One-round Strongly Secure Key Exchange with Perfect Forward Secrecy and Deniability

Version 3, October 26, 2011[⋆]

Cas Cremers and Michèle Feltz

Department of Computer Science
ETH Zurich, Switzerland

Abstract. Traditionally, secure one-round key exchange protocols in the PKI setting have either achieved perfect forward secrecy, or forms of deniability, but not both. On the one hand, achieving perfect forward secrecy against active attackers seems to require some form of authentication of the messages, as in signed Diffie-Hellman style protocols, that subsequently sacrifice deniability. On the other hand, using implicit authentication along the lines of MQV and descendants sacrifices perfect forward secrecy in one round and achieves only weak perfect forward secrecy instead.
We show that by reintroducing signatures, it is possible to satisfy both a very strong key-exchange security notion, which we call eCK-PFS, as well as a strong form of deniability, in one-round key exchange protocols. Our security notion for key exchange is stronger than, e. g., the extended-CK model, and captures perfect forward secrecy. Our notion of deniability, which we call peer-and-time deniability, is stronger than that offered by, e. g., the SIGMA protocol.
We propose a concrete protocol and prove that it satisfies our definition of key-exchange security in the random oracle model as well as peer-and-time deniability. The protocol combines a signed-Diffie-Hellman message exchange with an MQV-style key computation, and offers a remarkable combination of advanced security properties.

Keywords: Key Exchange, Perfect Forward Secrecy, Deniability, PKI

1 Introduction

We consider the problem of key exchange in the PKI setting. Numerous protocols have been proposed in this context, which can be classified into two main categories: those that explicitly provide authentication, e. g., by using signatures or additional authenticating message flows, and those that only implicitly provide authentication, e. g., by involving the participants' private keys in the key derivation. An example from the first category is the modified STS-protocol [4]:

$$
\begin{array}{llcl}
 & \hat{A} & & \hat{B} \\
1. & x \in_R \mathbb{Z}_q \; ; \; X := g^x & \xrightarrow{\quad X \quad} & y \in_R \mathbb{Z}_q \; ; \; Y := g^y \\
2. & \text{verify signature} & \xleftarrow{\; Y, Sign_{sk_{\hat{B}}}(Y, X, \hat{A}) \;} & K_{\hat{B}} := X^y \\
3. & K_{\hat{A}} := Y^x & \xrightarrow{\; Sign_{sk_{\hat{A}}}(X, Y, \hat{B}) \;} & \text{verify signature}
\end{array}
$$

[⋆] Important differences between the versions are summarized in Appendix D.

The core of the protocol is a basic Diffie-Hellman key exchange. Additionally, the protocol ensures authentication of the exchanged messages by adding signatures, which include the peer's identity to prevent unknown-key share (UKS) attacks.

The modified-STS protocol ensures perfect forward secrecy (PFS). This security property means that the compromise of long-term secret keys does not compromise past session keys [24]. However, the protocol requires three messages and lacks desirable security features, such as deniability: The signatures prove that $\hat{A}$ intended to talk to $\hat{B}$, and vice versa. If $\hat{A}$ is malicious, she can even replace x by a digest of today's newspaper, which allows her to prove that $\hat{B}$ was willing to communicate with her after a certain date.

In contrast, protocols from the second category only authenticate implicitly, and delay the authentication to the key derivation phase. This allows for the construction of very efficient one-round protocols. These protocols offer deniability of the message exchange and aim at providing strong security properties, such as resilience against UKS attacks and leakage of ephemeral keys (i. e., the random coins drawn by the parties). A basic example is the MQV protocol [21], shown below. Here, $G = \langle g \rangle$ denotes a subgroup of $\mathbb{Z}_q^*$ of prime order p where $p|q-1$ for some large prime q, $\overline{X}$ is defined as $X \bmod 2^w + 2^w$, and a and b denote $\hat{A}$'s and $\hat{B}$'s long-term private keys, respectively. The corresponding long-term public keys are $A = g^a$ and $B = g^b$.

$$\begin{array}{lccc} & \hat{A} & & \hat{B} \\ 1. & x \in_R \mathbb{Z}_q\ ;\ X := g^x & \xrightarrow{\quad X \quad} & \text{verify that } X^p \bmod q = 1 \\ 2. & \text{verify that } Y^p \bmod q = 1 & \xleftarrow{\quad Y \quad} & y \in_R \mathbb{Z}_q\ ;\ Y := g^y \\ & K_{\hat{A}} := (YB^{\overline{Y}})^{x+a\overline{X}} & & K_{\hat{B}} := (XA^{\overline{X}})^{y+b\overline{Y}} \end{array}$$

Due to the lack of message origin authentication, a recipient must check that the received element X belongs to the group G (that is, $X^p \bmod q = 1$) and that $X \neq 1$ to prevent attacks such as small subgroup attacks.

However, implicitly authenticated one-round protocols that are similar to MQV do not guarantee perfect forward secrecy, as witnessed by the attack described by Krawczyk [19]. Assume that Alice starts the protocol, trying to talk to Bob. The adversary intercepts her message, and generates his own value y, and sends g^y to Alice. Next, Alice computes the session key and sends her confidential message to Bob, encrypted with the session key. If at any point in the future, the adversary manages to learn Bob's long-term secret key, he can decrypt Alice's message. This generic attack shows the impossibility of achieving perfect forward secrecy for a class of implicitly authenticated one-round protocols.

To prove a slightly weaker notion of forward secrecy for the HMQV protocol, Krawczyk introduces the notion of *weak* perfect forward secrecy [19]. 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. Hence, the strong level of deniability, which is achieved by implicitly-authenticated key-exchange protocols such as (H)MQV, is achieved at the expense of important security guarantees.

The desire to efficiently achieve both perfect forward secrecy and deniability has been first addressed in the context of "off-the-record messaging" by Borisov, Goldberg and Brewer in [6]. They suggest an instant-messaging protocol (OTR) that relies on an authenticated Diffie-Hellman key-exchange protocol using digital signatures. However, in [26], Di Raimondo, Gennaro and Krawczyk show that the OTR protocol suffers from a serious security flaw, namely it admits an unknown-key share attack. Fixing the flaw by including the identity of the intended peer within the signed messages, as suggested in [26], significantly weakens the level of deniability.

In this work, we propose a protocol that combines the message exchange of a signed Diffie-Hellman variant with an MQV-style key computation. The signatures offer two main benefits. First, they allow us to establish perfect forward secrecy in one round contradicting common belief. Second, message origin authentication is achieved through the signatures on the exchanged ephemeral public keys and the group element check on received data.

It may seem at first glance that the use of signatures has drastic consequences on the level of deniability of our protocol. However, we show that it is possible to establish a very strong form of deniability, which we call peer-and-time deniability. In practice, this means that participants cannot deny that they may have participated in partial protocol runs with some party at some time in the past, in either the initiator or responder role. But they can deny any more specific allegations.

Thus, our protocol achieves the best of two worlds for one-round authenticated key-exchange protocols using explicit signatures, that is, perfect forward secrecy defined within a remarkably strong indistinguishability-based security model as well as a high well-defined level of deniability.

Contributions. First, we define a new security notion for key-exchange protocols, which we call eCK-PFS. The eCK-PFS model is a strengthening of the extended-CK model [20] and additionally requires (full) perfect forward secrecy.

Second, we introduce a strong notion of deniability, called peer-and-time deniability. This notion is strictly weaker than full deniability [27], but stronger than the deniability offered by protocols such as SIGMA [18]. Peer-and-time deniability allows parties to deny communicating with particular peers as well as being alive at specific times.

Third, we propose a one-round key exchange protocol, which combines signatures on self-generated data with an MQV-style key computation. We prove that the protocol satisfies our notion of key-exchange security as well as peer-and-time deniability. To the best of our knowledge, our key-exchange protocol is the first to satisfy this strong key-exchange security property, even without considering deniability.

Organization. We recall standard definitions in Section 2. In Section 3 we define our notion of key exchange (KE) security, and introduce peer-and-time deniability in Section 4. We propose a concrete protocol and discuss design choices in Section 5. We sketch proofs of the deniability and KE security of our protocol in Section 6. We discuss the efficiency of our protocol in Section 7 and related work in Section 8. Finally, we conclude in Section 9. We give detailed proofs of our results in the appendix.

2 Preliminaries

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

Definition 1 (CDH-Assumption). *The computational Diffie-Hellman assumption (CDH) 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}.*

Definition 2 (GAP-CDH-Assumption). *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 $w = uv \bmod p$).*

We denote by $Adv_C^{GAP-CDH}$ the probability of a probabilistic polynomial time (PPT) adversary C to break the $GAP-CDH$ assumption in G. If the $GAP-CDH$ assumption in G holds, then $Adv_C^{GAP-CDH}$ is negligible for any PPT adversary C.

Definition 3 (Signature Scheme [16]). *A* signature scheme *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 public/private key pair (pk, sk).*
2. *The (possibly probabilistic) signing algorithm $Sign$ takes as input a private 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 (EU-CMA [16]). *A signature scheme $\Sigma = (Gen, Sign, Vrfy)$ is* 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 on a message which has not been previously queried to a signing oracle (returning a signature for any message of the adversary's choice).*

Key exchange terminology. Key exchange protocols are specified as a set of *roles*. In particular, for two-party protocols as considered here, there is an *initiator* role and a *responder* role. Roles are performed by *parties* such as $\hat{A}$ or $\hat{B}$. Each party may execute multiple roles, and even multiple instances of each role, concurrently. We refer to a particular instance of a role at a party as a *session*.

3 Key exchange security notion: eCK-PFS

We define a security notion for key exchange (KE) protocols, which we refer to as eCK-PFS. Our work builds on Bellare-Rogaway style indistinguishability-based security notions for key exchange protocols [3]. As illustrated in Appendix A, our KE-security notion is stronger than the extended-CK security notion [20], that is, it captures strictly more attack scenarios than the extended-CK model. Instead of only requiring weak perfect forward secrecy (as in extended-CK), we require (full) perfect forward secrecy.

We associate to each session a unique session identifier. The session identifier of a session s is defined as a quintuple $(s_{actor}, s_{peer}, s_{send}, s_{recv}, s_{role})$, where s_{actor}, s_{peer} denote the identities of the owner and intended peer of the session s, $s_{role} \in \{i \text{ (initiator)}, r \text{ (responder)}\}$ denotes the role that the session is executing, and s_{send}, s_{recv} are sequences of timely ordered messages as sent/received by s_{actor} during session s. For incomplete sessions, these sequences are defined as the messages sent/received so far. From now on, we assume that each session is represented by its session identifier.

In order to specify functional requirements (e. g., two parties can successfully establish a shared key in the absence of adversaries) as well as security requirements (e. g., keys from different sessions should be independent) we need to specify when two sessions are intended to be communication partners. In KE security notions this is commonly done by defining a notion of *matching*. Here we adopt the matching sessions definition from the extended-CK model [20].

