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 that satisfy even stronger security properties than provided by the extended Canetti-Krawczyk (eCK) security model. In particular, we consider perfect forward secrecy in the presence of adversaries that can reveal ephemeral secret keys and the long-term secret keys of the actor of a session (similar to Key Compromise Impersonation).
We propose two new game-based security models for KE protocols. First, we formalize a slightly stronger variant of the eCK security model that we call eCK^w. Second, we integrate perfect forward secrecy into eCK^w, which gives rise to the even stronger eCK-PFS model. We propose a security-strengthening transformation (i. e., a *compiler*) between our new models. Given a two-message Diffie-Hellman type protocol secure in eCK^w, our transformation yields a two-message protocol that is secure in eCK-PFS. As an example, we show how our transformation can be applied to the NAXOS protocol.

Keywords: Key Exchange, Security Models, Protocol Transformations, Perfect Forward Secrecy, Ephemeral-key reveal, Key Compromise Impersonation, Actor compromise

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, 8,15,17]. The first such security model was introduced by Bellare and Rogaway [3]. In this 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

* An extended abstract of this paper appears in ESORICS 2012.

** This work was supported by ETH Research Grant ETH-30 09-3 and the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), which is supported by the Swiss National Science Foundation.

that (b) the adversary does not learn the session-key with more than negligible probability. Building on this work, Canetti and Krawczyk [8] 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. [17] 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 learns the long-term secret keys of all the parties [21]. The designers of the eCK model argued that this property cannot be achieved by two-message KE protocols, based on [15]. In particular, in [15, 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. 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) [15]. When the long-term keys are compromised, weak perfect forward secrecy guarantees secrecy of previously established 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 [6, 10, 15, 17]. As a result, even though the eCK security model [17] guarantees only weak perfect forward secrecy, it is currently described in the literature as the strongest possible security model for two-message KE protocols [9, 17, 19].

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 [17] 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 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 on a per-session basis).

Our second contribution is a generic security-strengthening transformation (a so-called *compiler*) that contributes towards the modular design approach of KE

protocols. Given a two-message Diffie-Hellman (DH) type KE protocol that is secure in eCK^w, our transformation yields a two-message protocol that is secure in the eCK-PFS model. The transformation does not introduce additional message dependencies. 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. As an example we show that NAXOS [17], the first key exchange protocol proven secure in the eCK model, is also secure in eCK^w and use our transformation to construct a protocol that is secure in eCK-PFS. Thus, we demonstrate that it is possible for two-message and even one-round KE protocols to achieve PFS, even under actor compromise (i. e. disclosure of the long-term secret keys of the actor of a session) and leakage of ephemeral secret keys.

Related Work. The majority of related works claim that perfect forward secrecy cannot be achieved in a two-message KE protocol [6, 10, 15–17]. There are two notable exceptions. First, the two-message modified-Okamoto-Tanaka (mOT) protocol by Gennaro et al. [12] 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 [12], 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 [7], Boyd and Gonzalez 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. In Section 4 we show that, e. g., $\mathcal{C}$(NAXOS) [7] is insecure in eCK-PFS.

In [13], 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], protocol TS3 satisfies a stronger forward secrecy property than protocol TS2. Whereas protocol TS2 only achieves weak-PFS, we conjecture that protocol TS3 achieves PFS under the same assumptions as stated in [13, Theorem 3].

The eCK variant for protocols with more than two messages, defined in [16], guarantees perfect forward secrecy. However, this eCK variant cannot be met by any of the protocols from the class we are considering here because it uses the concept of matching session instead of origin-session.

Organization. In Section 2 we recall some standard definitions used in this paper. In Section 3 we motivate and define 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 eCK^w into a two-message KE protocol secure in

eCK-PFS. We show how this transformation can be applied to the NAXOS protocol in Section 5. Finally, we conclude in Section 6.

2 Preliminaries

Let $G = \langle g \rangle$ be a finite cyclic group of large prime order p with generator g.

Similar to the discrete logarithm experiment [14], 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 $\text{GAP-DLog}_{\mathcal{A},\mathcal{G}}(k)$**:**

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 [20]).** *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 [22]). *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 [14]). *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 [5]). *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 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 eCK model. The second model called eCK-PFS integrates PFS directly into eCK^w.

3.1 Motivation for the New Models

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 [15]. We briefly recall the 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). 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 [15]. Additionally, the attack also works on protocols like NAXOS [17], where z is a hash of the sender's long-term secret key and a random value. The adversary can just replace this value by an arbitrary value.

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 [17, 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*. This allows us to capture the adversary's capability of 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 though no session matching to s exists. 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 [7]), or
- leave session s' incomplete (in case session s' is in the initiator role).

