's guess). Success of E in the experiment is expressed in terms of E's advantage in distinguishing whether he received the real or a random session key in response to the test-session query. The advantage of adversary E in the above security experiment against a key exchange protocol π for security parameter k is defined as $Adv^{\pi,E}_{W(X)}(k) = |2P(b = b') - 1|$.

Definition 36 (AKE security)**.** *A key exchange protocol π is said to be* secure in AKE security model (Q, F) *if, for all PPT adversaries E, it holds that*

- *if two users successfully complete matching sessions, then they compute the same session key,*
- *the probability of event* Multiple-Match$^{W(X)}_{\pi,E}(k)$ *is negligible, where* Multiple-Match$^{W(X)}_{\pi,E}(k)$ *denotes the event that there exists a session that has accepted with at least two matching sessions, and*
- *E has no more than a negligible advantage in winning the $W(X)$ security experiment, that is, there exists a negligible function negl in the security parameter k such that $Adv^{\pi,E}_{W(X)}(k) \leq negl(k)$.*

Informally, the second requirement in Definition 36 (see also [17]) states that, for a given session of protocol π that has accepted, it holds that its matching session, if it exists, is unique.

Remark 16. In comparison to the BR93 security definition [17, Definition 5.1] based on the notion of matching conversations [17, Definition 4.1],

- Definition 33 does not take into account the timing of sent and received messages and

- we do not require in Definition 36 that the probability of event No-Matching$^E(k)$ is negligible, where No-Matching$^E(k)$ denotes the event that there exists a session that has accepted with no matching session.

Let π be a *stateless* two-message AKE protocol (see Definition 40). Adapting [17, Definition 4.1] for two-message protocols by not requiring that the last message sent by a responder session must be received by the initiator session that sent the message received by the responder session, leads us to the conclusion that protocol π is an insecure AKE protocol according to [17, Definition 5.1] since the first requirement of [17, Definition 4.2] is violated; there is an adversary who establishes two sessions that have matching conversations and leaves the initiator session incomplete so that not both sessions accept. Adapting [17, Definition 4.1] for two-message protocols by adding the requirement that two sessions must be completed in order to have matching conversations, leads us to the conclusion that protocol π is an insecure AKE protocol according to [17, Definition 5.1] since the second requirement of [17, Definition 4.2] is violated for π. There is an adversary E such that the probability of event No-Matching$^E(k)$ is non-negligible; E first establishes an initiator session s and a responder session s' that have matching conversations, and then replays the first message from the initiator session s to activate another responder session s''. Clearly, session s'' and session s do not have matching conversations since $\tau_3 > \tau_2$, where the initiator session received the message from the first responder session at time τ_2 and the second responder session received the replayed message from the initiator session at time τ_3. Hence session s'' has accepted with no matching session. So the BR93 security definition combined with either of the two previous variants of the notion of matching conversations is too strong for analyzing stateless two-message AKE protocols as no such protocol can satisfy the BR93 security definition.

5.2. Protocol Classes

In this section we define a series of relevant protocol classes. The largest protocol class includes, e. g., one-round protocols and Diffie-Hellman type protocols. As we see in Section 5.3 and Section 5.5, the distinction between these classes allows the systematic development of strong security models for analyzing protocols belonging to the respective classes.

5.2.1. Classes AKE, INDP, and INDP-DH

We start by defining a global class of AKE protocols. Such protocols are required to be executable, i. e., if the messages of two users $\hat{A}$ and $\hat{B}$ are faithfully relayed to each other, then both users end up with a shared session key (see also [15,17,18]). A second requirement ensures that protocol messages depend on session-specific randomness. Both properties are used for proving impossibility results in Section 5.3.

Definition 37 (Protocol class AKE)**.** *We define* AKE *as the class of all h-message protocols that meet the following requirements: In the presence of an eavesdropping adversary,*

- *two users $\hat{A}$ and $\hat{B}$ can complete matching sessions, in which case they hold the same session key, and*
- *the probability that two sessions of the same user output in all protocol steps identical messages is negligible in the security parameter.*[1]

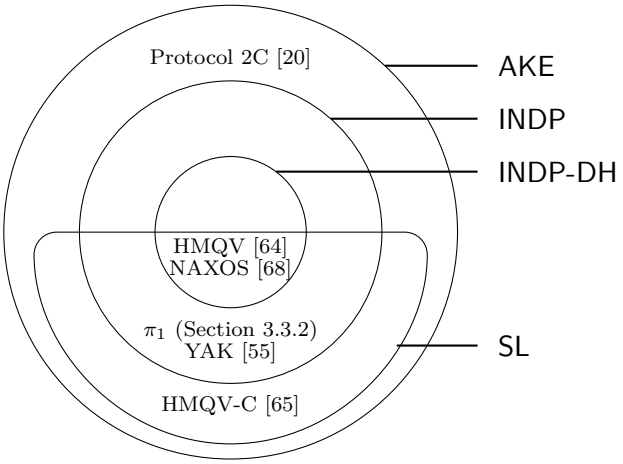

Figure 5.1.: Venn Diagram of the protocol classes and example protocols

We next consider a subclass of two-message AKE protocols, namely the class INDP of one-round AKE protocols, where the outgoing message can be computed before any (valid) message is received. Formally, we define the class of one-round protocols as follows.

Definition 38 (Protocol class INDP)**.** *The protocol class* INDP *consists of all two-message protocols in* AKE *for which the outgoing message of any session s with status $s_{status} \neq$* `rejected` *does not depend on the incoming message.*

We define the class INDP-DH of one-round Diffie-Hellman type protocols as a subclass of INDP as follows.

Definition 39 (Protocol class INDP-DH)**.** *Let $G = \langle g \rangle$ be a cyclic group of prime order p generated by g. Let* KeyGen *be the key generation algorithm defined as* KeyGen()*: Choose $a \in_R [0, p-1]$; Set $A \leftarrow g^a$; Return secret key* $\mathsf{sk} = a$ *and public key* $\mathsf{pk} = A$*.*

The protocol class INDP-DH *consists of all protocols in the class* INDP *with domain parameters (G, g, p), key generation algorithm* KeyGen*, and where the outgoing message of any initiator session s in status $s_{status} \neq$* `rejected` *is of the form $(\delta, g^{f_{\mathcal{I}}(1^k, st_s, st_{\hat{A}})})$ and the outgoing message of any responder session s' in status $s'_{status} \neq$* `rejected` *is of the form $(\delta', g^{f_{\mathcal{R}}(1^k, st_{s'}, st_{\hat{B}})})$, where $\hat{A}$ is the actor of session s, $\hat{B}$ is the actor of session s', $f_{\mathcal{I}}, f_{\mathcal{R}} : \{0,1\}^* \to \mathbb{Z}_p$ are two polynomial-time computable functions, and δ, δ' is optional publicly available information such as the identifiers of actor and peer of the session.*

[1] Note that the class AKE does not include static Diffie-Hellman protocols such as Protocol 2 in [23].

5.2.2. Stateless and stateful protocols

We distinguish between stateless and stateful protocols. Stateless protocols leave the state of a user's memory (i.e., the memory that is shared among sessions) invariant under execution of the protocol. In contrast, the state of a user's memory is modified when executing a protocol that is not stateless. Examples of stateless and stateful protocols are given in Figure 5.1 and in Section 5.6.

We define the class of stateless AKE protocols as a subclass of the class AKE as follows.

Definition 40 (Stateless protocol)**.** *Let $\mathcal{A}, \mathcal{B}$, and $\mathcal{C}$ be sets. Let* $\mathrm{proj}_3 : \mathcal{A} \times \mathcal{B} \times \mathcal{C} \to \mathcal{C}$ *be the map given by* $\mathrm{proj}_3(a, b, c) = c$ *for all* $(a, b, c) \in \mathcal{A} \times \mathcal{B} \times \mathcal{C}$*. Let π be a protocol in the class* AKE*. We say that π is a* stateless protocol *if*

$$\mathrm{proj}_3(\Psi(1^k, st_s, st_{\hat{P}}, m)) = st_{\hat{P}},$$

for all $(k, st_s, st_{\hat{P}}, m) \in \mathbb{N} \times \{st_s \mid s \in \mathcal{P} \times \mathbb{N}\} \times \{st_{\hat{P}} \mid \hat{P} \in \mathcal{P}\} \times \{0,1\}^*$*. We denote by* SL *the subclass of* AKE *containing all stateless protocols.*

Definition 41 (Stateful protocol)**.** *Let π be a protocol in the class* AKE*. We say that π is a* stateful protocol *if π is not stateless.*

Remark 17. Stateless protocols cannot provide message replay detection to reject messages that have been received in earlier sessions of the same user as this would require storing all previously received messages in a table in the user memory and, upon receipt of a valid message in a session, accessing the table in the user memory and checking whether the message corresponds to a message in the table.

Most recently proposed strong AKE protocols fall into the narrow class INDP-DH∩SL. As we illustrate next, protocols from the wider classes can achieve stronger security.

5.3. Impossibility results and strong models for stateless protocols

In this section we provide impossibility results for protocols in the classes AKE ∩ SL, INDP ∩ SL and INDP-DH ∩ SL with respect to an adversary who is given access to the queries in the set $Q_{\mathsf{noCR}} = Q_{\mathsf{R}} \cup (Q_{\mathsf{C}} \setminus \{\mathsf{cr\text{-}create}\})$. We then derive strong security models for reasoning about the security of protocols in the respective classes from these impossibility results. Each of our models can be satisfied by existing stateless protocols. In Section 5.4 we show that no stateless protocol can achieve security in stronger models, in which the adversary can additionally perform chosen-randomness attacks via the cr-create query.

We start by defining the notion of partially matching sessions in a similar way as matching conversations in [17, Definition 4.1].

Definition 42 (Partially matching sessions)**.** *Let π be an h-message protocol, where $h \geq 2$. Let s denote a session of π with $s_{status} =$* `accepted`*. We say that session s is* partially matching *session s' in status $s'_{status} \neq \perp$ if the following conditions hold:*

- $s_{role} \neq s'_{role} \wedge s_{actor} = s'_{peer} \wedge s_{peer} = s'_{actor}$ *and either*
- $s_{role} = \mathcal{I} \wedge s_{send}[1..m] = s'_{recv}[1..m] \wedge s_{recv}[1..m] = s'_{send}[1..m]$ *with* $m = \frac{h}{2}$ *if* h *is even and* $m = \frac{h-1}{2}$ *if* h *is odd, or*
- $s_{role} = \mathcal{R} \wedge s_{send}[1..(m-1)] = s'_{recv}[1..(m-1)] \wedge s_{recv}[1..m] = s'_{send}[1..m]$ *with* $m = \frac{h}{2}$ *if* h *is even and* $m = \frac{h+1}{2}$ *if* h *is odd,*

where $s_{send}[1..l]$ *denotes the concatenation of the first* l *messages sent by session* s *and* $s_{recv}[1..l]$ *denotes the concatenation of the first* l *messages received by session* s*.*

Remark 18. In contrast to the notion of matching sessions (see Definition 33), which is only defined for completed sessions, the notion of partially matching sessions does not require the last message sent by one session to be received by the partner session. This allows us to capture combinations of randomness and corrupt attacks on such sessions as well.

To relate a received message that was not constructed by the adversary to the session it originates from, we use the concept of origin-session, which was first introduced in Section 3.1.2. The existence of an origin-session for a given session implies integrity of the received messages.

Definition 43 (origin-session)**.** *We say that a session* s' *with* $s'_{status} \neq \perp$ *is an origin-session for a session* s *with* $s_{status} = \texttt{accepted}$ *if* $s'_{send} = s_{recv}$*.*

Theorem 5 (Impossibility result for $\mathsf{AKE} \cap \mathsf{SL}$)**.** *Let* π *be an arbitrary protocol in the class* $\mathsf{AKE} \cap \mathsf{SL}$*. Let* $X = (Q_{\mathsf{noCR}}, F)$ *be the AKE security model with* F *being true for all sessions* s *and all sequences of queries. Let* s^* *denote the test session and let* s' *denote a session such that* s^* *is partially matching session* s'*. There exist adversaries who win the security experiment* $W(X)$ *against protocol* π *with non-negligible probability by issuing either*

1. *a query* $\mathsf{session\text{-}key}(s^*)$*, or*
2. *a query* $\mathsf{session\text{-}key}(\tilde{s})$*, where* $\tilde{s}$ *and* s^* *are matching sessions, or*
3. *a query* $\mathsf{corrupt}(s^*_{actor})$ *and a query* $\mathsf{randomness}(s^*)$*, or*
4. *a query* $\mathsf{corrupt}(s^*_{peer})$ *as well as a query* $\mathsf{randomness}(s')$*, or*
5. *a query* $\mathsf{corrupt}(s^*_{peer})$ *before creation of session* s^* *via a* create *query or as long as* $s^*_{status} = \texttt{active}$ *and* $s^*_{recv} = \epsilon$*, and impersonating the peer to session* s^**.*

Proof. Let $\pi \in \mathsf{AKE} \cap \mathsf{SL}$. There exist PPT adversaries who win the security game $W(X)$ against protocol π with non-negligible probability, as follows.

Scenario 1**:** Adversary E_1 establishes two sessions s and s' that are matching (via a sequence of create and send queries) and chooses session s as the test session. Issuing a $\mathsf{session\text{-}key}(s)$ query reveals the session key of session s.

Scenario 2**:** Since $\pi \in \mathsf{AKE}$, adversary E_2 can establish via a sequence of create and send queries two sessions s and s' that are matching according to Definition 33. He then issues the test-session query to one of the two sessions, say to session s, and issues a session-key query to session s'. E_2 thereby learns the session key of session s.

Scenario 3**:** Adversary E_3 establishes two sessions s and s' that are matching and chooses session s as the test session. Issuing the queries $\mathsf{corrupt}(s_{actor})$ and

$\mathsf{randomness}(s)$ reveals the long-term secret key of $s_{actor} = \hat{P}$ and the randomness s_{rand} of session s, respectively. Together with the set $\mathcal{L}$ returned as response to the query public-info and the last message received during session s, which we denote by m', the adversary can compute the session key of session s by executing the algorithm Ψ on input $(1^k, st_s, st_{\hat{P}}, m')$. Note that the user state remains unchanged during protocol execution as $\pi \in \mathsf{SL}$.

Scenario 4: Assume that $\pi \in \mathsf{AKE} \cap \mathsf{SL}$ is an h-message protocol with h even. The proof works similarly for the case where h is odd.
Case 1: test session in initiator role. First E_4 establishes via a sequence of create and send queries an initiator session s and a responder session s' such that s and s' are matching sessions (this is possible since $\pi \in \mathsf{AKE}$). Clearly, session s is also partially matching session s'. He then chooses session s as the test session. Issuing the queries $\mathsf{corrupt}(s_{peer})$ and $\mathsf{randomness}(s')$ reveals the long-term secret key of $s_{peer} = \hat{P}$ and the randomness s'_{rand} of session s', respectively. Since $\pi \in \mathsf{SL}$ and the initial values of additional variables in the user memory are returned as response to the query public-info, the user state $st_{\hat{P}}$ is known to the adversary. Hence, the adversary can emulate the session key computation of a matching session and compute the session key of session s by executing Ψ on input $(1^k, st_{s'}, st_{\hat{P}}, m)$, where m denotes the last incoming message to session s'.
Case 2: test session in responder role. First E_4 establishes via a sequence of create and send queries an incomplete initiator session s and a responder session s' such that s' is partially matching session s (this is possible since $\pi \in \mathsf{AKE}$). He then chooses session s' as the test session. Issuing the queries $\mathsf{corrupt}(s'_{peer})$ and $\mathsf{randomness}(s)$ reveals the long-term secret key of s'_{peer} and the randomness of session s, respectively. Similar to the previous case, the adversary is able to compute the session key of session s'.

Scenario 5: Adversary E_5 issues a corrupt query to some user, say user $\hat{Q}$. He then creates a responder session s by issuing the query $\mathsf{create}(\hat{P}, \mathcal{R}, \hat{Q})$. The adversary now impersonates user $\hat{Q}$ to s_{actor} as follows. E_5 chooses randomness $r \in_R \{0,1\}^k$ and runs the protocol with $\hat{P}$ on behalf of $\hat{Q}$ by executing the algorithm Ψ. The algorithm Ψ executed by the adversary takes as input, among others, the user state of user $\hat{Q}$ containing its long-term secret key $\mathsf{sk}_{\hat{Q}}$ and the set $\mathcal{L}$ returned as response to the public-info query. Once session s has accepted, he chooses the latter as the test session. The adversary can compute the session key of session s, for which no origin-session exists, by emulating a matching session. Note that a similar attack also works against (a) an initiator session, (b) an initiator session s such that $s_{status} = \texttt{active}$ and $s_{recv} = \epsilon$, and (c) a responder session s such that $s_{status} = \texttt{active}$ and $s_{recv} = \epsilon$.

□

Theorem 5 gives rise to the security model $\Omega_{\mathsf{AKE}\cap\mathsf{SL}}$ defined as follows. The associated freshness notion restricts the adversary from performing the generic attacks specified in Theorem 5.

Definition 44 ($\Omega_{\mathsf{AKE}\cap\mathsf{SL}}$)**.** *The $\Omega_{\mathsf{AKE}\cap\mathsf{SL}}$ model is defined by (Q, F), where $Q = Q_{\mathsf{noCR}}$ and a session is said to satisfy F if all of the following conditions hold:*

1. *no* session-key(s) *query has been issued,*
2. *for all sessions* s^* *such that* s^* *matches* s*, no* session-key(s^*) *query has been issued,*
3. *not both queries* corrupt(s_{actor}) *and* randomness(s) *have been issued,*
4. *for all sessions* s' *such that* s *is partially matching session* s'*, not both queries* corrupt(s_{peer}) *and* randomness(s') *have been issued, and*
5. *if there exists no origin-session for session* s*, then no* corrupt(s_{peer}) *query has been issued before creation of session* s *via a* create *query or as long as* $s_{status} = \texttt{active}$ *and* $s_{recv} = \epsilon$.

To see that there exist protocols in the class $\mathsf{AKE} \cap \mathsf{SL}$ that are secure in the model $\Omega_{\mathsf{AKE}\cap\mathsf{SL}}$, consider the protocol SIG*(NAXOS) obtained by applying the signature transformation SIG with optional fields from Chapter 3 to the NAXOS protocol. Clearly, protocol SIG*(NAXOS) does not belong to the class $\mathsf{INDP} \cap \mathsf{SL}$ since the outgoing message of a responder session depends on the Diffie-Hellman exponential contained in the incoming message. Adapting the proof of Theorem 1, it can be shown that protocol SIG*(NAXOS) is secure in model $\Omega_{\mathsf{AKE}\cap\mathsf{SL}}$.

Even though security in model $\Omega_{\mathsf{AKE}\cap\mathsf{SL}}$ can be achieved by protocols in the class $\mathsf{AKE} \cap \mathsf{SL}$, we next show that no protocol in the class INDP can provide these strong security guarantees.

Proposition 9 (Impossibility result for INDP)**.** *No protocol in the class* INDP *can satisfy security in the model* $\Omega_{\mathsf{AKE}\cap\mathsf{SL}}$.

Proof. Let π be an arbitrary protocol in INDP. There exists an adversary E who wins the $W(\Omega_{\mathsf{AKE}\cap\mathsf{SL}})$ experiment against the challenger with non-negligible probability as follows. The adversary E first reveals the long-term secret key of some user $\hat{A}$ by issuing the query corrupt($\hat{A}$). Since E knows the values of all the variables stored in the user memory of user $\hat{A}$ (through the queries public-info and corrupt($\hat{A}$)), E can run the first protocol execution step on behalf of $\hat{A}$ by executing the deterministic algorithm Ψ on input $(1^k, st_s, st_{\hat{A}}, \epsilon)$, where the values of the state st_s of the emulated session s are as follows, $s_{actor} = \hat{A}$, $s_{peer} = \hat{B}$, $s_{role} = \mathcal{I}$, $s_{sent} = \epsilon$, $s_{recv} = \epsilon$, $s_{status} = \texttt{active}$, $s_{key} = \perp$, $s_{rand} = z$ (with $z \in_R \{0,1\}^k$), $s_{data} = \perp$, and $s_{step} = 1$. The algorithm Ψ outputs a message m, an updated session state, and an updated state of the user memory. The adversary then creates an initiator session s_1 of user $\hat{A}$ with peer $\hat{B}$ via the query create($\hat{A}, \mathcal{I}, \hat{B}$). The latter query returns a message m_1. He then creates a responder session s_2 of user $\hat{B}$ via the query create($\hat{B}, \mathcal{R}, \hat{A}$), and sends the message m, previously obtained by executing Ψ, to session s_2 via the query send($\hat{B}, 1, m$). As a response, the adversary receives the message m_2, which he sends to session s_1 via a send query. Upon receiving message m_2 in session s_1, $\hat{A}$ executes $\Psi(1^k, st_{s_1}, st_{\hat{A}}, m_2)$ and thereby completes the session. E now chooses the completed session s_1 as the test session, and reveals the long-term secret key of the peer of the test session, namely user $\hat{B}$, as well as the randomness of session s_2. This enables him to compute the session key of the test session by executing Ψ on input $(1^k, st_{\tilde{s}}, st_{\hat{B}}, m_1)$, where m_1 is the outgoing message of session s_1 and the values of the session state $st_{\tilde{s}}$ are as follows, $s_{actor} = \hat{B}$, $s_{peer} = \hat{A}$, $s_{role} = \mathcal{R}$, $s_{sent} = \epsilon$, $s_{recv} = \epsilon$, $s_{status} = \texttt{active}$, $s_{key} = \perp$, s_{rand} is the randomness of session s_2, $s_{step} = 2$, and s_{data} is obtained by executing the

first protocol step. E thereby emulates a matching session of user $\hat{B}$. The previous attack shows that protocol π is insecure in the $\Omega_{\mathsf{AKE}\cap\mathsf{SL}}$ model. Note that the test session is fresh in $\Omega_{\mathsf{AKE}\cap\mathsf{SL}}$ since there is no freshness condition on a session that is an origin-session for the test session but no partially matching session for the test session in the definition of $F_{\Omega_{\mathsf{AKE}\cap\mathsf{SL}}}$. Also, there is no matching session for the test session s_1 since the message m sent by the adversary to session s_2 is different, with overwhelming probability, from the message m_1 output by session s_1, by Definition 37. □

The notion of c-origin-session is a stronger notion than origin-session as the former additionally requires distinct roles and agreement on the communicating users.

Definition 45 (c-origin-session)**.** *We say that a session s' with $s'_{status} \neq \perp$ is a c-origin-session for a session s with $s_{status} = \texttt{accepted}$ if $s_{actor} = s'_{peer} \wedge s_{peer} = s'_{actor} \wedge s_{recv} = s'_{sent} \wedge s_{role} \neq s'_{role}$.*

Note that for protocols in the class INDP the last condition of Definition 44 can be simplified. This follows from the fact that a created initiator session s of any protocol in the class INDP completes upon receipt of a valid message m and the variable $s_{recv} = \epsilon$ is then updated with message m, i. e. $s_{recv} \leftarrow (s_{recv}, m)$. Consequently, the point in time of receipt of the first message m in an initiator session s coincides with the completion of session s. The same reasoning is applicable to responder sessions. It follows from (a) the previous observation, (b) the model $\Omega_{\mathsf{AKE}\cap\mathsf{SL}}$, and (c) Proposition 9 that a strong model for analyzing protocols in the class $\mathsf{INDP}\cap\mathsf{SL}$ is model $\Omega_{\mathsf{INDP}\cap\mathsf{SL}}$ defined as follows.

Definition 46 ($\Omega_{\mathsf{INDP}\cap\mathsf{SL}}$)**.** *The $\Omega_{\mathsf{INDP}\cap\mathsf{SL}}$ model is defined by (Q, F), where $Q = Q_{\mathsf{noCR}}$, and a session s is said to satisfy F if all of the following conditions hold:*

1. *no* session-key(s) *query has been issued,*
2. *for all sessions s^* such that s^* matches s, no* session-key(s^*) *query has been issued,*
3. *not both queries* corrupt(s_{actor}) *and* randomness(s) *have been issued,*
4. *for all sessions s' such that s' is a c-origin-session for session s, not both queries* corrupt(s_{peer}) *and* randomness(s') *have been issued, and*
5. *if there exists no origin-session for session s, then no* corrupt(s_{peer}) *query has been issued before the completion of session s.*

Remark 19. Compared to the notion of freshness in model $\Omega_{\mathsf{AKE}\cap\mathsf{SL}}$, freshness of a session s in $\Omega_{\mathsf{INDP}\cap\mathsf{SL}}$ requires that the adversary does not issue a corrupt query to the peer of session s as well as a randomness query to a c-origin-session for session s. This restriction prevents the attack described in the proof of Proposition 9.

It is not difficult to verify that protocol $\mathrm{SIG_t}$(NAXOS) obtained by applying a tagged version of the signature transformation SIG from Chapter 3 to the NAXOS protocol is secure in model $\Omega_{\mathsf{INDP}\cap\mathsf{SL}}$. The outgoing message of a $\mathrm{SIG_t}$(NAXOS) initiator session of user $\hat{A}$ is of the form $(X, Sign_{\hat{A}}(0, X, \hat{B}))$ while the outgoing message of a $\mathrm{SIG_t}$(NAXOS) responder session of user $\hat{B}$ is of the form $(Y, Sign_{\hat{B}}(1, Y, \hat{A}))$.

We next show that any protocol in the class INDP-DH is insecure in $\Omega_{\mathsf{INDP}\cap\mathsf{SL}}$.

Proposition 10 (Impossibility result for INDP-DH)**.** *No one-round Diffie-Hellman type protocol in the class* INDP-DH *can satisfy security in the model* $\Omega_{\mathsf{INDP}\cap\mathsf{SL}}$*.*

Proof. Let π be an arbitrary protocol in INDP-DH. There exists an adversary E who wins the $W(\Omega_{\mathsf{INDP}\cap\mathsf{SL}})$ experiment against the challenger with non-negligible probability as follows. The adversary E first creates an initiator session s at $\hat{A}$ with peer $\hat{B}$ via the query $\mathsf{create}(\hat{A}, \mathcal{I}, \hat{B})$ and receives as a response the message $m = (\delta, X)$, where X is the Diffie-Hellman exponential generated in session s and δ denotes optional public information. E chooses a value $z \in_R \mathbb{Z}_q$, computes $Z = g^z$, and sends message $\tilde{m} = (\delta', Z)$ to session s. Upon receiving message $\tilde{m}$ in session s, $\hat{A}$ executes $\Psi(1^k, st_s, st_{\hat{A}}, \tilde{m})$. E then chooses the completed session s as the test session and reveals the long-term secret key of user $\hat{B}$ via the query $\mathsf{corrupt}(\hat{B})$. This enables him to compute the session key of the test session by executing Ψ on input $(1^k, st_{\tilde{s}}, st_{\hat{B}}, m)$, where $st_{\tilde{s}}$ is the state of the emulated matching session $\tilde{s}$. Note that the query public-info returned the initial values of additional variables stored in the user memory. This attack shows that π is insecure in $\Omega_{\mathsf{INDP}\cap\mathsf{SL}}$. It is a generalized version of the attack described by Krawczyk in [64, p. 15]. □

Remark 20. Any protocol $\pi \in \mathsf{INDP\text{-}DH} \cap \mathsf{SL}$ that does not contain sufficient public information in the outgoing message is insecure in the model derived from model $\Omega_{\mathsf{INDP}\cap\mathsf{SL}}$ and Proposition 10. This follows from the fact that a redirect event of a message from a session of a different user than the test session's peer can cause the existence of an origin-session that is not a c-origin-session for the test session. However, we do not distinguish between the relative strength of AKE protocols based on public information within the outgoing messages; adding such information does not provide additional security guarantees as it can be altered by the adversary.