Definition 5 (matching sessions for two-message protocols). *Two completed sessions s and s' are said to be matching, denoted by $s \sim s'$, if the following condition holds*

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

In early models, the definition of matching based only on the exchanged messages [3], but for protocols that offer, e. g., identity protection, the names of the involved parties cannot be inferred from the messages although they clearly play a role when defining the intended communication partner. Hence they are explicit in the above definition. For similar reasons, the role is included in the definition.

To relate a received message that was not constructed by the adversary to the session it originates from, we introduce the concept of origin-session.

Definition 6 (origin-session). *We say that a (possibly incomplete) session s' is the origin-session for a completed session s when*

$$s'_{send} = s_{recv} \wedge s_{peer} = s'_{actor}.$$

In other works the previous two definitions are collapsed into a single matching predicate, by extending the definition of matching to also cover incomplete sessions. However, for clarity we choose to keep these concepts separate.

As we will see later in more detail, the key exchange security notion is defined as a game in which the adversary interacts with the parties. The adversary attempts to distinguish a real session key from a random one, and can activate parties to start sessions and he can send messages to parties. The parties are restricted to executing the roles. Any message sent by the parties is learned by the adversary. This interaction is modeled by an experiment in which the adversary can perform the following operations, known as *queries*.

1. send(s, v). This query models the adversary sending message v to an active session s, or initiating a session, via a send$(s,$"initiate"$)$ query. The adversary is given the response from s_{actor} according to the protocol.
2. corrupt(X). With this query E learns the long-term key of party X.
3. ephemeral-key(s). This query reveals the ephemeral keys (i. e., the random coins) of an incomplete session s.
4. session-key(s). This query returns the session key accepted during the session s. If s is incomplete or no key was accepted, the query returns empty.
5. test-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 output chosen from the probability distribution of keys generated by the protocol. This query can only be issued to a completed session that is a *fresh session* (defined below) by the time the query is issued.

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 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. We specify the restrictions on the queries made by the adversary by the concept of *fresh sessions.*

Definition 7 (Fresh session). *A completed session s in a key-exchange experiment W is said to be* fresh *if all of the following conditions hold:*

1. *W does not include the query session-key(s)*
2. *if a session $\tilde{s}$ exists that matches s, then W does not include session-key$(\tilde{s})$*
3. *W does not include both corrupt(s_{actor}) and ephemeral-key(s)*
4. *if s' is an origin-session for session s, then W does not include both corrupt(s_{peer}) and ephemeral-key(s')*
5. *if there is no origin-session for session s, then W does not include a corrupt(s_{peer}) query before the completion of session s.*

Observe that in experiments in which no ephemeral-key queries occur, the above definition allows the adversary to corrupt all parties after the session s is completed, specifying (full) perfect forward secrecy.

Security Experiment for eCK-PFS. Security of a key-exchange protocol Π is defined via an attack game played by the adversary E, modeled as an efficient probabilistic algorithm, against a challenger. Before the attack game starts, each of the involved parties $\hat{P}$ runs a key-generation algorithm Gen that takes as input a security parameter 1^k and outputs a static public/private key pair (pk_P, sk_P). The public key of each party is distributed in an authenticated way to all other parties. The adversary is given access to all public data.

First E is allowed to perform send, corrupt, ephemeral-key and session-key queries. The adversary then issues a test-session query to a fresh session of its choice. The challenger chooses a random bit b and provides the adversary with either the real session key (for $b = 0$) or a random key (for $b = 1$). The aim of the adversary is to correctly guess b, i. e., whether he got the real session key or not. To achieve this, he may continue with prevalent queries (session-key, corrupt, send, ephemeral-key) without rendering the test-session un-fresh. Finally, the adversary outputs a bit b' as his guess for b. He wins the game if $b' = b$.

The advantage of adversary E in the above security experiment with a key exchange protocol Π at security parameter k is defined as

$$Adv_E^{\Pi}(k) = |2P(b = b') - 1|.$$

Definition 8. *A key exchange protocol Π is called KE-secure if, for any efficient probabilistic adversary E, the following conditions hold:*

- *If two parties successfully complete matching sessions, then they both compute the same session key.*
- *E has no more than a negligible advantage in winning the above experiment, that is, there exists a negligible function negl in the security parameter k such that $Adv_E^{\Pi}(k) \leq negl(k)$.*

Rationale. Our KE security model captures the following security requirements. *Perfect forward secrecy* means that the compromise of long-term secret keys does not compromise past session keys. This property is reflected in the above security model by allowing the adversary to corrupt any party after the completion of the test-session. Further, the adversary is allowed to perform *key compromise impersonation attacks* by corrupting the actor of the test-session before its completion. Additionally, the adversary is allowed to *reveal ephemeral keys* of any session as in the extended-CK model. Note that if the adversary reveals the ephemeral keys of a session as well as the long-term secret key of the actor of that session, he can trivially reconstruct any computation performed by that party in any protocol. Hence we need to exclude this combination for the test-session and the origin-session if it exists (matching sessions are by definition also origin-sessions). Also, note that if the adversary chooses ephemeral-keys on behalf of a party and corrupts this party before the completion of the test-session, he can impersonate this party as peer to the test-session. Therefore we need to exclude this scenario by requiring that if there is no origin-session for the test-session, then the adversary should not be allowed to reveal the long-term key of the peer to the test-session before the latter is completed. Finally, *known-key attacks* are captured via the session-key queries; revealing session-keys of some sessions should not enable the adversary to obtain information on other session-keys. *Unknown key-share (UKS) attacks* lead to a special kind of known-key attacks. Informally, a key-exchange protocol is resilient against UKS attacks if no probabilistic polynomial-time (PPT) adversary can lead a party to sharing a session-key with a different party than its intended peer, with more than negligible probability (similar to the informal definition in [15]). In Appendix C we show in more detail how UKS attacks are captured in our model.

4 Peer-and-Time Deniability

Deniability is a privacy-related property that considers the scenario in which at least one of two parties involved in a protocol execution is dishonest and tries to convince a judge that some messages can be traced back to the other party.

Formal definitions of *full deniability* and *partial deniability* for key-exchange protocols were proposed in [27]. Full deniability allows a party to deny having been involved in a given run of the protocol. As a consequence, a recipient cannot convince a judge that the messages it received during a given execution were sent by the accused sender. In contrast, partial deniability does not allow the recipient to prove that the accused sender communicated *with him*: in other words, the transcripts are *peer-independent*.

As pointed out in [27], there is an inherent trade-off between deniability and authentication. This poses a direct problem for one-round key exchange problems. On the one hand, authentication of the exchanged key is critical, and on the other hand, there are only very limited options to establish authentication without violating deniability. In practice, many protocols do not achieve full deniability, and instead only weaker forms such as partial deniability of identity protection. In [27], the authors show that SKEME satisfies full deniability whereas the four-message version of SIGMA only satisfies partial deniability.

In order to achieve perfect forward secrecy in a one-round key exchange protocol, the protocol that we propose later relies on message-origin authentication of the exchanged messages, realized using signatures. This directly leads to a violation of full deniability. However, we manage to achieve a different form of deniability, which we call *peer-and-time deniability.* As opposed to partial deniability, our peer-and-time deniability allows parties to deny that they were alive during a certain time window. Though a judge can be convinced that a party signed some self-generated data at some point, there can be no proof of the party's role, its intended peer, whether the session was completed, or the time at which the message was signed happened. For many practical purposes, peer-and-time deniability seems to be a sufficiently strong form of deniability. In contrast, the SIGMA protocol allows others to obtain a signature on a peer-provided value. If this is chosen as, e. g., the hash of today's newspaper, a judge can be convinced that a party was alive after a certain date and time.

We formally define *peer-and-time* deniability by first formalizing the weaker property of *peer-deniability* and strengthening it to peer-and-time deniability. In our formal definitions, we assume an environment where an adversary has full control over the communication medium, can activate sessions through send queries and can corrupt parties, as in Section 3. Corrupted parties model malicious (or dishonest) insiders on behalf of whom the adversary can act. Given a protocol execution of an honest party $\hat{A}$ and a possibly dishonest party $\hat{B}$, $\hat{A}$ should be able to deny having been intentionally involved in a protocol run with $\hat{B}$. That is, party $\hat{A}$ should be able to deny its communication peer. Our notion of peer-deniability refers to the ability to produce a protocol transcript indistinguishable from a real protocol transcript between an honest party $\hat{A}$ and an adversary-controlled party $\hat{B}$. Informally, a protocol is said to be *peer-deniable* if, after a protocol execution between an adversary-controlled party $\hat{B}$ and some honest party $\hat{A}$, party $\hat{B}$ cannot convince a judge that party $\hat{A}$ was intentionally involved in that execution with $\hat{B}$. Party $\hat{A}$ should be able to deny interaction with $\hat{B}$ by arguing that its signed messages could have been generated while executing a protocol session with another party. If $\hat{A}$ is in the initiator role, then $\hat{A}$ could even deny having ever accomplished (completed) a protocol run with any other party. The message exchange of some protocol execution with a dishonest peer $\hat{B}$ can be efficiently simulated given access to an oracle of $\hat{A}$ in a protocol execution with some other party $\hat{C} \neq \hat{B}$.

Our formal definitions of deniability are inspired by the works of Dwork, Naor and Sahai [11] as well as of Di Raimondo, Gennaro and Krawczyk [27]. The main difference to the definition of *partial-deniability* [27] is that the simulator is additionally given access to the secret key of corrupted parties and that the simulation of the protocol transcript does not include the session-key. As in [27], we model the generation of signatures on behalf of uncorrupted parties by giving the simulator access to oracles. We denote by $\Sigma_I^{\hat{A},\hat{B}}$ the $I-$oracle representing either an incomplete initiator session at party $\hat{A}$ with intended peer $\hat{B}$ or a completed initiator session at party $\hat{A}$ with peer $\hat{B}$. Similarly, $\Sigma_R^{\hat{A},\hat{B}}$ denotes an $R-$oracle representing a responder session at party $\hat{A}$ with peer $\hat{B}$.

Definition 9 (peer-deniability). *Let Π be an (authenticated) key-exchange protocol. We say that an initiator role of Π is* peer-deniable with respect to an $I-$oracle (or $R-$oracle) *if there exists a polynomial time simulator S_I such that, for any adversary-controlled party $\hat{B}$ and any honest party $\hat{A}$, S_I can produce a transcript of sent and received messages between $\hat{A}$ as initiator and $\hat{B}$ as responder, indistinguishable from a real transcript between $\hat{A}$ and $\hat{B}$ in a setting where the adversary is allowed to issue send and corrupt queries, while given oracle access to a polynomial number of protocol role instances except to $\Sigma_I^{\hat{A},\hat{B}}$ and $\Sigma_R^{\hat{A},\hat{B}}$ (i.e. no oracle access to role instances with actor $\hat{A}$ and peer $\hat{B}$). Additionally, the simulator is given the secret keys of corrupted parties.*

An initiator role is said to be peer-deniable *if it is peer-deniable with respect to an $I-$oracle or with respect to an $R-$oracle. A responder role is defined to be* peer-deniable *analogously. Protocol Π is* peer-deniable *if both roles are* peer-deniable.

