Beyond eCK: Perfect Forward Secrecy under Actor Compromise and Ephemeral-Key Reveal⋆

Cas Cremers and Michèle Feltz

Institute of Information Security, ETH Zurich, Switzerland

Abstract. We show that it is possible to achieve perfect forward secrecy in two-message or one-round key exchange (KE) protocols even in the presence of very strong active adversaries that can reveal random values of sessions and compromise long-term secret keys of parties. We provide two new game-based security models for KE protocols with increasing security guarantees, namely, eCK^w and eCK-PFS. The eCK^w model is a slightly stronger variant of the extended Canetti-Krawczyk (eCK) security model. The eCK-PFS model captures perfect forward secrecy in the presence of eCK^w adversaries. We propose a security-strengthening transformation (i. e., a *compiler*) from eCK^w to eCK-PFS that can be applied to protocols that only achieve security in a weaker model than eCK^w, which we call eCK$^{\mathrm{passive}}$. We show that, given a two-message Diffie-Hellman type protocol secure in eCK$^{\mathrm{passive}}$, our transformation yields a two-message protocol that is secure in eCK-PFS.
We demonstrate how our transformation can be applied to concrete KE protocols. In particular, our methodology allows us to prove the security of the first known one-round protocol that achieves perfect forward secrecy under actor compromise and ephemeral-key reveal.

1 Introduction

The majority of recently developed key exchange protocols have been proven secure with respect to game-based security models for key exchange protocols [2, 3, 11, 25, 28]. The first such security model was introduced by Bellare and Rogaway [3]. In their model, the adversary is modeled as a probabilistic polynomial-time Turing machine that interacts with the protocol participants through *queries*. The queries specify the capabilities of the adversary. For instance, he can send messages to parties and reveal certain session-keys. The definition of security in the Bellare-Rogaway model requires that (a) two parties 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. Building on this work, Canetti and Krawczyk [11] developed a more complex security model that gives the adversary additional powers such as access to a session-state query that reveals the internal state of a session. LaMacchia et al. [28] adapted the Canetti-Krawczyk model to capture resilience to key compromise impersonation (KCI) attacks and resilience to the leakage of various combinations of long-term and ephemeral secret keys in a single security model. This model is known as the extended Canetti-Krawczyk (eCK) security model.

One important property of KE protocols that is not guaranteed by the eCK security model is *perfect forward secrecy* (PFS). This property holds if an adversary cannot learn the session-keys of past sessions, even if he compromises long-term keys of parties [34, p. 496]. The designers of the eCK model claimed that this property cannot be achieved by two-message KE protocols: "As noted

---

by Krawczyk [26], the PFS requirement is not relevant for 2-round AKE protocols since no 2-round protocol can achieve PFS" [28, p. 5]. In particular, in [25, p. 15], Krawczyk sketched a generic PFS attack, for which he claimed that it breaks the security of any "implicitly authenticated" two-message KE protocol. This class of protocols is not formally defined in [25]. The attack is sketched on a basic g^x, g^y message exchange, as used in the (H)MQV protocols. In the attack, the adversary actively interferes with the communication between the parties by injecting self-constructed messages. This enables him to compute the used session-key if he later learns the long-term secret keys of the parties. To prove a slightly weaker notion of forward secrecy for the HMQV protocol, Krawczyk introduced the notion of *weak perfect forward secrecy* (weak-PFS) [25]. When long-term secret keys are compromised, weak perfect forward secrecy guarantees secrecy of session-keys, but only for sessions in which the adversary did not actively interfere. Krawczyk's comments seem to have led to the incorrect belief that the best that can be achieved for two-message KE protocols is weak perfect forward secrecy, e.g., [28, pages 2 and 5], [14, p. 213]. As a result, even though the eCK security model [28] guarantees only weak perfect forward secrecy, it is currently described in the literature as the strongest possible security model for two-message KE protocols [12, 28, 31].

Contributions. Our first contribution is to push forward the theoretical limits of key exchange security notions. This contribution has two parts. First, we generalize the eCK security model [28] based on the observation that a restriction on the adversary in the eCK model, whose purpose it is to prevent Krawczyk's PFS attack, is stronger than needed. To weaken this restriction (while still preventing the attack) we introduce the concept of *origin-session*, which relaxes the notion of matching session. The resulting model, which we call eCK^w, specifies a slightly stronger variant of weak perfect forward secrecy than the eCK model. We then integrate (strong) perfect forward secrecy into the eCK^w model, which gives rise to the eCK-PFS model. The eCK-PFS model is strictly 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 a fully 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 ephemeral secret keys (i.e., random coins generated in a session).

Our second contribution is a generic security-strengthening transformation (a so-called *compiler*) that promotes the modular design approach of KE protocols. Given a two-message Diffie-Hellman (DH) type KE protocol that is secure in eCK^w or even in a weaker model that we call eCK$^{\text{passive}}$, our transformation yields a two-message protocol that is secure in the eCK-PFS model. The transformation introduces neither additional message dependencies nor additional protocol rounds. Consequently, if our transformation is applied to a one-round protocol, in which all outgoing messages can be computed before any message is received, the result is also a one-round protocol. We apply our transformation to two concrete KE protocols. First, we show that NAXOS [28], the first KE 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 the protocol π_1-core that is insecure in eCK^w can be turned into a protocol secure in eCK-PFS by proving it secure in the weaker eCK$^{\text{passive}}$ model. Both examples illustrate how our transformation enables the modular design of protocols. Overall, this article combines and extends results from our previous papers [16, 17].

Organization. In Section 2 we recall some standard definitions used in this paper. In Section 3 we formalize PFS and weak-PFS, and introduce our security notions eCK^w and eCK-PFS. In Section 4 we provide a transformation that turns any two-message Diffie-Hellman type KE protocol secure in

either eCK^w or eCK$^{\text{passive}}$ into a two-message KE protocol secure in eCK-PFS. We show how this transformation can be applied to concrete KE protocols in Section 5. We discuss related work in Section 6. Finally, we conclude in Section 7.

2 Preliminaries

Let $\|a\|$ denote the length of the binary representation of the integer a, where $\log a < \|a\| \leq \log a + 1$.

Let $G = \langle g \rangle$ be a finite cyclic group of large prime order p with generator g. Similar to the discrete logarithm experiment [22], we define the GAP discrete logarithm (GAP-DLog) experiment for a given group-generating algorithm $\mathcal{G}$, algorithm $\mathcal{A}$, and parameter k as follows.

The GAP discrete logarithm experiment GAP-DLog$_{\mathcal{A},\mathcal{G}}(k)$**:**

1. Run $\mathcal{G}(1^k)$ to obtain (G, p, g) with $\|p\| = k$.
2. Choose $h \in_R G$. (This can be done by choosing $x' \in_R \mathbb{Z}_p$ and setting $h := g^{x'}$.)
3. $\mathcal{A}$ is given G, p, g, h, and outputs $x \in \mathbb{Z}_p$. In addition, $\mathcal{A}$ is given access to a decisional Diffie-Hellman (DDH) oracle that, for any three elements $g^u, g^v, g^w \in G$, replies whether or not $w = uv \bmod p$.
4. The output of the experiment is defined to be 1 if $g^x = h$, and 0 otherwise.

Definition 1 (GAP-DLog Assumption [32]). *The* GAP-DLog *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 decisional Diffie-Hellman (DDH) oracle (that, for any three elements $g^u, g^v, g^w \in G$, replies whether or not $w = uv \bmod p$). More precisely, we say that the* GAP-DLog *assumption holds relative to $\mathcal{G}$, if for all probabilistic polynomial-time algorithms $\mathcal{A}$, there exists a negligible function negl such that*

$$P(\text{GAP-DLog}_{\mathcal{A},\mathcal{G}}(k) = 1) \leq negl(k).$$

Definition 2 (GAP-CDH Assumption [35]). *The GAP-CDH 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 decisional Diffie-Hellman (DDH) oracle (that, for any three elements $g^u, g^v, g^w \in G$, replies whether or not $w = uv \bmod p$).*

Definition 3 (Signature Scheme [22]). *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 1^k and outputs a secret/public key pair (sk, 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 $\sigma := Sign_{sk}(m)$.*
3. *The deterministic verification algorithm Vrfy takes as input a public key pk, a message m, and a signature σ. It outputs a bit b, with $b = 1$ meaning valid and $b = 0$ meaning invalid. We write $b = Vrfy_{pk}(m, \sigma)$.*

Definition 4 (SUF-CMA [7]). *A signature scheme $\Sigma = (Gen, Sign, Vrfy)$ is 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 $Adv_A^{Sig}(k) \leq negl(k)$, where $Adv_A^{Sig}(k)$ denotes the probability of successfully forging a valid signature σ on a message m and (m, σ) is not among the pairs (m_i, σ_i) $(i = 1, ..., q)$ generated during the query phase to a signature oracle $\mathcal{O}^{Sign}$ returning a signature for any message m_i of the adversary's choice.*

3 New Key Exchange Security Notions

We propose two new eCK-like security models for the analysis of key exchange 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 PFS directly into eCK^w. We first describe a framework for defining key exchange security models in Section 3.1. Using this framework, we define our new security notions in Sections 3.2 and 3.3. We then formally compare them in Section 3.4.