From the previous remark and Proposition 10 we derive the following security model.

Definition 47 ($\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{SL}}$)**.** *The* $\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{SL}}$ *model is defined by* (Q, F)*, where* $Q = Q_{\mathsf{noCR}}$ *and a session* s *is said to satisfy* F *if all of the following conditions hold:*

1. *no* $\mathsf{session\text{-}key}(s)$ *query has been issued, and*
2. *for all sessions* s^* *such that* s^* *matches* s*, no* $\mathsf{session\text{-}key}(s^*)$ *query has been issued, and*
3. *not both queries* $\mathsf{corrupt}(s_{actor})$ *and* $\mathsf{randomness}(s)$ *have been issued, and*
4. *for all sessions* s' *such that* s' *is an origin-session for session* s*, not both queries* $\mathsf{corrupt}(s_{peer})$ *and* $\mathsf{randomness}(s')$ *have been issued, and*
5. *if there exists no origin-session for session* s*, then no* $\mathsf{corrupt}(s_{peer})$ *query has been issued.*

The model $\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{SL}}$ is very similar to the eCK^w model defined in Section 3.1.2. The NAXOS protocol [68] provides an example of a protocol in the class INDP-DH∩SL that is secure in the eCK^w model (see Section 3.3.1).

5.4. Models capturing chosen-randomness attacks

5.4.1. Deriving models with chosen-randomness

As an immediate consequence of Theorem 5, we obtain Theorem 6, which generalizes our impossibility results on protocol class $\mathsf{AKE} \cap \mathsf{SL}$ to adversaries who are in addition given access to the query cr-create.

Theorem 6 (Impossibility result for $\mathsf{AKE} \cap \mathsf{SL}$ under chosen-randomness)**.** *Let π be an arbitrary protocol in the class $\mathsf{AKE} \cap \mathsf{SL}$. Let $X = (Q_{\mathsf{noCR}} \cup \{\mathsf{cr\text{-}create}\}, F)$ be the AKE security model with F being true for all sessions s and all sequences of queries. Let s^* denote the test session and let s' denote a session such that s^* is partially matching session s'. There exist adversaries who win the security experiment $W(X)$ against protocol π with non-negligible probability by issuing*

1. *a query* session-key(s^*)*, or*
2. *a query* session-key$(\tilde{s})$*, where $\tilde{s}$ and s^* are matching sessions, or*
3. *a query* corrupt(s^*_{actor}) *and a (*randomness *or* cr-create*) query to session s^*, or*
4. *a query* corrupt(s^*_{peer}) *as well as a (*randomness *or* cr-create*) query to session s', or*
5. *a query* corrupt(s^*_{peer}) *before creation of session s^* via a (*create *or* cr-create*) query or as long as $s^*_{status} =$* `active` *and $s^*_{recv} = \epsilon$, and impersonating the peer to the test session s^*.*

The previous theorem gives rise to the security model Ω^-_{AKE} defined as follows.

Definition 48 (Ω^-_{AKE})**.** *The Ω^-_{AKE} model is defined by (Q, F), where $Q = Q_{\mathsf{noCR}} \cup \{\mathsf{cr\text{-}create}\}$, and a session s is said to satisfy F if all of the following conditions hold:*

1. *no* session-key(s) *query has been issued,*
2. *for all sessions s^* such that s^* matches s, no* session-key(s^*) *query has been issued,*
3. *not both queries* corrupt(s_{actor}) *and (*randomness(s) *or* cr-create$(.)$ *creating session s) have been issued,*
4. *for all sessions s' such that s is partially matching session s', not both queries* corrupt(s_{peer}) *and (*randomness(s') *or* cr-create$(.)$ *creating session s') have been issued, and*
5. *if there exists no origin-session for session s, then no* corrupt(s_{peer}) *query has been issued before creation of session s via a (*create *or* cr-create*) query or as long as $s_{status} =$* `active` *and $s_{recv} = \epsilon$.*

The models Ω^-_{INDP} and $\Omega^-_{\mathsf{INDP\text{-}DH}}$, defined below, are obtained from the models $\Omega_{\mathsf{INDP} \cap \mathsf{SL}}$ and $\Omega_{\mathsf{INDP\text{-}DH} \cap \mathsf{SL}}$, respectively, in a similar way as model Ω^-_{AKE} is obtained from model $\Omega_{\mathsf{AKE} \cap \mathsf{SL}}$.

Definition 49 (Ω^-_{INDP})**.** *The Ω^-_{INDP} model is defined by (Q, F), where $Q = Q_{\mathsf{noCR}} \cup \{\mathsf{cr\text{-}create}\}$, and a session s is said to satisfy F if all of the following conditions hold:*

1. *no* session-key(s) *query has been issued,*
2. *for all sessions s^* such that s^* matches s, no* session-key(s^*) *query has been issued,*

3. *not both queries* corrupt(s_{actor}) *and (*randomness(s) *or* cr-create(.) *creating session* s*) have been issued,*
4. *for all sessions* s' *such that* s' *is a c-origin-session for session* s*, not both queries* corrupt(s_{peer}) *and (*randomness(s') *or* cr-create(.) *creating session* s'*) have been issued, and*
5. *if there exists no origin-session for session* s*, then no* corrupt(s_{peer}) *query has been issued before the completion of session* s*.*

Definition 50 ($\Omega^{-}_{\mathsf{INDP\text{-}DH}}$)**.** *The* $\Omega^{-}_{\mathsf{INDP\text{-}DH}}$ *model is defined by* (Q, F)*, where* $Q = Q_{\mathsf{noCR}} \cup \{\mathsf{cr\text{-}create}\}$ *and a session* s *is said to satisfy* F *if all of the following conditions hold:*

1. *no* session-key(s) *query has been issued, and*
2. *for all sessions* s^* *such that* s^* *matches* s*, no* session-key(s^*) *query has been issued, and*
3. *not both queries* corrupt(s_{actor}) *and (*randomness(s) *or* cr-create(.) *creating session* s*) have been issued, and*
4. *for all sessions* s' *such that* s' *is an origin-session for session* s*, not both queries* corrupt(s_{peer}) *and (*randomness(s') *or* cr-create(.) *creating session* s'*) have been issued, and*
5. *if there exists no origin-session for session* s*, then no* corrupt(s_{peer}) *query has been issued.*

We show in Section 5.4.3 that security in our extended models, which additionally capture attacks based on chosen randomness, implies security against attacks exploiting repeated randomness failures.

5.4.2. Insecurity of stateless protocols against chosen-randomness attacks

Even though the following proposition states that no stateless protocol is secure in model $\Omega^{-}_{\mathsf{INDP\text{-}DH}}$, we show in Section 3.3 that stateful protocols can in fact achieve these stronger security guarantees.

Proposition 11 (Impossibility result for $\mathsf{AKE} \cap \mathsf{SL}$ under chosen-randomness)**.** *No protocol in the class* $\mathsf{AKE} \cap \mathsf{SL}$ *can satisfy security in the model* $\Omega^{-}_{\mathsf{INDP\text{-}DH}}$*.*

Proof. Let π be an arbitrary protocol in $\mathsf{AKE} \cap \mathsf{SL}$. There exists an adversary E that wins the $\Omega^{-}_{\mathsf{INDP\text{-}DH}}$ game against the challenger with non-negligible probability as follows. The adversary E first completes a regular execution between two users $\hat{A}$ and $\hat{B}$, i.e. $\hat{A}$ and $\hat{B}$ complete matching sessions s and s', respectively. He then issues a randomness query against the responder session s' of user $\hat{B}$. E creates another responder session s'' of user $\hat{B}$ via the query cr-create($\hat{B}, \mathcal{R}, str, \hat{A}$), where str denotes the randomness that he revealed from session s', and replays the messages from session s to session s''. As the randomness used in session s'' is identical to the randomness used in session s' and $\pi \in \mathsf{SL}$, the messages that E receives from session s'' are the same as the messages sent by session s'. Now, E chooses the completed session s' as the test session, and reveals the session key computed in session s'' via a session-key(s'')

query. As the session keys computed in sessions s' and s'' are the same and both sessions are non-matching, the adversary learns the session key of the test session. Hence, the protocol π is insecure in the $\Omega^{-}_{\mathsf{INDP\text{-}DH}}$ model. A similar attack that involves cr-create queries on both sessions s' and s'' is sketched in [108, p. 119]. □

Corollary 5. *No protocol in the class* $\mathsf{AKE} \cap \mathsf{SL}$ *can satisfy security in either model* $\Omega^{-}_{\mathsf{INDP}}$ *or model* $\Omega^{-}_{\mathsf{AKE}}$.

Proof. By Proposition 11, we know that no protocol in the class $\mathsf{AKE} \cap \mathsf{SL}$ can satisfy security in the model $\Omega^{-}_{\mathsf{INDP\text{-}DH}}$. The corollary now follows from the fact that the models $\Omega^{-}_{\mathsf{INDP}}$ and $\Omega^{-}_{\mathsf{AKE}}$ are both at least as strong as model $\Omega^{-}_{\mathsf{INDP\text{-}DH}}$ (by Proposition 17). □

5.4.3. Repeated randomness failures

In this section we show that security in the $\Omega^{-}_{\mathsf{AKE}}$ model implies security against repeated randomness. To this end, we compare the relative strength of security between the model $\Omega^{-}_{\mathsf{AKE}}$ and a very similar model to $\Omega^{-}_{\mathsf{AKE}}$, where the adversary is not given access to the query cr-create, but to the query reset-create. The latter query allows the adversary to create a session that uses the same randomness as used in a previous session of the same user. Practically, this models a flawed RNG that produces the same value more than once.

The query reset-create creates a new session with the same randomness as used in a previous session of the same user.

- reset-create$(\hat{P}, r, i[, \hat{Q}])$. The query models the creation of a new session s, using the same randomness as in session $s' = (\hat{P}, i)$, for the user $\hat{P}$. The query requires that $\hat{P} \in \mathcal{P}, \hat{Q} \in \mathcal{P}, r \in \{\mathcal{I}, \mathcal{R}\}$, and that $s'_{status} \neq \perp$; otherwise, it returns $\perp$. Session variables are initialized as

  $$(s_{actor}, s_{role}, s_{sent}, s_{recv}, s_{status}, s_{key}, s_{rand}, s_{step}) \leftarrow (\hat{P}, r, \epsilon, \epsilon, \texttt{active}, \perp, s'_{rand}, 1) \ .$$

  If the optional peer identifier $\hat{Q}$ is provided, the variable s_{peer} is set to $\hat{Q}$.
  The key exchange algorithm Ψ is executed on input $(1^k, st_s, st_{\hat{P}}, \epsilon)$. The algorithm returns a triple $(m', st'_s, st'_{\hat{P}})$. We set $st_s \leftarrow st'_s$ and $st_{\hat{P}} \leftarrow st'_{\hat{P}}$. The query returns m'.

Consider the security model $X_{\mathsf{AKE}} = (Q_{\mathsf{noCR}} \cup \{\mathsf{reset\text{-}create}\}, F)$, where a session $s = (\hat{P}, i)$ is said to satisfy F if all of the following conditions hold:

1. no session-key(s) query has been issued,
2. for all sessions s^* such that s^* matches s, no session-key(s^*) query has been issued,
3. not both queries corrupt$(\hat{P})$ and (randomness(s) or reset-create(.) creating session s) have been issued,
4. for all sessions s' such that s is partially matching session s', not both queries corrupt(s_{peer}) and (randomness(s') or reset-create(.) creating session s') have been issued, and
5. if there exists no origin-session for session s, then no corrupt(s_{peer}) query has been issued before creation of session s or as long as $s_{status} = \texttt{active}$ and $s_{recv} = \epsilon$.

Proposition 12. *Let* $\Pi = \mathsf{AKE}$. *The model* $\Omega^{-}_{\mathsf{AKE}}$ *is at least as strong as the model* X_{AKE} *with respect to* Π *according to Definition 17.*

Proof. The first condition of Definition 36 is satisfied since matching is defined in the same way for both models $\Omega^{-}_{\mathsf{AKE}}$ and X_{AKE}. Let $\pi \in \Pi$. To show that the second and third condition of Definition 36 hold, we construct an adversary E' attacking protocol π in model $\Omega^{-}_{\mathsf{AKE}}$ using an adversary E attacking π in model X_{AKE}. Adversary E' proceeds as follows. Whenever E issues a query create, corrupt, randomness, session-key or test-session, adversary E' issues the same query and forwards the answer received to E. Whenever E issues a query reset-create$(\hat{P}, r, i[, \hat{Q}])$ to create a new session of user $\hat{P}$, adversary E' first checks whether the status of session $s = (\hat{P}, i)$ is different from $\perp$. If this is the case, then E' issues the following sequence of queries: 1. randomness$(\hat{P}, i)$, and 2. cr-create$(\hat{P}, r, s_{rand}[, \hat{Q}])$. At the end of E's execution, i. e. after it has output its guess bit b, E' outputs b as well. Note that if F holds for the test session, then the freshness condition of model $\Omega^{-}_{\mathsf{AKE}}$ is also satisfied. In particular, if there exists a partially matching session for the test session and the actors of both sessions are the same, then there is no corrupt query on that user in case a query reset-create was issued to create either of the two sessions. If there exists a partially matching session for the test session and the actors of both sessions are different, then a reset-create query on the test session only involves a query cr-create and a query randomness on another session of the test session's actor; issuing a corrupt query on the test session's peer does not render the test session un-fresh in model $\Omega^{-}_{\mathsf{AKE}}$ as long as there is no query randomness on the partially matching session. Hence, it holds that $Adv^{\pi,E}_{W(X_{\mathsf{AKE}})}(k) \leq Adv^{\pi,E'}_{W(\Omega^{-}_{\mathsf{AKE}})}(k)$, where k denotes the security parameter. Since by assumption protocol π is secure in $\Omega^{-}_{\mathsf{AKE}}$, there is a negligible function g such that $Adv^{\pi,E'}_{W(\Omega^{-}_{\mathsf{AKE}})}(k) \leq g(k)$. It follows that protocol π is secure in X_{AKE}. □

We obtain the following corollary as an immediate consequence of Proposition 12.

Corollary 6. *Let* $\Pi = \mathsf{AKE}$. *The model* $\Omega^{-}_{\mathsf{AKE}}$ *is at least as strong as the model* $Y_{\mathsf{AKE}} = (Q_{\mathsf{R}} \cup \{\mathsf{corrupt}, \mathsf{session\text{-}key}, \mathsf{reset\text{-}create}\}, F')$ *with respect to* Π *according to Definition 17, where* F' *is obtained from predicate* F *above by removing the* randomness *query from the conditions.*

Proof. Since $X_{\mathsf{AKE}} \leq^{\Pi}_{Sec} \Omega^{-}_{\mathsf{AKE}}$ (by Proposition 12) and $Y_{\mathsf{AKE}} \leq^{\Pi}_{Sec} X_{\mathsf{AKE}}$ (by a similar reduction proof as in the proof of Proposition 12), it follows that $Y_{\mathsf{AKE}} \leq^{\Pi}_{Sec} \Omega^{-}_{\mathsf{AKE}}$ by transitivity of Implication (3.1). □

5.5. Impossibility results and strong models for stateful protocols

We next present a generalized impossibility result and its derived model for the broad class AKE taking into account the adversary's ability to choose session-specific randomness. Recall that the completion of a session occurs at the time at which the status of the session is set to `accepted`.

Corollary 7 (Impossibility result for AKE)**.** *Let π be an arbitrary protocol in the class* AKE*. Let $X = (Q_{\mathsf{noCR}} \cup \{\mathsf{cr\text{-}create}\}, F)$ be the AKE security model with F being true for all sessions s and all sequences of queries. Let s^* denote the test session and let s' denote a session such that s^* is partially matching session s'. There exist adversaries who win the security experiment $W(X)$ against protocol π with non-negligible probability by issuing either*

1. *a query* session-key(s^*)*, or*
2. *a query* session-key$(\tilde{s})$*, where $\tilde{s}$ and s^* are matching sessions, or*
3. *a query* corrupt(s^*_{actor}) *as well as (*randomness *or* cr-create*) queries on all sessions s with $s_{actor} = s^*_{actor}$, where the query* create *or* cr-create *creating session s occurred before completion of session s^*, or*
4. *a query* corrupt(s^*_{peer}) *as well as (*randomness *or* cr-create*) queries on all sessions s with $s_{actor} = s'_{actor}$, where the query* create *or* cr-create *creating session s occurred before completion of session s', or*
5. *a query* corrupt(s^*_{peer}) *before creation of session s^* via a query (*create *or* cr-create*) or as long as $s^*_{status} =$* `active` *and $s^*_{recv} = \epsilon$, and impersonating the peer to the test session s^*.*

Proof. Corollary 7 follows from Theorem 5 and the following observation. Given the initial value of all the variables stored in the user memory of user $\hat{P}$ returned as response to the query public-info, the randomness used in all sessions of user $\hat{P}$ that are created before completion of the test session, and the long-term secret key of $\hat{P}$, the adversary can emulate the protocol execution steps on behalf of user $\hat{P}$ by executing Ψ sequentially to recompute the session-specific and user-specific data from the first session of user $\hat{P}$ to the relevant session of user $\hat{P}$. □

Remark 21 (On Corollary 7). Consider a protocol $\pi \in$ AKE storing previously seen messages to prevent replay attacks. Then, in case a session s receives a message m sent by some session s', the user memory of user s_{actor} gets updated after receiving this message. Thus, if another session s'' of user s_{actor}, created before s was created, is activated via a query send with the same message m later on, then session s'' is aborted as it detected a previously received message (even if the session was created before session s). Similar situations might occur if updates of the user state occurring in later steps of the protocol execution involve the randomness of the session. Therefore, some of the attacks in Corollary 7 require the randomness of all sessions created prior to completion of the target session or its partially matching session.

Corollary 7 gives rise to the model Ω_{AKE} defined as follows.

Definition 51 (Ω_{AKE})**.** *The model Ω_{AKE} is defined by (Q, F), where $Q = Q_{\mathsf{noCR}} \cup \{\mathsf{cr\text{-}create}\}$ and a session s is said to satisfy F if all of the following conditions hold:*

1. *no* session-key(s) *has been issued,*
2. *for all sessions s^* such that s^* matches s, no* session-key(s^*) *query has been issued,*
3. *not all queries* corrupt(s_{actor}) *as well as (*randomness *or* cr-create*) queries on all sessions $\tilde{s}$ with $\tilde{s}_{actor} = s_{actor}$, where the query* create *or* cr-create *creating session $\tilde{s}$ occurred before completion of session s, have been issued,*

4. *for all sessions s' such that s is partially matching session s', not all queries* corrupt(s_{peer}) *as well as (*randomness *or* cr-create*) on all sessions $\tilde{s}$ with $\tilde{s}_{actor} = s'_{actor}$, where the query* create *or* cr-create *creating session $\tilde{s}$ occurred before completion of session s', have been issued, and*
5. *if there exists no origin-session for session s, then no* corrupt(s_{peer}) *query has been issued before creation of session s via a query (*create *or* cr-create*) or as long as $s_{status} =$* `active` *and $s_{recv} = \epsilon$.*

In this work we restrict ourselves to the subclass ISM (Initial State Modification) of the class AKE and provide strong models for analyzing protocols in this subclass. The class ISM contains all AKE protocols that may *only* access and update user memory upon creation of sessions. It contains, e. g., the CNX protocol as well as the NXPR protocol, which we present in Section 5.6.[2] We leave the analysis of protocols that update user memory at later steps in the protocol execution as future work.

From the definition of the class ISM and from Corollary 7, we derive the following model.

Definition 52 ($\Omega_{\mathsf{AKE}\cap\mathsf{ISM}}$)**.** *The model $\Omega_{\mathsf{AKE}\cap\mathsf{ISM}}$ is defined by (Q, F), where $Q = Q_{\mathsf{noCR}} \cup \{\text{cr-create}\}$ and a session s is said to satisfy F if all of the following conditions hold:*

1. *no* session-key(s) *has been issued,*
2. *for all sessions s^* such that s^* matches s, no* session-key(s^*) *query has been issued,*
3. *not all queries* corrupt(s_{actor}) *as well as (*randomness *or* cr-create*) on all sessions $\tilde{s}$ with $\tilde{s}_{actor} = s_{actor}$, where the query* create *or* cr-create *creating session $\tilde{s}$ occurred before or at creation of session s, have been issued,*
4. *for all sessions s' such that s is partially matching session s', not all queries* corrupt(s_{peer}) *as well as (*randomness *or* cr-create*) on all sessions $\tilde{s}$ with $\tilde{s}_{actor} = s'_{actor}$, where the query* create *or* cr-create *creating session $\tilde{s}$ occurred before or at creation of session s', have been issued, and*
5. *if there exists no origin-session for session s, then no* corrupt(s_{peer}) *query has been issued before creation of session s via a query (*create *or* cr-create*) or as long as $s_{status} =$* `active` *and $s_{recv} = \epsilon$.*

Whereas in the models that we defined in Section 5.3 and Section 5.4, the adversary is not allowed to compromise both the randomness of the target session and the long-term secret key of the actor of the target session, the adversary is allowed to compromise the latter values in the $\Omega_{\mathsf{AKE}\cap\mathsf{ISM}}$ model and its descendants as long as the randomness of at least one of the previous sessions of the actor of the target session has not been compromised.

Definition 52 and Proposition 9 give rise to the model $\Omega_{\mathsf{INDP}\cap\mathsf{ISM}}$ defined below. The third condition in Definition 53 prevents the adversary from both corrupting the actor of the target session and revealing the randomness of all sessions of this user

[2] Note that this subclass does not contain protocols providing message replay detection as such protocols need to access and update the list of received messages stored in the user memory upon receipt of a message. However the subclass contains protocols such as NAXOS and CMQV, which do not modify the user memory.

that were created prior to creation of the target session as well as the randomness of the target session itself. As the user memory is accessed and updated upon creation of sessions only, the responses for the previous queries would allow the adversary to emulate the protocol execution steps for the target session. The fourth condition specifies a similar requirement for sessions that are c-origin-sessions for the target session.

Definition 53 ($\Omega_{\mathsf{INDP}\cap\mathsf{ISM}}$)**.** *The model* $\Omega_{\mathsf{INDP}\cap\mathsf{ISM}}$ *is defined by* (Q, F)*, where* $Q = Q_{\mathsf{noCR}} \cup \{\mathsf{cr\text{-}create}\}$ *and a session* s *is said to satisfy* F *if all of the following conditions hold:*

1. *no* $\mathsf{session\text{-}key}(s)$ *has been issued,*
2. *for all sessions* s^* *such that* s^* *matches* s*, no* $\mathsf{session\text{-}key}(s^*)$ *query has been issued,*
3. *not all queries* $\mathsf{corrupt}(s_{actor})$ *as well as (*$\mathsf{randomness}$ *or* $\mathsf{cr\text{-}create}$*) on all sessions* $\tilde{s}$ *with* $\tilde{s}_{actor} = s_{actor}$*, where the query* create *or* $\mathsf{cr\text{-}create}$ *creating session* $\tilde{s}$ *occurred before or at creation of session* s*, have been issued,*
4. *for all sessions* s' *such that* s' *is a c-origin-session for session* s*, not all queries* $\mathsf{corrupt}(s_{peer})$ *as well as (*$\mathsf{randomness}$ *or* $\mathsf{cr\text{-}create}$*) on all sessions* $\tilde{s}$ *with* $\tilde{s}_{actor} = s'_{actor}$*, where the query* create *or* $\mathsf{cr\text{-}create}$ *creating session* $\tilde{s}$ *occurred before or at creation of session* s'*, have been issued, and*
5. *if there exists no origin-session for session* s*, then no* $\mathsf{corrupt}(s_{peer})$ *query has been issued before the completion of session* s*.*

Definition 52 and Propositions 9 and 10 give rise to the model $\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{ISM}}$ defined as follows.

Definition 54 ($\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{ISM}}$)**.** *The model* $\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{ISM}}$ *is defined by* (Q, F)*, where* $Q = Q_{\mathsf{noCR}} \cup \{\mathsf{cr\text{-}create}\}$ *and a session* s *is said to satisfy* F *if all of the following conditions hold:*

1. *no* $\mathsf{session\text{-}key}(s)$ *has been issued,*
2. *for all sessions* s^* *such that* s^* *matches* s*, no* $\mathsf{session\text{-}key}(s^*)$ *query has been issued,*
3. *not all queries* $\mathsf{corrupt}(s_{actor})$ *as well as (*$\mathsf{randomness}$ *or* $\mathsf{cr\text{-}create}$*) on all sessions* $\tilde{s}$ *with* $\tilde{s}_{actor} = s_{actor}$*, where the query* create *or* $\mathsf{cr\text{-}create}$ *creating session* $\tilde{s}$ *occurred before or at creation of session* s*, have been issued,*
4. *for all sessions* s' *such that* s' *is an origin-session for session* s*, not all queries* $\mathsf{corrupt}(s_{peer})$ *as well as (*$\mathsf{randomness}$ *or* $\mathsf{cr\text{-}create}$*) on all sessions* $\tilde{s}$ *with* $\tilde{s}_{actor} = s'_{actor}$*, where the query* create *or* $\mathsf{cr\text{-}create}$ *creating session* $\tilde{s}$ *occurred before or at creation of session* s'*, have been issued, and*
5. *if there exists no origin-session for session* s*, then no* $\mathsf{corrupt}(s_{peer})$ *query has been issued.*

We formally study the relations between the different security models that we established in this chapter in Section 5.7.

Remark 22 (comparison with [31])*.* Boyd and González Nieto [31] show that one-round AKE protocols that do not provide message replay detection cannot achieve PFS if the adversary can reveal session-specific randomness of the target session's peer. The

attack on which their argument is based is disallowed in model $\Omega_{\mathsf{INDP}\cap\mathsf{ISM}}$, but only if the replayed message originates from the first created session of the target session's peer. In case the target session receives a message replayed from the n'th session of its peer, where $n > 1$, then the adversary is allowed to compromise the randomness of the n'th session of the peer as well as the long-term secret key of the peer after the end of the target session as long as he does not reveal the randomness of the previous $n-1$ sessions of the peer. Our results thus reflect the impossibility result of Boyd and González Nieto on the class of protocols ISM only if the replayed message originates from the first session of the target session's peer. In particular, the attack in the proof of Theorem 5 from which we derived the fourth condition in Definition 44 can be seen as a generalized version of Boyd and González Nieto's attack.

5.6. Construction of strongly secure stateful protocols

In the previous sections we derived strong models capturing chosen-randomness attacks from impossibility results on given protocol classes. We have seen that stateless protocols fail to achieve security in these models. In this section we provide stateful variants of the NAXOS protocol [68], which are secure against attacks based on chosen randomness.