Under the key-awareness assumption (see [27]) for the key derivation function used, the four-message version of the SIGMA protocol is peer-deniable with respect to Definition 9. Note that even the session-key (which is independent of public/private information relating to the parties involved in the simulation) could be simulated.

We now specialize the above definition by conditioning the proceedings of the simulator. The resulting definition additionally ensures timelessness of exchanged messages. Thus, it captures peer-independence as well as time-independence of sent and received messages during a protocol execution.

Definition 10 (peer-and-time deniability). *Let Π be an (authenticated) key-exchange protocol. We say that an initiator role of Π is* peer-and-time deniable with respect to an $I-$oracle (or $R-$oracle) *if there exists a polynomial time simulator S_I working as described below such that, for any adversary-controlled party $\hat{B}$ and any honest party $\hat{A}$, S_I can produce a transcript of sent and received messages between $\hat{A}$ as initiator and $\hat{B}$ as responder, indistinguishable from a real transcript between $\hat{A}$ and $\hat{B}$ in a setting where the adversary is allowed to* *issue send and corrupt queries, while given oracle access to a polynomial number of protocol role instances except to $\Sigma_I^{\hat{A},\hat{B}}$ and $\Sigma_R^{\hat{A},\hat{B}}$.*

The simulation should proceed in four steps, as follows:

1. *At time τ_0, the simulator is setup with public knowledge (such as the identity and public-key of a polynomial number of parties (in particular the public keys of the parties $\hat{A}$ and $\hat{B}$) as well as the secret keys of corrupted parties.*
2. *The simulator is allowed to activate and query the oracles is it given access to for a time period $\delta := \tau_1 - \tau_0$, where $\tau_1 > \tau_0$. It is not allowed to interact with $\hat{B}$ or to perform on behalf of $\hat{B}$ (since it knows its long-term secret key).*
3. *From time $\tau_2 > \tau_1$, the simulator can interact with $\hat{B}$ without interacting with oracles $\Sigma_I^{\hat{A},*}$ and $\Sigma_R^{\hat{A},*}$ for which $\hat{A}$ acts as initiator or responder. (Here $*$ can be replaced by any identity $\hat{C} \neq \hat{B}$.)*
4. *The simulator outputs a protocol transcript between $\hat{A}$ and $\hat{B}$.*

An initiator role is said to be peer-and-time deniable *if it is peer-and-time deniable with respect to an $I-$oracle or with respect to an $R-$oracle. A responder role is defined to be* peer-and-time deniable *analogously. Protocol Π is* peer-and-time deniable *if both roles are* peer-and-time deniable.

The SIGMA protocol does not satisfy peer-and-time deniability because it requires a signature on a peer-provided element.

5 A strongly-secure one round key exchange protocol

Let G be a finite cyclic group of prime order p with $p = O(2^k)$ for some security parameter k and let g be a generator of the group G. Further, let $KDF : \{0,1\}^* \to \{0,1\}^k$ denote a key derivation function. We assume that each party has a long-term public/private key pair ($pk = g^v, sk = v$) (where $v \in \mathbb{Z}_p$) for use in a digital signature scheme and that the public keys of other involved parties are known and distinct from each other. We assume that the owners of valid public keys know the corresponding private keys. For example, this can be realized by a CA that generates the pairs, or by requiring that users provide a proof-of-possession for the private key when they register a public key. If users can generate key pairs dynamically, i. e., they can register new key pairs during the lifetime of the protocol, then we require that the signature scheme is resilient against Duplicate Signature Key Selection (DSKS) attacks [17].

An execution of the protocol proceeds as follows. We denote by $x \in_R S$ the element x being chosen uniformly at random from the set S.

1. $\hat{A}$ (initiator) chooses an ephemeral secret key $x \in_R \mathbb{Z}_p$, signs the group element $X = g^x$ with her long-term secret key, and sends the message $(\hat{A}, X, \sigma_X = Sign_{sk_{\hat{A}}}(X))$ to $\hat{B}$.
2. $\hat{B}$ (responder) verifies that $X \in G$ and that $Vrfy_{pk_{\hat{A}}}(X, \sigma_X) = 1$. If the checks are successful, then $\hat{B}$ chooses an ephemeral secret key $y \in_R \mathbb{Z}_p$, signs the group element $Y = g^y$ with his long-term secret key, sends the message $(\hat{B}, Y, \sigma_Y = Sign_{sk_{\hat{B}}}(Y))$ to $\hat{A}$ and computes the session-key $K_{\hat{B}} = KDF(\hat{A}, \hat{B}, (XA)^{y+b}, X)$.

3. $\hat{A}$ verifies that $Y \in G$ and that $Vrfy_{pk_{\hat{B}}}(Y, \sigma_Y) = 1$. If the checks are successful, then $\hat{A}$ computes the session-key $K_{\hat{A}} = KDF(\hat{A}, \hat{B}, (YB)^{x+a}, X)$.

We assume that whenever a verification in a session s fails, all session-specific data is erased from memory and session s is aborted (that is, it terminates without establishing a session-key). Note that the role of the session specifies the order of the identities as input to the key derivation function KDF.

$$\begin{array}{ccc}
\underline{i}:\ \hat{A} : a, A = g^a & & \underline{r}:\ \hat{B} : b, B = g^b \\
x \in_R \mathbb{Z}_p & & \\
X = g^x & \xrightarrow{\hat{A}, X, Sign_{sk_{\hat{A}}}(X)} & \text{Check } X \in G \\
 & & \text{Verify signature} \\
 & & y \in_R \mathbb{Z}_p \\
\text{Check } Y \in G & \xleftarrow{\hat{B}, Y, Sign_{sk_{\hat{B}}}(Y)} & Y = g^y \\
\text{Verify signature} & & \\
K_{\hat{A}} = KDF(\hat{A}, \hat{B}, (YB)^{x+a}, X) & & K_{\hat{B}} = KDF(\hat{A}, \hat{B}, (XA)^{y+b}, X)
\end{array}$$

Design choices. Under the assumption that an attacker did not learn a party's long-term secret key and that forging signatures is hard, he cannot impersonate that party to a session. Without ephemeral public-key validation, an attacker could recover the long-term secret key of an uncorrupted party via a variant of the small-subgroup attack against HMQV described in [23, p. 136]. The attack involves corrupt, ephemeral-key and session-key queries and exploits the structure of certain cyclic groups.

To prevent unknown-key share attacks, we include the participants' identities (ordered according the their roles) in the input of the key derivation function KDF. The final component X in the key derivation function excludes two initiators from computing the same key, even in the case of self-communication.

An important design choice in our protocol is the secret group element $g^{(x+a)(y+b)}$ taken as input to the KDF. This provides resilience against the replay attack described in [18, p. 418]. Suppose that an adversary E learns the ephemeral private key x of an exponential g^x sent by party $\hat{A}$ in session s. Replaying the message $(\hat{A}, g^x, Sign_{sk_{\hat{A}}}(g^x))$ to a session s' at party $\hat{B}$, the adversary cannot impersonate $\hat{A}$ (or compute the same session-key as in session s') without revealing $\hat{A}$'s long-term secret key (under the $GAP-CDH$ assumption in the underlying group G). Thus, the leakage of an ephemeral private key used in a given session does not affect the security of other sessions. Moreover, revealing any subset of $\{x, y, a, b\}$ that does not contain $\{x, a\}$ or $\{y, b\}$, does not allow the adversary to compute the session-key. [1]

[1] In the security proof of our protocol, we assume that the signature scheme does not leak any information about the secret key. Alternatively, one can use two different independent public/secret key pairs. One for use in the digital signature scheme and the other one for the computation of the secret value in the key derivation function.

6 Security and Deniability Analysis

We focus on the most important steps in the security analysis of our protocol and discuss the reasons for achieving the desired security goals. Detailed proofs of Theorem 1 and Lemma 1 can be found in Appendix B. We first state the main security result.

Theorem 1. *Under the $GAP-CDH$ assumption in the cyclic group G of prime order p, using a deterministic signature scheme that is existentially unforgeable under adaptively chosen-message attacks, our protocol is a secure authenticated key-exchange protocol with respect to the eCK-PFS model (Definition 8), when KDF is modeled as a random oracle. The adversary E's advantage is bounded by*

$$Adv_E^{\Pi}(k) \leq \frac{(q_s+q_{ro})^2}{2^k} + \frac{q_s^2+2Nq_s}{p} + 2Nq_s Adv_M^{Sig}(k) + 2q_s^2 q_{ro} Adv_C^{GAP-CDH}(k) + \frac{q_s}{2^k},$$

where N is an upper bound on the number of parties and q_s, q_{ro} are upper bounds on the number of activated sessions and random oracle queries by the adversary.

The following lemma is an intermediate result that states that the adversary cannot break the security of the protocol through session-key reveal queries. First, these queries are not allowed on matching sessions. Second, since the session-keys are generated independently from each other and do only collide with negligible probability when the key derivation function *KDF* is modeled as a random oracle, session-key reveal queries on non-matching sessions give no information on the session-key of the fresh test-session. Both arguments guarantee resilience against known-key attacks as well as UKS attacks.

We write K_s to denote the session key computed by session s.

Lemma 1. *Assume that the KDF function in the protocol is modeled as a random oracle. If for any two sessions in a security experiment the chosen ephemeral public keys are distinct, then it holds that either $s \sim t$ or $K_s \neq K_t$ with overwhelming probability, for all completed sessions s, t with $s \neq t$.*

We proceed with two reduction arguments. First, recall that, while being allowed to compromise the long-term secret key of the actor of the test-session, the adversary can only reveal the long-term secret key of the peer to the test-session before its completion when he does not inject his own message. Hence, the adversary could only break the security of the protocol via insertion of a message of its choice by forging a signature with respect to the long-term public key of the test-session's peer. By assumption forgery events can only occur with negligible probability, so that the protocol is resilient against key compromise impersonation attacks.

Excluding a forgery event, we can assume that there exists an origin-session s for the test-session t. Revealing the long-term secret keys of actor and peer of the test-session, the adversary could break the security of the protocol by solving the instance of the $GAP-CDH$ problem given by the sent and received messages involved in the test-session. Similarly, the adversary could have issued other combinations of corrupt and ephemeral-key queries (without rendering the test-session un-fresh) and break the $GAP - CDH$ with respect to a corresponding instance. The hardness of the $GAP - CDH$ assumption in the cyclic group G of prime order allows to achieve resilience against such attacks.