3.1 Security notions for key exchange

Terminology. Let $\mathcal{P} = \left\{\hat{P}_1, \hat{P}_2, ..., \hat{P}_N\right\}$ be a finite set of N honest parties represented by binary strings. Each honest party can execute multiple instances of a KE protocol, called sessions, concurrently. We denote session i at honest party $\hat{P}$ 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 $T_s = (s_{actor}, s_{peer}, s_{role}, s_{sent}, s_{recv}) \in \mathcal{P} \times \{0,1\}^* \times \{\mathcal{I}, \mathcal{R}\} \times \{0,1\}^* \times \{0,1\}^*$. The variables s_{actor}, s_{peer} denote the identities of the actor and intended 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 defined by the protocol execution steps. A session can only be activated once.

Adversarial capabilities. As is standard for Bellare-Rogaway style security notions for AKE [3], we model the adversary as a probabilistic polynomial-time (PPT) Turing machine that controls all communications between parties. Similar to the eCK model [28], we consider the following queries:

1. $\mathsf{send}(s, v)$. This query models the adversary sending message v to session s of honest party 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 party $\hat{P}$. Otherwise the query returns $\perp$.
3. $\mathsf{ephemeral\text{-}key}(s)$. This query returns the ephemeral secret keys (i. e., the random coins) of session s.
4. $\mathsf{session\text{-}key}(s)$. This query returns the session key for a completed session s (i. e. a session that has accepted/computed a session-key).
5. $\mathsf{test\text{-}session}(s)$. To respond to this query, a random bit b is chosen. If $b = 0$, 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 KE 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 by the notion of *fresh sessions*. Formally, we define a freshness predicate, that holds if certain combinations of queries did not occur. Examples of such predicates will be given in the following two sections.

Security Model. A game-based security model M is defined by a set of adversary capabilities (queries) M_Q and a freshness notion M_{fresh}.

Security Experiment W in model M. 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 static secret/public key pair(s), for each party $\hat{P} \in \mathcal{P}$. The adversary is then given all public data, including the public keys of all the honest parties in $\mathcal{P}$. Then, the adversary can choose to register arbitrary valid public keys (even public keys of honest parties) on behalf of a set of adversary-controlled parties $\hat{L} \notin \mathcal{P}$.

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 M_Q.
2. At some point in the experiment, E issues a test-session query to a completed session that is M_{fresh} by the time the query is issued.
3. The adversary may continue with queries from M_Q, under the condition that the test session must remain M_{fresh}.
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 a key exchange 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 [28] which is based on matching conversations.

Definition 5 (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 [15]. 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 5 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 6 (security). *A key exchange protocol π is said to be* secure in model M *if, for all PPT adversaries E, it holds that*

- *if two honest parties 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 $Adv_E^\pi(k) \le negl(k)$.*

3.2 eCK^w: strengthening weak-PFS

As stated in the introduction, the eCK model captures weak perfect forward secrecy but not perfect forward secrecy, based on Krawczyk's generic PFS attack [25, 26]. 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 [34, 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 7 (PFS$_{\text{fresh}}$). *A completed session s in experiment W is PFS$_{fresh}$ if it holds that, before the completion of session s, no keys have been registered on behalf of adversary-controlled parties and no* corrupt *query has been issued.*

Definition 8 (PFS). *A key exchange protocol π is said to satisfy* PFS *if it is secure in the PFS model, where $PFS_Q = \{\mathsf{send}, \mathsf{corrupt}\}$ and PFS_{fresh} is defined as above.*

We now return to Krawczyk's generic attack. Consider a two-message protocol in which the agents exchange ephemeral public Diffie-Hellman keys, i. e., g^x and g^y, where x and y are chosen at random from $\mathbb{Z}_p$ (for some large prime p). Then, Krawczyk's attack proceeds as follows. The adversary, impersonating party $\hat{A}$, generates a random value x ($\in \mathbb{Z}_p$) and sends g^x to a responder session at party $\hat{B}$. $\hat{B}$ 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 party $\hat{A}$ 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 PFS.

Krawczyk's attack works directly for all two-message KE protocols that exchange DH keys of the form g^z, where z does not involve the sender's long-term secret key, such as HMQV [26].

Additionally, the attack also works on protocols like NAXOS [28], where z ($\in \mathbb{Z}_p$) is a hash of the sender's long-term secret key and a random value. The adversary can replace this value by an arbitrary value from $\mathbb{Z}_p$.

To still prove some form of forward secrecy for such protocols, 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 [28, 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 9 (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 10 (wPFS$_{fresh}$). *A completed session s in security experiment W is wPFS$_{fresh}$ 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 parties and no* corrupt *query has been issued.*

Definition 11 (weak-PFS). *A key exchange protocol π is said to satisfy* weak-PFS *if it is secure in the wPFS model, where wPFS$_Q = \{$send, corrupt$\}$ and wPFS$_{fresh}$ is defined as above.*

Summarizing, we capture weak-PFS by the compromise of long-term secret keys of parties 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 notion of matching session, e.g., as done in [26]).

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 [9]), 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 12 ($eCK^w{}_{\mathbf{fresh}}$). *A completed session s in security experiment W is $eCK^w{}_{fresh}$ if all of the following conditions hold:*

1. *s_{actor} and s_{peer} are honest parties, i.e. $(s_{actor}, s_{peer}) \in \mathcal{P} \times \mathcal{P}$,*
2. *W does not include the query session-key(s),*
3. *for all sessions s^* such that s^* matches s, W does not include session-key(s^*),*
4. *W does not include both corrupt(s_{actor}) and ephemeral-key(s),*
5. *for all sessions s' such that s' is an origin-session for session s, W does not include both corrupt(s_{peer}) and ephemeral-key(s'), and*
6. *if there exists no origin-session for session s, then W does not include a corrupt(s_{peer}) query.*

Definition 13 (eCK^w security). *A key exchange protocol π is said to satisfy eCK^w security if it is secure in the eCK^w model, where $eCK^w{}_{fresh}$ is defined as above, and $eCK^w{}_Q = \{\mathsf{send}, \mathsf{corrupt}, \mathsf{ephemeral\text{-}key}, \mathsf{session\text{-}key}\}$.*

In Section 5.1 we will show that the NAXOS protocol satisfies eCK^w security.

3.3 eCK-PFS: integrating PFS into eCK^w.

We next extend the eCK^w model by integrating perfect forward secrecy, which yields the strictly 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 [9] incorporates PFS, eCK-PFS additionally captures leakage of various combinations of ephemeral secret keys and long-term secret keys as well as perfect forward secrecy under actor compromise.

Definition 14 ($eCK\text{-}PFS_{\mathbf{fresh}}$). *A completed session s in security experiment W is $eCK\text{-}PFS_{fresh}$ if all of the following conditions hold:*

1. *s_{actor} and s_{peer} are honest parties, i.e. $(s_{actor}, s_{peer}) \in \mathcal{P} \times \mathcal{P}$,*
2. *W does not include the query session-key(s),*
3. *for all sessions s^* such that s^* matches s, W does not include session-key(s^*),*
4. *W does not include both corrupt(s_{actor}) and ephemeral-key(s),*

5. *for all sessions s' such that s' is an origin-session for session s, W does not include both* corrupt(s_{peer}) *and* ephemeral-key(s')*, and*
6. *if there exists no origin-session for session s, then W does not include a* corrupt(s_{peer}) *query before the completion of session s.*

Definition 15 (eCK-PFS security). *A key exchange protocol π is said to satisfy* eCK-PFS *security if it is secure in the eCK-PFS model, where eCK-PFS$_Q = \{$send, corrupt, ephemeral-key, session-key$\}$ and eCK-PFS$_{fresh}$ is defined as above.*

3.4 Relations between the Security Models

Let $secure(M, \pi)$ be a predicate that is true if and only if the protocol π is secure in security model M. Here we formalize the relative strengths of security between game-based KE security models given in [13] as follows.