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

eCK-PFS: integrating PFS into eCK^w. We 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. 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 CK-NSR model in [7] incorporating PFS, the eCK-PFS model additionally captures leakage of various combinations of ephemeral secret keys and long-term secret keys as well as perfect forward secrecy under actor compromise.

3.2 Defining eCK^w and eCK-PFS

Terminology. Let $\mathcal{P} = \left\{\hat{P}_1, \hat{P}_2, ..., \hat{P}_N\right\}$ be a finite set of N parties' identities. Each party can execute multiple instances of a KE protocol, called sessions, concurrently. We denote session i at 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 $T_s = (s_{actor}, s_{peer}, s_{role}, s_{sent}, s_{recv}) \in \mathcal{P} \times \mathcal{P} \times \{\mathcal{I}, \mathcal{R}\} \times \{0,1\}^* \times \{0,1\}^*$ (where the empty sequence in $\{0,1\}^*$ is denoted by $-$). 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.

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 [17] which is based on matching conversations.

Definition 5 (matching sessions). *Two completed sessions s and s' are said to be* matching *if*

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

To relate a message received (and accepted) by some session to the session it originates from (if the latter exists), we introduce the concept of origin-session. If an origin-session s' for some session s exists, then the messages received by session s have not been modified or injected (as in Krawczyk's PFS attack [15]) by the adversary.

Definition 6 (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.

Adversarial capabilities. Similar to the eCK model [17], we model the adversary as a probabilistic polynomial-time (PPT) Turing machine that controls all communications between parties through the following queries:

1. $\mathsf{send}(s, v)$. This query models the adversary sending message v to session s. 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. $\mathsf{corrupt}(\hat{P})$. This query reveals the long-term keys of party $\hat{P}$.
3. $\mathsf{ephemeral\text{-}key}(s)$. This query reveals 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.*

Definition 7 (Fresh session in eCK^w). *A completed session s in security experiment W is said to be* fresh in *eCK^w if all of the following conditions hold:*

1. *W does not include the query* session-key(s),
2. *for all sessions s^* such that s^* matches s, W does not include* session-key(s^*),
3. *W does not include both* corrupt(s_{actor}) *and* ephemeral-key(s),
4. *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*
5. *if there exists no origin-session for session s, then W does not include a* corrupt(s_{peer}) *query.*

Definition 8 (Fresh session in eCK-PFS). *A completed session s in experiment W is said to be* fresh in *eCK-PFS if all of the following conditions hold:*

1. *W does not include the query* session-key(s),
2. *for all sessions s^* such that s^* matches s, W does not include* session-key(s^*),
3. *W does not include both* corrupt(s_{actor}) *and* ephemeral-key(s),
4. *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*
5. *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.*

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, each party $\hat{P}$ runs a key-generation algorithm that takes as input a security parameter 1^k and outputs valid static secret/public key pair(s). The public key(s) of each party are distributed in an authenticated way to all other parties. The adversary E is given access to all public data. The setting of the security experiment W can be described in four successive stages, as follows:

1. The adversary E can perform send, corrupt, ephemeral-key, and session-key queries.
2. At some point in the experiment, E issues a test-session query to a completed session that is fresh in model M by the time the query is issued. The challenger chooses a random bit b and provides E with either the real session-key of the test-session (for $b = 0$) or a random key from the key space (for $b = 1$).
3. The adversary may continue with send, corrupt, ephemeral-key and session-key queries, without rendering the test-session un-fresh in model M.
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 $Adv_E^{\Pi}(k) = |2P(b = b') - 1|$.

Definition 9. *A key exchange protocol Π is said to be secure in model $M \in \{eCK^w, eCK\text{-}PFS\}$ if, for all PPT adversaries E, it holds that*

- *if two 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 $Adv_E^{\Pi}(k) \leq negl(k)$.*

Comparison between eCK^w and eCK-PFS. The eCK-PFS model is strictly stronger than eCK^w because it captures more attack scenarios. 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 there is no origin-session for the test-session, the adversary is not allowed to reveal the long-term secret key of the peer of the test-session in the eCK^w model. As an example, NAXOS is provably secure in eCK^w, as we show in Section 5, but insecure in eCK-PFS due to the PFS attack described in Subsection 3.1.

4 A Transformation from eCK^w to eCK-PFS

We define a class of two-message Diffie-Hellman type key exchange protocols (similar to the class of KE protocols in [7]). 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' identities, 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 1, 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 $r_{\hat{A}} \in_R S$ 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.