Theorem 2. *Our protocol is peer-and-time deniable conform Definition 10.*

Proof. We first show that an initiator role of our protocol is peer-and-time deniable with respect to an I-oracle. The simulator S_I is setup with public information, in particular the identity and public key of the parties $\hat{A}$,$\hat{B}$ and $\hat{C}$. Further, S_I is given oracle access to the initiator instance $\Sigma_I^{\hat{A},\hat{C}}$ where $\hat{C} \neq \hat{B}$. The simulator activates the oracle $\Sigma_I^{\hat{A},\hat{C}}$ and gets as response the message $m = (\hat{A}, X, Sign_{sk_{\hat{A}}}(X))$ at time τ_1. At time $\tau_2 > \tau_1$, it forwards the message m to $\hat{B}$. Upon receipt of $\hat{B}$'s response n of the form $(\hat{B}, Y, \sigma)$, S_I verifies whether $Y \in G$ and whether the signature σ on Y with respect to the public key of $\hat{B}$ is valid, and if the verifications are successful, S_I outputs the simulated protocol transcript (m, n). Note that the initiator session at $\hat{A}$ is still incomplete.

Next, we show that a responder role is peer-and-time deniable with respect to an I-oracle. The simulator S_R is setup as in the previous case. S_R is given oracle access to the initiator instance $\Sigma_I^{\hat{A},\hat{C}}$ where $\hat{C} \neq \hat{B}$. The simulator S_R first activates the oracle $\Sigma_I^{\hat{A},\hat{C}}$ and gets as response the message $m = (\hat{A}, X, Sign_{sk_{\hat{A}}}(X))$ at time τ_1. At time $\tau_2 > \tau_1$, it activates $\hat{B}$ and gets as response the message n of the form $(\hat{B}, Y, \sigma)$. If Y belongs to the group G and the verification of the signature σ on Y with respect to the public key of $\hat{B}$ succeeds, then S_I outputs the simulated protocol transcript (n, m). In a similar way, one can show that a responder role is peer-and-time deniable with respect to an R-oracle.

Known weaknesses. If the adversary learns the exponent used in the key derivation, e. g., $(x + a)$, as well as the signature on g^x, he can indefinitely impersonate $\hat{A}$. Similar attacks exist for (H)MQV [1] and Naxos [8]. The exponents $(x + a)$ and $(y + b)$ must therefore be similarly protected as the long-term private key.

7 Efficiency

Computational complexity. A run of the protocol requires for each party three exponentiations (one for the ephemeral public key, one for the session key and one for the group element check), one signature generation and one signature verification.

The computational cost of the signature scheme is therefore a large factor in the efficiency of our protocol. We require that the signature scheme does not reveal the long-term keys even if the used random coins are revealed. This can be realized, e. g., by using a deterministic signature scheme. For example, when using the GDH signature scheme from [5], the signature generation needs one exponentiation, and verification costs one DH-tuple check. As mentioned in [5], for the suggested signature scheme based on elliptic curves the signature verification is more expensive than the signature generation since it requires the computation of two pairings.

Alternatively, the "NAXOS trick" [20] can be applied to a non-deterministic signature scheme: the random coins x drawn by party $\hat{A}$ for use in the signature scheme can be replaced by $H(sk_{\hat{A}}, x)$, where H is a hash function. As a concrete example for Schnorr signatures, this means that in the computation of the signature we draw random coins x, and compute $g^{H(sk_{\hat{A}},x)}$ instead of just g^x. Consequently, revealing x (but not $sk_{\hat{A}}$) and the corresponding signature no longer reveals any information about the long term key.

Note that the ephemeral public keys and their signatures can be computed off-line. However, the signature verification and the exponentiation for the key computation need to be performed on-line.

Communication complexity. Our protocol requires that the two ephemeral public keys are sent together with the signatures. Depending on the signature scheme, this is about 2.5 times more bandwidth than (the two-message version of) MQV. However, two-message protocols that satisfy similar security notions, such as MQV or NAXOS, require additional communications to achieve perfect forward secrecy. Furthermore, at the expense of computational efficiency, it is possible to switch to short signatures, e. g. [5], to optimize communication complexity.

8 Related work

Our eCK-PFS security notion is a strengthening of extended-CK [20]. The extended-CK model only considers weak perfect forward secrecy. Hence, the model only allows revealing the long-term keys of the test session's participants if the adversary is passive during the session, which is captured in the model by requiring the existence of a matching session. This restriction does not occur in eCK-PFS.

Deniable authentication was first introduced by Dwork et al. [11] using the simulation paradigm. Deniability of key-exchange protocols has been formalized by Di Raimondo et al. [27]. Their definition of *partial-deniability* cannot be met by signed key-exchange protocols where the session-key computation depends on public data related to actor and peer of the session such as their identities or public/secret keys. An honest initiator $\hat{A}$ cannot pretend having established the same session-key with $\hat{B}$ as with any other party $\hat{C} \neq \hat{B}$ since these keys will be distinct with overwhelming probability (e. g., by collision-freeness of the key derivation function, and assuming that the computation of the session-keys relies on the same ephemeral public/private data). Further, partial-deniability only captures deniability for those sessions that computed a session-key. Thus an honest initiator $\hat{A}$ may not pretend never having completed a protocol session. Also, the dishonest party $\hat{B}$ in a protocol execution with $\hat{A}$ trying to trace a session-key back to $\hat{A}$ cannot convince a judge that he could not have computed the session-key himself given the sent and received messages.

The protocol that is closest to our protocol is the YAK protocol by Hao [13]. There are two main differences: YAK uses zero-knowledge proofs instead of signatures, and includes identity information in the messages whereas our protocol delays this to the key computation. Because YAK does not offer message origin authentication, it is vulnerable to the generic PFS attack sketched by Krawczyk, and only satisfies weak perfect forward secrecy. Compared to our protocol, YAK requires more communication (for the zero-knowledge proofs).

After we published the first version of our report, a closely related protocol was proposed in [14]. The protocol in [14] can be considered as an instance of our protocol (up to the computation of the session-key) by instantiating the deterministic signature scheme with Boneh's GDH Signature Scheme [5]. As stated in [5], this scheme was implicitly described in [25]. In the protocol description [14, Section 3.2], the "signatures" c_1 and c_2 on the messages X and Y are included within the key derivation (as part of the session identifier sid). These values c_1, c_2 are redundant in the computation of the session-key and only serve authentication purposes through the DDH check.

The modified-Okamoto-Tanaka (mOT) protocol by Gennaro, Krawczyk and Rabin [12] very efficiently provides perfect forward secrecy in the identity-based setting in one round. This is achieved by defining the messages as the product of the ephemeral public key and the ID-based private key. In the key derivation function, these values are subsequently divided by the ID-based public keys, allowing both parties to compute the same key. The security proof for mOT depends on a variant of the KEA1 assumption [2]. Additionally, they sketch variants of the protocol for the PKI-based setting. As noted by the authors, the mOT protocol is not resilient against loss of ephemeral keys. In particular, the loss of a single ephemeral key and corresponding message allows indefinite impersonation of that party to any other party.

9 Conclusions

At the expense of a small degree of deniability, the use of signatures in an MQV-style protocol yields several advantages for one-round key exchange protocols. The main advantage is that they allow us to prove a very strong security notion for our protocol. To the best of our knowledge, our protocol is the first one-round key exchange protocol that satisfies this strong security notion, which implies both extended-CK security and perfect forward secrecy.

In terms of efficiency, our protocol is more expensive than protocols that offer only weak perfect forward secrecy and do not use signatures. However, protocols that satisfy comparable security notions, including perfect forward secrecy (e. g., the three-message versions of (H)MQV and NAXOS), require significantly more communication than our protocol.

Finally, our concept of peer-and-time deniability may be of use in other contexts where full deniability cannot be achieved.

References

1. D. Basin and C. Cremers. Modeling and analyzing security in the presence of compromising adversaries. In *Computer Security - ESORICS 2010*, volume 6345 of *Lecture Notes in Computer Science*, pages 340–356. Springer, 2010.

2. M. Bellare and A. Palacio. The Knowledge-of-Exponent assumptions and 3-round Zero-Knowledge protocols. In *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 273–289. Springer, 2004.
3. M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Advances in Cryptology-CRYPTO 93*, volume 773, pages 232–249. Springer-Verlag, 1994.
4. 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 *Lecture Notes in Computer Science*, pages 154–170. Springer-Verlag Berlin / Heidelberg, 1999.
5. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil Pairing. In *ASIACRYPT'01*, pages 514–532, 2001.
6. N. Borisov, I. Goldberg, and E. Brewer. Off-the-record communication, or, why not to use pgp. In *Proceedings of the 2004 ACM workshop on Privacy in the electronic society*, pages 77–84. ACM Press, 2004.
7. 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.
8. C. Cremers. Session-StateReveal is stronger than eCK's EphemeralKeyReveal: Using automatic analysis to attack the NAXOS protocol. *International Journal of Applied Cryptography (IJACT)*, 2:83–99, 2010.
9. Cas 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.
10. A.W. Dent. A note on game-hopping proofs. Cryptology ePrint Archive, Report 2006/260, 2006. Available at `http://eprint.iacr.org/2006/260`.
11. C. Dwork, M. Naor, and A. Sahai. Concurrent zero-knowledge. In *IN 30TH STOC*, pages 409–418, 1999.
12. R. Gennaro, H. Krawczyk, and T. Rabin. Okamoto-Tanaka revisited: fully authenticated Diffie-Hellman with minimal overhead. In *Proceedings of the 8th international conference on Applied cryptography and network security*, ACNS'10, pages 309–328. Springer-Verlag, 2010.
13. F. Hao. On robust key agreement based on public key authentication. In *Financial Cryptography*, volume 6052 of *Lecture Notes in Computer Science*, pages 383–390. Springer, 2010.
14. H. Huang. Strongly secure one round authenticated key exchange protocol with perfect forward security. Cryptology ePrint Archive, Report 2011/346, 2011. `http://eprint.iacr.org/`.
15. B.S.JR. Kaliski. An unknown key-share attack on the MQV key agreement protocol. In *ACM Transactions on Information and System Security (TISSEC)*, volume 4, pages 275–288. ACM New York, NY, USA, August 2001.
16. J. Katz and Y. Lindell. *Introduction to Modern Cryptography*. Chapman Hall/CRC, 2008.
17. N. Koblitz and A. Menezes. Another look at security definitions. Cryptology ePrint Archive, Report 2011/343, 2011. `http://eprint.iacr.org/`.
18. H. Krawczyk. SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and Its Use in the IKE-Protocols. In *CRYPTO*, pages 400–425, 2003.
19. H. Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In *CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 546–566. Springer-Verlag, 2005.
20. B.A. LaMacchia, K. Lauter, and A. Mityagin. Stronger security of authenticated key exchange. In *ProvSec*, volume 4784 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, 2007.
21. L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone. An efficient protocol for authenticated key agreement. *Designs, Codes and Cryptography*, 28(2):119–134, 2003.
22. U. Maurer. Abstract models of computation in cryptography. In Nigel Smart, editor, *Cryptography and Coding 2005*, volume 3796 of *Lecture Notes in Computer Science*, pages 1–12. Springer-Verlag, December 2005.