5.6.1. Protocol CNX

The CNX protocol ("Counter-NaXos"), shown in Figure 5.2, is a variant of the NAXOS protocol [68] and provides an example of a protocol from the class INDP-DH \ SL, where $H_1 : \{0,1\}^* \to \mathbb{Z}_p$ and $H_2 : \{0,1\}^* \to \{0,1\}^k$ denote two hash functions. In contrast to the NAXOS protocol, protocol CNX additionally includes a global counter value, which is shared across the sessions of a user, as input to the hash function H_1. We assume that each user maintains a counter l, taking values in $\mathbb{N}$, initialized with 0 and incremented by one upon creation of a new session. This counter variable is stored in the user memory. We write $st_{\hat{P}}.l$ to access the counter variable l of user $\hat{P}$. The CNX protocol proceeds as follows.

1. Upon creation of a new initiator session $s = (\hat{A}, i)$ with a $\mathsf{create}(\hat{A}, \mathcal{I}, \hat{B})$ query, user $\hat{A}$ increments the counter variable $st_{\hat{A}}.l \leftarrow st_{\hat{A}}.l + 1$, sets $s_{data} \leftarrow st_{\hat{A}}.l$, computes an outgoing public key $X \leftarrow g^{H_1(s_{rand}, a, s_{data})}$, and returns X as an outgoing message.
2. Upon creation of a new responder session $s' = (\hat{B}, j)$ with a $\mathsf{create}(\hat{B}, \mathcal{R}, \hat{A})$ query, user $\hat{B}$ increments the counter variable $st_{\hat{B}}.l \leftarrow st_{\hat{B}}.l + 1$ and sets $s'_{data} \leftarrow st_{\hat{B}}.l$.
3. When the responder receives message X via a $\mathsf{send}(\hat{B}, j, X)$ query, he computes an outgoing public key $Y \leftarrow g^{H_1(s'_{rand}, b, s'_{data})}$ and returns Y as an outgoing message. He computes a session key $s'_{key} \leftarrow H_2(A^{H_1(s'_{rand}, b, s'_{data})}, X^b, X^{H_1(s'_{rand}, b, s'_{data})}, \hat{A}, \hat{B})$ and accepts $s'_{status} \leftarrow \texttt{accepted}$.
4. When the initiator receives message Y via a $\mathsf{send}(\hat{A}, i, Y)$ query, he computes a session key $s_{key} \leftarrow H_2(Y^a, B^{H_1(s_{rand}, a, s_{data})}, Y^{H_1(s_{rand}, a, s_{data})}, \hat{A}, \hat{B})$ and accepts $s_{status} \leftarrow \texttt{accepted}$.

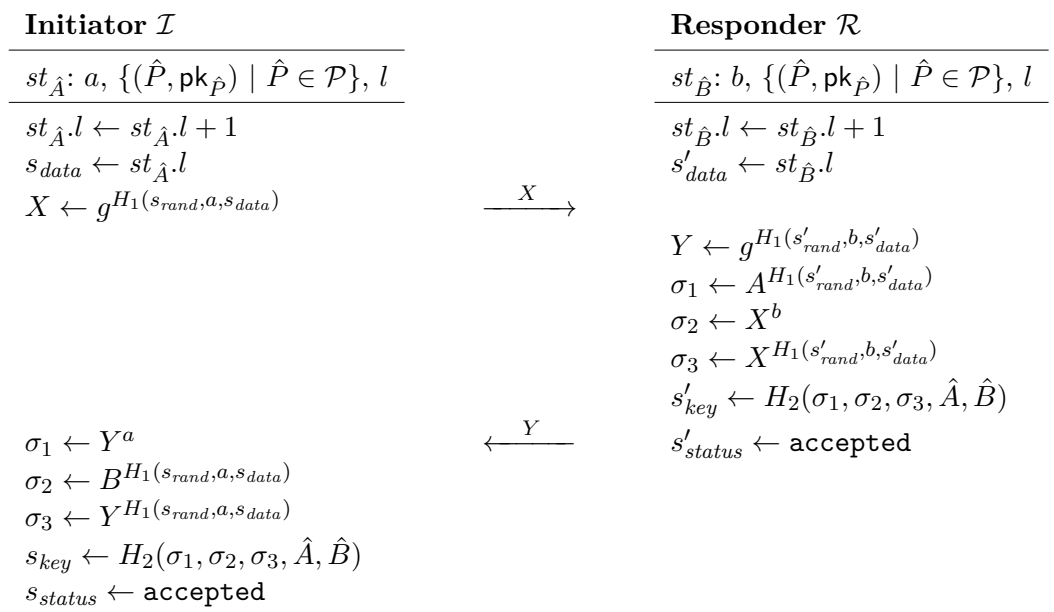

Figure 5.2.: CNX protocol

Remark 23. In the specification of the CNX protocol given in Figure 5.2, the variable s_{data} stores the current value of the counter. The H_1 values, which depend on the value of the counter, are recomputed in the session key computation. In particular, it is essential for initiator sessions to store the counter value in the variable s_{data} to prevent the use of a different counter value in the session key computation due to other activations of sessions of the same user in between creation of the session and completion of the session. As an alternative specification of the CNX protocol, one could consider storing the exponent $H_1(s_{rand}, a, st_{\hat{A}}.l)$ in the variable s_{data}. Then users would not need to recompute the H_1 value in the session key computation.

The following proposition states that the CNX protocol is secure in model $\Omega^{-}_{\mathsf{INDP\text{-}DH}}$.

Proposition 13. *Under the Gap Diffie-Hellman assumption in the cyclic group G of prime order p, the* CNX *protocol is secure in model $\Omega^{-}_{\mathsf{INDP\text{-}DH}}$, when H_1, H_2 are modeled as independent random oracles.*

We refer the reader to Appendix B.1 for the proof of Proposition 13.

By a straightforward adaptation of the proof of Theorem 1 via integration of the cr-create query into the security models, we can show that protocol $\mathrm{SIG_t}$(CNX) obtained by applying the tagged version of the signature transformation SIG suggested in Section 5.3 to the CNX protocol is secure in model $\Omega^{-}_{\mathsf{INDP}}$. Similarly, we can show that protocol SIG*(CNX), obtained by applying the signature transformation SIG with optional fields to the CNX protocol, is secure in model $\Omega^{-}_{\mathsf{AKE}}$.

Remark 24 (sequence numbers). In contrast to the use of sequence numbers to uniquely identify messages, the recipient of a message during execution of the CNX protocol does not need to store and maintain state information of each possible verifier to detect previously used sequence numbers (see also [84, p. 399]). However, as stated

in [31], protocols using sequence numbers to order messages are secure against Boyd and González Nieto's replay attack [31, p. 458]. Moreover such protocols are secure against reset-and-replay attacks. Thus, designers of protocols face a trade-off between efficiency and security against various types of replay attacks.

Remark 25 (comparison with [108]). Yang et al. [108] argue that whenever the randomness of one session is identical to the randomness of another session of the same user, the adversary can learn the session key of either of the two sessions by performing a replay attack combined with a session-key query (as both sessions compute the same session key, but are non-matching). While Proposition 11 confirms that this statement holds for all protocols in the class AKE∩SL, we have shown that there exists a protocol in AKE, namely CNX, that achieves security even under such reset-and-replay attacks against the target session.

5.6.2. Protocol NXPR

Even though the CNX protocol is secure in model $\Omega^{-}_{\mathsf{INDP\text{-}DH}}$, it fails to achieve security in the stronger model $\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{ISM}}$, as the following proposition shows.

Proposition 14. *The* CNX *protocol is insecure in model* $\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{ISM}}$.

Proof. The following attack shows that the CNX protocol is insecure in $\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{ISM}}$. The adversary creates an initiator session s of user $\hat{A}$ via the query $\mathsf{create}(\hat{A},\mathcal{I},\hat{B})$ and an initiator session of user $\hat{B}$ by issuing the query $\mathsf{create}(\hat{B},\mathcal{I},\hat{C})$. He then creates a responder session s' via the query $\mathsf{create}(\hat{B},\mathcal{R},\hat{A})$ and activates session s' by sending the message $X = g^x$ sent by session s to session s'. The adversary then sends message Y sent by session s' to session s. Session s accepts the key $s_{key} = H_2(Y^a, B^{H_1(s_{rand},a,s_{data})}, Y^{H_1(s_{rand},a,s_{data})}, \hat{A}, \hat{B})$ as the session key, while session s' accepts as its key $s'_{key} = H_2(A^{H_1(s'_{rand},b,s'_{data})}, X^b, X^{H_1(s'_{rand},b,s'_{data})}, \hat{A}, \hat{B})$. The completed session s is chosen as the test session. Now a randomness query to session s' revealing the randomness of session s' followed by a $\mathsf{corrupt}(\hat{B})$ query revealing the long-term secret key of user $\hat{B}$, allows the adversary to compute the session key of the test session s (as he knows the counter value used in session s'). Note that the test session is fresh in $\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{ISM}}$ since the adversary did not issue the query randomness or cr-create to the first created session of user $\hat{B}$. □

The NXPR protocol ("NaXos with Previous Randomness"), shown in Figure 5.3, is a variant of the NAXOS protocol [68] and provides an example of a protocol from the class INDP-DH ∩ ISM, where $H_1 : \{0,1\}^* \to \mathbb{Z}_p$ and $H_2 : \{0,1\}^* \to \{0,1\}^k$ denote two hash functions. In contrast to the NAXOS protocol, protocol NXPR additionally includes the randomness of all sessions that have been previously created as input to the hash function H_1. We assume that each user maintains a variable $l \in \{0,1\}^*$ initialized with the empty string ϵ. We write $st_{\hat{P}}.l$ to access the variable l stored in the user memory of user $\hat{P}$.

Proposition 15. *Under the Gap Diffie-Hellman assumption in the cyclic group* G *of prime order* p*, the* NXPR *protocol is secure in the* $\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{ISM}}$ *model, when* H_1, H_2 *are modeled as independent random oracles.*

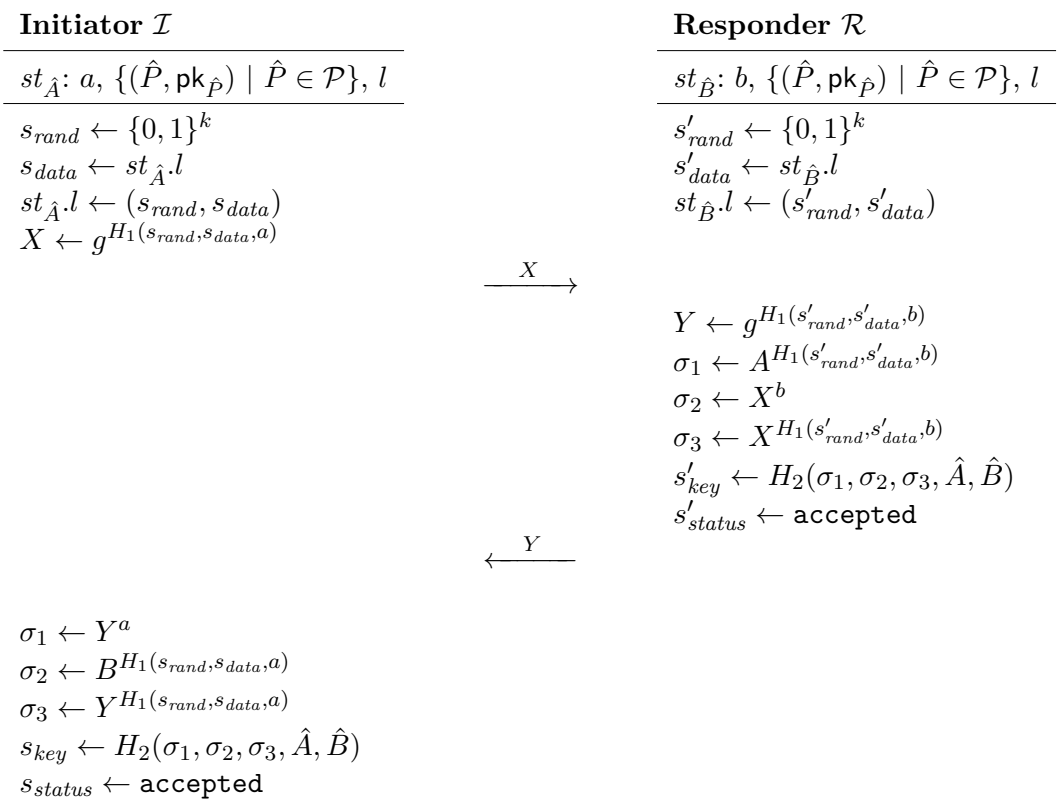

Figure 5.3.: NXPR Protocol

We refer the reader to Appendix B.2 for the proof of Proposition 15.

By a straightforward adaptation of the proof of Theorem 1 via integration of the cr-create query into the security models, we can show that protocol $\mathrm{SIG_t}$(NXPR) obtained by applying the tagged version of the signature transformation SIG suggested in Section 5.3 to the NXPR protocol is secure in model $\Omega_{\mathsf{INDP} \cap \mathsf{ISM}}$. Similarly, we can show that protocol SIG*(NXPR), obtained by applying the signature transformation SIG with optional fields to the NXPR protocol, is secure in model $\Omega_{\mathsf{AKE} \cap \mathsf{ISM}}$.

Remark 26 (user state comparison of NXPR to CNX). In contrast to the CNX protocol, the NXPR protocol requires each user to store the concatenation of the randomness generated in all his sessions in the user memory. Thus, before the n'th session, where $n > 1$, of user $\hat{P}$ is created, the user memory $st_{\hat{P}}$ contains its long-term secret key, the long-term public key of all users $\hat{Q} \in \mathcal{P}$, and the concatenation of $n-1$ bit strings of length k corresponding to the randomness generated in all previous sessions of user $\hat{P}$.

5.7. Relations between the security models

In this section we illustrate the relations between the different security models of this chapter. Recall that the relation of relative strength of security between security models has been formally defined in Section 3.1.4.

Proposition 16. *Let* Π *be protocol class* AKE*.*

- *The model* $\Omega_{\mathsf{INDP}\cap\mathsf{SL}}$ *is stronger than the model* $\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{SL}}$ *with respect to* Π *according to Definition 17.*
- *The model* $\Omega_{\mathsf{AKE}\cap\mathsf{SL}}$ *is stronger than the model* $\Omega_{\mathsf{INDP}\cap\mathsf{SL}}$ *with respect to* Π *according to Definition 17.*

Proof. We first show that $\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{SL}} \leq_{Sec}^{\mathsf{AKE}} \Omega_{\mathsf{INDP}\cap\mathsf{SL}}$. The proof that $\Omega_{\mathsf{INDP}\cap\mathsf{SL}} \leq_{Sec}^{\mathsf{AKE}} \Omega_{\mathsf{AKE}\cap\mathsf{SL}}$ proceeds in a very similar way. The first condition of Definition 36 is satisfied as matching is defined in the same way for both models $\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{SL}}$ and $\Omega_{\mathsf{INDP}\cap\mathsf{SL}}$. To see that the second condition of Definition 36 holds, it suffices to show that if there exists an adversary E such that the probability of event Multiple-Match$_{\pi,E}^{W(\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{SL}})}(k)$ is non-negligible, then there exists an adversary E' such that the probability of event Multiple-Match$_{\pi,E}^{W(\Omega_{\mathsf{INDP}\cap\mathsf{SL}})}(k)$ is non-negligible. This is straightforward. Let $\pi \in \Pi$. To show that the third condition of Definition 36 holds, we construct an adversary E' attacking protocol π in model $\Omega_{\mathsf{INDP}\cap\mathsf{SL}}$ using an adversary E attacking π in $\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{SL}}$. Adversary E' proceeds as follows. Whenever E issues a query $q \in Q_{\mathsf{noCR}} \cup \{\mathsf{test\text{-}session}\}$, adversary E' issues the same query and forwards the answer received to E. At the end of E's execution, i. e., after it has output its guess bit b, E' outputs b as well. Note that if the freshness condition of $\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{SL}}$ holds for the test session, then by definition the freshness condition of $\Omega_{\mathsf{INDP}\cap\mathsf{SL}}$ also holds. First, if there is no origin-session, then the fifth condition of the freshness condition of model $\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{SL}}$ requires that there is no corrupt query has been issued on the peer of the test session, which implies the fifth condition of the freshness condition of model $\Omega_{\mathsf{INDP}\cap\mathsf{SL}}$. Second, if there is an origin-session s for the test session, then it is required in model $\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{SL}}$ that not both queries corrupt on the peer of the test session and randomness on session s have been issued. Now, if session s is a c-origin-session for the test session, then the fourth condition of the freshness definition of model $\Omega_{\mathsf{INDP}\cap\mathsf{SL}}$ is satisfied. However, if session s is not a c-origin-session for the test session, then freshness in the model $\Omega_{\mathsf{INDP}\cap\mathsf{SL}}$ is guaranteed as well. Even though both of the critical queries are allowed in the model $\Omega_{\mathsf{INDP}\cap\mathsf{SL}}$, they do not occur since otherwise freshness in the model $\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{SL}}$ would be violated. Hence, it holds that $Adv_{W(X)}^{\pi,E}(k) \leq Adv_{W(X)}^{\pi,E'}(k)$, where k denotes the security parameter. Since by assumption protocol π is secure in model $\Omega_{\mathsf{INDP}\cap\mathsf{SL}}$, there is a negligible function g such that $Adv_{W(X)}^{\pi,E'}(k) \leq g(k)$. It follows that protocol π is secure in model $\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{SL}}$.

The NAXOS protocol provides an example of a protocol that is secure in the model $\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{SL}}$, but insecure in $\Omega_{\mathsf{INDP}\cap\mathsf{SL}}$. The protocol $\mathrm{SIG_t}$(NAXOS) introduced in Section 5.3 provides an example of a protocol that is secure in $\Omega_{\mathsf{INDP}\cap\mathsf{SL}}$, but insecure in $\Omega_{\mathsf{AKE}\cap\mathsf{SL}}$ as the following replay attack shows.

1. First the adversary establishes via a sequence of create and send queries an initiator session s of user $\hat{A}$ and a responder session s' of user $\hat{B}$ such that s and s' are matching sessions.
2. The adversary creates another initiator session s'' of user $\hat{A}$ with peer $\hat{B}$, and replays the message sent by session s' to session s''. Clearly, session s' is a c-origin-session for session s'', but s'' is not partially matching session s'.
3. He then chooses session s'' as the test session and issues the queries corrupt($\hat{B}$) and randomness(s').

4. Since the adversary knows the user state of user $\hat{B}$ and the randomness of session s', he can emulate the session key computation of a matching session and compute the session key of session s'' by executing Ψ on input $(1^k, st_{s'_p}, st_{\hat{B}}, s''_{sent})$, where $st_{s'_p}$ denotes the session-specific memory of session s' after its creation, but before its completion.

□

Proposition 17. *Let* Π *be protocol class* AKE*.*

- *The model* $\Omega^{-}_{\mathsf{INDP}}$ *is stronger than the model* $\Omega^{-}_{\mathsf{INDP\text{-}DH}}$ *with respect to* Π *according to Definition 17.*
- *The model* $\Omega^{-}_{\mathsf{AKE}}$ *is stronger than the model* $\Omega^{-}_{\mathsf{INDP}}$ *with respect to* Π *according to Definition 17.*

Proof. The proofs that model $\Omega^{-}_{\mathsf{INDP}}$ is at least as strong as the model $\Omega^{-}_{\mathsf{INDP\text{-}DH}}$ and that model $\Omega^{-}_{\mathsf{AKE}}$ is at least as strong as the model $\Omega^{-}_{\mathsf{INDP}}$ are similar to the proof of Proposition 16.

The CNX protocol provides an example of a protocol that is secure in model $\Omega^{-}_{\mathsf{INDP\text{-}DH}}$, but insecure in model $\Omega^{-}_{\mathsf{INDP}}$ due to a PFS attack. The protocol $\mathrm{SIG_t}$(CNX) introduced in Section 5.6 provides an example of a protocol that is secure in model $\Omega^{-}_{\mathsf{INDP}}$, but insecure in model $\Omega^{-}_{\mathsf{AKE}}$ due to the same replay attack as the one on the protocol $\mathrm{SIG_t}$(NAXOS) described in the proof of Proposition 16. □

Proposition 18. *Let* Π *be protocol class* AKE*.*

- *The model* $\Omega^{-}_{\mathsf{INDP\text{-}DH}}$ *is stronger than the model* $\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{SL}}$ *with respect to* Π *according to Definition 17.*
- *The model* $\Omega^{-}_{\mathsf{INDP}}$ *is stronger than the model* $\Omega_{\mathsf{INDP}\cap\mathsf{SL}}$ *with respect to* Π *according to Definition 17.*
- *The model* $\Omega^{-}_{\mathsf{AKE}}$ *is stronger than the model* $\Omega_{\mathsf{AKE}\cap\mathsf{SL}}$ *with respect to* Π *according to Definition 17.*

Proof. The proof that model $\Omega^{-}_{\mathsf{INDP\text{-}DH}}$ is at least as strong as the model $\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{SL}}$ proceeds in a similar way as the proof of Proposition 16. The same holds for the other pairs of models.

The protocols SIG^*(NAXOS), $\mathrm{SIG_t}$(NAXOS), and NAXOS provide examples of protocols that are secure in the models $\Omega_{\mathsf{AKE}\cap\mathsf{SL}}$, $\Omega_{\mathsf{INDP}\cap\mathsf{SL}}$, and $\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{SL}}$, respectively. As all of these protocols are stateless, they fail to provide security in our models that give the adversary access to the query cr-create, by Proposition 11. □

Proposition 19. *Let* Π *be protocol class* AKE*.*

- *The model* $\Omega_{\mathsf{INDP}\cap\mathsf{ISM}}$ *is stronger than the model* $\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{ISM}}$ *with respect to* Π *according to Definition 17.*
- *The model* $\Omega_{\mathsf{AKE}\cap\mathsf{ISM}}$ *is stronger than the model* $\Omega_{\mathsf{INDP}\cap\mathsf{ISM}}$ *with respect to* Π *according to Definition 17.*

Proof. The proof is similar to the proof of Proposition 16.

To prove that model $\Omega_{\mathsf{INDP}\cap\mathsf{ISM}}$ is stronger than model $\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{ISM}}$, we need to show, among others, that the third condition of Definition 36 holds. Thus, we

construct an adversary E' attacking protocol π in model $\Omega_{\mathsf{INDP}\cap\mathsf{ISM}}$ using an adversary E attacking π in model $\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{ISM}}$, where $\pi \in \mathsf{AKE}$. Whenever E issues a query $q \in Q_{\mathsf{noCR}} \cup \{\mathsf{cr\text{-}create}, \mathsf{test\text{-}session}\}$, adversary E' issues the same query and forwards the answer received to E. Note that if the freshness condition of $\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{ISM}}$ holds for the test session, then by definition the freshness condition of $\Omega_{\mathsf{INDP}\cap\mathsf{ISM}}$ also holds. First, if there is no origin-session, then the fifth condition of the freshness condition of model $\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{ISM}}$ requires that there is no $\mathsf{corrupt}$ query has been issued on the peer of the test session, which implies Condition 5 of the freshness condition of model $\Omega_{\mathsf{INDP}\cap\mathsf{ISM}}$. The case where there is an origin-session for the test session is treated is the same way as in the proof of Proposition 16. The proof that $\Omega_{\mathsf{INDP}\cap\mathsf{ISM}} \leq_{Sec}^{\mathsf{AKE}} \Omega_{\mathsf{AKE}\cap\mathsf{ISM}}$ proceeds in a very similar way.

The NXPR protocol presented in Section 5.6.2 provides an example of a protocol that is secure in the model $\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{ISM}}$, but insecure in $\Omega_{\mathsf{INDP}\cap\mathsf{ISM}}$ due to a PFS attack. The protocol $\mathrm{SIG_t}$(NXPR) introduced in Section 5.6.2 provides an example of a protocol that is secure in model $\Omega_{\mathsf{INDP}\cap\mathsf{ISM}}$, but insecure in model $\Omega_{\mathsf{AKE}\cap\mathsf{ISM}}$ due to the same replay attack as the one on the protocol $\mathrm{SIG_t}$(NAXOS) described in the proof of Proposition 16.

□

Proposition 20. *Let* Π *be protocol class* AKE*.*

- *The model* $\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{ISM}}$ *is stronger than the model* $\Omega^{-}_{\mathsf{INDP\text{-}DH}}$ *with respect to* Π *according to Definition 17.*
- *The model* $\Omega_{\mathsf{INDP}\cap\mathsf{ISM}}$ *is stronger than the model* $\Omega^{-}_{\mathsf{INDP}}$ *with respect to* Π *according to Definition 17.*
- *The model* $\Omega_{\mathsf{AKE}\cap\mathsf{ISM}}$ *is stronger than the model* $\Omega^{-}_{\mathsf{AKE}}$ *with respect to* Π *according to Definition 17.*

Proof. The proof that model $\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{ISM}}$ is at least as strong as the model $\Omega^{-}_{\mathsf{INDP\text{-}DH}}$ proceeds in a similar way as the proof of Proposition 16. Among others, we need to show that the third condition of Definition 36 holds. Thus, we construct an adversary E' attacking protocol π in model $\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{ISM}}$ using an adversary E attacking π in model $\Omega^{-}_{\mathsf{INDP\text{-}DH}}$, where $\pi \in \mathsf{AKE}$. Whenever E issues a query $q \in Q_{\mathsf{noCR}} \cup \{\mathsf{cr\text{-}create}, \mathsf{test\text{-}session}\}$, adversary E' issues the same query and forwards the answer received to E. Note that if the freshness condition of $\Omega^{-}_{\mathsf{INDP\text{-}DH}}$ is satisfied for the test session s, then the freshness condition of $\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{ISM}}$ is also satisfied. The third condition of freshness in model $\Omega^{-}_{\mathsf{INDP\text{-}DH}}$ requires that not both queries $\mathsf{corrupt}(s_{actor})$ and ($\mathsf{randomness}(s)$ or $\mathsf{cr\text{-}create}(s)$) have been issued, which implies Conditions 3 of freshness in model $\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{ISM}}$. The same argument applies to the fourth condition of the freshness definition.

The proofs that model $\Omega_{\mathsf{INDP}\cap\mathsf{ISM}}$ is at least as strong as model $\Omega^{-}_{\mathsf{INDP}}$ and that model $\Omega_{\mathsf{AKE}\cap\mathsf{ISM}}$ is at least as strong as model $\Omega^{-}_{\mathsf{AKE}}$ are similar to the proof of the previous statement.

The CNX protocol provides an example of a protocol that is secure in model $\Omega^{-}_{\mathsf{INDP\text{-}DH}}$, but insecure in model $\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{ISM}}$ (see Proposition 14). The protocols $\mathrm{SIG_t}$(CNX) and SIG^{*}(CNX) from Section 5.6.1 are secure in the models $\Omega^{-}_{\mathsf{INDP}}$ and $\Omega^{-}_{\mathsf{AKE}}$, respectively, but insecure in the corresponding lifted models $\Omega_{\mathsf{INDP}\cap\mathsf{ISM}}$ and

$\Omega_{\mathsf{AKE}\cap\mathsf{ISM}}$ due to a similar attack as in the proof of Proposition 14. □

5.7.1. Protocol-security hierarchy

In Figure 5.4 we show the *protocol-security hierarchy* [10] for AKE security of the protocols developed in this chapter with respect to security models of this chapter. Each node in Figure 5.4 lists a protocol and the security model in which the protocol is secure. Arrows indicate stronger protocols, i. e., the protocol at the end of an arrow is not only secure in the same models as the protocol at the start of that arrow, but also in a stronger model, listed below the protocol name. Further, if there is an arrow between two nodes, then the protocol at the start of the arrow is insecure in the model indicated in the node at the end of the arrow. Thus, the protocol at the top of the hierarchy is the only one that is secure in all models in the figure. We discuss below the protocols of the hierarchy in Figure 5.4.

