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

June 6, 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 strong key-exchange security notion 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 and efficiency.

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]:

\[
\begin{align*}
\hat{A} &\quad& \hat{B} \\
1. & x \in_R \mathbb{Z}_q ; & X := g^x &\quad& y \in_R \mathbb{Z}_q ; & Y := g^y \\
2. & \text{verify signature} & Y, \text{Sign}_{sk_B}(Y, X, \hat{A}) &\quad& K_B := X^y \\
3. & K_A := Y^x & \text{verify signature} & & \text{Sign}_{sk_A}(X, Y, \hat{B}) \\
\end{align*}
\]

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 [19]. 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 [18], shown below. Here, $G = \langle g \rangle$ denotes a subgroup of $\mathbb{Z}_p^*$ of prime order $q$ where $q \mid p - 1$ for some large prime $p$, $\pi$ is defined as $x \mod 2^w + 2^w$, 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 $B = g^b$.

Due to the lack of message origin authentication, a recipient must check that the received element $X$ belongs to the group $G$ (that is, $X^q \mod p = 1$) 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 [16]. 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 [16]. 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 other 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 [20], 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 [20], significantly weakens the level of deniability.

In this work, we propose a protocol that combines the message exchange of a (insecure) 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. Second, because the ephemeral public keys are authenticated, stronger guarantees can be directly derived, e.g., removing the need for group element checks, and providing message origin authentication.

It may seem at first glance that the use of signatures has two drawbacks: loss of deniability, and increased (computational and communication) complexity. With respect to the first potential drawback, we show that it is possible to establish a very strong form of deniability, which we call peer-and-time deniability, even though signatures are used. 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. However, they can deny any more specific allegations.

Addressing the second potential drawback, we show that the computational complexity of our protocol is close to that of MQV. Compared to such implicitly authenticated one-round protocols, our protocol requires more communication bandwidth (the signatures). However, our protocol additionally provides perfect forward secrecy as well as message authentication. Hence, when perfect forward secrecy is required, our protocol is also more communication efficient than the corresponding three-message variant of MQV.

**Contributions.** First, we define a new security notion for key-exchange protocols. Our security notion is a strengthening of the extended-CK model [17] 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 [21], but stronger than the deniability offered by protocols such as SIGMA [15]. Peer-and-time deniability allows parties to deny communicating with particular peers as well as being alive at specific times.

Third, we propose an efficient 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, no other one-round key exchange protocol satisfies the same strong key-exchange security properties, 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 $g^{uv}$.

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

**Definition 2 (Signature Scheme [14]).** A signature scheme is a tuple of three polynomial-time algorithms $(\text{Gen}, \text{Sign}, \text{Vrfy})$ satisfying the following:

1. The probabilistic key-generation algorithm $\text{Gen}$ takes as input a security parameter $1^k$ and outputs a public/private key pair $(pk, sk)$.
2. The (possibly probabilistic) signing algorithm $\text{Sign}$ takes as input a private key $sk$ and a message $m \in \{0, 1\}^*$. It outputs a signature $\sigma := \text{Sign}_{sk}(m)$.
3. The deterministic verification algorithm $\text{Vrfy}$ takes as input a public key $pk$, a message $m$, and a signature $\sigma$. It outputs a bit $b$, with $b = 1$ meaning valid and $b = 0$ meaning invalid. We write $b := \text{Vrfy}_{pk}(m, \sigma)$.

**Definition 3 (EU-CMA [14]).** A signature scheme $\Sigma = (\text{Gen}, \text{Sign}, \text{Vrfy})$ is existentially unforgeable under an adaptive chosen-message attack if for all probabilistic polynomial-time adversaries $A$, there exists a negligible function $\text{negl}$ such that $\text{Adv}_{A}^{\text{Sig}}(k) \leq \text{negl}(k)$, where $\text{Adv}_{A}^{\text{Sig}}(k)$ 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

We define a security notion for key exchange (KE) protocols. Our work builds on Bellare-Rogaway style indistinguishability-based security notions for key exchange protocols [3]. Intuitively, our KE-security notion is a strengthening of the extended-CK model [17]. 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 $(s_{\text{actor}}, s_{\text{peer}}, s_{\text{send}}, s_{\text{recv}}, s_{\text{role}})$, where
denote the identities of the owner and intended peer of the session \(s\), and \(s_{\text{role}} \in \{i\ \text{(initiator)}, r\ \text{(responder)}\}\) denotes the role that the session is executing, and \(s_{\text{send}}, s_{\text{recv}}\) are sequences of timely ordered messages as sent/received by \(s_{\text{actor}}\) 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 [17].

**Definition 4 (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

\[
s_{\text{actor}} = s'_{\text{peer}} \land s_{\text{peer}} = s'_{\text{actor}} \land s_{\text{send}} = s'_{\text{recv}} \land s_{\text{recv}} = s'_{\text{send}} \land s_{\text{role}} \neq s'_{\text{role}}.
\]

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

\[
s'_{\text{send}} = s_{\text{recv}} \land s_{\text{peer}} = s'_{\text{actor}}.
\]

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. \text{send}(s, v). This query models the adversary sending message \(v\) to an active session \(s\), or initiating a session, via a \text{send}(s, ”initiate”) query. The adversary is given the response from \(s_{\text{actor}}\) according to the protocol.
2. \text{corrupt}(X). With this query \(E\) learns the long-term key of party \(X\).
3. \text{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 6 (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 ˜s exists that matches s, then W does not include session-key(˜s)
3. W does not include a corrupt(s.peer) query before the completion of session s
4. W does not include both corrupt(s.actor) and ephemeral-key(s)
5. if s′ is an origin-session for the session s, then W does not include both corrupt(s.peer) and ephemeral-key(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.** 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 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

\[
\text{Adv}_E^\Pi(k) = |2P(b = b') - 1|.
\]
Definition 7. A key exchange protocol $\Pi$ 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 $\text{negl}$ in the security parameter $k$ such that $\text{Adv}_E^\Pi(k) \leq \text{negl}(k)$.