23. A. Menezes and B. Ustaoglu. On the Importance of Public-Key Validation in the MQV and HMQV Key Agreement Protocols. In R. Barua and T. Lange, editors, *INDOCRYPT 2006*, volume 4329 of *Lecture Notes in Computer Science*, pages 133–147. Springer-Verlag Berlin / Heidelberg, 2006.
24. A. Menezes, P. van Oorschot, and S. Vanstone. Handbook of applied cryptography, October 1996.
25. T. Okamoto and D. Pointcheval. The gap-problems: a new class of problems for the security of cryptographic schemes. In *Proceedings of the 2001 International workshop on Practive and Theory in Public Key Cryptography (PKC '2001)*, Lecture Notes in Computer Science, pages 104–118. Springer-Verlag, 2001.
26. M. Di Raimondo, R. Gennaro, and H. Krawczyk. Secure off-the-record messaging. In *WPES '05 Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pages 81–89. ACM New York, NY, USA, 2005.
27. M. Di Raimondo, R. Gennaro, and H. Krawczyk. Deniable authentication and key exchange. Cryptology ePrint Archive, Report 2006/280, 2006. `http://eprint.iacr.org/`.
28. V. Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2006. `http://eprint.iacr.org/`.

A Comparison of the eCK-PFS model to the eCK model

The eCK-PFS model is a strengthened eCK-like model that captures perfect forward secrecy by allowing the adversary to corrupt any party after the completion of the test-session. Further, it requires that, when an origin-session for the test-session exists, the adversary is not allowed to reveal both the long-term secret key of the peer to the test-session and the ephemeral-key of the origin-session. In Table 1, we compare the adversary's capabilities in the eCK model [20] and the eCK-PFS model with respect to the same definition for matching sessions (that is, Definition 5). The reader is referred to [9] for an in-depth comparison of other security models for key exchange protocols.

Adversary capability	Condition in eCK	Condition in eCK-PFS
reveal long-term key of actor s_{actor} of test-session s	if no ephemeral-key reveal on s	if no ephemeral-key reveal on s
reveal long-term key of peer s_{peer} to test-session s	if there is a session s' that eCK-matches s, then no ephemeral-key reveal on s', else not allowed	if there is an origin-session s' for the test-session s, then no ephemeral-key reveal on s', else only allowed after the completion of session s
reveal ephemeral-key of test-session s	if no long-term key reveal on s_{actor}	if no long-term key reveal on s_{actor}
reveal ephemeral-key of non-test session s'	if s' eCK-matches the test-session s, then no long-term key reveal on s'_{actor}	if s' is an origin-session for the test-session s, then no long-term key reveal on s'_{actor}
reveal session-key of non-test session s'	if s' does not eCK-match the test-session s	if s' does not eCK-match the test-session s

Table 1. Comparison of the necessary conditions for adversary capabilities between the security models eCK and eCK-PFS

The following proposition states that the eCK-PFS model is stronger than the eCK model with respect to the adversary's capabilities.

Proposition 1. *The eCK-PFS model is stronger than the eCK model, that is, eCK-PFS captures strictly more attacks than the eCK model.*

Proof. Let $valid(W, M)$ be the predicate that is true if and only if security experiment W is valid in security model M, that is, the session chosen by the adversary to be the test-session must be fresh by the time he issues the test-session query and remains fresh until the end of the experiment with respect to the freshness definition of model M.

We have to show that

- $\forall W\ (valid(W, \text{eCK}) \rightarrow valid(W, \text{eCK} - \text{PFS}))$, and
- $\exists W$ such that $valid(W, \text{eCK} - \text{PFS})$ and $\neg valid(W, \text{eCK})$.

Claim. It holds that $\forall W\ (valid(W, \text{eCK}) \rightarrow valid(W, \text{eCK} - \text{PFS}))$.

Proof. Most cases of the proof are straightforward. We focus on the non-trivial cases that revolve around ephemeral-key reveals on non-test sessions. Let W be a security experiment such that $valid(W, \text{eCK})$. Suppose that in experiment W there exists a session s' that eCK-matches the test-session s and that W includes a corrupt(s_{peer}) query before the completion of the test-session. Note that session s' is also an origin-session for session s (by definition). Since $valid(W, \text{eCK})$, the experiment W does not include an ephemeral-key(s') query. It follows that $valid(W, \text{eCK} - \text{PFS})$.

Now let W' be a security experiment such that $valid(W', \text{eCK})$. Suppose that in experiment W' there exists a session s' that eCK-matches the test-session s and that W' includes an ephemeral-key(s') query. By assumption $valid(W', \text{eCK})$, so that the experiment W' does not include a corrupt(s_{peer}) query. Since session s' is an origin-session for the test-session s and W' does not include a corrupt(s_{peer}) query, it holds that $valid(W, \text{eCK} - \text{PFS})$.

Claim. It holds that $\exists W$ such that $valid(W, \text{eCK} - \text{PFS})$ and $\neg valid(W, \text{eCK})$.

Proof. Let W be a security experiment such that $valid(W, \text{eCK} - \text{PFS})$. Suppose that in experiment W there exists a unique origin-session s' for the test-session s such that s' is not extended to an eCK-matching during experiment W. Further, suppose that the adversary issued a corrupt(s_{peer}). Note that, since by assumption $valid(W, \text{eCK} - \text{PFS})$, experiment W does not include the query ephemeral-key(s'). Clearly, it holds that $\neg valid(W, \text{eCK})$. This follows from the fact that in the eCK model, the query corrupt(s_{peer}) with $s_{peer} = s'_{actor}$ and s' does not eCK-match s in experiment W renders the test-session s un-fresh.

B Security proofs

In this appendix we provide detailed proofs of Lemma 1 and Theorem 1.

B.1 Preliminaries

We first recall some cryptographic assumption and then we state a new definition for resilience against key-recovery attacks within security experiments.

Similar to the discrete logarithm experiment [16], we define the GAP discrete logarithm[2] experiment for a given group-generating algorithm G, algorithm A, and parameter k as follows.

The GAP discrete logarithm experiment $GAP - DLog_{A,G}(k)$**: (see also [16])**

1. Run $G(1^k)$ to obtain (G, p, g), where G is a cyclic group of order p (with $||p|| = k$) and g is a generator of G.
2. Choose $h \leftarrow G$. (This can be done by choosing $x' \in \mathbb{Z}_p$ and setting $h := g^{x'}$.)
3. A is given G, p, g, h, and outputs $x \in \mathbb{Z}_p$. In addition, 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 11 **(**$GAP - DLog$**).** *We say that the* GAP discrete logarithm problem *is hard relative to G if for all probabilistic polynomial-time algorithms A there exists a negligible function negl such that*

$$P(GAP - DLog_{A,G}(k) = 1) \leq negl(k).$$

Consider the following experiment for a given group G, algorithm A, protocol Π, adversarial environment Env_M (where M denotes a security model)[3] specifying the queries that an adversary can issue against the protocol instances and the parties involved, and parameter k:

The key-recovery experiment $KR_{A,G,\Pi,Env_M}(k)$**:**

Let G be a cyclic group of order p (with $||p|| = k$) and g a generator of G.

1. Choose party $\hat{H}$ whose public key is an element $h \leftarrow G$ such that $h := g^{x'}$ for some $x' \in \mathbb{Z}_p$. (e.g. choose a party uniformly at random)
2. A is given G, p, g, h where h is the public key of party P and other public data (such as the public keys of other parties). Additionally, A can interact with a polynomial number of sessions of protocol Π and perform queries as specified in Env_M (e.g. send, corrupt, ephemeral-key and session-key queries when $M = eCK - PFS$). However, A is not allowed to corrupt party $\hat{H}$.
3. A outputs $x \in \mathbb{Z}_p$.
4. The output of the experiment is defined to be 1 if $g^x = h$, and 0 otherwise.

Definition 12 (key-recovery attack). *We say that a key-exchange protocol Π relative to a security model M and group G is* resilient against key-recovery attacks *if for all probabilistic polynomial-time algorithms A there exists a negligible function negl such that*

$$P(KR_{A,G,\Pi,Env_M}(k) = 1) \leq negl(k).$$

Finally, we recall the Difference Lemma introduced by Shoup in [28]. Let $P(X)$ denote the probability that event X occurs. We denote the complement of an event F by F^c (also often denoted by $\neg F$).

[2] The GAP discrete logarithm problem has been studied by Maurer [22] in an abstract model of computation.

[3] When $M = eCK - PFS$, the adversarial environment Env_M comprises send, corrupt, ephemeral-key and session-key queries. Note that the test-session query is not part of Env_M.

Lemma 2 (Difference Lemma [28]). *Let A, B, F be events defined on some probability space, and suppose that $A \wedge F^c \Leftrightarrow B \wedge F^c$. Then*

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

B.2 Security statements and proofs

Lemma 3. *Assume that the key derivation function KDF used in our protocol is modeled as a random oracle. If for any two sessions in a security experiment the chosen ephemeral public keys are distinct, then it holds that either $s \sim t$ (i. e., s and t are matching sessions) or $K_s \neq K_t$ with overwhelming probability, for all completed sessions s, t with $s \neq t$.*

Proof. By contrapositive. Suppose that there exist two completed sessions s, t with $s \neq t$ such that $K_s = K_t$ and $s \not\sim t$. We have to show that there are two distinct sessions in the experiment that generate the same ephemeral keys.

We denote by Z_s, Z_t the Diffie-Hellman exponentials sent in sessions s, t, respectively. Let Q_s, Q_t denote the Diffie-Hellman exponentials received in sessions s, t, respectively. Observe that $s \not\sim t$ if and only if ($s_{actor} \neq t_{peer} \vee s_{peer} \neq t_{actor} \vee s_{send} \neq t_{recv} \vee s_{recv} \neq t_{send} \vee s_{role} = t_{role}$). We exclude collisions in the key derivation function since they can only occur with negligible probability (by assumption).

We distinguish between the following cases.
Case 1: $t_{role} = s_{role} = i$. Then we must have that $Z_s = Z_t$ so that $K_s = K_t$.
Case 2: $t_{role} = s_{role} = r$. Both sessions must receive the same Diffie-Hellman value, i. e., $Q_s = Q_t$ (otherwise the last input to the *KDF* would be different). Suppose that session t receives a message at time τ_n and session s receives a message at time τ_m with $m > n$. Then session t establishes the session-key first which involves the computation of the group element $(Q_t * pk_{t_{peer}})^{(z_t + sk_{t_{actor}})}$ (where z_t denotes the discrete logarithm of $Z_t = g^{z_t}$ and $pk_{t_{peer}}, sk_{t_{actor}}$ denote the public key of t_{peer} and secret key of t_{actor}, respectively). Necessary conditions to have $K_s = K_t$ are that $t_{actor} = s_{actor}$ and $t_{peer} = s_{peer}$. This implies that $Z_s = Z_t$.

Note that we do not need to consider the case where $s_{role} = i$ and $t_{role} = r$. To get $K_s = K_t$, it must hold that $s_{actor} = t_{peer}$ and $s_{peer} = t_{actor}$ (by the ordering of the identities as input to the *KDF* according to their roles and the the inclusion of the first message's ephemeral public key). Together with the fact that we must have $Z_s = Q_t$ and $Z_t = Q_s$, it follows that $s_{send} = t_{recv}$ and $t_{send} = s_{recv}$. Hence, the sessions s and t are matching (that is, $s \sim t$).