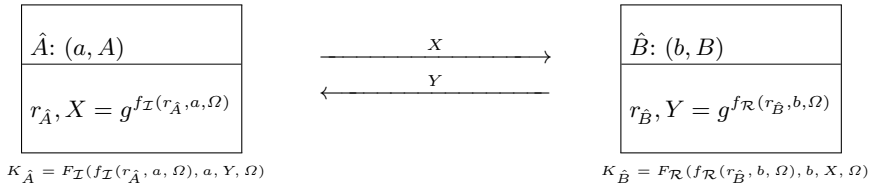

Fig. 1. A generic two-message DH type protocol

Protocol description. The generic two-message DH type protocol, depicted in Figure 1, 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. identities of peer and actor of the session) to $\hat{B}$.
   - Initialize T_s to $(\hat{A}, \hat{B}, \mathcal{I}, m, -)$, 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 10 (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 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 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 [17], NAXOS+ [19], NETS [18] and CMQV [24] belong to the class $\mathcal{DH}$-2.

Protocol transformation. We now show how to transform any protocol $\Pi \in \mathcal{DH}$-2 into a two-message protocol SIG(Π), shown in Figure 2, 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 2 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.

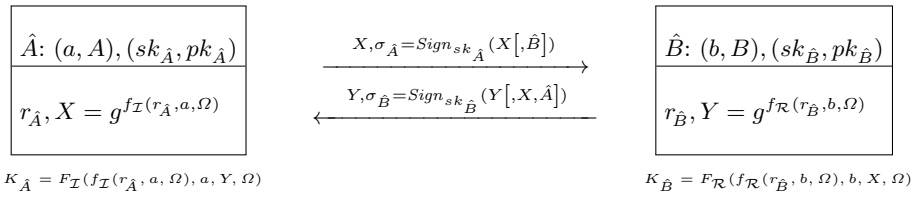

Fig. 2. A transformed generic protocol SIG(Π)

Security analysis. We show in Theorem 1 below that the SIG transformation is a security-strengthening transformation from the eCK^w model to the stronger model eCK-PFS provided that the digital signature scheme is strongly existentially unforgeable under an adaptive chosen-message attack (SUF-CMA) as well as deterministic. 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 [17]).

The following lemma is used in the proof of Theorem 1.

Lemma 1 (Difference Lemma [23]). *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. Then $|P(A) - P(B)| \leq P(F)$.*

Theorem 1. *Let $\Pi \in \mathcal{DH}$-2 be a protocol secure in the eCK^w model. Under the assumption that the signature scheme is deterministic and SUF-CMA, the protocol* SIG(Π) *is a secure key-exchange protocol in the eCK-PFS model.*

Proof. It is straightforward to verify the first condition of Definition 9, i. e., that matching sessions of protocol SIG(Π) compute the same key (since matching sessions of protocol Π compute the same key). We show next that the second condition of Definition 9 holds, i. e., the adversary 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 [23]. 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 Subsection 3.2, played by a PPT adversary E against the protocol SIG(Π).

Game 1 [Transition based on a small failure event] Let $Coll_{\mathrm{SIG}(\Pi)}$ be the small failure event that a collision for protocol SIG(Π) occurs (e.g. in ephemeral secret keys). As soon as event $Coll_{\mathrm{SIG}(\Pi)}$ 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 $Coll_{\mathrm{SIG}(\Pi)}$ occurs for the first time. The Difference Lemma yields that $|P(S_0) - P(S_1)| \leq P(Coll_{\mathrm{SIG}(\Pi)})$. Hence,

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

Game 2 [Transition based on a large failure event (see [6, 11])] Before the adversary E starts the attack game, the challenger chooses a random value $m \in_R \{1, 2, ..., q_s\}$. The m-th session activated by E, denoted by s^*, is the session on which the challenger wants the adversary to be tested. Let T be the event that the test-session is not session s^*. If event T occurs, then the attack game halts and the adversary outputs a random bit.