Definition 16. *Let Π be a class of KE protocols. We say that a security model M' is* stronger *than a security model M with respect to Π, denoted by $M \leq_{Sec}^{\Pi} M'$, if*

$$\forall \ \pi \in \Pi. \ secure(M', \pi) \rightarrow secure(M, \pi). \tag{1}$$

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 stronger than model M, denoted by $M \nleq_{Sec}^{\Pi} M'$, it suffices to find a protocol $\pi \in \Pi$ such that π is secure in model M' and insecure in model M, as in [13,15]. We say that model M' is *strictly stronger* than model M with respect to protocol class Π, denoted by $M <_{Sec}^{\Pi} M'$, if $M \leq_{Sec}^{\Pi} M'$ and $M' \nleq_{Sec}^{\Pi} M$.

Informally, the eCK-PFS model is stronger than eCK^w because the eCK-PFS model allows the adversary to corrupt all parties *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 1. *Let Π be the class of two-message KE protocols. The eCK-PFS model is strictly stronger than the eCK^w model with respect to Π.*

The first part of the proof of Proposition 1, namely that eCK-PFS is stronger than eCK^w, proceeds in a similar way as the reduction proofs in [13].

Proof. We first show that the eCK-PFS model is stronger than the eCK^w model with respect to Π. The first condition of Definition 6 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 6 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, ephemeral-key, 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 eCK$^w_{fresh}$ holds for the test session, then by definition eCK-PFS$_{fresh}$ also holds. In particular, if there is no origin session, then the sixth condition of eCK$^w_{fresh}$ requires that there is no corrupt of the peer, which implies the sixth condition of eCK-PFS$_{fresh}$. Hence, it holds that $Adv_E^{\pi}(k) \leq Adv_{E'}^{\pi}(k)$, where k denotes the security parameter. Since by assumption protocol π
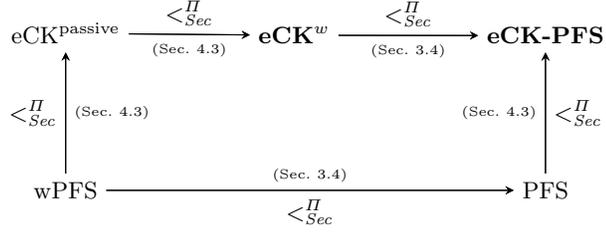
Fig. 1. Relations between the security models for the class of two-message KE protocols.

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 strictly stronger than eCK^w since, e.g., the NAXOS protocol is secure in eCK^w, as we show in Section 5, but insecure in eCK-PFS due to the PFS attack described in Section 3.2. □

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

Proposition 2. *Let Π be the class of two-message KE protocols. The PFS model is strictly stronger than the wPFS model with respect to Π.*

Proof. The proof that the PFS model is stronger than the wPFS model is similar to the corresponding proof of Proposition 1. Note that the test session being wPFS$_{fresh}$ implies that it is PFS$_{fresh}$ since we have a further freshness requirement in the wPFS model. The PFS model is strictly stronger than the wPFS model since, e.g., the NAXOS protocol achieves weak-PFS as can be easily deduced from the proof of Proposition 7, but does not satisfy PFS due to the generic PFS attack described in Section 3.2. □

The relations between these four models and the eCK^{passive} model that we will define in Section 4.3 are depicted in Figure 1.

4 A Security-Strengthening Transformation from eCK^w to eCK-PFS

4.1 Protocol Class $\mathcal{DH}$-2

We define a class of two-message Diffie-Hellman type key exchange protocols (similar to the class of KE protocols in [9]). Then, we present a security-strengthening transformation (compiler) that can be applied to any such protocol. Finally we show that this transformation turns any KE protocol secure in eCK^w into a KE protocol secure in eCK-PFS.

Let k be a security parameter and let G be a finite cyclic group of prime order p with generator g, where $\|p\| = k$. Let Ω be static publicly known data such as parties' 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 (e.g. $S = \mathbb{Z}_p$ or $S = \{0,1\}^k$). We denote by $x \in_R S$ that x is chosen uniformly at random from the set S. In the generic two-message DH type protocol, illustrated in Figure 2, party $\hat{A}$'s long-term secret key is $a \in_R \mathbb{Z}_p$ and $\hat{A}$'s long-term public key is $A = g^a$. The session-specific ephemeral secret key of the session at party $\hat{A}$ is denoted by

$$\boxed{\begin{array}{c} \hat{A} : (a, A) \\ \hline r_{\hat{A}}, X := g^{f_{\mathcal{I}}(r_{\hat{A}}, a, \Omega)} \end{array}}$$

$$\xrightarrow{\quad X \quad}$$
$$\xleftarrow{\quad Y \quad}$$

$$\boxed{\begin{array}{c} \hat{B} : (b, B) \\ \hline r_{\hat{B}}, Y := g^{f_{\mathcal{R}}(r_{\hat{B}}, b, \Omega)} \end{array}}$$

$$K_{\hat{A}} := F_{\mathcal{I}}(f_{\mathcal{I}}(r_{\hat{A}}, a, \Omega), a, Y, \Omega) \qquad\qquad K_{\hat{B}} := F_{\mathcal{R}}(f_{\mathcal{R}}(r_{\hat{B}}, b, \Omega), b, X, \Omega)$$
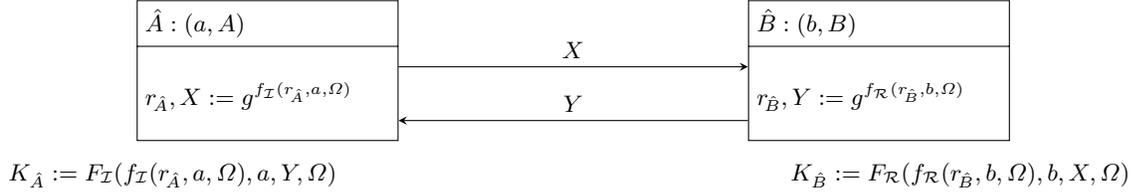
Fig. 2. A generic two-message DH type protocol

$r_{\hat{A}} \in_R S$ and the corresponding ephemeral public key is denoted by X. Similarly, party $\hat{B}$'s long-term secret/public key pair is (b, B) and the ephemeral secret/public key pair of the session at $\hat{B}$ is denoted by $(r_{\hat{B}}, Y)$. The public functions $f_{\mathcal{I}}, f_{\mathcal{R}} : \{0, 1\}^* \to \mathbb{Z}_p$ depend on the ephemeral secret key and may depend on the long-term secret key or on public information. The public functions $F_{\mathcal{I}}, F_{\mathcal{R}} : \{0, 1\}^* \to \{0, 1\}^k$ depend on the Diffie-Hellman exponent the long-term secret key, the received Diffie-Hellman exponential and other public information. We assume that the public keys of all parties are known to all other participants in the protocol.

Protocol description. The generic two-message DH type protocol, depicted in Figure 2, proceeds as follows:

1. Upon activation of session $s = (\hat{A}, i) \in \mathcal{P} \times \mathbb{N}$ with peer $\hat{B}$, $\hat{A}$ (the initiator) performs the steps:
   – Choose an ephemeral secret key $r_{\hat{A}} \in_R S$ and compute $X = g^{f_{\mathcal{I}}(r_{\hat{A}}, a, \Omega)}$.
   – Send X (and possibly other public data, e.g. identifiers of peer and actor of the session) to $\hat{B}$.
   – Initialize T_s to $(\hat{A}, \hat{B}, \mathcal{I}, m, \epsilon)$, where m denotes the message sent by session s.
2. Upon activation of session $s' = (\hat{B}, j) \in \mathcal{P} \times \mathbb{N}$ with message X (and possibly other data) on behalf of $\hat{A}$, party $\hat{B}$ (the responder) performs the steps:
   – Check that $X \in G$.
   – Choose an ephemeral secret key $r_{\hat{B}} \in_R S$ and compute $Y = g^{f_{\mathcal{R}}(r_{\hat{B}}, b, \Omega)}$.
   – Compute $K_{\hat{B}} = F_{\mathcal{R}}(f_{\mathcal{R}}(r_{\hat{B}}, b, \Omega), b, X, \Omega)$.
   – Send Y (and possibly other public data) to $\hat{A}$.
   – Set $T_{s'}$ to $(\hat{B}, \hat{A}, \mathcal{R}, m', n')$, where m' denotes the message sent by session s' and n' the message received by session s', and complete the session by accepting $K_{\hat{B}}$ as the session-key.
3. Upon receiving message Y (with possibly other data) in session s , party $\hat{A}$ performs the steps:
   – Check that $Y \in G$.
   – Compute $K_{\hat{A}} = F_{\mathcal{I}}(f_{\mathcal{I}}(r_{\hat{A}}, a, \Omega), a, Y, \Omega)$.
   – Update T_s to $(\hat{A}, \hat{B}, \mathcal{I}, m, n)$ and complete 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.