Lemma 4. *Assume that the signature scheme does not leak any information about the secret key and that KDF is modeled as a random oracle. Under the $GAP - DLog$ assumption in the finite cyclic group G of prime order p, our protocol is resilient against key-recovery attacks.*

Proof. We present a proof structured as a sequence of games where $KDF : \{0,1\}^* \rightarrow \{0,1\}^k$ is modeled as a random oracle. Let N, q_s be upper bounds on the number of parties and activated sessions, respectively. We denote by S_i the event that the adversary E succeeds in a key-recovery attack in Game i. Recall that we assume that the public/secret key pairs of the parties are distinct from each other.

Game 0 Fix an efficient adversary E. This game reflects the real attack game against E in the key-recovery experiment.

Game 1 [Transition based on a bridging step] Let Event N be the session-specific failure event that there exist two distinct sessions s and s' that choose the same ephemeral private key. As soon as event N occurs, the attack game stops.
`Analysis of Game` 1: Game 1 is identical to Game 0 up to the point in the experiment where event N occurs for the first time. Moreover, we have that

$$P(N) = \binom{q_s}{2}\frac{1}{p} \leq \frac{q_s^2}{2p}.$$

The Difference Lemma yields that

$$|P(S_0) - P(S_1)| \leq P(N) \leq \frac{q_s^2}{2p}.$$

Game 2 [Transition based on a small failure event] Let Event U be the session-specific failure event that there exists a session t and a party P such that the long-term public key of party P equals the ephemeral public key chosen in session t. We need to prevent the scenario where revealing the ephemeral secret key of some session implies learning the long-term secret key of some party, because they are identical. When event U occurs, the attack game stops.

Game 1 and Game 2 simplify the analysis of the success probability in a key-recovery attack in Game 2.
`Analysis of Game` 2: Game 1 is identical to Game 2 up to the point where event U occurs. We have that

$$P(U) \leq \frac{Nq_s}{p}.$$

The Difference Lemma yields that

$$|P(S_1) - P(S_2)| \leq P(U) \leq \frac{Nq_s}{p}.$$

Claim. If the GAP discrete logarithm problem is hard relative to the group G, then the probability of the adversary succeeding in a key-recovery attack is negligible. Moreover, in Game 2, it holds that $P(S_2) = N * P(GAP - DLog_{C,G}(k) = 1)$.

Proof. We define the minimal session identifier, denoted by min-sid, of a session s as a quintuple $(s_{actor}, s_{peer}, X_1, X_2, s_{role})$, where s_{actor}, s_{peer} denote the identities of the owner and intended peer of the session s, $s_{role} \in \{i \text{ (initiator)}, r \text{ (responder)}\}$ denotes the role that the session is executing, and X_1, X_2 are the Diffie-Hellman exponentials (belonging to the group G) contained in the messages s_{send}, s_{recv}, respectively. For incomplete sessions, the element X_2 is undefined (denoted by $*$).

We solve the $GAP - DLog$ problem with probability $\frac{1}{N}P(S_2)$ where $P(S_2)$ denotes the probability in succeeding in a key-recovery attack in Game 2. Consider the following algorithm C (against $GAP - DLog$) which uses adversary E as a subroutine.

ALGORITHM C: The algorithm is given a group element $B = g^b$ of G as an instance of the $GAP - DLog$ problem. C chooses public keys for all parties except for some party $\hat{B}$ and stores the associated secret keys. It sets the public key of party $\hat{B}$ to $B = g^b$. Additionally, it is given access to a signing oracle O^{Sign} that on input an ephemeral public key W outputs the signature on W with respect to the public key of $\hat{B}$.

1. Run E on input 1^k and the public keys for all of the N parties.
2. If E issues a send(z, "initiate") query to session z with $z_{actor} = \hat{B}$, answer it in the following way. Choose $x \in_R \mathbb{Z}_p$ and query the signing oracle on message $X = g^x$ to get $\sigma(X)$. Return $(z_{actor}, X, \sigma(X))$ to E.
   If E issues a send(z, "initiate") query to session z with $z_{actor} = \hat{P} \neq \hat{B}$, then choose $x \in_R \mathbb{Z}_p$ and return $(z_{actor}, X, \sigma(X))$ to E (C knows the long-term secret key of $\hat{P}$).
3. If E issues a send$(z, m = (\hat{P}, X, \sigma))$ query to session z with $z_{actor} = \hat{B}$ and $z_{role} = r$, answer it in the following way. Check whether $X \in G$ and whether σ is a valid signature on message X with respect to the public key of party $\hat{P}$. Choose $y \in_R \mathbb{Z}_p$ and query the signing oracle on message $Y = g^y$ to get $\sigma(Y)$.
   - If an origin-session $\tilde{z}$ exists for session z and $\tilde{z}_{actor} = \hat{B}$, then check whether there is an entry (x, h) in Table 2 such that $x = \hat{B}, \hat{B}, Z, X$ with $Z \in G$ and $DDH(BY, BX, Z) = 1$. If there is such an entry, then store the tuple $((\hat{B}, \hat{B}, Y, X, r), x, h)$ in Table 1. Else (if there is no such entry), choose a random $h \in \{0,1\}^k$ (uniformly at random) and store the following entry in Table 1:

min-sid(s)	x	KDF(x)
...	...	...
$(\hat{B}, \hat{B}, Y, X, r)$	$\hat{B}, \hat{B}, *, X$	h

Table 2. Table 1

     Return the message $(z_{actor}, Y, \sigma(Y))$ to E.
   - If an origin-session $\tilde{z}$ exists for session z and $\tilde{z}_{actor} = \hat{P} \neq \hat{B}$, then compute the secret $Z = (BY)^{x+p}$.
     Check whether there is an entry $(x = (\hat{P}, \hat{B}, Z, X), h)$ in Table 2. If there is such an entry, then store the tuple $((\hat{B}, \hat{P}, Y, X, r), x, h)$ in Table 1. Else (if there is no such entry), choose a random $l \in \{0,1\}^k$ (uniformly at random) and store the entry $((\hat{B}, \hat{P}, Y, X, r), (\hat{P}, \hat{B}, Z, X), l)$ in Table 1.
     Return the message $(z_{actor}, Y, \sigma(Y))$ to E.

  - If no origin-session exists for session z, then check whether there is an entry (x, h) in Table 2 such that $x = \hat{P}, \hat{B}, Z, X$ with $Z \in G$ and $DDH(BY, PX, Z) = 1$.[4] If there is such an entry, then store the tuple $((\hat{B}, \hat{P}, Y, X, r), x, h)$ in Table 1. Else (if there is no such entry), choose a random $l \in \{0,1\}^k$ (uniformly at random) and store the entry $((\hat{B}, \hat{P}, Y, X, r), (\hat{P}, \hat{B}, *, X), l)$ in Table 1.
    Return the message $(z_{actor}, Y, \sigma(Y))$ to E.

   Return the message $(z_{actor}, Y, \sigma(Y))$ to E.[5]
4. If E issues a send$(z, m = (\hat{P}, Y, \sigma))$ query to session z with $z_{actor} = \hat{B}$ and $z_{role} = i$, answer it in the following way. Check whether $X \in G$ and whether σ is a valid signature on message X with respect to the public key of party $\hat{P}$. Check in Table 1 whether the matching session exists. If yes, copy the $(x, KDF(x))$ entry in the row for $\min - \text{sid} = (\hat{B}, \hat{P}, X, Y, i)$ where X denotes the Diffie-Hellman exponential sent in session z. If no, proceed in a similar way as before (previous point).
5. If E issues a send$(z, m = (\hat{Q}, X, \sigma))$ query to session z with $z_{actor} = \hat{P} \neq \hat{B}$ and $z_{role} = r$, answer it in the following way. Check whether $X \in G$ and whether σ is a valid signature on message X with respect to the public key of party $\hat{Q}$. Choose $y \in_R \mathbb{Z}_p$ and compute the secret $Z = (QX)^{y+p}$. Check whether there is an entry $((\hat{Q}, \hat{P}, Z, X), h)$ in Table 2. If yes, then copy the entry to the corresponding min-sid $(\hat{P}, \hat{Q}, Y, X, r)$ in Table 1. If no, then select a random $l \in \{0,1\}^k$ (uniformly at random) and store the entry $((\hat{P}, \hat{Q}, Y, X, r), (\hat{Q}, \hat{P}, Z, X), l)$ in Table 1.
6. If E issues a send$(z, m = (\hat{Q}, Y, \sigma))$ query to session z with $z_{actor} = \hat{P} \neq \hat{B}$ and $z_{role} = i$, answer it in the following way. Check whether $Y \in G$ and whether σ is a valid signature on message Y with respect to the public key of party $\hat{Q}$. Check in Table 1 whether the matching session exists. If yes, copy the $(x, KDF(x))$ entry in the row for $\min - \text{sid} = (\hat{P}, \hat{Q}, X, Y, i)$ where X denotes the Diffie-Hellman exponential sent in session z. If no, compute the secret $Z = (QY)^{x+p}$ and check whether there is an entry $((\hat{P}, \hat{Q}, Z, X), h)$ in Table 2. If this is the case, then copy this entry into Table 1. If not, then select a random $l \in \{0,1\}^k$ (uniformly at random) and store the entry $((\hat{P}, \hat{Q}, X, Y, i), (\hat{P}, \hat{Q}, Z, X), l)$ in Table 1.
7. When E makes a query x of the form $x = (\hat{P}_1, \hat{P}_2, Z, X)$ to the random oracle KDF, where $\hat{P}_1, \hat{P}_2$ are identities of parties and $Z, X \in G$, answer it as follows:
   Check whether there is an entry for x in Table 1 for which $DDH(XA, YB, Z) = 1$ for some corresponding min-sid that contains Y.

   - If yes, then update the x-entry for min-sid with $(\hat{P}_1, \hat{P}_2, Z, X)$ (only if x-entry incomplete) and return the corresponding $KDF(x)$-entry.

[4] If one omits the check $X \in G$ in the protocol description, then the consistency between the tables could not be guaranteed anymore due to the DDH oracle requiring three group elements as input. In case the group element check is omitted, the adversary can succeed in a key-recovery attack via a variant of a small-subgroup attack.

[5] In these three subcases, the matching session cannot exist yet, because the session-key is computed/chosen by C before the message is sent to E.