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

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

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

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

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

1. If E issues a $\mathsf{corrupt}(\hat{P})$ query before the completion of session s^*, then this query would render session s^* un-fresh. This would have caused Game 2 to abort since session s^* would not be the test-session. Recall that the $\mathsf{test\text{-}session}$ query can only be issued to a session that is fresh by the time the query is issued. Hence this case can be excluded.
2. If E does not issue a $\mathsf{corrupt}(\hat{P})$ 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, $P(F) \leq N Adv^{Sign}_M(k)$, where $Adv^{Sign}_M(k)$ 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 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 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 $\mathsf{send}(z, \hat{Q})$ query to activate session z with peer $\hat{Q} \in \mathcal{P}$, then answer it as follows.
   - If $z_{actor} \neq \hat{P}$, then choose $x \in_R \mathbb{Z}_p$ to get $X = g^x$, compute the signature σ on message $m = (X[, \hat{Q}])$ on behalf of z_{actor} and return the message (X, σ) to E.
   - If $z_{actor} = \hat{P}$, then choose $x \in_R \mathbb{Z}_p$ to get $X = g^x$ and query the signature oracle on message $m = (X[, \hat{Q}])$ which returns the signature σ on message m. Store the pair (m, σ) in a table L, initially empty, and return the message (X, σ) to E.
3. If E issues a $\mathsf{send}(z, \hat{Q}, m)$ query to activate session z, then answer it as follows. First check whether message m is of the form (X, σ) for some $X \in G$ and σ a valid signature on message $(X[, z_{actor}])$ with respect to the public key of $\hat{Q}$. If the checks succeed, then:
   - If $z_{actor} \neq \hat{P}$, then choose $y \in_R \mathbb{Z}_p$ to get $Y = g^y$, compute the signature σ on message $m = (Y[, X, \hat{Q}])$ on behalf of z_{actor} and return the message (Y, σ) to E.
   - If $z_{actor} = \hat{P}$, then choose $y \in_R \mathbb{Z}_p$ to get $Y = g^x$ 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 $\mathsf{corrupt}(\hat{P}_i)$, where $\hat{P}_i \in \mathcal{P}$ and $\hat{P}_i \neq \hat{P}$, are answered in the appropriate way since M knows the secret key pairs of the parties $\hat{P}_i \neq \hat{P}$.
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 abortion as in Step 8 does 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 parties, that is, $P(F) \leq NAdv_M^{Sign}(k)$.

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

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 $Adv_E^{\mathrm{SIG}(\Pi),\text{Game } 3,O}(k)$ 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^w succeeding in Game 3 against protocol Π with non-negligible advantage using adversary E as a subroutine. Moreover, it holds that $Adv_E^{\mathrm{SIG}(\Pi),\text{Game } 3,O}(k) \leq Adv_{E'}^{\Pi,\text{Game } 3,O}(k)$.

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^w 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.
2. When E issues a $\mathsf{corrupt}(\hat{P})$ 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 $\mathsf{ephemeral\text{-}key}$ or a $\mathsf{session\text{-}key}$ 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 $\mathsf{send}(z, \hat{Q})$ 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}.

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

- If E issues a $\mathsf{send}(z, \hat{Q}, m)$ 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 $\mathsf{send}(z, W)$ 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 $\mathsf{send}(z, m)$ 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 $\mathsf{send}(z, V)$ to session z.

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

5. In case E issues the $\mathsf{test\text{-}session}$ query to session s^*, E' issues the $\mathsf{test\text{-}session}$ 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.

Thus, it holds that $Adv_E^{\mathrm{SIG}(\Pi),\text{Game } 3,O}(k) \leq Adv_{E'}^{\Pi,\text{Game } 3,O}(k)$.

Finally,

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

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

□

Remark 1. Let M^w and M-PFS be the security models obtained from eCK^w and eCK-PFS (respectively) by removing the $\mathsf{ephemeral\text{-}key}$ query 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 M^w, the transformed protocol $\mathrm{SIG}(\Pi)$ is secure in M-PFS using either a deterministic or a randomized SUF-CMA signature scheme.

Remark 2. In contrast to the SIG transformation, the MAC transformation $\mathcal{C}$ suggested in [7] applied to any protocol $\pi \in \mathcal{DH}$-2 does not yield a two-message key-exchange protocol secure in eCK-PFS since the transformed protocol is vulnerable to an attack that combines revealing the long-term secret keys of the actor of the test-session with revealing the long-term secret keys of the peer of the test-session after its completion. More precisely, an attacker can impersonate the peer of the test-session by first revealing the long-term secret keys of the actor (which allows him to create valid MACs on messages of his choice) and after the completion of the test-session revealing the long-term secret keys of the peer. For example, this attack shows that $\mathcal{C}$(NAXOS) [7] is insecure in eCK-PFS.

5 NAXOS Revisited

The NAXOS protocol [17], shown in Figure 3, 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 1, 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 $F_{\mathcal{R}}(f_{\mathcal{R}}(r_{\hat{B}}, b, \Omega), b, X, \Omega) = H_2(A^{H_1(r_{\hat{B}},b)}, X^b, X^{H_1(r_{\hat{B}},b)}, \hat{A}, \hat{B})$.

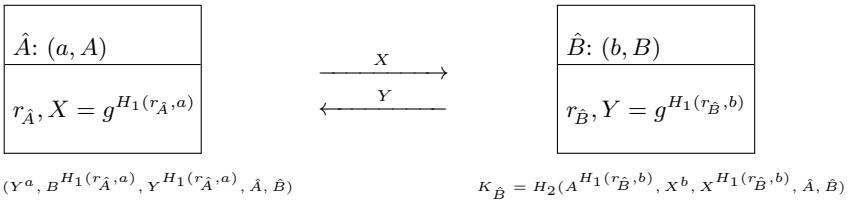

Fig. 3. NAXOS protocol [17]

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

Proposition 1. *Under the GAP-CDH assumption in the cyclic group G of prime order p,* NAXOS *is secure in the eCK^w model, when H_1 and H_2 are modeled as independent random oracles.*

In contrast to the proof of NAXOS in the eCK model [17], the proof of Proposition 1 distinguishes between the cases whether or not an origin-session (instead of a matching session) exists for the test-session.