Definition 17 (Protocol Class $\mathcal{DH}$-2). *We define $\mathcal{DH}$-2 as the class of all two-message key-exchange protocols that follow the description of a generic DH type protocol and meet the following validity requirement:*

– In the presence of an eavesdropping adversary, two honest parties $\hat{A}$ and $\hat{B}$ can complete matching sessions (in the sense of Definition 5), in which case they hold the same session-key.

The validity requirement requires that if the messages of two honest parties $\hat{A}$ and $\hat{B}$ are faithfully relayed to each other, then both parties end up with a shared session-key (see also [2–4]). Note that, e. g., the KE protocols NAXOS [28], NAXOS+ [31], NETS [30] and CMQV [37] belong to the class $\mathcal{DH}$-2.

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

4.2 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, by applying the signature transformation SIG. Party $\hat{A}$ 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, party $\hat{B}$'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 proceeds as protocol π except that each party 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.

$\hat{A} : (a, A), (sk_{\hat{A}}, pk_{\hat{A}})$	$X, \sigma_{\hat{A}} = Sign_{sk_{\hat{A}}}(X \left[, \hat{B}\right])$	$\hat{B} : (b, B), (sk_{\hat{B}}, pk_{\hat{B}})$
$r_{\hat{A}}, X := g^{f_{\mathcal{I}}(r_{\hat{A}}, a, \Omega)}$	$Y, \sigma_{\hat{B}} = Sign_{sk_{\hat{B}}}(Y \left[, X, \hat{A}\right])$	$r_{\hat{B}}, Y := g^{f_{\mathcal{R}}(r_{\hat{B}}, b, \Omega)}$

$$K_{\hat{A}} := F_{\mathcal{I}}(f_{\mathcal{I}}(r_{\hat{A}}, a, \Omega), a, Y, \Omega) \qquad\qquad K_{\hat{B}} := F_{\mathcal{R}}(f_{\mathcal{R}}(r_{\hat{B}}, b, \Omega), b, X, \Omega)$$
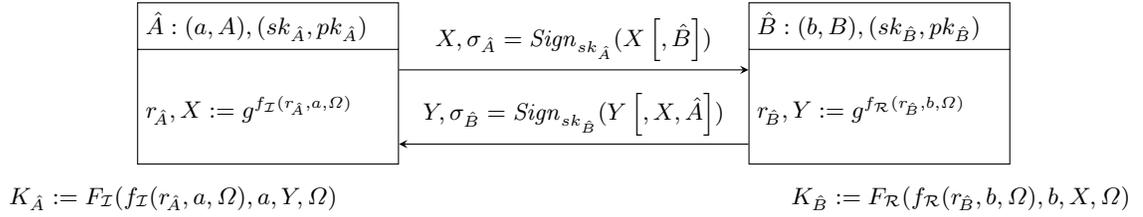
Fig. 3. A transformed generic protocol SIG(π)

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

Definition 18 (Security-strengthening protocol transformation). *Let Π_1 and Π_2 be two classes of KE protocols. We say that a function $f : \Pi_1 \to \Pi_2$ is a security-strengthening protocol transformation from the model M to the model M' if*

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

The previous definition implies that if a KE 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 16, it follows that protocol $f(\pi)$ is secure in model M.

4.3 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 security-strengthening from a weaker version of the eCK^w model, which we call eCK^{passive}, to eCK-PFS.

Weakening eCK^w to $eCK^{passive}$. Informally, the eCK^{passive} model weakens eCK^w by capturing only passive attacks on the test session (i. e., the session under attack). 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 ephemeral values of sessions and long-term secret keys of parties, thus it also captures, e. g., weak perfect forward secrecy. Formally, the eCK^{passive} model only differs from eCK^w in its definition of freshness. In eCK^{passive}, there must exist an origin session for the test session.

Definition 19 ($eCK^{\text{passive}}_{\text{fresh}}$). *A completed session s in security experiment W is $eCK^{passive}_{fresh}$ if all of the following conditions hold:*

1. *s_{actor} and s_{peer} are honest parties, i. e. $(s_{actor}, s_{peer}) \in \mathcal{P} \times \mathcal{P}$,*
2. *there exists an origin-session s' for session s,*
3. *W does not include the query session-key(s),*
4. *for all sessions s^* such that s^* matches s, W does not include session-key(s^*),*
5. *W does not include both corrupt(s_{actor}) and ephemeral-key(s), and*
6. *for all sessions s' such that s' is an origin-session for session s, W does not include both corrupt(s_{peer}) and ephemeral-key(s').*

Definition 20 (eCK^{passive} security). *A key exchange protocol π is said to satisfy eCK^{passive} security if it is secure in the $eCK^{passive}$ model, where $eCK^{passive}_Q = \{\text{send}, \text{corrupt}, \text{ephemeral-key}, \text{session-key}\}$ and $eCK^{passive}_{fresh}$ is defined as above.*

Proposition 3. *Let Π be the class of two-message KE protocols.*

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

Proof. The proof that the eCK^{passive} model is stronger than the wPFS model is similar to the corresponding proof of Proposition 1. Note that the test session being $wPFS_{\text{fresh}}$ implies that it is $eCK^{\text{passive}}_{\text{fresh}}$ since in the wPFS model the adversary is not given access to the queries ephemeral-key and session-key. Also, in the wPFS model the adversary is not allowed to either register keys on behalf of adversary-controlled parties 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 stronger than the PFS model.

The eCK^{passive} model is strictly stronger than the wPFS model. It can be easily shown that protocol TS2 achieves weak-PFS by adapting the proof of [21, 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 ephemeral secret key of the origin-session for the test session. Hence, TS2 is insecure in eCK^{passive}. The eCK-PFS model is strictly stronger than the PFS model. By Theorem 1, it holds that SIG(TS2) satisfies PFS (see Remark 1). However, SIG(TS2) is insecure in eCK-PFS for a similar reason as TS2 is insecure in eCK^{passive}. □

Propositions 4 and 5 will be used in the proofs of Theorem 1 and Corollary 1.

Proposition 4. *Let Π be the class of two-message KE protocols. The eCK^w model is strictly stronger than the $eCK^{passive}$ model with respect to Π.*

Proof. The proof that the eCK^w model is stronger than the eCK^{passive} model is similar to the corresponding proof of Proposition 1. Note that if the test session is eCK^{passive}$_{fresh}$, then it is also eCK^w$_{fresh}$. This follows from the fact that compared to the eCK^w model, we have a further freshness condition on the test session in eCK^{passive}, namely that an origin session exists for the test session.

The model eCK^w is strictly stronger than the eCK^{passive} model since, e. g., the KE protocol π_1-core is secure in eCK^{passive}, as we show in Section 5, but insecure in eCK^w as observed in [25]. $\square$

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

Proof. Since eCK^{passive} $<^{\Pi}_{Sec}$ eCK^w and eCK^w $<^{\Pi}_{Sec}$ eCK-PFS, it follows that eCK^{passive} $<^{\Pi}_{Sec}$ eCK-PFS by transitivity of the implication (1). The eCK-PFS model is strictly stronger than eCK^{passive} since, e. g., the protocol π_1-core is secure in eCK^{passive} but insecure in eCK-PFS. $\square$

How to Provably Achieve eCK-PFS Security. We show in Theorem 1 below that the SIG transformation is a security-strengthening protocol transformation from eCK^{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 [6]. We require a deterministic signature scheme so that we do not have to consider additional random coins from the signature generation procedure when reasoning about ephemeral-key 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 random coins involved in the signature generation learned through an ephemeral-key query (as noted in [28]). The following lemma is used in the proof of Theorem 1.

Lemma 1 (Difference Lemma [36]). *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*

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

We now give the main theorem and corollary before proceeding with their proofs.