  - If no, then check whether there is an entry (x, h) in Table 2. If this is the case, then return h. Else choose a random $l \in \{0,1\}^k$ (uniformly at random) and store (x, l) in Table 2 and return l to E.

x	KDF(x)
...	...
x	l

Table 3. Table 2

8. When E makes a query x not of the above form to the random oracle KDF answer it as follows: Check whether there is an entry (x, h) in Table 2.
   - If yes, then return h.
   - If no, then choose a random $l \in \{0,1\}^k$ (uniformly at random), store (x, l) in Table 2 and return l to E.
9. Corrupt and Ephemeral-key queries are answered in the appropriate way (C knows the secret keys of the parties, except for party $\hat{B}$, and has chosen the ephemeral secret keys in the sessions (as answer to a Send query)).
10. Session-Key-Reveal queries are answered by lookup in Table 1.
11. When E outputs an element $w \in \mathbb{Z}_q$, output w as well.

C correctly solves the GAP discrete logarithm problem with probability at least $\frac{1}{N}P(S_2)$ which implies that $P(S_2) \leq N * P(GAP - DLog_{C,G}(k) = 1)$. Since, by assumption, the GAP discrete logarithm problem is hard relative to G, we conclude that $P(S_2)$ is negligible.

Theorem 3. *Under the $GAP - CDH$ assumption in the cyclic group G of prime order p, using a deterministic signature scheme that is existentially unforgeable under adaptively chosen-message attacks, our protocol is a secure authenticated key-exchange protocol according to Definition 8, when KDF is modeled as a random oracle. The adversary E's advantage for distinguishing a session key from a random key is bounded by*

$$Adv_E^{\Pi}(k) \leq \frac{(q_s + q_{ro})^2}{2^k} + \frac{q_s^2 + 2Nq_s}{p} + 2Nq_s Adv_M^{Sig}(k) + 2q_s^2 q_{ro} Adv_C^{GAP-CDH}(k) + \frac{q_s}{2^k},$$

where N is an upper bound on the number of parties and q_s, q_{ro} are upper bounds on the number of activated sessions and random oracle queries by the adversary.

It is straightforward to verify the first condition of Definition 8, i. e., that matching sessions compute the same key. We show next that the second condition of Definition 8 holds, i. e., the adversary has the above advantage in distinguishing the session key from a random key.

We present a security proof structured as a sequence of games where $KDF : \{0,1\}^* \to \{0,1\}^k$ is modeled as a random oracle. Let N, q_s, q_{ro} be upper bounds on the number of parties, activated sessions and random oracle queries by the adversary. We denote by S_i the event that the adversary E correctly guesses the bit chosen by the challenger to answer the test-session query in Game i and by $\alpha_i := |2P(S_i) - 1|$ the advantage of adversary E in Game i.

Proof. The proof proceeds by the following sequence of games.

Game 0 This game reflects the real interaction of adversary E with the protocol. The challenger chooses a bit b at random. When $b = 0$, he returns the real session-key to E in answer to the test-session query, otherwise he returns a random key from the set $\{0,1\}^k$.

Game 1 [Transition based on a small failure event] Let Event R be the event that the random oracle for KDF produces a collision. When Event R occurs, the attack game halts.
`Analysis of Game` 1: Game 0 is identical to Game 1 up to the point in the experiment where event R occurs for the first time. Moreover, we have that

$$P(R) = \binom{q_{ro} + q_s}{2} \frac{1}{2^k} \leq \frac{(q_s + q_{ro})^2}{2 * 2^k}.$$

Hence, by the Difference Lemma,

$$|P(S_0) - P(S_1)| \leq P(R) \leq \frac{(q_s + q_{ro})^2}{2 * 2^k},$$

and therefore

$$\begin{aligned}\alpha_0 &= |2P(S_0) - 1| = 2|P(S_0) - P(S_1) + P(S_1) - 1/2| \\ &\leq 2(|P(S_0) - P(S_1)| + |P(S_1) - 1/2|) \\ &\leq \frac{(q_s + q_{ro})^2}{2^k} + \alpha_1.\end{aligned}$$

Game 2 [Transition based on a small failure event] Let Event N be the session-specific failure event that there exist two distinct sessions s and s' that choose the same ephemeral private key. As soon as event N occurs, the attack game stops.
`Analysis of Game` 2: Game 1 is identical to Game 2 up to the point in the experiment where event N occurs for the first time. Moreover, we have that

$$P(N) = \binom{q_s}{2} \frac{1}{p} \leq \frac{q_s^2}{2p}.$$

The Difference Lemma yields that

$$|P(S_1) - P(S_2)| \leq P(N) \leq \frac{q_s^2}{2p}.$$

So

$$\begin{aligned}\alpha_1 &= |2P(S_1) - 1| = 2|P(S_1) - P(S_2) + P(S_2) - 1/2| \\ &\leq 2(|P(S_1) - P(S_2)| + |P(S_2) - 1/2|) \\ &\leq \frac{(q_s)^2}{p} + \alpha_2.\end{aligned}$$

Game 3 [Transition based on a small failure event] Let Event U be the session-specific failure event that there exists a session t and a party $\hat{P}$ such that the long-term public key of $\hat{P}$ equals the ephemeral public key chosen in session t. When event U occurs, the attack game stops.

As we will see later, Game 3 is useful for the analysis of Game 6. We need to prevent the scenario where revealing the long-term secret key of some party implies learning the ephemeral secret key used in some session, because they are identical, without explicitly issuing an ephemeral-key query against the session and vice-versa.

`Analysis of Game` 3: Game 2 is identical to Game 3 up to the point in the experiment where event U occurs. We have that

$$P(U) \leq \frac{Nq_s}{p}.$$

The Difference Lemma yields that

$$|P(S_2) - P(S_3)| \leq P(U) \leq \frac{Nq_s}{p}.$$

So

$$\begin{aligned}\alpha_2 &= |2P(S_2) - 1| = 2|P(S_2) - P(S_3) + P(S_3) - 1/2| \\ &\leq 2(|P(S_2) - P(S_3)| + |P(S_3) - 1/2|) \\ &\leq \frac{2Nq_s}{p} + \alpha_3.\end{aligned}$$

Game 4 [Transition based on a large failure event (see [10], [7])] 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 is the target session on which the challenger wants the adversary to be tested. We denote the m-th activated session by s^*. Let event T be the event that the target session is not the test session. If event T occurs, then the attack game halts and the adversary outputs a random bit.

`Analysis of Game` 4: The transition from Game 3 to Game 4 is based on a large failure event, as introduced by Dent in [10]. Event T is non-negligible, the environment can efficiently detect it and T is independent of the output in Game 3 (i.e. $P(S_3|T) = P(S_3)$). If T does not occur, then the attacker E will output the same bit in Game 4 as it did in Game 3 (so that $P(S_4|T^c) = P(S_3|T^c) = P(S_3)$). If event T occurs in Game 4, then the attack game halts and the adversary E outputs a random bit (so that $P(S_4|T) = 1/2$). We have,

$$\begin{aligned}P(S_4) &= P(S_4|T)P(T) + P(S_4|T^c)P(T^c) \\ &= \frac{1}{2}P(T) + P(S_3)P(T^c) \\ &= P(T^c)(P(S_3) - \frac{1}{2}) + \frac{1}{2}.\end{aligned}$$

Hence,

$$\alpha_4 = |2P(S_4) - 1| = P(T^c)|2P(S_3) - 1| = \frac{1}{q_s}\alpha_3.$$

Game 5 [Transition based on a small failure event] This game is the same as the previous one except that when a forgery event with respect to the long-term public key of party s^*_{peer} occurs, the attack game halts.

A forgery event F with respect to the public long-term key $pk_{\hat{P}}$ of some party $\hat{P}$ occurs when adversary E issues a send$(s^*, (\hat{P}, h, \sigma))$ query such that

- $Vrfy_{pk_{\hat{P}}}(h, \sigma) =$ accept, i.e. σ is a valid signature on h with respect to $pk_{\hat{P}}$,
- $(\hat{P}, h, \sigma)$ has never been output by party $\hat{P}$ in response to a send(.) query (i.e. h is a fresh message),
- the message $(\hat{P}, h, \sigma)$ is accepted by s^*_{actor} during the test-session s^* (according to the protocol) and leads to its completion.

In the subsequent game, we can therefore assume that there exists a (unique, by distinct randomness assumption) origin-session s with $s_{actor} = s^*_{peer}$ and $s^*_{recv} = s_{send}$.
`Analysis of Game 5`: Suppose that the adversary E chooses an ephemeral secret/public key pair on his own with intent to impersonate party s^*_{peer} to session s^*. Then we can distinguish between the following two cases:

1. If E issues a corrupt(s^*_{peer}) query before the completion of session s^*, then this query would render session s^* un-fresh. This would cause Game 4 to abort since the target session would be different to the test-session. Recall that the test-session query can only be issued to a completed session that is fresh by the time the query is issued. Hence this case can be excluded.
2. If E does not issue a corrupt(s^*_{peer}) query before the completion of session s^*, then he can only impersonate party s^*_{peer} to session s^* by forging a signature on his message with respect to the long-term public key of party s^*_{peer}.

Claim. We have $|P(S_4) - P(S_5)| \leq P(F)$.

Proof. It is obvious that if event F does not occur, then Games 4 and 5 proceed identically (i.e. $S_4 \wedge F^c \Leftrightarrow S_5 \wedge F^c$). The Difference Lemma yields that $|P(S_4) - P(S_5)| \leq P(F)$.

Claim. If the signature scheme is existentially unforgeable under adaptively chosen message attacks, then $P(F)$ is negligible.

Proof. Consider the following algorithm M which uses adversary E as a subroutine. The algorithm is given a public key pk. It selects at random one of the N parties and sets its public key to pk. We denote this party by $\hat{P}$, its public key by $pk_{\hat{P}} = pk$ and the corresponding secret key by $sk_{\hat{P}}$. Further, the algorithm M chooses public keys for all other parties and stores the associated secret keys. Additionally, it is given access to a conditional oracle O^{Sign} which works as follows.[6]

[6] 0 represents a corrupt(P) query by E. If the test-session is incomplete, then the forger M can monitor the end of the test-session (i.e. point in time where (test-)session-key is established) since he computes the session-key on behalf of the party.

```
input: ($g^x$ for some $x \in \mathbb{Z}_p$) or 0)
 if test-session incomplete and $m \neq 0$ then
   output $Sign_{sk_{\hat{P}}}(m)$
 else
   output $sk_{\hat{P}}$
 end if
```

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 query to session z, answer it in the following way.
   - If $z_{actor} \neq \hat{P}$, then choose $x \in_R \mathbb{Z}_p$, compute $\sigma(g^x) = Sign_{sk_{z_{actor}}}(g^x)$ and return $(z_{actor}, g^x, \sigma(g^x))$ to E.
   - If $z_{actor} = \hat{P}$, then choose $b \in_R \mathbb{Z}_p$ and query the signature oracle on message g^b to get $\sigma(g^b)$. Store the pair $(g^b, \sigma(g^b))$ in a table L (initially empty) and return $(z_{actor}, g^b, \sigma(g^b))$ to E.
3. If E makes a send query of the form send$(s^*, (\hat{P}, h, \sigma))$ (where σ is a valid signature on h with respect to $pk_{\hat{P}}$) before the completion of the test-session s^* and $(h, \sigma) \notin L$, then store (h, σ) as a forgery.
4. When E makes a query to the random oracle KDF, answer it in the following way. Store pairs of strings $(.,.)$ in a table, initially empty. When E makes a query x to the random oracle KDF, answer it as follows:
   - If there is an entry (x, h) in the table, return h.
   - If there is no entry for x, then do the following: choose a random $h \in \{0,1\}^k$, store (x, h) in the table and return h to E.
5. Corrupt, Session-Key-Reveal, Ephemeral-key and Send queries are answered in the appropriate way (M knows the secret keys of the parties and has chosen the ephemeral secret keys for all the sessions). A corrupt$(\hat{P})$ query can be answered by M by querying the oracle O^{Sign} on message 0 and get as response the long-term secret key of $\hat{P}$).
6. At the end of E's execution (after it has output its guess b') output "failed" if no forgery has been detected and stored, otherwise return the forgery.