Proof (Sketch). Similar to [17, 24], we analyze 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 that there exists a party $\hat{C}$ such that the adversary M, during its execution, queries H_1 with $(*, c)$ before issuing a $\mathsf{corrupt}(\hat{C})$ query and K denotes the event that M wins the security experiment against NAXOS by querying H_2 with $(\sigma_1, \sigma_2, \sigma_3, \hat{A}, \hat{B})$, where $\sigma_1 = CDH(Y, A), \sigma_2 = CDH(B, X)$ and $\sigma_3 = CDH(X, Y)$ given that the test-session is s^* with $T_{s^*} = (\hat{A}, \hat{B}, \mathcal{I}, X, Y)$. □

Applying the SIG transformation on the NAXOS protocol yields the protocol SIG(NAXOS), depicted in Figure 4. Combining Proposition 1 with Theorem 1, we obtain the following result.

Corollary 1. *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 is secure in the eCK-PFS model, when H_1, H_2 are modeled as independent random oracles.*

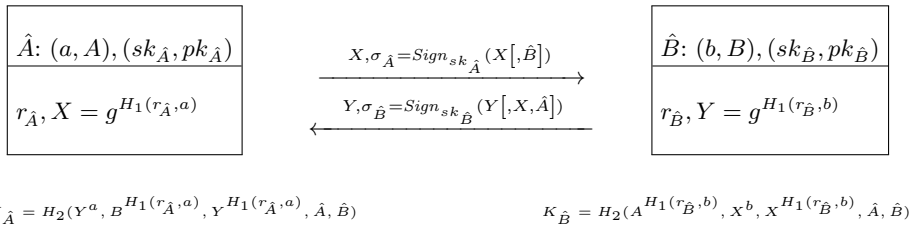

Fig. 4. SIG(NAXOS) protocol

6 Conclusions

We provided two new eCK-like security notions, namely eCK^w and eCK-PFS. The eCK^w model slightly strengthens eCK by a more precise modeling of weak-PFS. The stronger eCK-PFS notion guarantees PFS, even in the presence of eCK-like adversaries. Proving security in eCK-PFS provides strictly more guarantees than separately proving eCK^w-security and PFS. Existing two-message KE protocols such as CMQV [24], NAXOS [17], or $\mathcal{C}$(NAXOS) [7] 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^w into a two-message protocol secure in eCK-PFS. As our transformation does not introduce message dependencies, it also allows turning a one-round protocol secure in eCK^w into a one-round protocol secure in eCK-PFS. As future work, we would like to specify further transformations on KE protocols that are based on the newly developed security models in this work. It remains an open question whether there exist more efficient transformations that yield two-message KE protocols secure in eCK-PFS.

Acknowledgments. We would like to thank Colin Boyd for constructive comments on an earlier version of this work.

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. 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.
6. C. Boyd, Y. Cliff, J.M. Gonzalez Nieto, and K.G. Paterson. One-round key exchange in the standard model. *Int. J. Applied Cryptography*, 1:181–199, 2009.
7. C. Boyd and J. Gonzalez. On Forward Secrecy in One-Round Key Exchange. In *13th IMA International Conference, IMACC 2011*, volume 7089 of *LNCS*, pages 451–468. Springer, 2011.
8. 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.
9. 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.
10. 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.
11. A.W. Dent. A note on game-hopping proofs. Cryptology ePrint Archive, Report 2006/260, 2006. `http://eprint.iacr.org/2006/260`.
12. 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.
13. 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`.
14. J. Katz and Y. Lindell. *Introduction to Modern Cryptography*. Chapman Hall/CRC, 2008.
15. 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.
16. 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/`.
17. 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.
18. 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/`.
19. 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/`.
20. 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.
21. A. Menezes, P. van Oorschot, and S. Vanstone. Handbook of Applied Cryptography, October 1996.

22. 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.
23. V. Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2006. `http://eprint.iacr.org/`.
24. 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 Proof of Proposition 1

Proposition 1 *Under the GAP-CDH assumption in the cyclic group G of prime order p, the* NAXOS *protocol is secure in the eCK^w model, 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 [24] as it is more detailed than the proof of NAXOS in [17].