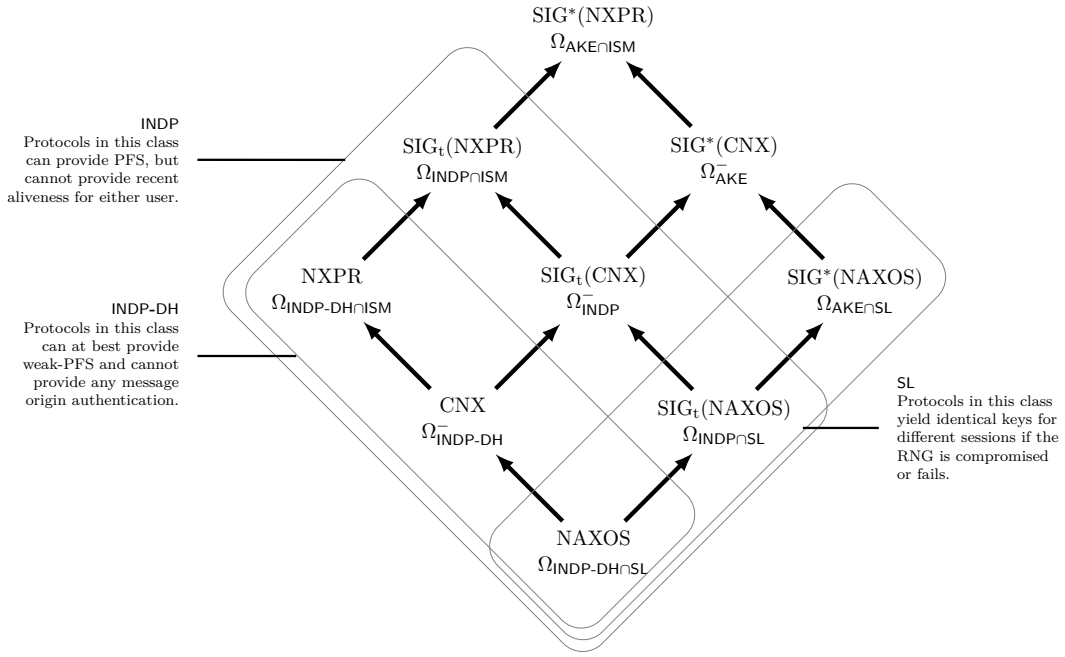

Figure 5.4.: Annotated protocol-security hierarchy

The rounded rectangles in Figure 5.4 identify protocol classes. For each class, we identify the security guarantees that cannot be achieved by any of its members. For example, because NAXOS is a member of $\mathsf{INDP\text{-}DH} \cap \mathsf{SL}$, it cannot provide PFS nor message origin authentication. Additionally, NAXOS is insecure if the RNG is compromised or fails (see Proposition 11). The protocols in the figure that belong to the class $\mathsf{AKE} \setminus \mathsf{SL}$ are secure even against attacks based on bad randomness such as reset-and-replay attacks. In contrast to the protocols CNX, SIG$_t$(CNX), and SIG*(CNX), the protocols NXPR, SIG$_t$(NXPR), and SIG*(NXPR), achieve security even under compromise of the randomness of the target session and the long-term secret key of the actor of that session as long as the randomness of at least one of the

previous sessions of the same user has not been compromised.

The protocols in the class INDP can provide PFS, but cannot provide *recent aliveness*. Recent aliveness means that, after completion of a session with a certain peer, the user executing the session has a guarantee that its peer has been alive during the execution of the protocol [63]. For example, the protocol $\mathrm{SIG}_{\mathrm{t}}$(NAXOS) provides PFS as it is secure in the model $\Omega_{\mathsf{INDP}\cap\mathsf{SL}}$, but it does not provide recent aliveness, because the messages of initiator and responder can be generated independently of each other. In contrast, the protocols SIG*(NAXOS), SIG*(CNX), and SIG*(NXPR) provide recent aliveness to the initiator in the models $\Omega_{\mathsf{AKE}\cap\mathsf{SL}}$, $\Omega^{-}_{\mathsf{AKE}}$, and $\Omega_{\mathsf{AKE}\cap\mathsf{ISM}}$, respectively. The responder not only signs his own Diffie-Hellman exponential but also the exponential that he received from the initiator. Thus, the latter protocols also achieve security against replay attacks to the initiator. Recent aliveness for both initiator and responder can be achieved in three-message protocols, e. g., by adding a third message that contains a signature on the Diffie-Hellman exponential that the initiator received from his peer.

5.8. Summary

In this chapter we clarified the limits of AKE security by giving formal impossibility results on the security of protocols that belong to different classes and deriving security models for the respective protocol classes from these impossibility results. If a protocol designer aims to develop a protocol in a certain class, our results demonstrate which strong guarantees can be achieved by such protocols. Conversely, if a certain security guarantee is required, our results indicate in which protocol classes it can be achieved. For example, PFS under weak security assumptions on the RNG can be achieved in the protocol class $\mathsf{AKE} \setminus (\mathsf{SL} \cup \mathsf{INDP\text{-}DH})$.

We provided new variants of the NAXOS protocol, which are secure against attacks based on chosen randomness; in these protocol variants, the state of a user's memory is modified during execution of the protocol. The CNX protocol is secure in the model $\Omega^{-}_{\mathsf{INDP\text{-}DH}}$. The NXPR protocol is secure in the stronger model $\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{ISM}}$. Our results show that it is possible to construct secure AKE protocols under significantly weaker assumptions on the RNG than previously thought possible.

6. Related Work

In this chapter we discuss further related work relevant for this thesis. In Section 6.1 we give an overview of various other formal security models to evaluate the security of AKE protocols. Then, in Section 6.2, we present additional related work for Chapter 5.

6.1. Security protocol analysis

In this section we first survey various computational security models. We then discuss how adversary capabilities and security properties captured in the computational setting have been dealt with in the symbolic setting.

Computational AKE security models

Bellare and Rogaway [17] introduced the first game-based security model whereby an adversary is modeled as a probabilistic polynomial-time Turing machine that controls all communication and interacts with users through queries. Their definition of security requires that (a) two users who complete *matching sessions* (i. e., the intended communication partners) compute the same session key and that (b) the adversary cannot learn the session key with more than negligible probability. Later variants of the original Bellare-Rogaway model were proposed by Bellare and Rogaway [18], and Bellare et al. [15]. The three models mainly differ in the definition of partnership of protocol sessions. Further, the model by Bellare and Rogaway [18] provides a security definition for secrecy of session keys, whereas the other two models provide a definition for entity authentication as well.

Building on prior work, Canetti and Krawczyk [34] developed a more complex security model, called unauthenticated-links adversarial model (UM), which gives the adversary additional powers such as access to a session-state query revealing the internal state of a session. The corruption query in the UM model not only reveals a user's long-term secret keys, but also the internal state and the session key of all activated sessions of the user. To model perfect forward secrecy, the UM model includes the query session expiration whose effect it is to delete the session key of a specific session.

To analyze the security of the HMQV protocol, Krawczyk [65] defined distinct security models providing different security guarantees. Based on a variant of the UM model [34] that does not incorporate the query session expiration, Krawczyk defined three related models that capture key compromise impersonation attacks, weak perfect forward secrecy, and leakage of session-specific random values, respectively. It is an open problem whether the HMQV protocol can be proven secure in a single model that integrates all of the guarantees provided by these individual models.

LaMacchia et al. [68] adapted the Canetti-Krawczyk model [34] to capture key compromise impersonation (KCI) attacks and the leakage of various combinations of long-term secret keys and session-specific random values in a *single* security model. Their model is known as the eCK security model. We refer the reader to Section 1.2 for further details on the eCK model.

In Chapter 3 we defined the eCK^w model using the concept of origin-session, which relaxes the notion of matching session. The eCK^w model captures a slightly stronger form of weak perfect forward secrecy than the eCK model. We then integrated perfect forward secrecy into the eCK^w model, which gave rise to the stronger model eCK-PFS. In particular, security in the eCK-PFS model implies perfect forward secrecy under actor compromise and randomness reveal. In Chapter 5 we derived even stronger models than eCK-PFS, in terms of authentication guarantees provided, from impossibility results on the security of protocols belonging to the classes INDP and AKE. We then extended our models to capture chosen-randomness attacks, in addition to randomness reveal; security in the resulting models implies security under repeated randomness.

The relations between some of the previous security models have been investigated, e. g., by Choo et al. [37], Cremers [41], and Menezes and Ustaoglu [82]. They showed that several security models are in fact formally incomparable due to sometimes subtle differences between them, not only in the capabilities of the adversary, but also in the restrictions posed on the target session, and the notions for relating sessions of the protocol. As these differences have a considerable impact on the guarantees provided by the models, we incrementally specified security models to objectively measure the security guarantees provided by our protocols and transformations.

Symbolic AKE security models

In the symbolic setting, the capabilities of an adversary attacking the protocol are limited since cryptographic primitives are modelled as black boxes; e. g., a ciphertext does not reveal any information about the plaintext unless the attacker knows the corresponding secret decryption key. Security guarantees are obtained by showing the absence of attacks for every possible execution scenario of the protocol.

Basin and Cremers [11] translated existing computational AKE security models into a symbolic framework. This framework separates the adversary's capabilities modelled by adversary rules, and the security property such as secrecy of the session key. For example, the rule reflecting perfect forward secrecy allows the corruption of all user's long-term secret keys after the target session ends. They extended the symbolic protocol analysis tool Scyther [39] with their adversary models, which resulted in the first tool that allows the automatic verification of protocols supporting, e. g., perfect forward secrecy, key compromise impersonation, and compromise of session-specific information [11]. Several new attacks on protocols were found and attacks reported in the cryptographic literature were rediscovered automatically [11]. There exist however attacks such as small-subgroup attacks or attacks based on adversarial key registration that cannot be captured in their symbolic framework. In Chapter 4, we analyze the security of AKE protocols in the computational setting in the presence of adversaries who can register arbitrary public keys with a certification authority.

In [10], Basin and Cremers introduced the concept of *protocol-security hierarchy*. Such a hierarchy classifies protocols according to their relative strength against adversaries with different capabilities. The protocol-security hierarchies generated with an extension of the tool developed in [11] revealed several interesting insights on the design of protocols and the security guarantees they provide. For instance, as observed in [10, p. 14], it seems that protocol TS3 satisfies a stronger forward secrecy property than protocol TS2, although this is not stated or proven in [56]. The security-strengthening transformations on protocols that we developed in this thesis naturally lead to protocol-security hierarchies in the computational setting (see, e. g., Chapter 5).

Schmidt et al. [94] developed the Tamarin prover for automated protocol verification, which has been used to automatically verify several AKE protocols in models that integrate various security properties in a flexible way. In contrast to the Scyther tool, Tamarin supports Diffie-Hellman exponentiation and has a more expressive property specification language. Users can specify their own security models without modifying the source code of Tamarin. However, as multiplication in the DH group as well as addition in the DH exponents are not modelled, some modern AKE protocols such as HMQV, CMQV, or π_1 from Chapter 3 cannot be analyzed using their approach. In contrast, we showed in Chapter 3 that the π_1 protocol achieves perfect forward secrecy under actor compromise and randomness reveal and demonstrated in Chapter 4 that a variant of the CMQV protocol is secure even in the presence of an adversary who can register arbitrary public keys.

Even though individual security protocols can be verified in the symbolic setting, it is unclear whether symbolic proofs of generic transformations that can be applied to a class of protocols can be established.

6.2. Stateless and stateful protocols

Most AKE protocols (e. g., HMQV [64], NAXOS [68], CMQV [104]) are stateless, i.e., they only modify session-specific memory, whereas the memory that is shared among sessions is invariant under protocol execution. Furthermore, the security of stateful AKE protocols, which update the memory that is shared among sessions during execution of the protocol, has not been considered in the context of randomness failures.

A few stateful protocols have been suggested. For example, Blake-Wilson et al. [20] propose to modify their Protocol 2 by concatenating the secret value that is used as the secret material to derive the session key with the value of a counter. We denote this new protocol by Protocol 2C. Instead of running the protocol each time a session key is required, a new session key is obtained by simply incrementing the counter and computing a new hash value [20]. The idea of using a counter variable is presented in the context of special applications for which it might not be desirable to run the protocol whenever a new session key has to be established. However, no security proof of Protocol 2C has been given. In Section 5.6.1 we proved the security of the CNX protocol, a stateful variant of the NAXOS protocol. The CNX protocol includes a global counter value, which is shared across the sessions of a user, as input to the

hash function H_1 used in the computation of the outgoing messages.

Stateful protocols preventing replay and interleaving attacks include protocols using timestamps such as Protocol-P1A [74], and key establishment protocols specified in the standard ISO/IEC 11770-2 [48]. These protocols can be considered stateful as each user needs to maintain a local clock in his memory. In contrast to protocols relying on a challenge-response mechanism via fresh random values, which only modify session-specific memory, protocols using timestamps are usually more efficient in terms of the round complexity of the protocol. However, the difficulty in modelling protocols with timestamps consists in keeping the local clocks of users synchronized as well as protected from adversarial modification. Barbosa and Farshim [6] extend the Bellare-Rogaway model and the UM model to analyze protocols using timestamps. In their extension, each user keeps a local clock variable initialized at zero and accessible to all sessions of the user. The adversary can increment the local clock via a special query. To capture synchronization of clocks, they introduce the notion of *δ-synchonisation*, which is satisfied if the adversary does not cause the local clocks of two distinct users to differ by more than δ. In [96], Schwenk defines a security model in which each user has a local time counter in his memory; this counter can be accessed and increased by all protocol sessions. When sending a message within a session, the user first requests a new timestamp from a special user $\mathcal{T}$ keeping a global time counter and compares it to its local counter before continuing the protocol execution. When receiving a message, the user compares the timestamp within the message to its local counter; in case the timestamp is greater than the local counter value, the latter is updated by setting it to the timestamp, otherwise the session is aborted [96]. In contrast to the work of Barbosa and Farshim [6], the adversary is not given access to a special query for incrementing local clocks. The models that we developed in this thesis do not take into consideration the notion of time as we do not analyze the security of protocols using timestamps.

The notions of stateless and stateful protocols have been introduced in different other contexts than AKE. In [107], van Deurson et al. provide a formal model for reasoning about stateful radio frequency identification (RFID) protocols in the symbolic setting. Whereas stateless protocols are modelled as protocols that do not share state between protocol executions, stateful protocols may use information from earlier protocol runs and update this information during execution of the protocol. Arapinis et al. [4] propose the process calculus StatVerif that allows the modelling of global state to analyze protocols updating the global state. Global state is not local to a session but is shared among different sessions of the protocol. Meier et al. [79] analyze protocols with mutable global state such as the security device and the contract signing protocol from [4] using the Tamarin tool [94]. The Tamarin analysis of these protocols subsumes and extends the analysis results from Arapinis et al. [4]. To the best of our knowledge, we are the first to analyze the security of stateful authenticated key exchange protocols that do not rely on timestamps. In particular, we show that stateful authenticated key exchange protocols can achieve security in the presence of adversaries who are able to perform chosen-randomness attacks even against the target session.

7. Conclusions

7.1. Summary

In this thesis we developed strong AKE security models which capture relevant attacks that lie outside the scope of current models. Additionally, we provided security-strengthening methods to achieve security in the respective models.

We analyzed for the first time the security of two-message AKE protocols in a model that captures PFS as well as key compromise impersonation attacks and the leakage of session-specific randomness. We have shown that it is possible for two-message protocols to achieve security in our eCK-PFS model by first providing a generic security-strengthening protocol transformation to achieve PFS, and then applying our transformation to concrete protocols that are secure in weaker models than eCK-PFS. Our transformation introduces no new message dependencies and does not increase the round complexity of the protocol it is applied on. Our generic approach gave rise to a new protocol that is secure in the eCK-PFS model.

Unlike current security models which do not explicitly model the certification authority (CA) and its behavior, we developed a framework that allows for explicit modelling of the certification process of public keys. Our framework, called the ASICS (AKE Security Incorporating Certification Systems) framework, can be instantiated to produce extensions of current security models that capture dynamic adversarial registration of arbitrary bitstrings as public keys. We provided a generic approach to achieve strong security guarantees against adversaries who can register arbitrary public keys with a CA that does not perform any checks. In particular, we show how to transform Diffie-Hellman type AKE protocols that are secure in a model where only honest key registration is allowed into protocols that are secure even when adversaries can register arbitrary valid or invalid public keys. Our results not only present a formal foundation for the importance of public key validation, but they also provide evidence that users can defend themselves against CAs that do not perform proper checks.

While the first part of this thesis provides stronger security guarantees than found in the state-of-the-art via extensions of current models, the second part of this thesis explores the limits of AKE security. We derived strong AKE security models from impossibility results on the security of protocols belonging to distinct protocol classes. Our results reveal that different security guarantees can be obtained from protocols belonging to different classes. In particular, we considered AKE security in the presence of active adversaries who can perform chosen-randomness attacks, whereby they control the randomness used in protocol sessions. While stateless protocols fail to achieve security against chosen-randomness attacks, we constructed stateful variants of the NAXOS protocol, which provide security against attacks based on this worst case randomness failure. Our new stateful protocols allow us to weaken the assumptions

made on the security of the RNG used to generate session-specific randomness.

7.2. Future work

In this section we describe different topics of interest for future work.

Deniability. In Chapter 3 we have provided a signature-based transformation to achieve PFS in two-message protocols. Applying this transformation on a concrete protocol results in a protocol for which *full deniability* is lost. Full deniability [90] allows a party to deny having been involved in a given run of the protocol. As a consequence, a recipient cannot convince a judge that the messages it received during a given execution were sent by the accused sender. At first glance it seems that the use of signatures has significant consequences on the level of deniability of the protocols resulting from our transformation [42]. However, various weaker forms of deniability are achievable depending on the information that is signed. For example, if a party $\hat{A}$ only signs its Diffie-Hellman exponential X, then the recipient can convince the judge that party $\hat{A}$ signed message X at some point in time, but there is no evidence of $\hat{A}$'s intended peer or of the point in time at which the message was signed [42].

Several research issues arise in this context. First, we could formalize various degrees of deniability for AKE protocols in order to develop a hierarchy of deniability properties. This would enable us to classify protocols according to their deniability feature. Second, we could develop a library of deniability-enhancing transformations on AKE protocols, and investigate the potential trade-offs between confidentiality, authentication, and deniability for AKE protocols. Third, we could investigate alternative methods to achieve eCK-PFS security while guaranteeing better deniability features. For example, it seems that replacing the signature of the responder by a MAC using a key that is derived from the responder's secret provides higher deniability for the responder; it is more difficult to relate this MAC to a specific party unless both long-term and session-specific secrets are disclosed. In addition, the resulting protocol provides unilateral key confirmation from the responder to the initiator. That is, if the MAC is valid, then the initiator is assured that its peer has derived the same secret value, and that its peer has been alive during the protocol execution [8].

CA compromise. The ASICS framework that we developed in Chapter 4 does not allow the adversary to corrupt the CA (i. e., to learn its secret key) or to issue fraudulent certificates. Turner et al. [102] identity four different attack scenarios on CA operations. While their first attack scenario, namely impersonation during the key registration phase, is captured in our framework, the other three attack scenarios resulting in the issuance of fraudulent certificates cannot be captured yet. It would therefore be worthwhile to incorporate the adversarial ability of corrupting the CA and issuing rogue certificates into our framework. Further, we would like to analyze the security of AKE protocols under these circumstances, and to provide appropriate countermeasures to repair vulnerable protocols. In particular, we could prove our conjecture that AKE protocols satisfy *CA forward secrecy*. This property has been introduced by Boldyreva et al. [24] in the context of certification and PKI for public

key encryption and signatures schemes. In the context of AKE, CA forward secrecy guarantees secrecy of session keys even if the adversary later compromises the CA's secret key.

Generalizing the key registration phase. In Chapter 4 we only considered non-interactive verification procedures (i. e., two-message registration protocols) between the user and the CA. As future work we envision to generalize our ASICS framework to allow for interactive verification procedures, reflecting n-message registration protocols (with $n > 2$), between the user and the CA. This generalized framework would allow us to consider, e. g., challenge-response registration protocols, where the user signs messages received from the CA. However, in this more complex setting, we suspect that interleavings between the registration protocol and adversarial queries that model the corruption of a user's secrets give rise to new attacks that affect the security of the registration protocol and, consequently, the security of the AKE protocol. Before dealing with these complex scenarios, we could choose to model the registration protocol as atomic in the sense that this protocol is not interleaved with corruption queries.

Session state reveal. The randomness query in our security models only reveals the randomness used in a particular session, but not the results of intermediate computations done by a user as part of a session. It may however be possible that the adversary not only obtains the randomness, but also other inputs (excluding the user's long-term secret key) as well as results of intermediate protocol computations [30]. In practice, such a scenario occurs when the long-term secret key of the user resides in secure memory (e. g., in a HSM), whereas protocol computations are done in unprotected memory. It would therefore be interesting to define models that incorporate a session state reveal query, similar to the query session-state in the CK model, which returns the internal state information associated with a session. The resulting models would allow us to capture the attacks on the NAXOS protocol pointed out by Cremers [40] as well as the authentication flaw in the SIGMA protocol exposed by Mao and Paterson [75]. In addition, we could develop countermeasures to these vulnerabilities and design new protocols that are secure in models that integrate attacks based on session state reveal.

Reset attacks. In Chapter 5 we defined security models that capture attacks based on chosen-randomness. In particular, security in these models implies security against repeated randomness failures as well as security against reset-and-replay attacks in which the adversary first sets the randomness of a session to the same randomness as used in a previous session of the same user and then replays messages to the session so that both sessions compute the same session key without being intended communication partners. We constructed stateful protocols that achieve security against chosen-randomness attacks. Thus, our protocols are secure even if a flawed RNG that produces the same value more than once is used to generate session-specific randomness. However, e. g., our CNX protocol is insecure in the presence of adversaries who can reset the value of the global counter stored in the user memory. It would

therefore be worthwhile to define models that incorporate a reset query that not only resets the randomness of a session to the same randomness as used in a previous session, but also resets the state of the user memory to a prior state, and to investigate countermeasures to these reset attacks.

7.3. Final remarks

To conclude this thesis, we present our vision on AKE protocol design and discuss the impact of our results.

A major part of this thesis was devoted to the development of new security models that capture relevant attacks that fall outside the scope of current models and to the investigation of countermeasures to these attacks. Instead of developing an ultimate security model and trying to construct protocols secure in this model, we analyzed the different types of attack in separate frameworks. We consider our approach a major step towards our long-term vision on AKE protocol design: Given a set of desired security properties, a secure base protocol, and a library of security-strengthening methods, protocol designers should be able to construct protocols by applying these methods on the base protocol until the desired security guarantees are achieved. However, we do not expect it to be possible to build protocols that are secure in an arbitrary security model starting from a secure base protocol via *generic* security-strengthening transformations only.

Our work has an impact on the design of AKE protocols as well as on the understanding of the security mechanisms that are employed in these protocols. First, our generic security-strengthening transformations are useful to protocol designers to build their protocols in an incremental way, which simplifies the proof process of the protocols. Second, our impossibility results show the impossibility of achieving certain security guarantees in given protocol classes. If specific security guarantees are required, then these results reveal the protocol class to be considered. Conversely, if a protocol designer aims to construct a protocol in a given class, then the associated model that we derived from our impossibility results reflects the security guarantees that can be achieved by such a protocol. Third, the security-strengthening methods that we used to counter attacks based on chosen randomness allow for the development of a range of new stateful protocols that are secure even under bad session-specific randomness.

As authenticated key exchange protocols are central building blocks of many important Internet applications, we hope that the research community as well as industry will use and build on our work to develop stronger protocols and applications.

Appendix

A. Analysis of CMQV'

Let $\mathsf{eCK}' = (\mathsf{M2}, Q', F')$ be the ASICS model where $Q' = Q \cup \{\mathsf{pkregister}\}$ and F' is defined as F with the additional requirement that no $\mathsf{pkregister}(s_{\text{pcert}}.\mathsf{pk}, s_{\text{pcert}}.\mathsf{id})$ query has been issued.

Lemma 2. *Let* eCK *and* eCK' *be as above. CMQV' has strong partnering in the* $\mathsf{ASICS}_{\mathsf{eCK}'}$ *experiment under the assumption that H is a random oracle.*

Proof. Suppose otherwise. Namely, suppose there exists two sessions s and s' of CMQV' that hold the same session key but are not M2-matching. Since the session key in CMQV' is derived by applying a random oracle, except with negligible probability, the input to the random oracle in both sessions must be the same. Since they are not M2 matching, either $s_{\text{acert}}.\mathsf{id} \neq s'_{\text{pcert}}.\mathsf{id}$, or $s_{\text{acert}}.\mathsf{pk} \neq s'_{\text{pcert}}.\mathsf{pk}$, or $s_{\text{pcert}}.\mathsf{id} \neq s'_{\text{acert}}.\mathsf{id}$, or $s_{\text{pcert}}.\mathsf{pk} \neq s'_{\text{acert}}.pk$, or $s_{\text{sent}} \neq s'_{\text{rcvd}}$, or $s_{\text{rcvd}} \neq s'_{\text{sent}}$, or $s_{\text{role}} = s'_{\text{role}}$.

First suppose $s_{\text{role}} \neq s'_{\text{role}}$. Then either the public keys, identifiers, or transcripts of the two sessions do not correspond. But these are all inputs to the random oracle, so except with negligible probability the outputs of the random oracle will be different, contradicting that the two sessions hold the same session key.

Now suppose $s_{\text{role}} = s'_{\text{role}}$. Except with negligible probability, two distinct honest sessions will have $s_{\text{rand}} \neq s'_{\text{rand}}$, and hence $s_{\text{sent}} \neq s'_{\text{sent}}$. But since both s and s' think of themselves as the initiator, they will each put their own sent ephemeral public key in the second component of the call to H, and these values are different, so except with negligible probability the outputs of the random oracle will be different, contradicting that the two sessions hold the same key. □

Let (G, g, q) be as in Definition 30. Let $\phi \subseteq (G \times G) \times G$ be the *Diffie–Hellman relation* on $G = \langle g \rangle$. In particular, (g^a, g^b) is related under ϕ to g^c if and only if $ab \equiv c \bmod q$.

Lemma 3. *The* cNR-eCK *security of the variant of CMQV' in which the session string is output as the session key is polynomial-time reducible to the computational problem of the Diffie–Hellman relation ϕ, under the assumption that $\mathcal{H}_1$ and $\mathcal{H}_2$ are random oracles.*

The basic idea of the proof is as follows.

- If the adversary happens to figure out a long-term secret key without issuing a corrupt query (event E), it must ask that value to a random oracle $\mathcal{H}_1$, and we can immediately use that value to solve the CDH problem by having embedded one of the CDH challenge values in that public key.
- If the adversary is passive in the test session (event $\overline{E} \wedge M$) we can embed the CDH challenge values U, V as the ephemeral public keys X and Y of the test session. The adversary's view can be simulated perfectly unless the adversary

asks either $(\tilde{x}, a)$ or $(\tilde{y}, b)$ as a query for $\mathcal{H}_1$. But the freshness condition prevents the adversary from finding both elements of either pair. Therefore the adversary cannot do better than guess the session string unless it can compute σ. Here the CDH of U and V can be extracted from σ.