Theorem 1. *Let Π denote the class of two-message KE protocols. Under the assumption that the signature scheme is deterministic and SUF-CMA, the transformation $\mathrm{SIG} : \mathcal{DH}\text{-}2 \to \Pi$ is a security-strengthening protocol transformation from $eCK^{passive}$ to eCK-PFS according to Definition 18.*

Corollary 1. *Let Π denote the class of two-message KE protocols. Under the assumption that the signature scheme is deterministic and SUF-CMA, the transformation $\mathrm{SIG} : \mathcal{DH}\text{-}2 \to \Pi$ is a security-strengthening protocol transformation from eCK^w to eCK-PFS according to Definition 18.*

Proof (Theorem 1). The first condition of Definition 18 is satisfied by Proposition 5. We next verify whether the second condition of Definition 18 holds. Let $\pi \in \mathcal{DH}\text{-}2$ be secure in model eCK^{passive}. It is straightforward to verify the first condition of Definition 6, i. e., that matching sessions of protocol $\mathrm{SIG}(\pi)$ compute the same key (since matching sessions of protocol π compute the same key). We show next that the second condition of Definition 6 holds, i. e., an adversary against $\mathrm{SIG}(\pi)$ 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 [36]. 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 parties and activated sessions, respectively.

Game 0. This game reflects the security experiment W in model eCK-PFS, as defined in Section 3.1, played by a PPT adversary E against the protocol $\text{SIG}(\pi)$.

Game 1. [Transition based on a small failure event] Let $Coll_{\text{SIG}(\pi)}$ be the small failure event that a collision for protocol $\text{SIG}(\pi)$ occurs (e.g., in ephemeral secret keys). 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 [8, 19])] 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,

$$P(S_2) = P(S_2|T)P(T) + P(S_2|T^c)P(T^c) = \frac{1}{2}P(T) + P(S_1)P(T^c)$$

$$= P(T^c)(P(S_1) - \frac{1}{2}) + \frac{1}{2}.$$

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 party $\hat{P}$, that is, adversary E issues a send(s^*, V, σ) 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 party $\hat{P}$ 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 party $\hat{P} \in \mathcal{P}$ 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 corrupt($\hat{P}$) query before the completion of session s^* and no origin-session exists for s^*, then session s^* is not eCK-PFS$_{fresh}$. 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 is eCK-PFS$_{fresh}$ by 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 be eCK-PFS$_{fresh}$ and Game 2 would have aborted. Hence this case can be excluded as well.
3. If E does not issue a corrupt($\hat{P}$) query before the completion of session s^*, then he can only impersonate party $\hat{P}$ to session s^* by forging a signature on a message with respect to the long-term public key of $\hat{P}$.

Claim. 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. 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 parties and sets its public key to pk. We denote this party 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 parties $\hat{P}_i \in \mathcal{P}$ 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 parties $\hat{P}_i \in \mathcal{P}$ as needed for protocol π and stores the associated secret keys.
ALGORITHM M:

1. Run E on input 1^k and the public keys for all of the N parties.
2. If E issues a 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 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 $Y = g^x$ 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 $W \in G$ is contained in message s^*_{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 party $\hat{P}$ 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 $\mathsf{session\text{-}key}$, $\mathsf{ephemeral\text{-}key}$ are answered in the appropriate way since M has chosen the ephemeral secret keys for all the sessions and the long-term secret keys for use in protocol π for all the parties.

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 parties in the set $\mathcal{P} \setminus \{\hat{P}\}$. In case $\hat{Q}_i \notin \mathcal{P}$, M returns $\bot$. 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 parties, that is, $P(F) \le N Adv_M^{Sign}(k)$.

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

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 $Adv_E^{\mathrm{SIG}(\pi), \mathrm{Game\ 3}, O}(k)$ in terms of the security of protocol π.

Claim. 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 $\mathrm{SIG}(\pi)$ with non-negligible advantage, then we can construct an efficient adversary E' in eCK$^{\mathrm{passive}}$ succeeding in Game 3 against protocol π with non-negligible advantage using adversary E as a subroutine. Moreover, it holds that $Adv_E^{\mathrm{SIG}(\pi), \mathrm{Game\ 3}, O}(k) \le Adv_{E'}^{\pi, \mathrm{Game\ 3}}(k)$.

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

1. Run E against $\mathrm{SIG}(\pi)$ on input 1^k and the public key pairs for all of the N parties.

---

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

2. When E issues a corrupt($\hat{P}$) query to some party $\hat{P}$, E' issues that query to party $\hat{P}$ 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 an ephemeral-key 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 is the Diffie-Hellman exponential contained in z_{sent}. If the checks succeed, then E' issues the query 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 being eCK-PFS$_{\text{fresh}}$ is also eCK$^{\text{passive}}{}_{\text{fresh}}$. Thus, it holds that

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

Finally,

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

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

$\square$

Proof (Corollary 1). The first condition of Definition 18 is satisfied by Proposition 1. We next verify the second requirement. Let $\pi \in \mathcal{DH}\text{-}2$ secure in eCK^w. Since by Proposition 4 we have eCK$^{\text{passive}} \leq_{Sec}^{\Pi}$ eCK^w, it follows that protocol π is secure in eCK$^{\text{passive}}$. By Theorem 1, SIG is a security strengthening protocol transformation from eCK$^{\text{passive}}$ to eCK-PFS. Therefore, the transformed protocol SIG(π) is secure in eCK-PFS. $\square$

Remark 2. Let eCK-NEK$^{\text{passive}}$ and eCK-NEK-PFS be the security models obtained from eCK$^{\text{passive}}$ and eCK-PFS (respectively) by removing the ephemeral-key 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 KE protocol $\pi \in \mathcal{DH}$-2 secure in eCK-NEK$^{\text{passive}}$, 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$^{\text{passive}}$ by wPFS and eCK-NEK-PFS by PFS.

Remark 3. Blake-Wilson and Menezes [5, 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, i.e. before interacting with the parties 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 [5, p. 159] as well as on KEA [29, p. 380]) are captured in our models eCK$^{\text{passive}}$, 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.

5 Application of SIG to Concrete KE Protocols

In Section 5.1 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 5.2 we show how to prove the security of the π_1 protocol from [16] in eCK-PFS by proving the much weaker protocol π_1-core secure in eCK$^{\text{passive}}$ and applying the SIG transformation to π_1-core.

5.1 NAXOS Revisited

The NAXOS protocol [28], shown in Figure 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 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 $F_{\mathcal{R}}(f_{\mathcal{R}}(r_{\hat{B}}, b, \Omega), b, X, \Omega) = H_2(A^{H_1(r_{\hat{B}}, b)}, X^b, X^{H_1(r_{\hat{B}}, b)}, \hat{A}, \hat{B})$.

$$\hat{A} : (a, A)$$
$$r_{\hat{A}}, X := g^{H_1(r_{\hat{A}}, a)}$$

$$X \longrightarrow$$
$$\longleftarrow Y$$

$$\hat{B} : (b, B)$$
$$r_{\hat{B}}, Y := g^{H_1(r_{\hat{B}}, b)}$$

$$K_{\hat{A}} := H_2(Y^a, B^{H_1(r_{\hat{A}}, a)}, Y^{H_1(r_{\hat{A}}, a)}, \hat{A}, \hat{B}) \qquad K_{\hat{B}} := H_2(A^{H_1(r_{\hat{B}}, b)}, X^b, X^{H_1(r_{\hat{B}}, b)}, \hat{A}, \hat{B})$$
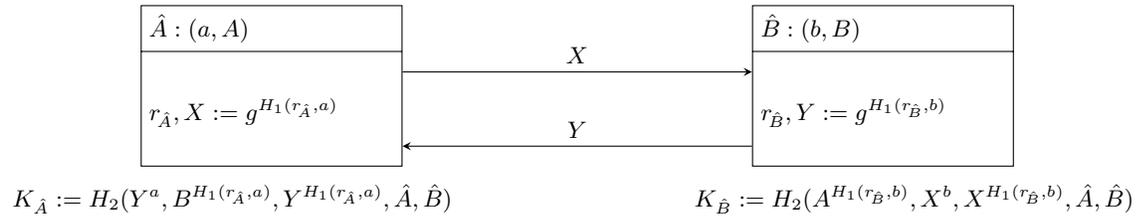
Fig. 4. NAXOS protocol [28]

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

Proposition 6. *Under the GAP-CDH assumption in the cyclic group G of prime order p, NAXOS satisfies eCK^w security, when H_1 and H_2 are modeled as independent random oracles.*

In contrast to the proof of NAXOS in the eCK model [28], the proof of Proposition 6 distinguishes between the cases whether or not an origin-session (instead of a matching session) exists for the test session.