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 $\sigma_1 = CDH(Y, A), \sigma_2 = CDH(B, X)$ and $\sigma_3 = CDH(X, Y)$.
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 exist two sessions s_1, s_2 such that $H_1(r_{s_1}, sk_{actor,s_1}) = H_1(r_{s_2}, sk_{actor,s_2})$ and $r_{s_1} \neq r_{s_2}$.
3. A_3 : there exists a session s' such that $H_2(\text{input}_{s'}) = H_2(\text{input}_{s^*})$ with $\text{input}_{s'} \neq \text{input}_{s^*}$.
4. A_4 : there exists an adversarial query input_M to the oracle H_2 such that $H_2(\text{input}_M) = H_2(\text{input}_{s^*})$ with $\text{input}_M \neq \text{input}_{s^*}$.

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

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

which is a negligible function of the security parameter k.

In the subsequent events (and their analyses) we assume that no collisions in the queries to the oracle H_1 occur and that none of the events $A_1, ..., A_4$ occurs. Similar to [17, 24], 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 $\hat{C}$ such that the adversary M, during its execution, queries H_1 with $(*, c)$ before issuing a $\mathsf{corrupt}(\hat{C})$ query and K denotes the event that M wins the security experiment against NAXOS by querying H_2 with $(\sigma_1, \sigma_2, \sigma_3, \hat{A}, \hat{B})$, where $\sigma_1 = CDH(Y, A), \sigma_2 = CDH(B, X)$ and $\sigma_3 = CDH(X, Y)$.

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. Since it might be possible that a redirect event occurs such that the actor of the origin-session for the test-session is different to the peer of the test-session, the study of this event applies to any party $\hat{C} \in \mathcal{P}$ (not only to $\hat{A}$ or $\hat{B}$).

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 $\hat{C}$ at random and sets its long-term public key to C. S chooses long-term secret/public key pairs for the remaining 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 $\left(\hat{P}_i, \hat{P}_j, r, U, V, \lambda\right) \in \mathcal{P} \times \mathcal{P} \times \{\mathcal{I}, \mathcal{R}\} \times G \times G \times \{0,1\}^k$ in a table T, initially empty. Upon completion of session s with $T_s = \left(\hat{P}_i, \hat{P}_j, \mathcal{I}, U, V\right)$, S does the following:
   - If there exists an entry $\left(\hat{P}_j, \hat{P}_i, \mathcal{R}, V, U, \lambda\right)$ in table T, then S stores $\left(\hat{P}_i, \hat{P}_j, \mathcal{I}, U, V, \lambda\right)$ in table 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.

- Else if there exists an entry $\left(\sigma_1, \sigma_2, \sigma_3, \hat{P}_i, \hat{P}_j, \lambda\right)$ in table L, for some $\lambda \in \{0,1\}^k$, such that $DDH(V, U, \sigma_3) = 1$ and
  - $V^{sk_{\hat{P}_i}} = \sigma_1$ (in case $\hat{P}_i \neq \hat{C}$) or $DDH(V, P_i, \sigma_1) = 1$ (in case $\hat{P}_i = \hat{C}$), and
  - $U^{sk_{\hat{P}_j}} = \sigma_2$ (in case $\hat{P}_j \neq \hat{C}$) or $DDH(U, P_j, \sigma_2) = 1$ (in case $\hat{P}_j = \hat{C}$),

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

The session-key of a completed session s with $T_s = \left(\hat{P}_j, \hat{P}_i, \mathcal{R}, V, U\right)$ 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 $\left(\sigma_1, \sigma_2, \sigma_3, \hat{P}_i, \hat{P}_j, \lambda\right) \in G \times G \times G \times \mathcal{P} \times \mathcal{P} \times \{0,1\}^k$ in a table L, initially empty. When M makes a query of the form $\left(\sigma_1, \sigma_2, \sigma_3, \hat{P}_i, \hat{P}_j\right)$ to the random oracle for H_2, answer it as follows:
   - If $\left(\sigma_1, \sigma_2, \sigma_3, \hat{P}_i, \hat{P}_j, \lambda\right) \in L$ for some $\lambda \in \{0,1\}^k$, then S returns λ to M.
   - Else if there exist entries $\left(\hat{P}_i, \hat{P}_j, \mathcal{I}, U, V, \lambda\right)$ or $\left(\hat{P}_j, \hat{P}_i, \mathcal{R}, V, U, \lambda\right)$ in table T, for some $\lambda \in \{0,1\}^k$ and $U, V \in G$, such that $DDH(V, U, \sigma_3) = 1$ and
     - $V^{sk_{\hat{P}_i}} = \sigma_1$ (in case $\hat{P}_i \neq \hat{C}$) or $DDH(V, P_i, \sigma_1) = 1$ (in case $\hat{P}_i = \hat{C}$), and
     - $U^{sk_{\hat{P}_j}} = \sigma_2$ (in case $\hat{P}_j \neq \hat{C}$) or $DDH(U, P_j, \sigma_2) = 1$ (in case $\hat{P}_j = \hat{C}$),