- If the adversary is active in the test session (event $\overline{E} \wedge \overline{M}$) we can embed the CDH challenge values in the long-term key of the partner of the test session and the ephemeral public key of the session. As before the simulation is perfect unless the adversary asks $(\tilde{x}, a)$ as a query for $\mathcal{H}_1$. Note that, since the adversary is active, the adversary cannot change or corrupt the secret long-term key of the peer. This time the value of σ is similar to a signature forgery and we can apply the Forking Lemma [14, 89] to extract the CDH of U and V.

Proof. Recall that cNR-eCK security means security in an ASICS model that omits the session-key query, so the allowed queries are $\mathcal{Q}_N \cup \{\mathsf{corrupt}, \mathsf{randomness}\}$. The freshness condition remains unchanged.

Recall further that the goal of the adversary is to *recover* the session string; let S be the event that an algorithm $\mathcal{M}$ computes the session string. The security proof largely follows the original proof of Ustaoglu that CMQV is eCK-secure, but can be simplified somewhat as the queries in the cNR-eCK game are restricted compared to full eCK security.

Consider the following two complementary events:

- E. There exists a certificate C' (created using hregister) such that $\mathcal{M}$, during its execution, queries $\mathcal{H}_1(*, b)$ (where $C'.\mathsf{pk} = g^b$) before issuing any $\mathsf{corrupt}(C'.\mathsf{pk})$ query (if it issues one at all).
- $\overline{E}$. During its execution, for every certificate C' (created using hregister) for which $\mathcal{M}$ queries $\mathcal{H}_1(*, b)$ (where $C'.\mathsf{pk} = g^b$), it issued a $\mathsf{corrupt}(C'.\mathsf{pk})$ query before the $\mathcal{H}_1(*, b)$ query.

Since the events are complementary, if $\mathcal{M}$ succeeds in computing the session string, it succeeded either when E occurred or when $\overline{E}$ occurred.

We will see how, when each event occurs, the required polynomial-time reduction exists.

E. Suppose event E occurs and $\mathcal{M}$ succeeds in computing the session string.

Here, the simulator $\mathcal{S}$ guesses one public key pk^* at random and assigns $\mathsf{pk}^* \leftarrow V$, where (U, V) is the Diffie–Hellman challenge. All other public keys are generated according to the protocol specification.

For all sessions and queries where the session actor is not using pk^*, $\mathcal{S}$ follows the protocol specification exactly.

For sessions where the session actor is using pk^*, $\mathcal{S}$ responds to queries as follows:

- $\mathsf{hregister}(\mathsf{pk}^*, \hat{P})$: $\mathcal{S}$ outputs a certificate as normal.
- $\mathsf{corrupt}(\mathsf{pk}^*)$: $\mathcal{S}$ aborts.
- $\mathsf{randomness}(s = (C, i))$ where $C.\mathsf{pk} = \mathsf{pk}^*$: Return s_{rand}.

For sessions where the session actor is using pk^* and is the initiator, $\mathcal{S}$ responds to queries as follows:

- $\mathsf{create}(s = (C, i), \mathcal{I}, C')$ where $C.\mathsf{pk} = \mathsf{pk}^*$: $\mathcal{S}$ selects $x \in_R \mathbb{Z}_q$, computes $X \leftarrow g^x$, and responds with X. Note that s_{rand} is not used in the calculation.

- $\mathsf{send}(s = (C, i), M)$ where $s_{\text{acert}}.\mathsf{pk} = \mathsf{pk}^*$ and $s_{\text{role}} = \mathcal{I}$: $\mathcal{S}$ does not need to simulate anything here, since there is not outgoing message required, and since the only variable updated is the session string ss but no session-key reveal query is allowed.

For sessions where the session actor is using pk^* and is the responder, $\mathcal{S}$ responds to queries as follows:

- $\mathsf{create}(s = (C, i), \mathcal{R}, C')$ where $C.\mathsf{pk} = \mathsf{pk}^*$: no response required.
- $\mathsf{send}(s = (C, i), M)$ where $s_{\text{acert}}.\mathsf{pk} = \mathsf{pk}^*$ and $s_{\text{role}} = \mathcal{R}$: $\mathcal{S}$ selects $y \in_R \mathbb{Z}_q$, computes $Y \leftarrow g^y$, and responds with Y. Note that s_{rand} is not used in the calculation.

$\mathcal{S}$ responds to $\mathcal{H}_2$ queries as normal. $\mathcal{S}$ responds to $\mathcal{H}_1(*, b)$ queries as normal for all b such that $g^b \neq \mathsf{pk}^*$. When $\mathcal{M}$ queries $\mathcal{H}_1(*, b)$ where $g^b = \mathsf{pk}^* = V$, $\mathcal{S}$ outputs the solution to the Diffie–Hellman challenge (U, V) as U^b.

Note that $\mathcal{S}$'s simulation is perfect up until an abort event from the corrupt query occurs. Given that event E occurs, there exists some public key $\mathsf{pk} = g^b$ for which the query $\mathcal{H}_1(*, b)$ occurs before any $\mathsf{corrupt}(\mathsf{pk})$ query occurs. With probability at least $1/n_{\mathsf{kgen}}$, where n_{kgen} is the number of kgen queries made by $\mathcal{M}$, this condition holds for pk^*. When $\mathcal{S}$ guesses correctly, $\mathcal{M}$ will indeed query $\mathcal{H}_1(*, b)$ before any $\mathsf{corrupt}(\mathsf{pk}^*)$ query, and thus $\mathcal{S}$ will solve the computational Diffie–Hellman problem.

Thus, when event E occurs, there exists a polynomial-time reduction from a cNR-eCK adversary for the session string variant of CMQV' to the computational Diffie–Hellman problem under the assumption that $\mathcal{H}_1$ is a random oracle, with a tightness factor of n_{kgen}.

$\overline{E}$. We divide this event into two complementary cases:

- M. The session s for which the adversary output the session string has an M2-matching session s'.
- $\overline{M}$. The session s for which the adversary output the session string does not have an M2-matching session.

When $\overline{E}$ occurs, either M or $\overline{M}$ must also occur.

$\overline{E} \wedge M$. Suppose event $\overline{E}$ occurs and there is an M2-matching session s' for the target session s.

Here, the simulator guesses two sessions s and s'; assume without loss of generality that $s_{\text{role}} = \mathcal{I}$ and $s'_{\text{role}} = \mathcal{R}$. $\mathcal{S}$ responds to all kgen, hregister, corrupt, and randomness queries as specified by the protocol. For all sessions other than s and s', $\mathcal{S}$ responds to create and send as specified by the protocol. For s and s', $\mathcal{S}$ responds to create and send as follows:

- $\mathsf{create}(s = (C, i), \mathcal{I}, C')$: Return $X \leftarrow U$, where (U, V) is the Diffie–Hellman challenge. Note that s_{rand} is not used.
- $\mathsf{create}(s' = (C', i), \mathcal{R}, C)$: No response required.
- $\mathsf{send}(s', M)$: Return $Y \leftarrow V$, where (U, V) is the Diffie–Hellman challenge. Note that s'_{rand} is not used.

$\mathcal{S}$ responds to $\mathcal{H}_2$ queries as normal. $\mathcal{S}$ responds to $\mathcal{H}_1$ queries as normal except for the queries $(\tilde{x}, a)$ or $(\tilde{y}, b)$, where a and b are the secret keys corresponding to the public keys in sessions s and s'; when this occurs, the simulation aborts.

Note that $\mathcal{S}$'s simulation is perfect unless a $\mathcal{H}_1(\tilde{x}, a)$ or $\mathcal{H}_1(\tilde{y}, b)$ query occurs. Because of event $\overline{E}$, $\mathcal{M}$ issues a $\mathsf{corrupt}(g^a)$ query before any $\mathcal{H}_1(\tilde{x}, a)$ query, and a $\mathsf{corrupt}(g^b)$ query before any $\mathcal{H}_1(\tilde{y}, b)$ query. Since $\tilde{x}$ and $\tilde{y}$ are used in only one session and $\mathcal{H}_1$ is a random function, no information can be learned about $\tilde{x}$ and $\tilde{y}$ without $\mathsf{randomness}(s)$ or $\mathsf{randomness}(s')$ queries. By the freshness condition, it cannot be that both $\mathsf{randomness}(s)$ and $\mathsf{corrupt}(g^a)$ occurred, or that both $\mathsf{randomness}(s')$ and $\mathsf{corrupt}(g^b)$ occurred. Thus, if $\mathcal{S}$ correctly guess s and s', the simulation is perfect and does not abort. This happens with probability at least $2/n^2_{\mathsf{create}}$.

Assuming the simulation is perfect and does not abort, and that $\mathcal{M}$ outputs the session string, $\mathcal{S}$ can use this to solve the Diffie–Hellman problem. In particular, let σ be the shared secret in the session string output by $\mathcal{M}$. Then $\mathcal{S}$ outputs $\sigma g^{-abed} U^{-be} V^{-ad}$ as the solution to the computational Diffie–Hellman challenge (U, V).

Thus, when event $\overline{E} \wedge M$ occurs, there exists a polynomial-time reduction from a $\mathsf{cNR\text{-}eCK}$ adversary for the session string variant of CMQV' to the computational Diffie–Hellman problem under the assumption that $\mathcal{H}_1$ is a random oracle, with a tightness factor of n^2_{create}.

$\overline{E} \wedge \overline{M}$. Suppose event $\overline{E}$ occurs but there is no M2-matching session for the target session s.

Here, the simulator guesses integers $j \in_R \{1, \ldots, n_{\mathsf{kgen}}\}$, and a session s^*. Assume without loss of generality that $s^*_{\text{role}} = \mathcal{I}$.

For the jth query to kgen, $\mathcal{S}$ assigns $\mathsf{pk}^* \leftarrow V$ from the Diffie–Hellman challenge (U, V) to be the public key; for all other kgen queries it responds as specified by the protocol.

All $\mathsf{hregister}$ queries are responded to as normal. All $\mathsf{corrupt}$ queries are responded to as normal, except for $\mathsf{corrupt}(\mathsf{pk}^*)$, in which case $\mathcal{S}$ aborts.

Suppose that $\mathcal{M}$ selects s^* as the target session and furthermore that $s^*_{\text{pcert}}.\mathsf{pk} = V$.

For all sessions and queries where the session actor or peer is not using pk^*, $\mathcal{S}$ follows the protocol specification exactly.

For sessions where the session actor is using pk^*, $\mathcal{S}$ responds as in event E.

For sessions where the session peer is using pk^*, $\mathcal{S}$ responds as specified by the protocol, except for the target session s^*. In s^*, $\mathcal{S}$ responds as follows:

- $\mathsf{create}(s^*, \mathcal{I}, C')$: $\mathcal{S}$ returns $X \leftarrow U$, where (U, V) is the Diffie–Hellman challenge. Note that s^*_{rand} is not used.
- $\mathsf{send}(s^*, M)$: No response required.
- $\mathsf{randomness}(s^*)$: Return s^*_{rand}.
- $\mathsf{session\text{-}key}(s^*)$: $\mathcal{S}$ aborts. Assuming that $\mathcal{S}$ correctly guesses s^* as the target session, this abort will never occur.

$\mathcal{S}$ responds to $\mathcal{H}_2$ queries as normal. $\mathcal{S}$ responds to $\mathcal{H}_1(\tilde{x}, b)$ queries as normal except for the following case:

- If $\tilde{x} = s^*_{\text{rand}}$ and $g^a = s^*_{\text{acert}}.\mathsf{pk}$: $\mathcal{S}$ aborts.

Note that $\mathcal{S}$'s simulation is perfect up until an abort event from the $\mathsf{corrupt}$ or the $\mathcal{H}_2$ query occurs. Given that s^* is fresh and no matching session exists, no $\mathsf{corrupt}(\mathsf{pk}^*)$ query is allowed and hence $\mathcal{S}$ does not abort for that reason. Given that event $\overline{E}$ occurs, if $\mathcal{M}$ queries $\mathcal{H}_1(s^*_{\text{rand}}, a)$ such that $g^a = s^*_{\text{acert}}.\mathsf{pk}$, $\mathcal{M}$ must have issued a

$\mathsf{corrupt}(g^a)$ query first. But it is also the case that s^* is fresh, so $\mathcal{M}$ cannot have also issued a $\mathsf{randomness}(s^*)$ query, and thus cannot know s^*_{rand} unless it guessed it correctly, which can be done only with negligible probability.

Assume the simulation is perfect and does not abort, and that $\mathcal{M}$ outputs the session string containing the correct shared secret $\sigma = g^{uy}g^{ady}g^{uve}g^{adev}$. $\mathcal{S}$ can then compute $\eta = \sigma Y^{-ad}V^{-ade} = g^{uy+uve}$. But the peer's ephemeral secret key y was chosen by the adversary, so without y $\mathcal{S}$ cannot directly compute g^{uv} from η.

Using the Forking Lemma, $\mathcal{S}$ runs $\mathcal{M}$ on the same input and the same random coins but with modified answers to $\mathcal{H}_2$ queries. Note that $\mathcal{M}$ must have queried $\mathcal{H}_2(Y, s^*_{\text{acert}}.\mathsf{id}, s^*_{\text{pcert}}.\mathsf{id})$ to obtain e, because otherwise $\mathcal{M}$ would be unable to compute σ except with negligible probability. For the second run of $\mathcal{M}$, $\mathcal{S}$ responds to $\mathcal{H}_2(Y, s^*_{\text{acert}}.\mathsf{id}, s^*_{\text{pcert}}.\mathsf{id})$ with $e' \neq e$ selected uniformly at random.

If $\mathcal{M}$ succeeds in the second run, it outputs $\sigma = g^{uy}g^{ad'y}g^{uve'}g^{ad'e'v}$. $\mathcal{S}$ can then compute $\eta' = \sigma' Y^{-ad'}V^{-ad'e'} = g^{uy}g^{uve'}$. $\mathcal{S}$ can furthermore compute

$$(\eta/\eta')^{1/(e-e')} = (g^{uy}g^{uve}g^{-uy}g^{-uve'})^{1/(e-e')} = g^{uv(e-e')/(e-e')} = g^{uv}$$

which is the solution to the computational Diffie–Hellman challenge (U, V).

Thus, when event $\overline{E} \wedge \overline{M}$ occurs, there exists a polynomial-time reduction from a $\mathsf{cNR\text{-}eCK}$ adversary for the session string variant of CMQV' to the computational Diffie–Hellman problem under the assumption that $\mathcal{H}_1$ and $\mathcal{H}_2$ are random oracles, with a tightness factor of $n_{\mathsf{create}}n_{\mathsf{kgen}}n_{\mathcal{H}_2}c$, where c is a constant from the Forking Lemma. □

Remark 27. Because in the above lemma we do not have to prove full session key indistinguishability security of CMQV', instead proving the hardness of session string computation of a variant of CMQV', we can make a few simplifications from Ustaoglu's original proof:

- We do not have to worry about key replication attacks (when the adversary causes two non-matching sessions to have the same session key (that is, session string), and then reveals the session key at one of the sessions) because there is no $\mathsf{session\text{-}key}$ query.
- In event E, we do not have to worry about setting the session string correctly for any session involving the user whose public key has been injected with the CDH challenge, because there is no $\mathsf{session\text{-}key}$ query. Thus we do not need a DDH oracle here.
- In event $\overline{E} \wedge M$, we do not have to use the DDH oracle to test which of the many $\mathcal{H}$ random oracle queries is the solution we need: we simply output the CDH value derived directly from the output of $\mathcal{M}$.

Lemma 4. *The session string decision problem for CMQV' is polynomial-time reducible to the decisional problem of the Diffie–Hellman relation ϕ.*

Proof. Let D be a polynomial-time algorithm that can distinguish real CMQV' session strings $(g^{(y+eb)(x+da)} \parallel X \parallel Y \parallel \mathsf{id} \parallel A \parallel \mathsf{id}' \parallel B)$ from random session strings $(g^r \parallel g^x \parallel g^y \parallel \mathsf{id} \parallel g^a \parallel \mathsf{id}' \parallel g^b)$, for randomly chosen $a, b, x, y, r \in_R \mathbb{Z}_q$, id and id' are arbitrary binary strings, $d = \mathcal{H}_2(g^x \parallel \mathsf{id} \parallel \mathsf{id}')$, and $e = \mathcal{H}_2(g^y \parallel \mathsf{id} \parallel \mathsf{id}')$.

We claim that there exists an algorithm E that can distinguish real Diffie–Hellman triples (g^u, g^v, g^{uv}) from random triples (g^u, g^v, g^w) for randomly chosen $u, v, w \in_R \mathbb{Z}_q$.

First, note that $g^{(y+eb)(x+da)} = g^{xy+ady+bex+abde}$. Using D construct E_D as follows. Let (U, V, W) be a Diffie–Hellman challenge. Pick arbitrary $\mathsf{id}, \mathsf{id}'$. Do one of the following, each with equal probability:

1. Set $A \leftarrow U$ and $B \leftarrow V$. Choose $x, y \in_R \mathbb{Z}_q$.
   Run D on the session string $(g^{xy}A^{dy}B^{ex}W^{de} \parallel g^x \parallel g^y \parallel \mathsf{id} \parallel A \parallel \mathsf{id}' \parallel B)$.
2. Set $A \leftarrow U$ and $Y \leftarrow V$. Choose $x, b \in_R \mathbb{Z}_q$.
   Run D on the session string $(Y^xW^dg^{bex}A^{bde} \parallel g^x \parallel Y \parallel \mathsf{id} \parallel A \parallel \mathsf{id}' \parallel g^b)$.
3. Set $X \leftarrow U$ and $B \leftarrow V$. Choose $a, y \in_R \mathbb{Z}_q$.
   Run D on the session string $(X^yg^{ady}W^eB^{ade} \parallel X \parallel g^y \parallel \mathsf{id} \parallel g^a \parallel \mathsf{id}' \parallel B)$.
4. Set $X \leftarrow U$ and $Y \leftarrow V$. Choose $a, b \in_R \mathbb{Z}_q$.
   Run D on the session string $(WY^{ad}X^{be}g^{abde} \parallel X \parallel Y \parallel \mathsf{id} \parallel g^a \parallel \mathsf{id}' \parallel g^b)$.

E outputs the result of D.

Note that in each of the above cases, if (U, V, W) is a real Diffie–Hellman triple, then D is run on a real CMQV' session string, whereas if (U, V, W) is a random triple, then D is run on a random session string. Thus, if D is a distinguisher for CMQV' session strings, then E is a distinguisher for the Diffie–Hellman relation. □

B. Proofs of Chapter 5

B.1. Proof of Proposition 13

Proof. It is straightforward to verify the first condition of Definition 36. We next verify that the second condition of Definition 36 holds. Let E denote a PPT adversary against protocol $\pi :=$ CNX. We show that the probability of event Multiple-Match$_{\pi,E}^{W(\Omega^{-}_{\mathsf{INDP\text{-}DH}})}(k)$ is bounded above by a negligible function in the security parameter k, where Multiple-Match$_{\pi,E}^{W(\Omega^{-}_{\mathsf{INDP\text{-}DH}})}(k)$ denotes the event that, in the security experiment, there exist a session s with $s_{status} = \texttt{accepted}$ and at least two distinct sessions s' and s'' that are matching session s. Note that, if both sessions s' and s'' are matching session s, then it must hold that $s''_{actor} = s'_{actor}$ and $s''_{role} = s'_{role}$). In addition, the counter value in two different sessions of the same user are distinct. For some fixed session s that has accepted, let Ev denote the event that there exist two distinct sessions s' and s'' such that s and s' are matching as well as s and s''. We have:

$$\begin{aligned} P(Ev) &\leq P(\bigcup_{\substack{s',s'' \\ s'\neq s''}} \{H_1(s''_{rand}, \mathsf{sk}_{\hat{P}}, i) = H_1(s'_{rand}, \mathsf{sk}_{\hat{P}}, j)\}) \\ &\leq \sum_{\substack{s',s'' \\ s'\neq s''}} P(\{H_1(s''_{rand}, \mathsf{sk}_{\hat{P}}, i) = H_1(s'_{rand}, \mathsf{sk}_{\hat{P}}, j)\}) \\ &\leq q_s^2 \frac{1}{p}, \end{aligned}$$

where $\hat{P} = s''_{actor} = s'_{actor}$, $i \neq j$ and q_s denotes the number of created sessions (either via the create or the cr-create query) by the adversary. Therefore, $P(\text{Multiple-Match}_{\pi,E}^{W(\Omega_{\mathsf{INDP\text{-}DH}})}(k)) \leq q_s^3 \frac{1}{p}$.

The third condition of Definition 36 is implied by an adaptation of the security proof of NAXOS in the eCK^w model from Section 3.3.1. Let s^* denote the test session. Consider first the event K^c where the adversary M wins the security experiment against π with non-negligible advantage and does not query H_2 with $(\sigma_1, \sigma_2, \sigma_3, \hat{A}, \hat{B})$, where $\sigma_1 = \text{CDH}(Y, A), \sigma_2 = \text{CDH}(B, X)$ and $\sigma_3 = \text{CDH}(X, Y)$.

Event K^c
If event K^c occurs, then the adversary M must have issued a session-key query to some session s such that $K_s = K_{s^*}$ (where K_s and K_{s^*} denote the session keys computed in sessions s and s^*, respectively) and s does not match s^*. We consider the following four events:

1. A_1 : there exist two distinct sessions s', s'' created via a create query such that $s'_{rand} = s''_{rand}$.
2. A_2 : there exists a session $s \neq s^*$ such that $H_1(s_{rand}, \mathsf{sk}_{s_{actor}}, i) = H_1(s^*_{rand}, \mathsf{sk}_{s^*_{actor}}, j)$.

3. A_3 : there exists a session $s' \neq s^*$ such that $H_2(\text{input}_{s'}) = H_2(\text{input}_{s^*})$ with $\text{input}_{s'} \neq \text{input}_{s^*}$.
4. A_4 : there exists an adversarial query input_M to the oracle H_2 such that $H_2(\text{input}_M) = H_2(\text{input}_{s^*})$ with $\text{input}_M \neq \text{input}_{s^*}$.

In contrast to the NAXOS protocol with respect to model $\Omega_{\mathsf{INDP\text{-}DH}}$, the adversary cannot force two sessions of protocol π of the same user with the same role to compute the same session key via a chosen-randomness replay attack, as the H_1 values in both sessions will be different with overwhelming probability due to different counter values. The latter event is included in event A_2.

Analysis of event K^c
We denote by q_s the number of created sessions (either via the create or the cr-create query) by the adversary and by q_{ro2} the number of queries to the random oracle H_2. We have that

$$\begin{aligned} P(K^c) &\leq P(A_1 \vee A_2 \vee A_3 \vee A_4) \leq P(A_1) + P(A_2) + P(A_3) + P(A_4) \\ &\leq \frac{q_s^2}{2}\frac{1}{2^k} + \frac{q_s}{p} + \frac{q_s + q_{\text{ro2}}}{2^k}, \end{aligned}$$

which is a negligible function of the security parameter k.

In the subsequent events (and their analyses) we assume that no collisions in the queries to the oracle H_1 occur and that none of the events $A_1, ..., A_4$ occurs. As in the proof of Proposition 7, we next consider the following three events:

1. $DL \wedge K$,
2. $T_O \wedge DL^c \wedge K$, and
3. $(T_O)^c \wedge DL^c \wedge K$, where

T_O denotes the event that there exists an origin-session for the test session, DL denotes the event where there exists a user $\hat{C} \in \mathcal{P}$ such that the adversary M, during its execution, queries H_1 with $(*, c, *)$ before issuing a $\mathsf{corrupt}(\hat{C})$ query and K denotes the event that M wins the security experiment against NAXOS by querying H_2 with $(\sigma_1, \sigma_2, \sigma_3, \hat{A}, \hat{B})$, where $\sigma_1 = \text{CDH}(Y, A), \sigma_2 = \text{CDH}(B, X)$ and $\sigma_3 = \text{CDH}(X, Y)$.

Event $DL \wedge K$
Let the input to the GDL challenge be C. Suppose that event $DL \wedge K$ occurs with non-negligible probability. In this case, the simulator S chooses one user $\hat{C} \in \mathcal{P}$ at random and sets its long-term public key to C. S chooses long-term secret/public key pairs for the remaining honest parties and stores the associated long-term secret keys. Additionally S chooses a random value $m \in_R \{1, 2, ..., q_s\}$. We denote the m'th activated session by adversary M by s^*. Suppose further that $s^*_{actor} = \hat{A}, s^*_{peer} = \hat{B}$ and $s^*_{role} = \mathcal{I}$, w.l.o.g. We now define S's responses to M's queries for the pre-specified peer setting; the post-specified peer case proceeds similarly. Algorithm S maintains tables Q, J, T and L, all of which are initially empty. S also maintains a variable ω initialized with 1 and a table CV maintaining for each user the current counter value. Initially, table CV contains an entry $(\hat{P}, 0)$ for each user $\hat{P} \in \mathcal{P}$.

1. $\mathsf{create}\left(\hat{P}, r, \hat{Q}\right)$ to create session s: S checks whether $\hat{P} \in \mathcal{P}, \hat{Q} \in \mathcal{P}$, and $r \in \{\mathcal{I}, \mathcal{R}\}$. If one of the checks fails, then S returns $\perp$. Else, S initializes the

session variables according to the protocol specification, and stores an entry of the form $(s, s_{rand}, \mathsf{sk}_{s_{actor}}, l_s, \kappa) \in (\mathcal{P} \times \mathbb{N}) \times \{0,1\}^k \times (\mathbb{Z}_p \cup \{*\}) \times \mathbb{N} \times \mathbb{Z}_p$ in table Q as follows:

- S retrieves the counter value c for the user with identifier $\hat{P}$ from table CV, increments c by 1, and updates the counter value for $\hat{P}$ stored in table CV with $c+1$,
- S chooses $s_{rand} \in_R \{0,1\}^k$ (i. e. the randomness of session s),
- S chooses $\kappa \in_R \mathbb{Z}_p$,
- if $s_{actor} \neq \hat{C}$, then S stores the entry $(s, s_{rand}, \mathsf{sk}_{s_{actor}}, c+1, \kappa)$ in Q, else S stores the entry $(s, s_{rand}, *, c+1, \kappa)$ in Q,[1] and
- if $r = \mathcal{I}$, then S returns the Diffie-Hellman exponential g^κ to M, else S returns $\star$.