Proof (Sketch). Similar to [28, 37], we analyze the following events:

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

T_O denotes the event that there exists an origin-session for the test session, DL denotes the event that there exists a party $\hat{C} \in \mathcal{P}$ (with long-term secret key c) 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 NAXOS by querying 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 $\sigma_3 = \mathrm{CDH}(X, Y)$, given that the test session is s^* with $T_{s^*} = (\hat{A}, \hat{B}, \mathcal{I}, X, Y)$. □

The full proof of Proposition 6 is given in the appendix. Applying the SIG transformation to the NAXOS protocol yields the protocol SIG(NAXOS), depicted in Figure 5.

$\hat{A} : (a, A), (sk_{\hat{A}}, pk_{\hat{A}})$	$X, \sigma_{\hat{A}} = Sign_{sk_{\hat{A}}}(X \left[, \hat{B}\right])$	$\hat{B} : (b, B), (sk_{\hat{B}}, pk_{\hat{B}})$
$r_{\hat{A}}, X := g^{H_1(r_{\hat{A}}, a)}$	$Y, \sigma_{\hat{B}} = Sign_{sk_{\hat{B}}}(Y \left[, X, \hat{A}\right])$	$r_{\hat{B}}, Y := g^{H_1(r_{\hat{B}}, b)}$

$K_{\hat{A}} := H_2(Y^a, B^{H_1(r_{\hat{A}}, a)}, Y^{H_1(r_{\hat{A}}, a)}, \hat{A}, \hat{B})$ $\qquad\qquad$ $K_{\hat{B}} := H_2(A^{H_1(r_{\hat{B}}, b)}, X^b, X^{H_1(r_{\hat{B}}, b)}, \hat{A}, \hat{B})$
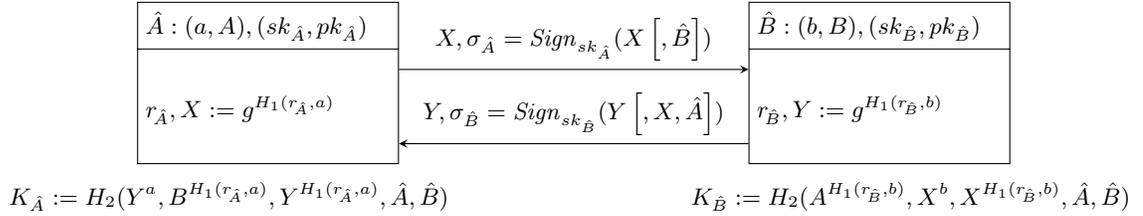
Fig. 5. SIG(NAXOS) protocol

Combining Proposition 6 with Theorem 1, we obtain the following result.

Corollary 2. *Under the GAP-CDH assumption in the cyclic group G of prime order p, using a deterministic SUF-CMA signature scheme, the* SIG(NAXOS) *protocol satisfies eCK-PFS security, when H_1, H_2 are modeled as independent random oracles.*

5.2 Proving π_1 Secure in eCK-PFS via π_1-core

Figure 6 shows the protocol π_1-core $\in \mathcal{DH}$-2, where $KDF : \{0, 1\}^* \to \{0, 1\}^k$ denotes a key derivation function and $x, y \in_R \mathbb{Z}_p$. As observed in [25], 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 $i \in_R \mathbb{Z}_p$) to $\hat{A}$. $\hat{A}$ would compute the secret value for the session-key as $(BB^{-1}Z)^{x+a} = Z^{x+a}$ which can also be 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 eCK$^{\mathrm{passive}}$ model, as the following proposition shows.
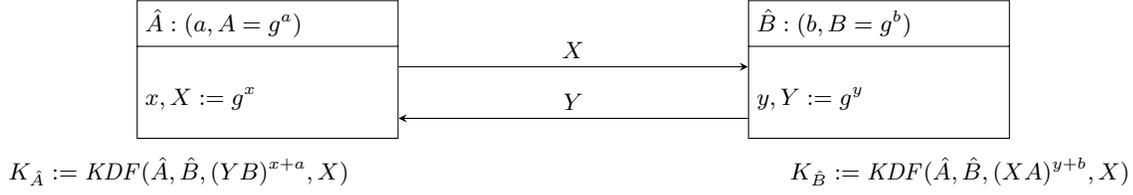
Fig. 6. Protocol π_1-core

Proposition 7. *Under the GAP-CDH assumption in the cyclic group G of prime order p, the protocol π_1-core satisfies $eCK^{passive}$ security, when KDF is modeled as a random oracle.*

Proof (Sketch). We analyze the following events:

1. K^c, and
2. $T_O \wedge K$, where

T_O denotes the event that there exists an origin-session for the test session, and K denotes the event that the adversary M wins the security experiment against π_1-core by querying KDF with $(\hat{A}, \hat{B}, \sigma, X)$, where $\sigma = \mathrm{CDH}(YB, XA)$, given that the test session is s^* with $T_{s^*} = (\hat{A}, \hat{B}, \mathcal{I}, X, Y)$. Recall that in case there is no origin-session for the test session, the test session is not $eCK^{passive}_{fresh}$. $\square$

The full proof of Proposition 7 is given in the appendix. Applying the SIG transformation to the π_1-core protocol yields the π_1 protocol from [16], depicted in Figure 7.
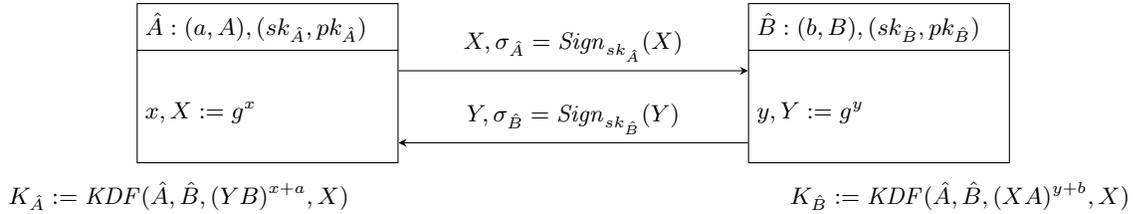
Fig. 7. π_1 protocol [16]

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

Corollary 3. *Under the GAP-CDH assumption in the cyclic group G of prime order p, using a deterministic SUF-CMA signature scheme, the protocol π_1 satisfies eCK-PFS security, when KDF is modeled as a random oracle.*

6 Related Work

6.1 Perfect Forward Secrecy

The majority of related works claim that perfect forward secrecy cannot be achieved in a two-message KE protocol [8, 14, 25, 27, 28]. There are two notable exceptions. First, the two-message modified-Okamoto-Tanaka (mOT) protocol by Gennaro et al. [20] 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 [20], the mOT protocol and its variants are not resilient against loss of ephemeral keys, and they are therefore insecure in eCK-like models. Second, in [9], Boyd and González Nieto suggest a transformation $\mathcal{C}$ based on adding MACs on the message exchange of a key-exchange protocol that satisfies weak perfect forward secrecy, to achieve perfect forward secrecy. However, the MAC transformation does not ensure security in eCK-PFS, because it does not guarantee perfect forward secrecy under actor compromise and leakage of ephemeral secret keys, as we show in Section 6.2. It is important to note that our security model eCK-PFS prevents the attack scenario described in [9, p. 458] since we restrict the adversary from revealing the ephemeral secret keys of the origin-session for the test session (see Definition 14).

In [21], Jeong, Katz and Lee introduce the one-round KE 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 [3] and captures forward secrecy by allowing the adversary to corrupt both actor and peer of some target session in case the adversary is passive during the execution of the target session (which corresponds to weak-PFS). As observed in [1], it seems that protocol TS3 satisfies a stronger forward secrecy property than protocol TS2, although this is not stated or proven in [21]. We conjecture that protocol TS3 in fact achieves PFS, under the same assumptions as stated in [21, Theorem 3].

In [27], LaMacchia et al. describe an eCK variant 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 we are considering here, because it uses the concept of matching session instead of origin-session. Boyd and González Nieto's replay attack [9, p. 458] serves as a generic counterexample: it shows that no two-message key exchange 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 5.