The probability that E breaks the protocol by 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, $P(F) \leq NAdv_M^{Sign}(k)$.

Game 6 [Transition based on a small failure event] In this game, we replace the session-key of the test-session K_{s^*} by a key chosen uniformly at random from the set $\{0,1\}^k$. The session key of the matching session (if it exists) is also replaced by the same random key.[7]

`Analysis of Game` 6: Here we assume w.l.o.g. that $s^*_{role} = i$. The analysis works similarly for $s^*_{role} = r$.

[7] As a reminder, note that, for all completed sessions $s \neq s^*$ it holds that either $K_s \neq K_{s^*}$ or $s \sim s^*$. If $K_s \neq K_{s^*}$, then a session-key query on session s presents no problem (through key-independence) and if $s \sim s^*$, then a session-key query on session s is not allowed by definition of the security model (i.e. session-key query not allowed on matching session).

Claim. Let us denote by X, Y the ephemeral public keys sent, received during the test-session s^*. Then we have

$$|P(S_5) - P(S_6)| \leq P(Q) \leq q_s q_{ro} Adv_C^{GAP-CDH}(k),$$

where Q denotes the event that at any point during its execution, adversary E queries message $(s^*_{actor}, s^*_{peer}, Z, X)$ to the random oracle for KDF (where $Z = (Y * pk_{s^*_{peer}})^{x+sk_{s^*_{actor}}}$ for $x = DLOG_g(X)$). We will analyze the probability of event Q with respect to Game 6.

Proof. It is obvious that if event Q does not occur, then Games 5 and 6 proceed identically (i.e. $S_5 \wedge Q^c \Leftrightarrow S_6 \wedge Q^c$). The Difference Lemma yields that $|P(S_5) - P(S_6)| \leq P(Q)$.

Claim (A). If the $GAP - CDH$ problem is hard relative to our experiment and KDF is modeled as a random oracle, then $P(Q)$ is negligible.

We denote by s the origin-session for the test-session s^* (which exists in this game). There are four different scenarios to consider.

1. The adversary E issued the queries corrupt(s^*_{actor}) and corrupt(s^*_{peer}).
2. The adversary E issued the queries corrupt(s^*_{actor}) and ephemeral-key(s).
3. The adversary E issued the queries corrupt(s^*_{peer}) and ephemeral-key(s^*).
4. The adversary E issued the queries ephemeral-key(s) and ephemeral-key(s^*).

`Analysis of scenario 1 and proof of Claim B.2 w.r.t. scenario 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 $DH_g(X, Y) = g^{xy}$ (and thus, breaking the CDH assumption in the group G) since

$$g^{xy} = (Y * pk_{s^*_{peer}})^{x+sk_{s^*_{actor}}} * (Y^{-sk_{s^*_{actor}}} * X^{-sk_{s^*_{peer}}} * pk_{s^*_{peer}}^{-sk_{s^*_{actor}}}).$$

Proof. We solve the CDH problem with probability $\frac{1}{q_{ro}q_s}P(Q)$ where $P(Q)$ must be negligible since 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 $(X = g^x, Y = g^y)$ of elements from G as an instance of the 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 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. When the test-session at s^*_{actor} is initiated (by E), set the ephemeral public key of session s^* to X and answer the query with the message $(s^*_{actor}, X, \sigma(X)) = (s^*_{actor}, X, Sign_{s^*_{actor}}(X))$.
3. When session s is activated (either by an incoming message or by an initiation request), set the ephemeral public key of session s to Y and answer the query with the message $(s_{actor}, Y, \sigma(Y)) = (s_{actor}, Y, Sign_{s_{actor}}(Y))$.

4. Store pairs of strings $(.,.)$ in a table, initially empty. When E makes a query x to the random oracle KDF, answer it as follows:
   – If there is an entry (x, h) in the table, return h.
   – If there is no entry for x, then do the following: choose a random $h \in \{0,1\}^k$, store (x, h) in the table and return h to E.
5. In case of the test-session query, return either the real session-key or a random value.
6. Corrupt, Session-Key-Reveal, Ephemeral-key and Send queries are answered in the appropriate way (C knows the secret keys of the parties and has chosen the ephemeral secret keys except for the test-session and its origin-session).
7. At the end of E's execution (after it has output its guess b'), let $x_1, ..., x_w$ (with $w \leq q_{ro}$) be the list of all oracle queries made by E. Choose a random i for which x_i is of the form $(s^*_{actor}, s_{actor}, Z, X)$ and output $Z * Y^{-sk_{s^*_{actor}}} * X^{-sk_{s_{actor}}} * pk_{s_{actor}}^{-sk_{s^*_{actor}}}$.

C correctly computes the CDH instance with probability at least $\frac{1}{q_s q_{ro}} P(Q)$ which implies that $P(Q) \leq q_s q_{ro} Adv_C^{CDH}(k)$.

`Analysis of scenario 2 and proof of Claim B.2 w.r.t. scenario 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 s^*_{actor} and the ephemeral key of the origin-session s to 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 $s_{actor} = s^*_{peer}$, since

$$g^{xb} = (Y * pk_{s^*_{peer}})^{x + sk_{s^*_{actor}}} * (X^{-y} * Y^{-sk_{s^*_{actor}}} * pk_{s^*_{peer}}^{-sk_{s^*_{actor}}}).$$

Proof. We solve the $GAP - CDH$ problem with probability $\frac{1}{q_{ro} q_s} P(Q)$ 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 $(X = g^x, B = g^b)$ 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 public keys for all parties except for party s^*_{peer} and stores the associated secret keys. It sets the public key of party s^*_{peer} to $B = g^b$. Additionally, it is given access to a signing oracle O^{Sign} that on input an ephemeral public key Y outputs the signature on Y with respect to the public key of s^*_{peer}.

1. Run E on input 1^k and the public keys for all of the N parties.
2. When the test-session at s^*_{actor} is initiated (by E), set the ephemeral public key of session s^* to X and answer the query with the message $(s^*_{actor}, X, \sigma(X)) = (s^*_{actor}, X, Sign_{s^*_{actor}}(X))$.
3. Send, Session-Key and random oracle queries by the adversary E are processed as in the proof of Lemma 4. In particular, when session s is activated (either by an incoming message or by an initiation request), set the ephemeral public key of session s to Y and answer the query with the message $(s_{actor}, Y, \sigma(Y)) = (s_{actor}, Y, Sign_{s_{actor}}(Y))$.

4. In case of the test-session query, return either the real session-key or a random value.
5. Corrupt and Ephemeral-key queries are answered in the appropriate way (C' knows the secret keys of the parties, except for party s^*_{peer}, and has chosen the corresponding ephemeral secret keys except for the test-session s^*).
6. At the end of E's execution (after it has output its guess b'), let $x_1, ..., x_w$ (with $w \leq q_{ro}$) be the list of all oracle queries made by E. Choose a random i for which x_i is of the form $(s^*_{actor}, s_{actor}, Z, X)$ and output $Z * X^{-y} * Y^{-sk_{s^*_{actor}}} * pk_{s^*_{peer}}^{-sk_{s^*_{actor}}}$.

C' correctly computes the $GAP-CDH$ instance with probability at least $\frac{1}{q_s q_{ro}} P(Q)$ which implies that $P(Q) \leq q_s q_{ro} Adv_C^{GAP-CDH}(k)$.

Remark 1. The analyses and proofs of scenario 3 and 4 are similar to the previous analyses and proofs.

Claim. It holds that $P(S_6) = \frac{1}{2} + \frac{1}{2^{k+1}}$.

Proof. Let D be the event that E correctly guesses the session-key of the test-session. By definition of the underlying security model, the adversary is not allowed to perform a session-key query on the test-session or a matching session (if it exists). This implies that $P(D) = \frac{1}{2^k}$ (since it is generated uniformly at random via the random oracle) and $P(S_6|D^c) = \frac{1}{2}$. Thus,

$$P(S_6) = P(S_6|D)P(D) + P(S_6|D^c)P(D^c) = \frac{1}{2} + \frac{1}{2^{k+1}}.$$

This completes the proof of Theorem 1.

C UKS attacks in our model

We define resilience of a key-exchange protocol against UKS attacks.

Definition 13 (UKS attack). *A key-exchange protocol Π relative to a security model M is said to be resilient against UKS attacks if no PPT adversary can establish, with more than negligible probability, a fresh session s and a session s' between uncorrupted (i. e., the adversary does not know their long-term secret keys) parties such that*

1. *the computed session-keys $K_s, K_{s'}$ are identical,*
2. *$s'_{peer} \neq s_{actor}$ (where s'_{peer} denotes the intended peer of session s' and s_{actor} denotes the actor of session s),*
3. *no session-specific private data from both sessions (such as ephemeral private data or session-keys) is leaked to the adversary.*

As a straightforward consequence of being secure according to Definition 8, a key-exchange protocol is resilient against UKS attacks, as the following proposition shows.

Proposition 2. *If a protocol Π is secure with respect to Definition 8, then it is resilient against UKS attacks in the sense of Definition 13.*

Proof. If there is a PPT adversary who creates a UKS attack for some fresh session s with non-negligible probability, then there exists a PPT adversary who can break the security of the protocol in a security experiment with non-negligible probability by issuing a session-key reveal query on some session s' for which $K_s = K_{s'}$ and $s'_{peer} \neq s_{actor}$. Notice that session s' is non-matching to session s since $s'_{peer} \neq s_{actor}$, hence the query session-key(s') is allowed.

D Version history

First version: June 6, 2011

Second version: July 22, 2011

– Updated the security model (and proof) to allow corruption of the peer to the test-session before the completion of the test-session under some condition.
– Added remark related to dynamic generation of key pairs and DSKS attacks to the introduction of Section 5.
– Added additional related work that was published after our initial report was released.

Third version: October 26, 2011

– Added group check in protocol description to prevent variants of small-subgroup attacks (see also [23]), updated proof and efficiency accordingly.
– Added Appendix A on the comparison between the eCK-PFS model and the extended-CK model.
– In Appendix B, added proof that, under the $GAP - DLog$ assumption in the group G, our protocol is now resilient against key-recovery attacks.
– Minor changes in the analysis of Game 6 of the main security proof.