2. $\mathsf{cr\text{-}create}\left(\hat{P}, r, str, \hat{Q}\right)$ to create session s: S checks whether $\hat{P} \in \mathcal{P}, \hat{Q} \in \mathcal{P}$, and $r \in \{\mathcal{I}, \mathcal{R}\}$. If one of the checks fails, then S returns $\perp$. Else, S initializes the session variables according to the protocol specification, and stores an entry of the form $(s, s_{rand}, \mathsf{sk}_{s_{actor}}, l_s, \kappa) \in (\mathcal{P} \times \mathbb{N}) \times \{0,1\}^k \times (\mathbb{Z}_p \cup \{*\}) \times \mathbb{N} \times \mathbb{Z}_p$ in table Q as follows:
   - S retrieves the counter value c for the user with identifier $\hat{P}$ from table CV, increments c by 1, and updates the counter value for $\hat{P}$ stored in table CV with $c+1$,
   - if there is an entry $(r_i, h_i, l_i, \kappa_i)$ in table J such that $r_i = str$, $h_i = \mathsf{sk}_{\hat{P}}$, and $l_i = c+1$, then S sets $\omega \leftarrow \kappa_i$, else S chooses $\kappa \in_R \mathbb{Z}_p$, and sets $\omega \leftarrow \kappa$.[2]
   - if $s_{actor} \neq \hat{C}$, then S stores the entry $(s, s_{rand}, \mathsf{sk}_{s_{actor}}, c+1, x_5)$ in Q, else S stores the entry $(s, s_{rand}, *, c+1, x_5)$ in Q, where x_5 denotes the value of variable ω,
   - if $r = \mathcal{I}$, then S returns the Diffie-Hellman exponential g^κ to M, else S returns $\star$.
3. S stores entries of the form $(r, h, l, \kappa) \in \{0,1\}^k \times \mathbb{Z}_p \times \mathbb{N} \times \mathbb{Z}_p$ in table J. When M makes a query of the form (r, h, l) to the random oracle for H_1, answer it as follows:
   - If $C = g^h$, then S aborts M and is successful by outputting $\mathrm{DLog}_g(C) = h$.
   - Else if $(r, h, l, \kappa) \in J$ for some $\kappa \in \mathbb{Z}_p$, then S returns κ to M.
   - Else if there exists an entry $(s, s_{rand}, \mathsf{sk}_{s_{actor}}, l_s, \kappa)$ in Q, for some $s \in \mathcal{P} \times \mathbb{N}, s_{rand} \in \{0,1\}^k, \mathsf{sk}_{s_{actor}} \in \mathbb{Z}_p$, $l_s \in \mathbb{N}$ and $\kappa \in \mathbb{Z}_p$, such that $s_{rand} = r$, $\mathsf{sk}_{s_{actor}} = h$ and $l_s = l$, then S returns κ to M and stores the entry (r, h, l, κ) in table J.
   - Else, S chooses $\kappa \in_R \mathbb{Z}_p$, returns it to M and stores the entry (r, h, l, κ) in

[1] We do not need to keep consistency with H_1 queries via lookup in table J since the probability that the adversary guesses the randomness of a session created via a query create is negligible.

[2] Here we need to keep consistency with H_1 queries via lookup in table J to be able to consistently answer all possible combinations of queries. Consider, e. g., the following scenario. The adversary first issues a query $(x, \mathsf{sk}_{\hat{P}}, i)$ to H_1 and then issues the query $\mathsf{cr\text{-}create}(\hat{P}, r, x, \hat{Q})$, which increments the current counter value $i-1$ by 1 so that the counter value used in session $s = (\hat{P}, i)$ is i. So, in contrast to the NAXOS proof with respect to model eCK^w, we need to additionally keep consistency between cr-create queries and queries to the random oracle for H_1.

table J.

4. $\mathsf{send}(\hat{P}, i, V)$ to send message V to session $s = (\hat{P}, i)$: If $s_{status} \neq \texttt{active}$, then S returns $\perp$. Else if $s_{role} = \mathcal{I}$, then S does the following. If $V \notin G$, then the status of session s is set to `rejected`. Else, the status of session s is set to `accepted`, the variable $recv$ is updated to $s_{recv} \leftarrow (s_{recv}, V)$ and
   - If there exists an entry $(s_{peer}, s_{actor}, \mathcal{R}, s_{recv}, s_{sent}, \lambda)$ in table T, then S stores the entry $(s_{actor}, s_{peer}, \mathcal{I}, s_{sent}, s_{recv}, \lambda)$ in table T.
   - Else if there exists an entry $(\sigma_1, \sigma_2, \sigma_3, s_{actor}, s_{peer}, \lambda)$ in table L, for some $\lambda \in \{0,1\}^k$, such that $\mathrm{DDH}(s_{recv}, s_{sent}, \sigma_3) = 1$, $\mathrm{DDH}(s_{sent}, \mathsf{pk}_{s_{peer}}, \sigma_2) = 1$ and $\mathrm{DDH}(s_{recv}, \mathsf{pk}_{s_{actor}}, \sigma_1) = 1$, then S stores $(s_{actor}, s_{peer}, \mathcal{I}, s_{sent}, s_{recv}, \lambda)$ in table T.
   - Else, S chooses $\mu \in_R \{0,1\}^k$ and stores the entry $(s_{actor}, s_{peer}, \mathcal{I}, s_{sent}, s_{recv}, \mu)$ in T.

   Else if $s_{role} = \mathcal{R}$, then S does the following. If $V \notin G$, then the status of session s is set to `rejected`. Else, S sets the status of session s to `accepted`, and the variable $recv$ to (s_{recv}, V). S returns g^{κ} to M, where κ denotes the last element of the entry $(s, r, \mathsf{sk}_{s_{actor}}, l, \kappa)$ in table Q, and proceeds in a similar way as in the previous case.
5. When M makes a query of the form $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j\right)$ to the random oracle for H_2, answer it as follows:
   - If $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j, \lambda\right) \in L$ for some $\lambda \in \{0,1\}^k$, then S returns λ to M.
   - Else if there exist entries $\left(\hat{Q}_i, \hat{Q}_j, \mathcal{I}, U, V, \lambda\right)$ or $\left(\hat{Q}_j, \hat{Q}_i, \mathcal{R}, V, U, \lambda\right)$ in table T, for some $\lambda \in \{0,1\}^k$ and $U, V \in G$, such that $\mathrm{DDH}(V, U, \sigma_3) = 1$, $\mathrm{DDH}(V, \mathsf{pk}_{\hat{Q}_i}, \sigma_1) = 1$ and $\mathrm{DDH}(U, \mathsf{pk}_{\hat{Q}_j}, \sigma_2) = 1$, then S returns λ to M and stores the entry $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j, \lambda\right)$ in table L.
   - Else, S chooses $\mu \in_R \{0,1\}^k$, returns it to M and stores the entry $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j, \mu\right)$ in L.
6. $\mathsf{randomness}(s)$: If $s_{status} = \perp$, then S returns $\perp$. Otherwise, S returns s_{rand} (via lookup in table Q).
7. $\mathsf{session\text{-}key}(s)$: If $s_{status} \neq \texttt{accepted}$, then S returns $\perp$. Otherwise, S answers this query by lookup in table T.
8. $\mathsf{test\text{-}session}(s)$: If $s \neq s^*$, then S aborts; otherwise S answers the query in the appropriate way.
9. $\mathsf{corrupt}(\hat{P})$: If $\hat{P} \notin \mathcal{P}$, then S returns $\perp$. Else if $\hat{P} = \hat{C}$, then S aborts. Else S returns $\mathsf{sk}_{\hat{P}}$.
10. M outputs a guess: S aborts.

Analysis of event $DL \wedge K$
Similar to the analysis of the related event $DL \wedge K$ in the proof of Proposition 7.

Event $T_O \wedge DL^c \wedge K$
Let s^* and s' denote the test session and the origin-session for the test session, respectively. We split event $Evt := T_O \wedge DL^c \wedge K$ into the following events $B_1, ..., B_3$ so that $Evt = B_1 \vee B_2 \vee B_3$:

1. B_1 : *Evt* occurs and $s^*_{peer} = s'_{actor}$.
2. B_2 : *Evt* occurs and $s^*_{peer} \neq s'_{actor}$ and M does issue neither a $\mathsf{randomness}(s')$ query nor a $\mathsf{cr\text{-}create}(s', \times)$ query to the origin-session s' of s^*, but may issue a $\mathsf{corrupt}(s^*_{peer})$ query.
3. B_3 : *Evt* occurs and $s^*_{peer} \neq s'_{actor}$ and M does not issue a $\mathsf{corrupt}(s^*_{peer})$ query, but may issue either a $\mathsf{randomness}(s')$ query or a $\mathsf{cr\text{-}create}(s', \times)$ query to the origin-session s' of s^*.

Event B_1

Let the input to the GDH challenge be (X_0, Y_0). Suppose that event B_1 occurs with non-negligible probability. In this case S chooses long-term secret/public key pairs for all the honest parties and stores the associated long-term secret keys. Additionally S chooses two random values $m, n \in_R \{1, 2, ..., q_s\}$. The m'th activated session by adversary M will be called s^* and the n'th activated session will be called s'. Suppose further that $s^*_{actor} = \hat{A}, s^*_{peer} = \hat{B}$ and $s^*_{role} = \mathcal{I}$, w.l.o.g.. The simulation of M's environment proceeds as follows:

1. $\mathsf{create}(\hat{A}, \mathcal{I}, \hat{B})$ or $\mathsf{cr\text{-}create}(\hat{A}, \mathcal{I}, str, \hat{B})$ to create session s^*: If create is issued, then S chooses $s^*_{rand} \in_R \{0,1\}^k$. Else, S sets $s^*_{rand} \leftarrow str$. Then, S (a) returns the message X_0, where (X_0, Y_0) is the GDH challenge, (b) increments by 1 the counter value c for the user with identifier $\hat{A}$ (stored in table CV), and (c) stores the updated counter value $c + 1$ for $\hat{A}$ in table CV.[3]
2. $\mathsf{create}(\hat{B}, r, \hat{Q})$ or $\mathsf{cr\text{-}create}(\hat{B}, r, str, \hat{Q})$ to create session s': If create is issued, then S chooses $s'_{rand} \in_R \{0,1\}^k$. Else, S sets $s'_{rand} \leftarrow str$. S then increments by 1 the counter value c for the user with identifier $\hat{B}$ (stored in table CV), and stores the updated counter value $c + 1$ for $\hat{B}$ in table CV.If $r = \mathcal{I}$, then S returns message Y_0 to M, where (X_0, Y_0) is the GDH challenge. Else, $\star$ is returned.
3. $\mathsf{send}(\hat{B}, i, Z)$ with $(\hat{B}, i) = s'$: If $s'_{status} \neq \texttt{active}$, then S returns $\perp$. Else if $s'_{role} = \mathcal{R}$ and $Z \in G$, then S returns message Y_0 to M, where (X_0, Y_0) is the GDH challenge, sets the status of session s' to `accepted`, and proceeds as in the previous simulation for completing the session. Else, S proceeds as in the previous simulation.
4. $\mathsf{send}(\hat{A}, i, Y_0)$ with $(\hat{A}, i) = s^*$: S proceeds as in the previous simulation for completing the session.
5. Other $\mathsf{create}, \mathsf{cr\text{-}create}$ and send queries are answered as in the previous simulation.
6. $\mathsf{randomness}(s)$: If $s_{status} = \perp$, then S returns $\perp$. Else, S returns s_{rand}.
7. $\mathsf{session\text{-}key}(s)$: If $s_{status} \neq \texttt{accepted}$, then S returns $\perp$. Otherwise, S answers this query by lookup in table T.
8. $\mathsf{test\text{-}session}(s)$: If $s \neq s^*$ or if s' is not the origin-session for session s^*, then S aborts; otherwise S answers the query in the appropriate way.
9. $H_1(r, h, *)$: If $h = a$ and $r = s^*_{rand}$ or if $h = b$ and $r = s'_{rand}$, then S aborts. Otherwise S simulates a random oracle as in the previous simulation.
10. $\mathsf{corrupt}(\hat{P})$: If $\hat{P} \notin \mathcal{P}$, then S returns $\perp$. Else, S returns $\mathsf{sk}_{\hat{P}}$.

[3] Note that s^*_{rand} is not used in the calculation.

11. When M makes a query of the form $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j\right)$ to the random oracle for H_2, answer it as follows:
    - If $\left\{\hat{Q}_i, \hat{Q}_j\right\} = \left\{\hat{A}, \hat{B}\right\}$, $\sigma_1 = Y_0^a$, $\sigma_2 = X_0^b$ and $\mathrm{DDH}(X_0, Y_0, \sigma_3) = 1$, then S aborts M and is successful by outputting $\mathrm{CDH}(X_0, Y_0) = \sigma_3$.
    - Else if $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j, \lambda\right) \in L$ for some $\lambda \in \{0,1\}^k$, then S returns λ to M.
    - Else if there exist entries $\left(\hat{Q}_i, \hat{Q}_j, \mathcal{I}, U, V, \lambda\right)$ or $\left(\hat{Q}_j, \hat{Q}_i, \mathcal{R}, V, U, \lambda\right)$ in table T, for some $\lambda \in \{0,1\}^k$ and $U, V \in G$, such that $\mathrm{DDH}(V, U, \sigma_3) = 1$, $\mathrm{DDH}(V, \mathsf{pk}_{\hat{Q}_i}, \sigma_1) = 1$ and $\mathrm{DDH}(U, \mathsf{pk}_{\hat{Q}_j}, \sigma_2) = 1$, then S returns λ to M and stores the entry $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j, \lambda\right)$ in table L.
    - Else, S chooses $\mu \in_R \{0,1\}^k$, returns it to M and stores the entry $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j, \mu\right)$ in L.
12. M outputs a guess: S aborts.

Analysis of event B_1
Similar to the analysis of the related event B_1 in the proof of Proposition 7.

Event B_2
Let the input to the GDH challenge be (X_0, Y_0). Suppose that event B_2 occurs with non-negligible probability. The simulation of S proceeds in the same way as for event B_1 with the following changes:

- $\mathsf{create}(\hat{B}, r, \hat{Q})$ or $\mathsf{cr\text{-}create}(\hat{B}, r, str, \hat{Q})$ to create session s': If $\mathsf{cr\text{-}create}$ is issued, then S aborts. Else, S proceeds as described before.
- $\mathsf{randomness}(s)$: If $s_{status} = \perp$, then S returns $\perp$. Else if $s = s'$, then S aborts. Else, S returns s_{rand}.
- $H_1(r, h, *)$: If $h = a$ and $r = s^*_{rand}$, then S aborts. Otherwise S simulates a random oracle as in the previous simulation.

Analysis of event B_2
Similar to the analysis of the related event B_2 in the proof of Proposition 7.

Event B_3
Let the input to the GDH challenge be (X_0, B). Suppose that event B_3 occurs with non-negligible probability. In this case, S chooses one user $\hat{B} \in \mathcal{P}$ at random from the set $\mathcal{P}$ and sets its long-term public key to B. S chooses long-term secret/public key pairs for the remaining parties in $\mathcal{P}$ and stores the associated long-term secret keys. Additionally S chooses two random values $m, n \in_R \{1, 2, ..., q_s\}$. We denote the m'th activated session by adversary M by s^* and the n'th activated session by s'. Suppose further that $s^*_{actor} = \hat{A}, s^*_{peer} = \hat{B}$ and $s^*_{role} = \mathcal{I}$, w.l.o.g.. Algorithm S maintains tables Q, J, T and L, all of which are initially empty. S also maintains a variable ω initialized with 1 and a table CV maintaining for each user the current counter value. Initially, table CV contains an entry $(\hat{P}, 0)$ for each user $\hat{P} \in \mathcal{P}$. The simulation of M's environment proceeds as follows:

1. $\mathsf{create}(\hat{A}, \mathcal{I}, \hat{B})$ or $\mathsf{cr\text{-}create}(\hat{A}, \mathcal{I}, str, \hat{B})$ to create session s^*: If create is issued, then S chooses $s^*_{rand} \in_R \{0,1\}^k$. Else, S sets $s^*_{rand} \leftarrow str$. Then, S (a) returns

the message X_0, (b) increments by 1 the counter value c for the user with identifier $\hat{A}$ (stored in table CV), and (c) stores the updated counter value $c+1$ for $\hat{A}$ in table CV.

2. $\mathsf{create}\left(\hat{P}, r, \hat{Q}\right)$ to create session s: S checks whether $\hat{P} \in \mathcal{P}, \hat{Q} \in \mathcal{P}$, and $r \in \{\mathcal{I}, \mathcal{R}\}$. If one of the checks fails, then S returns $\perp$. Else, S initializes the session variables according to the protocol specification, and stores an entry of the form $(s, s_{rand}, \mathsf{sk}_{s_{actor}}, l_s, \kappa) \in (\mathcal{P} \times \mathbb{N}) \times \{0,1\}^k \times (\mathbb{Z}_p \cup \{*\}) \times \mathbb{N} \times \mathbb{Z}_p$ in table Q as follows:
   - S retrieves the counter value c for the user with identifier $\hat{P}$ from table CV, increments c by 1, and updates the counter value for $\hat{P}$ stored in table CV with $c+1$,
   - S chooses $s_{rand} \in_R \{0,1\}^k$ (i. e. the randomness of session s),
   - S chooses $\kappa \in_R \mathbb{Z}_p$,
   - if $s_{actor} \neq \hat{B}$, then S stores the entry $(s, s_{rand}, \mathsf{sk}_{s_{actor}}, c+1, \kappa)$ in Q, else S stores the entry $(s, s_{rand}, *, c+1, \kappa)$ in Q, and
   - if $r = \mathcal{I}$, then S returns the Diffie-Hellman exponential g^κ to M, else S returns $\star$.
3. $\mathsf{cr\text{-}create}\left(\hat{P}, r, str, \hat{Q}\right)$ to create session s: S checks whether $\hat{P} \in \mathcal{P}, \hat{Q} \in \mathcal{P}$, and $r \in \{\mathcal{I}, \mathcal{R}\}$. If one of the checks fails, then S returns $\perp$. Else, S initializes the session variables according to the protocol specification, and stores an entry of the form $(s, s_{rand}, \mathsf{sk}_{s_{actor}}, l_s, \kappa) \in (\mathcal{P} \times \mathbb{N}) \times \{0,1\}^k \times (\mathbb{Z}_p \cup \{*\}) \times \mathbb{N} \times \mathbb{Z}_p$ in table Q as follows:
   - S retrieves the counter value c for the user with identifier $\hat{P}$ from table CV, increments c by 1, and updates the counter value for $\hat{P}$ stored in table CV with $c+1$,
   - if there is an entry $(r_i, h_i, l_i, \kappa_i)$ in table J such that $r_i = str$, $h_i = \mathsf{sk}_{\hat{P}}$, and $l_i = c+1$, then S sets $\omega \leftarrow \kappa_i$, else S chooses $\kappa \in_R \mathbb{Z}_p$, and sets $\omega \leftarrow \kappa$.
   - if $s_{actor} \neq \hat{B}$, then S stores the entry $(s, s_{rand}, \mathsf{sk}_{s_{actor}}, c+1, x_5)$ in Q, else S stores the entry $(s, s_{rand}, *, c+1, x_5)$ in Q, where x_5 denotes the value of variable ω,
   - if $r = \mathcal{I}$, then S returns the Diffie-Hellman exponential g^κ to M, else S returns $\star$.
4. S stores entries of the form $(r, h, l, \kappa) \in \{0,1\}^k \times \mathbb{Z}_p \times \mathbb{N} \times \mathbb{Z}_p$ in table J. When M makes a query of the form (r, h, l) to the random oracle for H_1, answer it as follows:
   - If $r = s^*_{rand}$ and $h = a$, then S aborts,
   - Else if $(r, h, l, \kappa) \in J$ for some $\kappa \in \mathbb{Z}_p$, then S returns κ to M.
   - Else if there exists an entry $(s, s_{rand}, \mathsf{sk}_{s_{actor}}, l_s, \kappa)$ in Q, for some $s \in \mathcal{P} \times \mathbb{N}, s_{rand} \in \{0,1\}^k, \mathsf{sk}_{s_{actor}} \in \mathbb{Z}_p$, $l_s \in \mathbb{N}$ and $\kappa \in \mathbb{Z}_p$, such that $s_{rand} = r$, $\mathsf{sk}_{s_{actor}} = h$ and $l_s = l$, then S returns κ to M and stores the entry (r, h, l, κ) in table J.
   - Else, S chooses $\kappa \in_R \mathbb{Z}_p$, returns it to M and stores the entry (r, h, l, κ) in table J.
5. $\mathsf{send}(\hat{P}, i, V)$ to send message V to session $s = (\hat{P}, i)$: If $s_{status} \neq \mathtt{active}$, then

S returns $\perp$. Else if $s_{role} = \mathcal{I}$, then S does the following. If $V \notin G$, then the status of session s is set to `rejected`. Else, the status of session s is set to `accepted`, the variable $recv$ is updated to $s_{recv} \leftarrow (s_{recv}, V)$ and

- If there exists an entry $(s_{peer}, s_{actor}, \mathcal{R}, s_{recv}, s_{sent}, \lambda)$ in table T, then S stores the entry $(s_{actor}, s_{peer}, \mathcal{I}, s_{sent}, s_{recv}, \lambda)$ in table T.
- Else if there exists an entry $(\sigma_1, \sigma_2, \sigma_3, s_{actor}, s_{peer}, \lambda)$ in table L, for some $\lambda \in \{0,1\}^k$, such that $\mathrm{DDH}(s_{recv}, s_{sent}, \sigma_3) = 1$, $\mathrm{DDH}(s_{sent}, \mathsf{pk}_{s_{peer}}, \sigma_2) = 1$ and $\mathrm{DDH}(s_{recv}, \mathsf{pk}_{s_{actor}}, \sigma_1) = 1$, then S stores $(s_{actor}, s_{peer}, \mathcal{I}, s_{sent}, s_{recv}, \lambda)$ in table T.
- Else, S chooses $\mu \in_R \{0,1\}^k$ and stores the entry $(s_{actor}, s_{peer}, \mathcal{I}, s_{sent}, s_{recv}, \mu)$ in T.

Else if $s_{role} = \mathcal{R}$, then S does the following. If $V \notin G$, then the status of session s is set to `rejected`. Else, S sets the status of session s to `accepted`, and the variable $recv$ to (s_{recv}, V). S returns g^{κ} to M, where κ denotes the last element of the entry $(s, r, \mathsf{sk}_{s_{actor}}, l, \kappa)$ in table Q, and proceeds in a similar way as in the previous case.

6. When M makes a query of the form $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j\right)$ to the random oracle for H_2, answer it as follows:
   - If $s'_{status} \neq \perp$, $\left\{\hat{Q}_i, \hat{Q}_j\right\} = \left\{\hat{A}, \hat{B}\right\}$, $\sigma_1 = A^{\kappa}$, $\mathrm{DDH}(X_0, B, \sigma_2) = 1$, and $\sigma_3 = X_0^{\kappa}$, where κ denotes the last element of the entry $(s', s'_{rand}, \mathsf{sk}_{s'_{actor}}, l, \kappa)$ in table Q[4], then S aborts M and is successful by outputting $\mathrm{CDH}(X_0, B) = \sigma_2$.
   - Else if $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j, \lambda\right) \in L$ for some $\lambda \in \{0,1\}^k$, then S returns λ to M.
   - Else if there exist entries $\left(\hat{Q}_i, \hat{Q}_j, \mathcal{I}, U, V, \lambda\right)$ or $\left(\hat{Q}_j, \hat{Q}_i, \mathcal{R}, V, U, \lambda\right)$ in table T, for some $\lambda \in \{0,1\}^k$ and $U, V \in G$, such that $\mathrm{DDH}(V, U, \sigma_3) = 1$, $\mathrm{DDH}(V, \mathsf{pk}_{\hat{Q}_i}, \sigma_1) = 1$ and $\mathrm{DDH}(U, \mathsf{pk}_{\hat{Q}_j}, \sigma_2) = 1$, then S returns λ to M and stores the entry $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j, \lambda\right)$ in table L.
   - Else, S chooses $\mu \in_R \{0,1\}^k$, returns it to M and stores the entry $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j, \mu\right)$ in L.
7. $\mathsf{randomness}(s)$: If $s_{status} = \perp$, then S returns $\perp$. Else, S returns s_{rand}.
8. $\mathsf{session\text{-}key}(s)$: If $s_{status} \neq \mathtt{accepted}$, then S returns $\perp$. Otherwise, S answers this query by lookup in table T.
9. $\mathsf{test\text{-}session}(s)$: If $s \neq s^*$ or if s' is not the origin-session for session s^*, then S aborts; otherwise S answers the query in the appropriate way.
10. $\mathsf{corrupt}(\hat{P})$: If $\hat{P} \notin \mathcal{P}$, then S returns $\perp$. Else if $\hat{P} = \hat{B}$, then S aborts. Else, S returns $\mathsf{sk}_{\hat{P}}$.
11. M outputs a guess: S aborts.

Analysis of event B_3
Similar to the analysis of the related event B_3 in the proof of Proposition 7.

Event $(T_O)^c \wedge DL^c \wedge K$

[4]This entry exists in table Q since the status of the session is different to $\perp$.

The simulation and analysis are very similar to the simulation and analysis related to event B_3. □

B.2. Proof of Proposition 15

Proof. It is straightforward to verify the first condition of Definition 36. We next verify that the second condition of Definition 36 holds. Let E denote a PPT adversary against protocol $\pi := \text{NXPR}$. We show that the probability of event $\text{Multiple-Match}_{\pi,E}^{W(\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{ISM}})}(k)$ is bounded above by a negligible function in the security parameter k, where $\text{Multiple-Match}_{\pi,E}^{W(\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{ISM}})}(k)$ denotes the event that, in the security experiment, there exist a session s with $s_{status} = \texttt{accepted}$ and at least two distinct sessions s' and s'' that are matching session s. Note that, if both sessions s' and s'' are matching session s, then it must hold that $s''_{actor} = s'_{actor}$ and $s''_{role} = s'_{role}$. In addition, it is easy to see that the value of the variable *data* in two different sessions of the same user are distinct (since of different length). For some fixed session s that has accepted, let Ev denote the event that there exist two distinct sessions s' and s'' such that s and s' are matching as well as s and s''. We have:

$$\begin{aligned} P(Ev) &\leq P(\bigcup_{\substack{s',s'' \\ s'\neq s''}} \{H_1(s''_{rand}, s''_{data}, \mathsf{sk}_{\hat{P}}) = H_1(s'_{rand}, s'_{data}, \mathsf{sk}_{\hat{P}})\}) \\ &\leq \sum_{\substack{s',s'' \\ s'\neq s''}} P(\{H_1(s''_{rand}, s''_{data}, \mathsf{sk}_{\hat{P}}) = H_1(s'_{rand}, s'_{data}, \mathsf{sk}_{\hat{P}})\}) \\ &\leq \frac{q_s^2}{p}, \end{aligned}$$

where $\hat{P} = s''_{actor} = s'_{actor}$ and q_s denotes the number of created sessions (either via the create or the cr-create query) by the adversary.

In the above computation, we distinguished between the following two events:

1. $D_1 := \{s''_{rand} \neq s'_{rand} \wedge s''_{data} \neq s'_{data}\}$; the probability that the two hash values are identical given D_1 is the probability of a collision in the hash function, and
2. $D_2 := \{s''_{rand} = s'_{rand} \wedge s''_{data} \neq s'_{data}\}$; the probability that the two hash values are identical given D_2 is the probability of a collision in the hash function.

The events $D_3 := \{s''_{rand} = s'_{rand} \wedge s''_{data} = s'_{data}\}$ and $D_4 := \{s''_{rand} \neq s'_{rand} \wedge s''_{data} = s'_{data}\}$ both occur with probability zero.

Even though the value of the variable *rand* can be the same for two different session of the same user due to the queries cr-create and randomness, the value of the variable *data* of two different sessions s' and s'' of the same user is always different since the bit strings s'_{data} and s''_{data} differ in length. Given a created session s, the length of the bit string s_{data} depends on the number of sessions of user s_{actor} that have already been created either via create or cr-create.