6.2 Protocol Transformations

Several protocol transformations (often called *compilers*) that aim towards a modular approach in the design and analysis of protocols have been proposed in the literature (see e. g. [9–11, 23]). In particular, applying such transformations to concrete protocols should lead to protocols that are secure in a stronger security model.

In [11], Canetti and Krawczyk 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. The idea is that a protocol secure in the AM can be translated into a protocol secure in the stronger model UM using an *authenticator*. Thus, authenticators can be seen as compilers which translate protocols secure in the AM into protocols secure in the UM.

In the context of authenticated group key exchange, Katz and Yung [23] propose a compiler 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. Note that the Katz and Yung compiler is however not a security-strengthening transformation in the sense of Definition 18 due to the counterexample given in [10, p. 13]. Bresson et al. [10] propose a similar signature-based compiler than [23] which 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 [10] is slightly stronger than the one in [23] since in addition to eavesdropping on regular protocol executions, he may delay or delete messages, or change their delivery order. The security model eCK$^{\text{passive}}$ introduced in Section 4 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.

The $\mathcal{C}$ transformation by Boyd and González Nieto [9] aims to provide message origin authentication based on 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 context, the static Diffie-Hellman key does not provide any 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). After the completion of the test session he can reveal the long-term secret keys of the peer, effectively performing a variant of Krawczyk's attack. We next detail a concrete instance of this attack, showing that $\mathcal{C}(\text{NAXOS})$ [9] is insecure in eCK-PFS. We denote by $S = g^{a'b'}$ the shared static DH key between the parties $\hat{A}$ and $\hat{B}$.

1. The adversary E first reveals the long-term secret keys of party $\hat{A}$.
2. He then activates an initiator session s at $\hat{A}$ with peer $\hat{B}$ via the query $\mathsf{send}(s, \hat{B})$ and receives as a response the message $m = X, \text{MAC}_S(\hat{A}, \hat{B}, X)$, 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 an arbitrary $z \in \mathbb{Z}_p$, computes $Z = g^z$ and sends message $\tilde{m} = Z, \text{MAC}_S(\hat{B}, \hat{A}, Z)$ to session s. 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 completes the session by accepting $K_{\hat{A}}$ as the session key.
4. Now, E chooses the completed session s as the test session, and reveals the long-term secret keys of party $\hat{B}$. 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, the MAC transformation does not achieve security in eCK-PFS.

7 Conclusions

We provided two new eCK-like security notions, namely eCK^w and eCK-PFS. The eCK^w model slightly strengthens eCK by including a more precise modeling of weak-PFS. The strictly stronger eCK-PFS notion guarantees PFS, even in the presence of eCK-like adversaries. Note that 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 key exchange protocols such as CMQV [37], NAXOS [28], $\mathcal{C}(\text{NAXOS})$ [9], or HMQV [25, 26], fail to achieve security in eCK-PFS. We specified a security-strengthening

transformation that transforms any two-message DH type KE protocol secure in eCK$^{\text{passive}}$ or 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 eCK$^{\text{passive}}$ model. As a concrete example we have proven that the π_1-core protocol is secure in eCK$^{\text{passive}}$. Subsequent application of the SIG transformation to π_1-core implies that the efficient π_1 protocol from [16] is secure in eCK-PFS. This example illustrates the use of SIG in the modular design of KE protocols.

It remains an open question whether there exist more efficient generic transformations that yield two-message KE protocols secure in eCK-PFS.

References

1. 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.
2. 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.
3. 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.
4. 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.
5. 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.
6. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil Pairing. In *ASIACRYPT'01*, pages 514–532, 2001.
7. 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.
8. 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.
9. 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.
10. 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/.
11. 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.
12. 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.

13. 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.

14. 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.

15. 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.

16. 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/.

17. 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, Berlin, Heidelberg, 2012. Springer-Verlag.

18. 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.

19. A.W. Dent. A note on game-hopping proofs. Cryptology ePrint Archive, Report 2006/260, 2006. http://eprint.iacr.org/2006/260.

20. 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.

21. 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.

22. J. Katz and Y. Lindell. *Introduction to Modern Cryptography*. Chapman Hall/CRC, 2008.

23. J. Katz and M. Yung. Scalable protocols for authenticated group key exchange. In Dan Boneh, editor, *Advances in Cryptology-CRYPTO 2003*, volume 2729, pages 110–125. Springer, 2003.

24. M. Kim, A. Fujioka, and B. Ustaoglu. Strongly secure authenticated key exchange without naxos' approach. In *IWSEC'09*, pages 174–191, 2009.

25. H. Krawczyk. HMQV: A High-Performance Secure Diffie-Hellman Protocol. Cryptology ePrint Archive, Report 2005/176, 2005. http://eprint.iacr.org/.

26. 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.

27. 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/.

28. 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.

29. 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, New York, NY, USA, April 24-26, 2006. Proceedings*, volume 3958/2006 of *LNCS*, pages 378–394. Springer, 2006.

30. 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/.

31. 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/.

32. U. Maurer. Abstract models of computation in cryptography. In Nigel Smart, editor, *Cryptography and Coding 2005*, volume 3796 of *LNCS*, pages 1–12. Springer, December 2005.

33. A. Menezes. Another look at HMQV. In *Journal of Mathematical Cryptology JMC*, volume 1, pages 47–64, 2008.

34. A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, FL, USA, 1996.

35. 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.

36. V. Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2006. `http://eprint.iacr.org/`.

37. 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.

A On the eCK model [28]

There are two main aspects in which the eCK model is underspecified.

First, the eCK model specifies that in the setup phase of the security experiment, the adversary may register arbitrary public keys [28, p. 8]. This can be interpreted in at least two ways: (a) the adversary may register arbitrary *valid* public keys (e.g., elements of a given group G), or (b) the adversary may register arbitrary bit strings (e.g., elements that do not belong to a given group G). As the security proof of NAXOS in the eCK model [28, pp. 12-16] is incomplete, it is unclear whether the result of LaMacchia et al. [28, Theorem 1] holds under the second interpretation. In addition, LaMacchia et al. [28, Figure 1] state that the HMQV protocol achieves CK-security under arbitrary key registration, (the same key registration as in the eCK model). This statement is only correct under interpretation (a), because HMQV is vulnerable to small-subgroup attacks as described in [33, p. 53], a reference cited in [28]. We therefore assume the literal interpretation (a), i.e., the adversary may register arbitrary valid public keys from the key space. This is also in line with the descriptions in [18, 24, 37].

Second, the eCK model puts no explicit restrictions on the corrupt query. For honest parties the intent of the query is clear. However, it is unclear in [28] whether the query is allowed on adversary-controlled parties on behalf of those the adversary registered a public key. Consider the following two cases for adversary-controlled parties. On the one hand, if the adversary already knows the secret key corresponding to the valid public key he registered, then the corrupt query is redundant. On the other hand, if the adversary were allowed to perform this query when he does not know the secret key of the corresponding valid public key, then no protocol would be secure in the eCK model: the adversary would be able to obtain the secret key of any honest party by simply re-registering the public key for an adversary-controlled party, and then corrupting the latter party. In particular, this gives the adversary access to the secret keys of honest parties without performing a corrupt on these honest parties, which can then be combined with an ephemeral-key query to compute the session key of the test session. The previous observations lead us to the conclusion that the query corrupt($\hat{P}$) should be defined in such a way that it returns the secret keys of party $\hat{P}$ if $\hat{P}$ is honest, and $\perp$ otherwise.

We show in Proposition 8 that our eCK^w model is stronger than the eCK model with respect to Π.

Proposition 8. *Let Π be the class of two-message KE protocols. The eCK^w model is stronger than the eCK model with respect to Π.*

Proof. The first condition of Definition 6 is satisfied since matching is defined in the same way for both models eCK^w and eCK. Let $\pi \in \Pi$. To show that the second condition of Definition 6 holds, we construct an adversary E' attacking protocol π in model eCK^w using an adversary E attacking π in eCK. In the setup phase of the eCK experiment, the adversary selects N distinct binary strings

$\hat{P}_1, \hat{P}_2, ..., \hat{P}_N$ for N honest parties. Define $\mathcal{P} = \{\hat{P}_1, \hat{P}_2, ..., \hat{P}_N\}$ for the eCK^w experiment. During the registration phase at the onset of the experiment, in case E registers valid public keys on behalf of adversary-controlled parties $\hat{L} \notin \mathcal{P}$, E' proceeds with the same registration of keys. Whenever E issues a query send, corrupt, ephemeral-key, 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 eCK$_{\text{fresh}}$ holds for the test session, then by definition eCK$^w{}_{\text{fresh}}$ also holds. In particular, if there is no matching session, then the last condition in the freshness definition of the eCK model [28, p. 9] requires that there is no corrupt of the peer, which implies the sixth condition of eCK$^w{}_{\text{fresh}}$. 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^w, there is a negligible function g such that $Adv_{E'}^{\pi}(k) \leq g(k)$. It follows that protocol π is secure in eCK. $\qquad\square$

B   Proof of Proposition 6

Proposition 6 *Under the GAP-CDH assumption in the cyclic group G of prime order p, the* NAXOS *protocol is satisfies eCK^w security, when H_1, H_2 are modeled as independent random oracles.*

Proof. Here we show that NAXOS is secure in eCK^w. We use the structure of the security proof of the CMQV protocol in [37] as it is more detailed than the proof of NAXOS in [28].