     , then S returns λ to M and stores the entry $\left(\sigma_1, \sigma_2, \sigma_3, \hat{P}_i, \hat{P}_j, \lambda\right)$ in table L.
   - Else, S chooses $\mu \in_R \{0,1\}^k$, returns it to M and stores the entry $\left(\sigma_1, \sigma_2, \sigma_3, \hat{P}_i, \hat{P}_j, \mu\right)$ 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 $\frac{1}{q_s}$. 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 $\mathsf{corrupt}(\hat{C})$ query. In this case, S is successful as described in Step 7 and does not abort in Steps 6 and 9. Hence, if event $DL \wedge K$ occurs, then the success probability of S is given by $P(S) \geq \frac{1}{Nq_s} P(DL \wedge K)$.

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

1. B_1 : Ev occurs and $s^*_{peer} = s'_{actor}$.
2. B_2 : Ev occurs and $s^*_{peer} \neq s'_{actor}$ and M does not issue an $\mathsf{ephemeral\text{-}key}(s')$ query to the origin-session s' of s^*, but may issue a $\mathsf{corrupt}(s^*_{peer})$ query.
3. B_3 : Ev occurs and $s^*_{peer} \neq s'_{actor}$ and M does not issue a $\mathsf{corrupt}(s^*_{peer})$ query, but may issue an $\mathsf{ephemeral\text{-}key}(s')$ 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 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 $\left(\hat{P}_i, \hat{P}_j, r, U, V, \lambda\right) \in \mathcal{P} \times \mathcal{P} \times \{\mathcal{I}, \mathcal{R}\} \times G \times G \times \{0,1\}^k$ in a table T, initially empty. Upon completion of session s with $T_s = \left(\hat{P}_i, \hat{P}_j, \mathcal{I}, U, V\right)$, S does the following:
   - If there exists an entry $\left(\hat{P}_j, \hat{P}_i, \mathcal{R}, V, U, \lambda\right)$ in table T, then S stores $\left(\hat{P}_i, \hat{P}_j, \mathcal{I}, U, V, \lambda\right)$ in table T.

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

- Else if there exists an entry $\left(\sigma_1, \sigma_2, \sigma_3, \hat{P}_i, \hat{P}_j, \lambda\right)$ in table L, for some $\lambda \in \{0,1\}^k$, such that $V^{sk_{\hat{P}_i}} = \sigma_1$, $U^{sk_{\hat{P}_j}} = \sigma_2$ and $DDH(V, U, \sigma_3) = 1$, then S stores $\left(\hat{P}_i, \hat{P}_j, \mathcal{I}, U, V, \lambda\right)$ in table T.
- Else, S chooses $\mu \in_R \{0,1\}^k$, and stores the entry $\left(\hat{P}_i, \hat{P}_j, \mathcal{I}, U, V, \mu\right)$ in T.

The session-key of a completed session s with $T_s = \left(\hat{P}_j, \hat{P}_i, \mathcal{R}, V, U\right)$ is determined and stored similarly.

8. ephemeral-key(s): S answers this query in the appropriate way.
9. session-key(s): S answers this query by look-up in table T.
10. test-session(s): If $s \neq s^*$ or if s' is not the origin-session for session s^*, then S aborts; otherwise S answers the query in the appropriate way.
11. $H_1(r_{\hat{C}}, c)$: S simulates a random oracle in the usual way except if $\hat{C} = \hat{A}$ (i.e. $c = a$) and $r_{\hat{C}} = \tilde{x}_0$ or if $\hat{C} = \hat{B}$ (i.e. $c = b$) and $r_{\hat{C}} = \tilde{y}_0$, in which case S aborts with failure.
12. corrupt($\hat{P}$): S answers this query in the appropriate way.
13. S stores entries of the form $\left(\sigma_1, \sigma_2, \sigma_3, \hat{P}_i, \hat{P}_j, \lambda\right) \in G \times G \times G \times \mathcal{P} \times \mathcal{P} \times \{0,1\}^k$ in a table L, initially empty. When M makes a query of the form $\left(\sigma_1, \sigma_2, \sigma_3, \hat{P}_i, \hat{P}_j\right)$ to the random oracle for H_2, answer it as follows:
    - If $\left\{\hat{P}_i, \hat{P}_j\right\} = \left\{\hat{A}, \hat{B}\right\}$, $\sigma_1 = Y_0^a$, $\sigma_2 = X_0^b$ and $DDH(X_0, Y_0, \sigma_3) = 1$, then S aborts M and is successful by outputting $CDH(X_0, Y_0) = \sigma_3$.
    - Else if $\left(\sigma_1, \sigma_2, \sigma_3, \hat{P}_i, \hat{P}_j, \lambda\right) \in L$ for some $\lambda \in \{0,1\}^k$, then S returns λ to M.
    - Else if there exist entries $\left(\hat{P}_i, \hat{P}_j, \mathcal{I}, U, V, \lambda\right)$ or $\left(\hat{P}_j, \hat{P}_i, \mathcal{R}, V, U, \lambda\right)$, for some $\lambda \in \{0,1\}^k$ and $U, V \in G$, such that $V^{sk_{\hat{P}_i}} = \sigma_1$, $U^{sk_{\hat{P}_j}} = \sigma_2$ and $DDH(V, U, \sigma_3) = 1$ in table T, then S returns λ to M and stores the entry $\left(\sigma_1, \sigma_2, \sigma_3, \hat{P}_i, \hat{P}_j, \lambda\right)$ in table L.
    - Else, S chooses $\mu \in_R \{0,1\}^k$, returns it to M and stores the entry $\left(\sigma_1, \sigma_2, \sigma_3, \hat{P}_i, \hat{P}_j, \mu\right)$ 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 $\frac{1}{(q_s)^2}$. 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 $CDH(X_0, Y_0)$, 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. Step 11 needs to be replaced by the following:

- $H_1(r_{\hat{C}}, c)$: 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 $\mathsf{ephemeral\text{-}key}(s^*)$ query before making an H_1 query that includes $\tilde{x}_0$. Under event DL^c, the adversary first issues a $\mathsf{corrupt}(\hat{P})$ query to 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 K, except with negligible probability of guessing $CDH(X_0, Y_0)$, 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 $\hat{B}$ at random and sets its long-term public key to B. S chooses long-term secret/public key pairs for the remaining parties 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 $\left(\hat{P}_i, \hat{P}_j, r, U, V, \lambda\right) \in \mathcal{P} \times \mathcal{P} \times \{\mathcal{I}, \mathcal{R}\} \times G \times G \times \{0,1\}^k$ in a table T, initially empty. Upon completion of session s with $T_s = \left(\hat{P}_i, \hat{P}_j, \mathcal{I}, U, V\right)$, 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 $\left(\sigma_1, \sigma_2, \sigma_3, \hat{P}_i, \hat{P}_j, \lambda\right) \in G \times G \times G \times \mathcal{P} \times \mathcal{P} \times \{0,1\}^k$ in a table L, initially empty. When M makes a query of the form $\left(\sigma_1, \sigma_2, \sigma_3, \hat{P}_i, \hat{P}_j\right)$ to the random oracle for H_2, answer it as follows:
    - If $\left\{\hat{P}_i, \hat{P}_j\right\} = \left\{\hat{A}, \hat{B}\right\}$, $\sigma_1 = A^{H_1(r_{s'}, sk_{s'_{actor}})}$, $DDH(X_0, B, \sigma_2) = 1$ and $\sigma_3 = X_0^{H_1(r_{s'}, sk_{s'_{actor}})}$, then S aborts M and is successful by outputting $CDH(X_0, B) = \sigma_2$.
    - Else if $\left(\sigma_1, \sigma_2, \sigma_3, \hat{P}_i, \hat{P}_j, \lambda\right) \in L$ for some $\lambda \in \{0,1\}^k$, then S returns λ to M.
    - Else if there exist entries $\left(\hat{P}_i, \hat{P}_j, \mathcal{I}, U, V, \lambda\right)$ or $\left(\hat{P}_j, \hat{P}_i, \mathcal{R}, V, U, \lambda\right)$ in table T, for some $\lambda \in \{0,1\}^k$ and $U, V \in G$, such that $DDH(V, U, \sigma_3) = 1$ and
        - $V^{sk_{\hat{P}_i}} = \sigma_1$ (in case $\hat{P}_i \neq \hat{B}$) or $DDH(V, P_i, \sigma_1) = 1$ (in case $\hat{P}_i = \hat{B}$), and
        - $U^{sk_{\hat{P}_j}} = \sigma_2$ (in case $\hat{P}_j \neq \hat{B}$) or $DDH(U, P_j, \sigma_2) = 1$ (in case $\hat{P}_j = \hat{B}$),

      , then S returns λ to M and stores the entry $\left(\sigma_1, \sigma_2, \sigma_3, \hat{P}_i, \hat{P}_j, \lambda\right)$ in table L.
    - Else, S chooses $\mu \in_R \{0,1\}^k$, returns it to M and stores the entry $\left(\sigma_1, \sigma_2, \sigma_3, \hat{P}_i, \hat{P}_j, \mu\right)$ 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 $\frac{1}{(q_s)^2}$. 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 $P(S) \geq \frac{1}{N(q_s)^2} P(B_3)$.

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 [16,17]. Note that consistency between $\mathsf{session\text{-}key}$ and H_2 queries as well as between send and H_1 queries is kept in a similar way as in the first proof step. □