Finally, $P(\text{Multiple-Match}_{\pi,E}^{W(\Omega_{\mathsf{INDP\text{-}DH}\cap\mathsf{ISM}})}(k)) \leq q_s^3 \frac{1}{p}$.

The third condition of Definition 36 is implied by an adaptation of the security proof of protocol CNX in the $\Omega^{-}_{\mathsf{INDP\text{-}DH}}$ model (see Appendix B.1). Let s^* denote the test session. Consider first the event K^c where the adversary M wins the security experiment against π with non-negligible advantage and does not query H_2 with $(\sigma_1, \sigma_2, \sigma_3, \hat{A}, \hat{B})$, where $\sigma_1 = \text{CDH}(Y, A), \sigma_2 = \text{CDH}(B, X)$ and $\sigma_3 = \text{CDH}(X, Y)$.

Event K^c

If event K^c occurs, then the adversary M must have issued a session-key query to some session s such that $K_s = K_{s^*}$ (where K_s and K_{s^*} denote the session keys computed in sessions s and s^*, respectively) and s does not match s^*. We consider the following four events:

1. A_1 : there exist two distinct sessions s', s'' created via a create query such that $s'_{rand} = s''_{rand}$.[5]
2. A_2 : there exists a session $s \neq s^*$ such that $H_1(s_{rand}, s_{data}) = H_1(s^*_{rand}, s^*_{data})$.
3. A_3 : there exists a session $s' \neq s^*$ such that $H_2(\text{input}_{s'}) = H_2(\text{input}_{s^*})$ with $\text{input}_{s'} \neq \text{input}_{s^*}$.
4. A_4 : there exists an adversarial query input_M to the oracle H_2 such that $H_2(\text{input}_M) = H_2(\text{input}_{s^*})$ with $\text{input}_M \neq \text{input}_{s^*}$.

Analysis of event K^c

We denote by q_s the number of created sessions (either via the query create or the query cr-create) by the adversary and by q_{ro2} the number of queries to the random oracle H_2. We have that

$$\begin{aligned} P(K^c) &\leq P(A_1 \vee A_2 \vee A_3 \vee A_4) \leq P(A_1) + P(A_2) + P(A_3) + P(A_4) \\ &\leq \frac{q_s^2}{2}\frac{1}{2^k} + \frac{q_s}{p} + \frac{q_s + q_{\text{ro2}}}{2^k}, \end{aligned}$$

which is a negligible function of the security parameter k.

In contrast to the NAXOS protocol analyzed with respect to model $\Omega_{\text{INDP-DH}}$, the adversary cannot force two sessions of protocol π of the same user with the same role to compute the same session key via a chosen-randomness replay attack since the H_1 values in both sessions will be different with overwhelming probability. The latter event is included in event A_2.

In the subsequent events (and their analyses) we assume that no collisions in the queries to the oracle H_1 occur and that none of the events $A_1, ..., A_4$ occurs. As in the proof of Proposition 7, we next consider the following three events:

1. $DL \wedge K$,
2. $T_O \wedge DL^c \wedge K$, and
3. $(T_O)^c \wedge DL^c \wedge K$, where

T_O denotes the event that there exists an origin-session for the test session, DL denotes the event where there exists a user $\hat{C} \in \mathcal{P}$ such that the adversary M, during its execution, queries H_1 with $(*, c)$ before issuing a corrupt($\hat{C}$) query and K denotes the event that M wins the security experiment against NXPR by querying H_2 with $(\sigma_1, \sigma_2, \sigma_3, \hat{A}, \hat{B})$, where $\sigma_1 = \text{CDH}(Y, A), \sigma_2 = \text{CDH}(B, X)$ and $\sigma_3 = \text{CDH}(X, Y)$.

Event $DL \wedge K$

Let the input to the GDL challenge be C. Suppose that event $DL \wedge K$ occurs with non-negligible probability. In this case, the simulator S chooses one user $\hat{C} \in \mathcal{P}$ at random and sets its long-term public key to C. S chooses long-term secret/public

[5] Under event A_1 the query randomness (e. g., for two sessions of different users) together with other queries might enable the adversary to learn all the information necessary to compute the session key of the target session without violating the freshness condition.

key pairs for the remaining honest parties and stores the associated long-term secret keys. Additionally S chooses a random value $m \in_R \{1, 2, ..., q_s\}$. We denote the m'th activated session by adversary M by s^*. Suppose further that $s^*_{actor} = \hat{A}, s^*_{peer} = \hat{B}$ and $s^*_{role} = \mathcal{I}$, w.l.o.g. We now define S's responses to M's queries for the pre-specified peer setting; the post-specified peer case proceeds similarly. Algorithm S maintains tables Q, J, T and L, all of which are initially empty. S also maintains a variable ω initialized with 1.

1. $\mathsf{create}\left(\hat{P}, r, \hat{Q}\right)$ to create session s: S checks whether $\hat{P} \in \mathcal{P}$, $\hat{Q} \in \mathcal{P}$, and $r \in \{\mathcal{I}, \mathcal{R}\}$. If one of the checks fails, then S returns $\perp$. Else, S initializes the session variables according to the protocol specification, and stores an entry of the form $(s, s_{rand}, l_s, \mathsf{sk}_{s_{actor}}, \kappa) \in (\mathcal{P} \times \mathbb{N}) \times \{0,1\}^k \times \{0,1\}^* \times (\mathbb{Z}_p \cup \{*\}) \times \mathbb{Z}_p$ in table Q as follows:
   - S chooses $s_{rand} \in_R \{0,1\}^k$ (i. e. the randomness of session s),
   - S chooses $\kappa \in_R \mathbb{Z}_p$,
   - if there is no entry $(s, s_{rand}, l_s, \mathsf{sk}_{s_{actor}}, \kappa)$ in table Q such that $s_{actor} = \hat{P}$, then S sets the value of l_s to s_{rand}, else S sets the value of l_s to $(s_{rand}, l_{s'})$, where s' is the previous session with $s'_{actor} = s_{actor}$ for which an entry in table Q has been made.[6]
   - if $s_{actor} \neq \hat{C}$, then S stores the entry $(s, s_{rand}, s_{data}, \mathsf{sk}_{s_{actor}}, \kappa)$ in Q, else S stores the entry $(s, s_{rand}, s_{data}, *, \kappa)$ in Q, and
   - if $r = \mathcal{I}$, then S returns the Diffie-Hellman exponential g^κ to M, else S returns $\star$.
2. $\mathsf{cr\text{-}create}\left(\hat{P}, r, str, \hat{Q}\right)$ to create session s: S checks whether $\hat{P} \in \mathcal{P}$, $\hat{Q} \in \mathcal{P}$, and $r \in \{\mathcal{I}, \mathcal{R}\}$. If one of the checks fails, then S returns $\perp$. Else, S initializes the session variables according to the protocol specification, and stores an entry of the form $(s, s_{rand}, l_s, \mathsf{sk}_{s_{actor}}, \kappa) \in (\mathcal{P} \times \mathbb{N}) \times \{0,1\}^k \times \{0,1\}^* \times (\mathbb{Z}_p \cup \{*\}) \times \mathbb{Z}_p$ in table Q as follows:
   - if there is an entry (r_i, h_i, κ_i) in table J such that $r_i = (str, l_{s'})$, and $h_i = \mathsf{sk}_{\hat{P}}$, where s' is the previous session with $s'_{actor} = s_{actor}$ for which an entry in table Q has been made, then S sets $\omega \leftarrow \kappa_i$, else S chooses $\kappa \in_R \mathbb{Z}_p$ and sets $\omega \leftarrow \kappa$.
   - if $s_{actor} \neq \hat{C}$, then S stores the entry $(s, s_{rand}, r_i, \mathsf{sk}_{s_{actor}}, \omega)$ in Q, else S stores the entry $(s, s_{rand}, l_s, *, \omega)$ in Q with $l_s = (str, l_{s'})$,
   - if $r = \mathcal{I}$, then S returns the Diffie-Hellman exponential g^κ to M, else S returns $\star$.
3. S stores entries of the form $(r, h, \kappa) \in \{0,1\}^* \times \mathbb{Z}_p \times \mathbb{Z}_p$ in table J. When M makes a query of the form (r, h) to the random oracle for H_1, answer it as follows:
   - If $C = g^h$, then S aborts M and is successful by outputting $\mathrm{DLog}_g(C) = h$.
   - Else if $(r, h, \kappa) \in J$ for some $\kappa \in \mathbb{Z}_p$, then S returns κ to M.
   - Else if there exists an entry $(s, s_{rand}, l_s, \mathsf{sk}_{s_{actor}}, \kappa)$ in table Q with $l_s = r$ and $\mathsf{sk}_{s_{actor}} = h$, then S returns κ to M and stores the entry (r, h, κ) in

[6] The value of $l_{s'}$ is the concatenation of the randomness of the current and the previous sessions of the same user.

table J.
- Else, S chooses $\kappa \in_R \mathbb{Z}_p$, returns it to M and stores the entry (r, h, κ) in table J.

4. $\mathsf{send}(\hat{P}, i, V)$ to send message V to session $s = (\hat{P}, i)$: If $s_{status} \neq \texttt{active}$, then S returns $\perp$. Else if $s_{role} = \mathcal{I}$, then S does the following. If $V \notin G$, then the status of session s is set to `rejected`. Else, the status of session s is set to `accepted`, and
   - If there exists an entry $(s_{peer}, s_{actor}, \mathcal{R}, s_{recv}, s_{sent}, \lambda)$ in table T, then S stores $(s_{actor}, s_{peer}, \mathcal{I}, s_{sent}, s_{recv}, \lambda)$ in table T.
   - Else if there exists an entry $(\sigma_1, \sigma_2, \sigma_3, s_{actor}, s_{peer}, \lambda)$ in table L, for some $\lambda \in \{0,1\}^k$, such that $\mathrm{DDH}(s_{recv}, s_{sent}, \sigma_3) = 1$, $\mathrm{DDH}(s_{sent}, \mathsf{pk}_{s_{peer}}, \sigma_2) = 1$ and $\mathrm{DDH}(s_{recv}, \mathsf{pk}_{s_{actor}}, \sigma_1) = 1$, then S stores $(s_{actor}, s_{peer}, \mathcal{I}, s_{sent}, s_{recv}, \lambda)$ in table T.
   - Else, S chooses $\mu \in_R \{0,1\}^k$ and stores the entry $(s_{actor}, s_{peer}, \mathcal{I}, s_{sent}, s_{recv}, \mu)$ in T.

   Else if $s_{role} = \mathcal{R}$, then S does the following. If $V \notin G$, then the status of session s is set to `rejected`. Else, S sets the status of session s to `accepted`, returns g^{κ} to M, where κ denotes the last element of the entry $(s, s_{rand}, l_s, \mathsf{sk}_{s_{actor}}, \kappa)$ in table Q, and proceeds in a similar way as in the previous case.
5. When M makes a query of the form $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j\right)$ to the random oracle for H_2, answer it as follows:
   - If $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j, \lambda\right) \in L$ for some $\lambda \in \{0,1\}^k$, then S returns λ to M.
   - Else if there exist entries $\left(\hat{Q}_i, \hat{Q}_j, \mathcal{I}, U, V, \lambda\right)$ or $\left(\hat{Q}_j, \hat{Q}_i, \mathcal{R}, V, U, \lambda\right)$ in table T, for some $\lambda \in \{0,1\}^k$ and $U, V \in G$, such that $\mathrm{DDH}(V, U, \sigma_3) = 1$, $\mathrm{DDH}(V, \mathsf{pk}_{\hat{Q}_i}, \sigma_1) = 1$ and $\mathrm{DDH}(U, \mathsf{pk}_{\hat{Q}_j}, \sigma_2) = 1$, then S returns λ to M and stores the entry $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j, \lambda\right)$ in table L.
   - Else, S chooses $\mu \in_R \{0,1\}^k$, returns it to M and stores the entry $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j, \mu\right)$ in L.
6. $\mathsf{randomness}(s)$: If $s_{status} = \perp$, then S returns $\perp$. Otherwise, S returns s_{rand}.
7. $\mathsf{session\text{-}key}(s)$: If $s_{status} \neq \texttt{accepted}$, then S returns $\perp$. Otherwise, S answers this query by lookup in table T.
8. $\mathsf{test\text{-}session}(s)$: If $s \neq s^*$, then S aborts; otherwise S answers the query in the appropriate way.
9. $\mathsf{corrupt}(\hat{P})$: If $\hat{P} \notin \mathcal{P}$, then S returns $\perp$. Else if $\hat{P} = \hat{C}$, then S aborts. Else, S returns $\mathsf{sk}_{\hat{P}}$.
10. M outputs a guess: S aborts.

Analysis of event $DL \wedge K$
Similar to the analysis of the related event $DL \wedge K$ in the proof of Proposition 7.

Event $T_O \wedge DL^c \wedge K$
Let s^* and s' denote the test session and the origin-session for the test session, respectively. We split event $Evt := T_O \wedge DL^c \wedge K$ into the following events $B_1, ..., B_3$ so that $Evt = B_1 \vee B_2 \vee B_3$:

1. B_1 : *Evt* occurs and $s^*_{peer} = s'_{actor}$.
2. B_2 : *Evt* occurs and $s^*_{peer} \neq s'_{actor}$ and M does not issue the queries randomness or cr-create to all sessions of s'_{actor} that were created prior to creation of the origin-session s' of s^*, including the origin-session itself, but may issue a corrupt(s^*_{peer}) query.
3. B_3 : *Evt* occurs and $s^*_{peer} \neq s'_{actor}$ and M does not issue a corrupt(s^*_{peer}) query, but may issue the queries randomness or cr-create to all session created prior to creation of the origin-session, including the origin-session s' itself.

Event B_1

Let the input to the GDH challenge be (X_0, Y_0). Suppose that event B_1 occurs with non-negligible probability. In this case S chooses long-term secret/public key pairs for all the honest parties and stores the associated long-term secret keys. Additionally S chooses two random values $m, n \in_R \{1, 2, ..., q_s\}$. The m'th activated session by adversary M will be called s^* and the n'th activated session will be called s'. Suppose further that $s^*_{actor} = \hat{A}, s^*_{peer} = \hat{B}$ and $s^*_{role} = \mathcal{I}$, w.l.o.g.. We now define S's responses to M's queries. S maintains tables Q, J, T and L, all of which are initially empty, as well as a variable ω initialized with 1.

1. create$(\hat{A}, \mathcal{I}, \hat{B})$ or cr-create$(\hat{A}, \mathcal{I}, str, \hat{B})$ to create session s^*: If create is issued, S chooses $s^*_{rand} \in_R \{0,1\}^k$. Else, S sets $s^*_{rand} \leftarrow str$. S (a) returns the message X_0, where (X_0, Y_0) is the GDH challenge, and (b) stores the entry $(s^*, s^*_{rand}, l_{s^*}, \mathsf{sk}_{\hat{A}}, *)$ in table Q, where $l_{s^*} = (s^*_{rand}, l_s)$ if there exists a previously created session s of user $s_{actor} = \hat{A}$ with an entry in table Q, and $l_{s^*} = s^*_{rand}$ if there no such session exists.
2. create$(\hat{B}, r, \hat{Q})$ or cr-create$(\hat{B}, r, str, \hat{Q})$ with $r \in \{\mathcal{I}, \mathcal{R}\}$ to create session s': If create is issued, S chooses $s'_{rand} \in_R \{0,1\}^k$. Else, S sets $s'_{rand} \leftarrow str$. S stores the entry $(s', s'_{rand}, l_{s'}, \mathsf{sk}_{\hat{B}}, *)$ in table Q, where $l_{s'} = (s'_{rand}, l_s)$ if there exists a previously created session s of user $s_{actor} = \hat{A}$ with an entry in table Q, and $l_{s'} = s'_{rand}$ if there no such session exists. If $r = \mathcal{I}$, then S returns message Y_0 to M, where (X_0, Y_0) is the GDH challenge. Else, $\star$ is returned.
3. send$(\hat{B}, i, Z)$ with $(\hat{B}, i) = s'$: If $s'_{status} \neq$ `active`, then S returns $\perp$. Else if $s'_{role} = \mathcal{R}$ and $Z \in G$, then S returns message Y_0 to M, where (X_0, Y_0) is the GDH challenge, sets the status of session s' to `accepted`, and proceeds as in the previous simulation for completing the session. Else, S proceeds as in the previous simulation.
4. send$(\hat{A}, i, Y_0)$ with $(\hat{A}, i) = s^*$: S proceeds as in the previous simulation for completing the session.
5. Other create, cr-create and send queries are answered as in the simulation relative to event $DL \wedge K$.
6. When M makes a query of the form $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j\right)$ to the random oracle for H_2, answer it as follows:
   - If $\left\{\hat{Q}_i, \hat{Q}_j\right\} = \left\{\hat{A}, \hat{B}\right\}$, $\sigma_1 = Y_0^a$, $\sigma_2 = X_0^b$ and $\mathrm{DDH}(X_0, Y_0, \sigma_3) = 1$, then S aborts M and is successful by outputting $\mathrm{CDH}(X_0, Y_0) = \sigma_3$.
   - Else if $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j, \lambda\right) \in L$ for some $\lambda \in \{0,1\}^k$, then S returns λ to M.

- Else if there exist entries $\left(\hat{Q}_i, \hat{Q}_j, \mathcal{I}, U, V, \lambda\right)$ or $\left(\hat{Q}_j, \hat{Q}_i, \mathcal{R}, V, U, \lambda\right)$ in table T, for some $\lambda \in \{0,1\}^k$ and $U, V \in G$, such that $\mathrm{DDH}(V, U, \sigma_3) = 1$, $\mathrm{DDH}(V, \mathsf{pk}_{\hat{Q}_i}, \sigma_1) = 1$ and $\mathrm{DDH}(U, \mathsf{pk}_{\hat{Q}_j}, \sigma_2) = 1$, then S returns λ to M and stores the entry $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j, \lambda\right)$ in table L.
- Else, S chooses $\mu \in_R \{0,1\}^k$, returns it to M and stores the entry $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j, \mu\right)$ in L.

7. $\mathsf{randomness}(s)$: If $s_{status} = \perp$, then S returns $\perp$. Otherwise, S returns s_{rand}.
8. $\mathsf{session\text{-}key}(s)$: If $s_{status} \neq \mathtt{accepted}$, then S returns $\perp$. Otherwise, S answers this query by lookup in table T.
9. $\mathsf{test\text{-}session}(s)$: If $s \neq s^*$ or if s' is not the origin-session for session s^*, then S aborts; otherwise S answers the query in the appropriate way.
10. $H_1(r, h)$: If $r = l_{s^*}$ and $h = \mathsf{sk}_{\hat{A}}$ or if $r = l_{s'}$ and $h = \mathsf{sk}_{\hat{B}}$, then S aborts. Otherwise S simulates a random oracle as in the simulation relative to event $DL \wedge K$.
11. $\mathsf{corrupt}(\hat{P})$: If $\hat{P} \notin \mathcal{P}$, then S returns $\perp$. Else, S returns $\mathsf{sk}_{\hat{P}}$.
12. M outputs a guess: S aborts.

Analysis of event B_1
S's simulation of M's environment is perfect except with negligible probability. The probability that M selects s^* as the test-session and s' as the origin-session for the test-session is $\frac{1}{(q_s)^2}$. Assuming that this is indeed the case, S does not abort in Step 9. Under event DL^c, the adversary first issues a $\mathsf{corrupt}(\hat{P})$ query to party $\hat{P}$ before making an H_1 query that involves the long-term secret key of party $\hat{P}$. Freshness of the test session guarantees that the adversary can reveal/determine either l_{s^*} or $\mathsf{sk}_{\hat{A}}$, but not both. Similar for $l_{s'}$ and $\mathsf{sk}_{\hat{B}}$. Hence S does not abort in Step 10. Under event K, except with negligible probability of guessing $CDH(X_0, Y_0)$, S is successful as described in the first case of Step 6 and does not abort as in Step 12. Hence, if event B_1 occurs, then the success probability of S is given by $P(S) \geq \frac{1}{(q_s)^2} P(B_1)$.

Event B_2
Let the input to the GDH challenge be (X_0, Y_0). Suppose that event B_2 occurs with non-negligible probability. The simulation of S proceeds in the same way as for event B_1 with the following changes. S additionally keeps a history H of M's queries.

- $\mathsf{randomness}(s)$: If $s_{status} = \perp$, then S returns $\perp$. Else if $s = s'$ and there were queries ($\mathsf{randomness}$ or $\mathsf{cr\text{-}create}$) to all previous sessions of the same user s'_{actor}, then S aborts. Else, S returns s_{rand}.
- $H_1(r, h)$: If $r = l_{s^*}$ and $h = \mathsf{sk}_{\hat{A}}$, then S aborts. Otherwise S simulates a random oracle as in the previous simulation.

Analysis of event B_2
Similar to the analyses of the related event B_2 in the proof of Proposition 7 and event B_1.

Event B_3
Let the input to the GDH challenge be (X_0, B). Suppose that event B_3 occurs with non-negligible probability. In this case, S chooses one user $\hat{B} \in \mathcal{P}$ at random from

the set $\mathcal{P}$ and sets its long-term public key to B. S chooses long-term secret/public key pairs for the remaining parties in $\mathcal{P}$ and stores the associated long-term secret keys. Additionally S chooses two random values $m, n \in_R \{1, 2, ..., q_s\}$. We denote the m'th activated session by adversary M by s^* and the n'th activated session by s'. Suppose further that $s^*_{actor} = \hat{A}, s^*_{peer} = \hat{B}$ and $s^*_{role} = \mathcal{I}$, w.l.o.g.. Algorithm S maintains tables Q, J, T and L, all of which are initially empty. S also maintains a variable ω initialized with 1.

1. $\mathsf{create}(\hat{A}, \mathcal{I}, \hat{B})$ or $\mathsf{cr\text{-}create}(\hat{A}, \mathcal{I}, str, \hat{B})$ to create session s^*: If create is issued, S chooses $s^*_{rand} \in_R \{0,1\}^k$. Else, S sets $s^*_{rand} \leftarrow str$. S (a) returns the message X_0, where (X_0, B) is the GDH challenge, and (b) stores the entry $(s^*, s^*_{rand}, l_{s^*}, \mathsf{sk}_{\hat{A}}, *)$ in table Q, where $l_{s^*} = (s^*_{rand}, l_s)$ if there exists a previously created session s of user $s_{actor} = \hat{A}$ with an entry in table Q, and $l_{s^*} = s^*_{rand}$ if there no such session exists.
2. $\mathsf{create}\left(\hat{P}, r, \hat{Q}\right)$ to create session s: S checks whether $\hat{P} \in \mathcal{P}$, $\hat{Q} \in \mathcal{P}$, and $r \in \{\mathcal{I}, \mathcal{R}\}$. If one of the checks fails, then S returns $\perp$. Else, S initializes the session variables according to the protocol specification, and stores an entry of the form $(s, s_{rand}, l_s, \mathsf{sk}_{s_{actor}}, \kappa) \in (\mathcal{P} \times \mathbb{N}) \times \{0,1\}^k \times \{0,1\}^* \times (\mathbb{Z}_p \cup \{*\}) \times \mathbb{Z}_p$ in table Q as follows:
   - S chooses $s_{rand} \in_R \{0,1\}^k$ (i. e. the randomness of session s),
   - S chooses $\kappa \in_R \mathbb{Z}_p$,
   - if there is no entry $(s, s_{rand}, l_s, \mathsf{sk}_{s_{actor}}, \kappa)$ in table Q such that $s_{actor} = \hat{P}$, then S sets the value of l_s to s_{rand}, else S sets the value of l_s to $(s_{rand}, l_{s'})$, where s' is the previous session with $s'_{actor} = s_{actor}$ for which an entry in table Q has been made.
   - if $s_{actor} \neq \hat{B}$, then S stores the entry $(s, s_{rand}, s_{data}, \mathsf{sk}_{s_{actor}}, \kappa)$ in Q, else S stores the entry $(s, s_{rand}, s_{data}, *, \kappa)$ in Q, and
   - if $r = \mathcal{I}$, then S returns the Diffie-Hellman exponential g^κ to M, else S returns $\star$.
3. $\mathsf{cr\text{-}create}\left(\hat{P}, r, str, \hat{Q}\right)$ to create session s: S checks whether $\hat{P} \in \mathcal{P}$, $\hat{Q} \in \mathcal{P}$, and $r \in \{\mathcal{I}, \mathcal{R}\}$. If one of the checks fails, then S returns $\perp$. Else, S initializes the session variables according to the protocol specification, and stores an entry of the form $(s, s_{rand}, l_s, \mathsf{sk}_{s_{actor}}, \kappa) \in (\mathcal{P} \times \mathbb{N}) \times \{0,1\}^k \times \{0,1\}^* \times (\mathbb{Z}_p \cup \{*\}) \times \mathbb{Z}_p$ in table Q as follows:
   - if there is an entry (r_i, h_i, κ_i) in table J such that $r_i = (str, l_{s'})$, and $h_i = \mathsf{sk}_{\hat{P}}$, where s' is the previous session with $s'_{actor} = s_{actor}$ for which an entry in table Q has been made, then S sets $\omega \leftarrow \kappa_i$, else S chooses $\kappa \in_R \mathbb{Z}_p$ and sets $\omega \leftarrow \kappa$.
   - if $s_{actor} \neq \hat{B}$, then S stores the entry $(s, s_{rand}, r_i, \mathsf{sk}_{s_{actor}}, \omega)$ in Q, else S stores the entry $(s, s_{rand}, l_s, *, \omega)$ in Q with $l_s = (str, l_{s'})$,
   - if $r = \mathcal{I}$, then S returns the Diffie-Hellman exponential g^κ to M, else S returns $\star$.
4. S stores entries of the form $(r, h, \kappa) \in \{0,1\}^* \times \mathbb{Z}_p \times \mathbb{Z}_p$ in table J. When M makes a query of the form (r, h) to the random oracle for H_1, answer it as follows:

- If $r = l_{s^*}$ and $h = \mathsf{sk}_{\hat{A}}$, then S aborts.
- Else if $(r, h, \kappa) \in J$ for some $\kappa \in \mathbb{Z}_p$, then S returns κ to M.
- Else if there exists an entry $(s, s_{rand}, l_s, \mathsf{sk}_{s_{actor}}, \kappa)$ in table Q with $l_s = r$ and $\mathsf{sk}_{s_{actor}} = h$, then S returns κ to M and stores the entry (r, h, κ) in table J.
- Else, S chooses $\kappa \in_R \mathbb{Z}_p$, returns it to M and stores the entry (r, h, κ) in table J.