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 $\sigma_1 = \text{CDH}(Y, A), \sigma_2 = \text{CDH}(B, X)$ 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 sessions s_1, s_2 such that $r_{s_1} = r_{s_2}$ (where r_{s_1} and r_{s_2} denote the random coins drawn in sessions s_1 and s_2, respectively).
2. A_2 : there exists a sessions s such that $H_1(r_s, sk_{actor,s}) = H_1(r_{s^*}, sk_{actor,s^*})$ 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

$$P(K^c) \leq P(A_1 \vee A_2 \vee A_3 \vee A_4) \leq P(A_1) + P(A_2) + P(A_3) + P(A_4)$$

$$\leq \frac{q_s^2}{2} \frac{1}{2^k} + \frac{q_s}{p} + \frac{q_s + q_{\text{ro2}}}{2^k},$$

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 [28, 37], 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 party $\hat{C} \in \mathcal{P}$ 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 NAXOS by querying 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 $\sigma_3 = \mathrm{CDH}(X, Y)$.

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 GAP-DLog challenge be C. Suppose that event $DL \wedge K$ occurs with non-negligible probability. In this case, the simulator S chooses one party $\hat{C} \in \mathcal{P}$ 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.. 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.

---

[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.

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. ephemeral-key(s): S answers this query in the appropriate way.
4. session-key(s): S answers this query by look-up in table T.
5. test-session(s): If $s \neq s^*$, then S aborts; otherwise S answers the query in the appropriate way.
6. 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 $DLog(C) = h$.
   - 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.
   - 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 $\text{DDH}(V, U, \sigma_3) = 1$, $\text{DDH}(V, Q_i, \sigma_1) = 1$ and $\text{DDH}(U, Q_j, \sigma_2) = 1$, 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 party $\hat{C}$ for whom M queries H_1 with $(*, h)$ such that $C = g^h$ before issuing a 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 $\text{Ev} := T_O \wedge DL^c \wedge K$ into the following events $B_1, ..., B_3$ so that $\text{Ev} = B_1 \vee B_2 \vee B_3$:

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 an ephemeral-key(s') query to the origin-session s' of s^*, but may issue a corrupt(s^*_{peer}) query.
3. B_3 : Ev occurs and $s^*_{peer} \neq s'_{actor}$ and M does not issue a corrupt(s^*_{peer}) query, but may issue an ephemeral-key(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 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'. The ephemeral secret key of session s^* is denoted by $\tilde{x}_0$ and the ephemeral secret key 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. $\mathsf{ephemeral\text{-}key}(s)$: S answers this query in the appropriate way.
9. $\mathsf{session\text{-}key}(s)$: S answers this query by look-up in table T.
10. $\mathsf{test\text{-}session}(s)$: 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.
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. $\mathsf{corrupt}(\hat{P})$: 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.

---

[3] Note that, if the group check fails, the session is aborted.

- 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 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 an ephemeral-key(s^*) query before making an H_1 query that includes $\tilde{x}_0$. Similarly, M can only obtain $\tilde{y}_0$ via an ephemeral-key(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 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 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:

- ephemeral-key(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 an ephemeral-key(s^*) query before making an H_1 query that includes $\tilde{x}_0$. Under event DL^c, the adversary first issues a 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 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 party $\hat{B} \in \mathcal{P}$ at random and sets its long-term public key to B. S chooses long-term secret/public key pairs for the remaining 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'. The ephemeral secret key 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. $\mathsf{ephemeral\text{-}key}(s)$: 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 [27, 28]. 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.
□

C Proof of Proposition 7

Proposition 7 *Under the GAP-CDH assumption in the cyclic group G of prime order p, the protocol π_1-core satisfies $eCK^{passive}$ security, when KDF 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 KDF 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 random coins drawn 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 random coins (possibly leading to ephemeral-key queries).
2. A_2 : there exists a session s such that $KDF(\mathrm{input}_s) = KDF(\mathrm{input}_{s^*})$ with $\mathrm{input}_s \neq \mathrm{input}_{s^*}$.
3. A_3 : there exists an adversarial query input_M to the oracle KDF such that $KDF(\mathrm{input}_M) = KDF(\mathrm{input}_{s^*})$ 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 KDF. We have that

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

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 KDF with $(\hat{A}, \hat{B}, \sigma, X)$, where $\sigma = \mathrm{CDH}(YB, XA)$. Recall that in case there is no origin-session for the test session, the test session is not $\mathrm{eCK^{passive}}_{\mathrm{fresh}}$.

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 $\mathrm{Ev} := T_O \wedge K$ into the following events $B_1, ..., B_4$ so that $\mathrm{Ev} = B_1 \vee B_2 \vee B_3 \vee B_4$:

1. B_1 : Ev occurs and the adversary does issue neither ephemeral-key(s') nor ephemeral-key(s^*), but may issue the queries corrupt(s^*_{actor}) and corrupt(s^*_{peer}).
2. B_2 : Ev occurs and the adversary does issue neither ephemeral-key(s^*) nor corrupt(s^*_{peer}), but may issue the queries corrupt(s^*_{actor}) and ephemeral-key(s').
3. B_3 : Ev occurs and the adversary does issue neither ephemeral-key(s') nor corrupt(s^*_{actor}), but may issue the queries corrupt(s^*_{peer}) and ephemeral-key(s^*).
4. B_4 : Ev occurs and the adversary does issue neither corrupt(s^*_{actor}) nor corrupt(s^*_{peer}), but may issue the queries ephemeral-key(s') and ephemeral-key(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 $\mathrm{CDH}(X, Y) = g^{xy}$ (where $X = g^x$ and $Y = g^y$) since

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

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

Consider the following algorithm C which uses adversary E as a subroutine.

ALGORITHM C: The algorithm is given a pair (X, Y) of elements from G as an instance of the GAP-CDH 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 parties and stores the associated secret keys.

1. Run E on input 1^k and the public keys for all of the N parties.
2. send$(s^*, \hat{B})$: C sets the ephemeral public key to X and answers the query with the message X.
3. send$(s', \hat{P})$ or 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. ephemeral-key(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 KDF, 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-CDH instance with probability at least $\frac{1}{(q_s)^2}P(Q)$ which implies that $P(Q) \le (q_s)^2 Adv_C^{\mathrm{GAP\text{-}CDH}}(k)$.

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-CDH problem with probability $\frac{1}{q_s N}P(Q)$ where $P(Q)$ must be negligible since GAP-CDH problem is hard in G.

Consider the following algorithm C' which uses adversary E as a subroutine.

ALGORITHM C': The algorithm is given a pair (X, B) of elements from G as an instance of the GAP-CDH problem. C' selects one party $\hat{B}$ (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 parties 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.

1. Run E on input 1^k and the public keys for all of the N parties.
2. 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 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. ephemeral-key(s): C' answers in the appropriate way, except if $s = s^*$ in which case C' aborts with failure.
5. corrupt$(\hat{P})$: C' answers in the appropriate way, except if $\hat{P} = \hat{B}$ in which case C' aborts with failure.
6. test-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 KDF, answer it as follows:
   - If $\left\{\hat{Q}_i, \hat{Q}_j\right\} = \left\{\hat{A}, \hat{B}\right\}$, $U = X$ and DDH$(XA, YB, Z) = 1$, then C' aborts E and is successful by outputting 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. session-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-CDH instance with probability at least $\frac{1}{q_s N} P(Q)$ which implies that $P(Q) \leq q_s N Adv_{C'}^{\text{GAP-CDH}}(k)$.

The analyses of events B_3 and B_4 are similar to the previous analyses. $\square$