5. $\mathsf{send}(\hat{P}, i, V)$ to send message V to session $s = (\hat{P}, i)$: If $s_{status} \neq \texttt{active}$, then S returns $\perp$. Else if $s_{role} = \mathcal{I}$, then S does the following. If $V \notin G$, then the status of session s is set to `rejected`. Else, the status of session s is set to `accepted`, and
   - If there exists an entry $(s_{peer}, s_{actor}, \mathcal{R}, s_{recv}, s_{sent}, \lambda)$ in table T, then S stores $(s_{actor}, s_{peer}, \mathcal{I}, s_{sent}, s_{recv}, \lambda)$ in table T.
   - Else if there exists an entry $(\sigma_1, \sigma_2, \sigma_3, s_{actor}, s_{peer}, \lambda)$ in table L, for some $\lambda \in \{0,1\}^k$, such that $\mathrm{DDH}(s_{recv}, s_{sent}, \sigma_3) = 1$, $\mathrm{DDH}(s_{sent}, \mathsf{pk}_{s_{peer}}, \sigma_2) = 1$ and $\mathrm{DDH}(s_{recv}, \mathsf{pk}_{s_{actor}}, \sigma_1) = 1$, then S stores $(s_{actor}, s_{peer}, \mathcal{I}, s_{sent}, s_{recv}, \lambda)$ in table T.
   - Else, S chooses $\mu \in_R \{0,1\}^k$ and stores the entry $(s_{actor}, s_{peer}, \mathcal{I}, s_{sent}, s_{recv}, \mu)$ in T.

   Else if $s_{role} = \mathcal{R}$, then S does the following. If $V \notin G$, then the status of session s is set to `rejected`. Else, S sets the status of session s to `accepted`, returns g^κ to M, where κ denotes the last element of the entry $(s, s_{rand}, l_s, \mathsf{sk}_{s_{actor}}, \kappa)$ in table Q, and proceeds in a similar way as in the previous case.
6. When M makes a query of the form $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j\right)$ to the random oracle for H_2, answer it as follows:
   - If $\left\{\hat{Q}_i, \hat{Q}_j\right\} = \left\{\hat{A}, \hat{B}\right\}$, $\sigma_1 = A^\kappa$, $\mathrm{DDH}(X_0, B, \sigma_2) = 1$, and $\sigma_3 = X_0^\kappa$, where κ denotes the last element of the entry $(s', s'_{rand}, l_{s'}, \mathsf{sk}_{s'_{actor}}, \kappa)$ in table Q, then S aborts M and is successful by outputting $\mathrm{CDH}(X_0, B) = \sigma_2$.
   - Else if $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j, \lambda\right) \in L$ for some $\lambda \in \{0,1\}^k$, then S returns λ to M.
   - If $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j, \lambda\right) \in L$ for some $\lambda \in \{0,1\}^k$, then S returns λ to M.
   - Else if there exist entries $\left(\hat{Q}_i, \hat{Q}_j, \mathcal{I}, U, V, \lambda\right)$ or $\left(\hat{Q}_j, \hat{Q}_i, \mathcal{R}, V, U, \lambda\right)$ in table T, for some $\lambda \in \{0,1\}^k$ and $U, V \in G$, such that $\mathrm{DDH}(V, U, \sigma_3) = 1$, $\mathrm{DDH}(V, \mathsf{pk}_{\hat{Q}_i}, \sigma_1) = 1$ and $\mathrm{DDH}(U, \mathsf{pk}_{\hat{Q}_j}, \sigma_2) = 1$, then S returns λ to M and stores the entry $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j, \lambda\right)$ in table L.
   - Else, S chooses $\mu \in_R \{0,1\}^k$, returns it to M and stores the entry $\left(\sigma_1, \sigma_2, \sigma_3, \hat{Q}_i, \hat{Q}_j, \mu\right)$ in L.
7. $\mathsf{randomness}(s)$: If $s_{status} = \perp$, then S returns $\perp$. Otherwise, S returns s_{rand}.
8. $\mathsf{session\text{-}key}(s)$: If $s_{status} \neq \texttt{accepted}$, then S returns $\perp$. Otherwise, S answers this query by lookup in table T.
9. $\mathsf{test\text{-}session}(s)$: If $s \neq s^*$ or if s' is not the origin-session for session s^*, then S aborts; otherwise S answers the query in the appropriate way.

10. $\mathsf{corrupt}(\hat{P})$: If $\hat{P} \notin \mathcal{P}$, then S returns $\perp$. Else if $\hat{P} = \hat{B}$, then S aborts. Else, S returns $\mathsf{sk}_{\hat{P}}$.
11. M outputs a guess: S aborts.

Analysis of event B_3
Similar to the analysis of the related event B_3 in the proof of Proposition 7.

Event $(T_O)^c \wedge DL^c \wedge K$
The simulation and analysis are very similar to the simulation and analysis related to event B_3. □

Bibliography

[1] P. Abeni, L. Bello, and M. Bertacchini. Exploiting DSA-1571: How to break PFS in SSL with EDH. `http://www.lucianobello.com.ar/exploiting_DSA-1571/index.html` (Accessed 05/11/2013).

[2] C. Adams, S. Farrell, T. Kause, and T. Mononen. Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP). RFC 4210 (Proposed Standard), September 2005. Updated by RFC 6712.

[3] J.H. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. In L.R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 83–107. Springer Berlin Heidelberg, 2002.

[4] M. Arapinis, E. Ritter, and M. D. Ryan. Statverif: Verification of Stateful Processes. In *Proceedings of the 2011 IEEE 24th Computer Security Foundations Symposium*, CSF '11. IEEE Computer Society, Washington, DC, USA, 2011.

[5] M. Artin. *Algebra*. Prentice–Hall, second edition, 2010.

[6] M. Barbosa and P. Farshim. Security analysis of standard authentication and key agreement protocols utilising timestamps. In B. Preneel, editor, *Progress in Cryptology AFRICACRYPT 2009*, volume 5580 of *LNCS*, pages 235–253. Springer Berlin Heidelberg, 2009.

[7] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid. Recommendation for key management — Part 1: General. NIST Special Publication, March 2007. `http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57-Part1-revised2_Mar08-2007.pdf`.

[8] E. Barker, D. Johnson, and M. Smid. Recommendation for pair–wise key establishment schemes using discrete logarithm cryptography (revised). NIST Special Publication 800-56A, March 2007. `http://csrc.nist.gov/publications/nistpubs/800-56A/SP800-56A_Revision1_Mar08-2007.pdf`.

[9] E. Barker, D. Johnson, and M. Smid. Recommendation for pair-wise key establishment schemes using discrete logarithm cryptography. NIST Special Publication, March 2007. `http://csrc.nist.gov/publications/nistpubs/800-56A/SP800-56A_Revision1_Mar08-2007.pdf`.

[10] D. Basin and C. Cremers. Degrees of security: Protocol guarantees in the face of compromising adversaries. In *Computer Science Logic, 24th International Workshop, CSL 2010, 19th Annual Conference of the EACSL*, volume 6247 of *LNCS*, pages 1–18. Springer, 2010.

[11] D. Basin and C. Cremers. Modeling and analyzing security in the presence of compromising adversaries. In *Computer Security – ESORICS 2010*, volume 6345 of *LNCS*, pages 340–356. Springer, 2010.

[12] M. Bellare, Z. Brakerski, M. Naor, T. Ristenpart, G. Segev, H. Shacham, and S. Yilek. Hedged Public-Key Encryption: How to protect against bad randomness. In M. Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 232–249. Springer Berlin Heidelberg, 2009.

[13] M. Bellare, R. Canetti, and H. Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols (extended abstract). In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, STOC '98, pages 419–428, New York, NY, USA, 1998. ACM.

[14] M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. In A. Juels, R. N. Wright, and S. Vimercati, editors, *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, pages 390–399. ACM, 2006.

[15] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In *19th International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'00, pages 139–155. Springer, 2000.

[16] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, CCS '93, pages 62–73, New York, NY, USA, 1993. ACM.

[17] M. Bellare and P. Rogaway. Entity authentication and key distribution. In *13th annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '93, pages 232–249. Springer New York, NY, USA, 1994.

[18] M. Bellare and P. Rogaway. Provably secure session key distribution: the three party case. In *27th annual ACM symposium on Theory of computing*, STOC '95, pages 57–66. ACM New York, NY, USA, 1995.

[19] S. Blake-Wilson, D. Johnson, and A. Menezes. Key agreement protocols and their security analysis. In M. Darnell, editor, *IMA Int. Conf.*, volume 1355 of *Lecture Notes in Computer Science*, pages 30–45. Springer, 1997.

[20] S. Blake-Wilson, D. Johnson, and A. Menezes. Key agreement protocols and their security analysis. In Michael Darnell, editor, *Crytography and Coding*, volume 1355 of *Lecture Notes in Computer Science*, pages 30–45. Springer Berlin Heidelberg, 1997.

[21] S. Blake-Wilson and A. Menezes. Entity authentication and authenticated key transport protocols employing asymmetric techniques. In B. Christianson,

B. Crispo, M. Lomas, and M. Roe, editors, *Security Protocols*, volume 1361 of *Lecture Notes in Computer Science*, pages 137–158. Springer Berlin Heidelberg, 1998.

[22] S. Blake-Wilson and A. Menezes. Unknown key-share attacks on the Station-to-Station (STS) protocol. In Imai H. and Zheng Y., editors, *PKC '99 Proceedings of the Second International Workshop on Practice and Theory in Public Key Cryptography*, volume 1560 of *LNCS*, pages 154–170. Springer, 1999.

[23] Simon Blake-Wilson and Alfred Menezes. Authenticated Diffie-Hellman key agreement protocols. In Stafford E. Tavares and Henk Meijer, editors, *Selected Areas in Cryptography*, volume 1556 of *LNCS*, pages 339–361. Springer, 1998.

[24] A. Boldyreva, M. Fischlin, A. Palacio, and B. Warinschi. A closer look at pki: Security and efficiency. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *Public Key Cryptography–PKC 2007*, volume 4450 of *Lecture Notes in Computer Science*, pages 458–475. Springer Berlin Heidelberg, 2007.

[25] D. Boneh. The decision diffie-hellman problem. In J.P. Buhler, editor, *Algorithmic Number Theory*, volume 1423 of *Lecture Notes in Computer Science*, pages 48–63. Springer Berlin Heidelberg, 1998.

[26] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil Pairing. In *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, ASIACRYPT '01, pages 514–532, London, UK, 2001. Springer–Verlag.

[27] D. Boneh, E. Shen, and B. Waters. Strongly unforgeable signatures based on computational Diffie-Hellman. In M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, editors, *PKC'06*, volume 3958 of *LNCS*, pages 229–240. Springer, 2006.

[28] N. Borisov, I. Goldberg, and E. Brewer. Off-the-record communication, or, why not to use PGP. In *Proceedings of the 2004 ACM workshop on Privacy in the electronic society*, pages 77–84. ACM Press, 2004.

[29] J.W. Bos, J.A. Halderman, N. Heninger, J. Moore, M. Naehrig, and E. Wustrow. Elliptic curve cryptography in practice. Cryptology ePrint Archive, Report 2013/734, 2013. `http://eprint.iacr.org/`.

[30] C. Boyd, Y. Cliff, J.M. González Nieto, and K.G. Paterson. One-round key exchange in the standard model. *Int. J. Applied Cryptography*, 1:181–199, 2009. Inderscience Publishers.

[31] C. Boyd and J. González Nieto. On Forward Secrecy in One-Round Key Exchange. In *13th IMA International Conference, IMACC 2011*, volume 7089 of *LNCS*, pages 451–468. Springer, 2011.

[32] E. Bresson, M. Manulis, and J. Schwenk. On security models and compilers for group key exchange protocols. Cryptology ePrint Archive, Report 2006/385, 2006. `http://eprint.iacr.org/`.

[33] C. Brzuska, M. Fischlin, B. Warinschi, and S.C. Williams. Composability of bellare-rogaway key exchange protocols. In *Proceedings of the 18th ACM conference on Computer and communications security*, CCS '11, pages 51–62, New York, NY, USA, 2011. ACM.

[34] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In B. Pfitzmann, editor, *EUROCRYPT'01*, volume 2045 of *LNCS*, pages 453–474. Springer London, UK, 2001. full version on eprint.

[35] S. Chatterjee, A. Menezes, and B. Ustaoglu. Combined security analysis of the one- and three-pass unified model key agreement protocols. In G. Gong and K.C. Gupta, editors, *Progress in Cryptology - INDOCRYPT 2010*, volume 6498 of *Lecture Notes in Computer Science*, pages 49–68. Springer Berlin Heidelberg, 2010.

[36] Q. Cheng, C. Ma, and X. Hu. A new strongly secure authenticated key exchange protocol. In J. H. Park, H-H. Chen, M. Atiquzzaman, C. Lee, T-H. Kim, and S-S. Yeo, editors, *ISA '09*, volume 5576 of *LNCS*, pages 135–144. Springer, 2009.

[37] K-K. R. Choo, C. Boyd, and Y. Hitchcock. Examining indistinguishability-based proof models for key establishment protocols. In *Proceedings of the 11th international conference on Theory and Application of Cryptology and Information Security*, ASIACRYPT'05, pages 585–604, Berlin, Heidelberg, 2005. Springer-Verlag.

[38] S. S. M. Chow and K-K. R. Choo. Strongly-secure identity-based key agreement and anonymous extension. In J. A. Garay, A. K. Lenstra, M. Mambo, and R. Peralta, editors, *Information Security, ISC'07*, volume 4779 of *LNCS*, pages 203–220. Springer, 2007.

[39] C. Cremers. The Scyther Tool: Verification, falsification, and analysis of security protocols. In *Proc. CAV*, volume 5123 of *LNCS*, pages 414–418. Springer, 2008.

[40] C. Cremers. Session-StateReveal is stronger than eCK's EphemeralKeyReveal: Using automatic analysis to attack the NAXOS protocol. *International Journal of Applied Cryptography (IJACT)*, 2:83–99, 2010. Inderscience Publishers.

[41] C. Cremers. Examining indistinguishability-based security models for key exchange protocols: the case of CK, CK-HMQV, and eCK. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '11, pages 80–91, New York, NY, USA, 2011. ACM.

[42] C. Cremers and M. Feltz. One-round strongly secure key exchange with perfect forward secrecy and deniability. Cryptology ePrint Archive, Report 2011/300, 2011. `http://eprint.iacr.org/`.

[43] O. Dagdelen and M. Fischlin. Security analysis of the extended access control protocol for machine readable travel documents. In *Proceedings of the 13th*

international conference on Information security, ISC'10, pages 54–68, Berlin, Heidelberg, 2011. Springer-Verlag.

[44] A.W. Dent. A note on game-hopping proofs. Cryptology ePrint Archive, Report 2006/260, 2006. `http://eprint.iacr.org/2006/260`.

[45] W. Diffie and M.E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. IEEE Press.

[46] FOX IT. Black Tulip: Report of the investigation into the DigiNotar Certificate Authority breach, 2012. `http://www.rijksoverheid.nl/bestanden/documenten-en-publicaties/rapporten/2012/08/13/black-tulip-update/black-tulip-update.pdf` (Accessed 12/03/2013).

[47] International Organization for Standardization, Genève, Switzerland. ISO/IEC IS 9798–3, Information technology–Security techniques–Entity authentication mechanisms–part 3: Entity authentication using asymmetric techniques, 1993.

[48] International Organization for Standardization, Genève, Switzerland. ISO/IEC 11770–2, Information technology–Security techniques–Key management–part 2: Mechanisms using symmetric techniques, 1996.

[49] R. Gennaro, S. Halevi, and T. Rabin. Secure hash-and-sign signatures without the random oracle. In J. Stern, editor, *Advances in Cryptology – EUROCRYPT 1999*, volume 1592 of *Lecture Notes in Computer Science*, pages 123–139. Springer Berlin Heidelberg, 1999.

[50] R. Gennaro, H. Krawczyk, and T. Rabin. Okamoto-Tanaka revisited: fully authenticated Diffie-Hellman with minimal overhead. In J. Zhou and M. Yung, editors, *ACNS'10*, pages 309–328. Springer, 2010.

[51] I. Goldberg, D. Stebila, and B. Ustaoglu. Anonymity and one-way authentication in key exchange protocols. *Designs, Codes and Cryptography*, 67(2):245–269, 2013. Springer US.

[52] O. Goldreich. On Post-Modern Cryptography. Cryptology ePrint Archive, Report 2006/461, 2006. `http://eprint.iacr.org/`.

[53] S. Goldwasser and S. Micali. Probabilistic encyption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984. Academic Press.

[54] S. Goldwasser, S. Micali, and R.L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, April 1988. Society for Industrial and Applied Mathematics.

[55] F. Hao. On robust key agreement based on public key authentication. In *Financial Cryptography*, volume 6052 of *LNCS*, pages 383–390. Springer, 2010.

[56] I.R. Jeong, J. Katz, and D.H. Lee. One-round Protocols for Two-Party Authenticated Key Exchange, 2008. `http://www.cs.umd.edu/~jkatz/papers/1round_AKE.pdf`.

[57] A. Joux and K. Nguyen. Separating Decision Diffie-Hellman from Computational Diffie-Hellman in cryptographic groups. *Journal of Cryptology*, 16(4):239–247, 2003. Springer-Verlag.

[58] M. Just and S. Vaudenay. Authenticated multi-party key agreement. In Kwangjo Kim and Tsutomu Matsumoto, editors, *Advances in Cryptology–ASIACRYPT '96*, volume 1163 of *Lecture Notes in Computer Science*, pages 36–49. Springer Berlin Heidelberg, 1996.

[59] B.S.JR. Kaliski. An unknown key-share attack on the MQV key agreement protocol. In *ACM Transactions on Information and System Security (TISSEC)*, volume 4, pages 275–288. ACM New York, NY, USA, August 2001.

[60] J. Katz and Y. Lindell. *Introduction to Modern Cryptography*. Chapman Hall/CRC, 2008.

[61] J. Katz and M. Yung. Scalable protocols for authenticated group key exchange. In D. Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729, pages 110–125. Springer, 2003.

[62] N. Koblitz and A.J. Menezes. Another look at "Provable Security". *Journal of Cryptology*, 20(1):3–37, 2007. Springer-Verlag.

[63] H. Krawczyk. SIGMA: The 'SIGn–and–MAc' Approach to authenticated diffie-hellman and its use in the IKE protocols. In D. Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 400–425. Springer Berlin Heidelberg, 2003.

[64] H. Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In V. Shoup, editor, *Advances in Cryptology - CRYPTO 2005*, volume 3621 of *LNCS*, pages 546–566. Springer, 2005.

[65] H. Krawczyk. HMQV: A High-Performance Secure Diffie-Hellman Protocol. Cryptology ePrint Archive, Report 2005/176, 2005. `http://eprint.iacr.org/`.

[66] C. Kudla and K. G. Paterson. Modular security proofs for key agreement protocols. In *Advances in Cryptology - ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 549–565. Springer Berlin Heidelberg, 2005.

[67] B.A. LaMacchia, K. Lauter, and A. Mityagin. Stronger security of authenticated key exchange. Cryptology ePrint Archive, Report 2006/073, 2006. `http://eprint.iacr.org/`.

[68] B.A. LaMacchia, K. Lauter, and A. Mityagin. Stronger security of authenticated key exchange. In W. Susilo, J. K. Liu, and Y. Mu, editors, *ProvSec'07*, volume 4784 of *LNCS*, pages 1–16. Springer, 2007.

[69] K. Lauter and A. Mityagin. Security analysis of KEA authenticated key exchange protocol. In *Public Key Cryptography – PKC 2006, 9th International Conference on Theory and Practice in Public-Key Cryptography*, volume 3958/2006 of *LNCS*, pages 378–394. Springer, 2006.

[70] L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone. An efficient protocol for authenticated key agreement. *Designs, Codes and Cryptography*, 28(2):119–134, 2003. Kluwer Academic Publishers.

[71] J. Lee and C.S. Park. An efficient authenticated key exchange protocol with a tight security reduction. Cryptology ePrint Archive, Report 2008/345, 2008. `http://eprint.iacr.org/`.

[72] J. Lee and J.H. Park. Authenticated key exchange secure under the computational Diffie-Hellman assumption. Cryptology ePrint Archive, Report 2008/344, 2008. `http://eprint.iacr.org/`.

[73] C. Lim and P. Lee. A key recovery attack on discrete log-based schemes using a prime order subgroup. In Jr. Kaliski, BurtonS., editor, *Advances in Cryptology–CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 249–263. Springer Berlin Heidelberg, 1997.

[74] C. H. Lim and P. J. Lee. Authenticated session keys and their server-aided computation. Technical report, Pohang University of Science and Technology, Korea, 2006.

[75] W. Mao and K. G. Paterson. On the plausible deniability feature of internet protocols. Preprint, `www.isg.rhul.ac.uk/~kp/IKE.ps` (Accessed 17/02/2014), 2002.

[76] R. Marvin. Google admits an Android crypto PRNG flaw led to Bitcoin heist (August 2013). `http://sdt.bz/64008` (Accessed 01/10/2013).

[77] U. Maurer. Abstract models of computation in cryptography. In N. Smart, editor, *Cryptography and Coding 2005*, volume 3796 of *LNCS*, pages 1–12. Springer, December 2005.

[78] K. S. McCurley. A key distribution system equivalent to factoring. *Journal of Cryptology*, 1(2):95–105, 1988. Springer-Verlag.

[79] S. Meier, B. Schmidt, C. Cremers, and D. Basin. The TAMARIN Prover for the Symbolic Analysis of Security Protocols. In N. Sharygina and H. Veith, editors, *Computer Aided Verification*, volume 8044 of *Lecture Notes in Computer Science*, pages 696–701. Springer Berlin Heidelberg, 2013.

[80] A. Menezes. Another look at HMQV. Cryptology ePrint Archive, Report 2005/205, 2005. `http://eprint.iacr.org/`.

[81] A. Menezes and B. Ustaoglu. On the Importance of Public-Key Validation in the MQV and HMQV Key Agreement Protocols. In R. Barua and T. Lange, editors, *INDOCRYPT 2006*, volume 4329 of *LNCS*, pages 133–147. Springer, 2006.

[82] A. Menezes and B. Ustaoglu. Comparing the pre- and post-specified peer models for key agreement. In *Proceedings of the 13th Australasian conference*

on Information Security and Privacy, ACISP '08, pages 53–68. Springer-Verlag Berlin, Heidelberg, 2008.

[83] A. Menezes and B. Ustaoglu. Security arguments for the UM key agreement protocol in the NIST SP 800-56A standard. In M. Abe and V. Gligor, editors, *ASIACCS*, pages 261–270. ACM, 2008.

[84] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, FL, USA, 1996.

[85] A. Menezes, S. Vanstone, and T. Okamoto. Reducing elliptic curve logarithms to logarithms in a finite field. In *Proceedings of the Twenty-third Annual ACM Symposium on Theory of Computing*, STOC '91, pages 80–89, New York, NY, USA, 1991. ACM.

[86] M. Mueller. Debian OpenSSL predictable PRNG Bruteforce SSH exploit, 2008. `http://www.exploit-db.com/exploits/5622/` (Accessed 23/05/2013).

[87] T. Okamoto and D. Pointcheval. The gap-problems: a new class of problems for the security of cryptographic schemes. In K. Kim, editor, *PKC'2001*, volume 1992 of *LNCS*, pages 104–118. Springer, 2001.

[88] N. Perlroth, J. Larson, and S. Scott. NSA able to foil basic safeguards of privacy on web (September 2013). `http://www.nytimes.com/2013/09/06/us/nsa-foils-much-internet-encryption.html?pagewanted=1&_r=0` (Accessed 01/10/2013).

[89] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000. Springer-Verlag.

[90] M. Di Raimondo, R. Gennaro, and H. Krawczyk. Deniable authentication and key exchange. Cryptology ePrint Archive, Report 2006/280, 2006. `http://eprint.iacr.org/`.

[91] T. Ristenpart and S. Yilek. The power of proofs-of-possession: Securing multi-party signatures against rogue-key attacks. In *Proceedings of the 26th Annual International Conference on Advances in Cryptology*, EUROCRYPT '07, pages 228–245, Berlin, Heidelberg, 2007. Springer-Verlag.

[92] T. Ristenpart and S. Yilek. When good randomness goes bad: Virtual machine reset vulnerabilities and hedging deployed cryptography. In *Proceedings of the Network and Distributed System Security Symposium*, NDSS'10. The Internet Society, 2010.

[93] J. Schaad. Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF). RFC 4211 (Proposed Standard), September 2005.

[94] B. Schmidt, S. Meier, C. Cremers, and D. Basin. Automated analysis of Diffie-Hellman Protocols and Advanced Security Properties. In *Proceedings of the 25th IEEE Computer Security Foundations Symposium (CSF)*, pages 78–94. IEEE Computer Society, 2012.

[95] B. Schneier. NSA surveillance: A guide to staying secure (September 2013). http://www.theguardian.com/world/2013/sep/05/nsa-how-to-remain-secure-surveillance (Accessed 27/09/2013).

[96] J. Schwenk. Modelling time, or a step towards reduction-based security proofs for OTP and Kerberos. Cryptology ePrint Archive, Report 2013/604, 2013. http://eprint.iacr.org/.

[97] Z. Shmuely. Composite Diffie-Hellman public-key generating systems are hard to break. Technical Report No. 356, Computer Science Department, Technion-Israel Institute of Technology, 1985.

[98] V. Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2006. http://eprint.iacr.org/.

[99] V. Shoup. On formal methods for secure key exchange (version 4), November 1999. revision of IBM Research Report RZ 3120 (April 1999) http://www.shoup.net/papers/skey.pdf.

[100] D. Shumow and N. Ferguson. On the Possibility of a Back Door in the NIST SP800-90 Dual Ec Prng. http://rump2007.cr.yp.to/15-shumow.pdf (Accessed 15/10/2013).

[101] N.P. Smart. The discrete logarithm problem on elliptic curves of trace one. *Journal of Cryptology*, 12(3):193–196, 1999. Springer-Verlag.

[102] P. Turner, W. Polk, and E. Barker. ITL Bulletin for July 2012: Preparing for and responding to certification authority compromise and fraudulent certificate issuance, 2012. http://csrc.nist.gov/publications/nistbul/july-2012_itl-bulletin.pdf (Accessed 12/03/2013).

[103] S. Turner. The application/pkcs10 Media Type. RFC 5967 (Informational), August 2010.

[104] B. Ustaoglu. Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. Cryptology ePrint Archive, Report 2007/123, 2007. Version June 22, 2009.

[105] B. Ustaoglu. Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. *Designs, Codes and Cryptography*, 46(3):329–342, 2008. Kluwer Academic Publishers.

[106] B. Ustaoglu. Comparing SessionStateReveal and EphemeralKeyReveal for Diffie-Hellman Protocols. In J. Pieprzyk and F. Zhang, editors, *ProvSec*, volume 5848 of *Lecture Notes in Computer Science*, pages 183–197. Springer, 2009.

[107] T. van Deursen, S. Mauw, S. Radomirovic, and P. Vullers. Secure Ownership and Ownership Transfer in RFID systems. In *Computer Security – ESORICS 2009*, volume 5789 of *LNCS*, pages 637–654. Springer Berlin Heidelberg, 2009.

[108] G. Yang, S. Duan, D.S. Wong, C.H. Tan, and H. Wang. Authenticated key exchange under bad randomness. In *Proceedings of the 15th international conference on Financial Cryptography and Data Security*, FC'11, pages 113–126, Berlin, Heidelberg, 2012. Springer-Verlag.

[109] S. Yilek, E. Rescorla, H. Shacham, B. Enright, and S. Savage. When private keys are public: results from the 2008 Debian OpenSSL vulnerability. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, IMC '09, pages 15–27. ACM New York, NY, USA, 2009.

[110] K. Zetter. How a Crypto 'Backdoor' Pitted the Tech World Against the NSA (September 2013). `http://www.wired.com/threatlevel/2013/09/nsa-backdoor/all/` (Accessed 01/10/2013).

Index