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, after the completion of the test-session, the adversary learns the ephemeral keys of a session as well as the long-term secret key of that party, 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 (matching sessions are by definition also origin sessions). 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 [13]). In Appendix B 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 [21]. 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 [21], 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 [21], 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 \( A \) and a possibly dishonest party \( B \), \( A \) should be able to deny having been intentionally involved in a protocol run with \( B \). That is, party \( A \) 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 \( A \) and an adversary-controlled party \( B \). Informally, a protocol is said to be peer-deniable if, after a protocol execution between an adversary-controlled party \( B \) and some honest party \( A \), party \( B \) cannot convince a judge that party \( A \) was intentionally involved in that execution with \( B \). Party \( A \) should be able to deny interaction with \( B \) by arguing that its signed messages could have been generated while executing a protocol session with another party. If \( A \) is in the initiator role, then \( A \) 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 \( B \) can be efficiently simulated given access to an oracle of \( A \) in a protocol execution with some other party \( C \neq B \).

Our formal definitions of deniability are inspired by the works of Dwork, Naor and Sahai [10] as well as of Di Raimondo, Gennaro and Krawczyk [21]. The main difference to the definition of partial-deniability [21] 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 [21], we model the generation of signatures on behalf of uncorrupted parties by giving the simulator access to oracles. We denote by \( \Sigma^{A,B}_I \) 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, \( \Sigma_{R}^{A,B} \) denotes an \( R \)–oracle representing a responder session at party \( \hat{A} \) with peer \( \hat{B} \).

**Definition 8 (peer-deniability).** Let \( \Pi \) be an (authenticated) key-exchange protocol. We say that an initiator role of \( \Pi \) 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 \( \Sigma_{I}^{A,B} \) and \( \Sigma_{R}^{A,B} \) (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 \( \Pi \) is peer-deniable if both roles are peer-deniable.

Under the key-awareness assumption (see [21]) for the key derivation function used, the four-message version of the SIGMA protocol is peer-deniable with respect to Definition 8. 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 9 (peer-and-time deniability).** Let \( \Pi \) be an (authenticated) key-exchange protocol. We say that an initiator role of \( \Pi \) 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 \( \Sigma_{I}^{A,B} \) and \( \Sigma_{R}^{A,B} \).

The simulation should proceed in four steps, as follows:

1. At time \( \tau_0 \), 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 \( A \) and \( B \)) as well as the secret keys of corrupted parties.
2. The simulator is allowed to activate and query the oracles in such 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 $\Sigma^A_1$ and $\Sigma^A_R$ 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 $\Pi$ is peer-and-time deniable if both roles are peer-and-time deniable.

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 $KDF : \{0,1\}^* \rightarrow \{0,1\}^k$ 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.

An execution of the protocol proceeds as follows. First, an initiator $\hat{A}$ chooses an ephemeral secret key $x \in \mathbb{Z}_p$ uniformly at random, signs the group element $g^x$ with her long-term secret key, and sends the message $(\hat{A}, g^x, \text{Sign}_{sk\hat{A}}(g^x))$ to some party, say $\hat{B}$. Upon receipt of this message, the responder $\hat{B}$ verifies the signature on the ephemeral public key received. If the verification is successful, $\hat{B}$ chooses an ephemeral secret key $y \in \mathbb{Z}_p$ uniformly at random, signs the group element $g^y$ with his long-term secret key, sends the message $(\hat{B}, g^y, \text{Sign}_{sk\hat{B}}(g^y))$ intentionally to $\hat{A}$ and computes a session-key. Receiving the latter message, $\hat{A}$ verifies whether the signature is valid and computes a session-key. The session-key $K$ is computed as $KDF(\hat{A}, \hat{B}, (YB)^{x+a}, g^x)$ where the role of the session specifies the order of the identities as input to the key derivation function $KDF$.

We assume that whenever a verification of a signature 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).

\[
\begin{align*}
\hat{A} : & \quad a, A = g^a \\
\hat{A} : & \quad X = g^x \\
\hat{A} \rightarrow & \quad \hat{A}, X, \text{Sign}_{sk\hat{A}}(X) \quad \text{Verify signature} \\
\hat{B} : & \quad b, B = g^b \\
\hat{B} : & \quad Y = g^y \\
\hat{B} \leftarrow & \quad \hat{B}, Y, \text{Sign}_{sk\hat{B}}(Y) \\
K_{\hat{A}} = & \quad KDF(\hat{A}, \hat{B}, (YB)^{x+a}, X) \\
K_{\hat{B}} = & \quad KDF(\hat{A}, \hat{B}, (XA)^{y+b}, X)
\end{align*}
\]
Design choices. Using signatures for the message exchange is a mechanism to prevent impersonation and man-in-the-middle attacks. If an attacker does not know a party’s long-term secret key and forging signatures is hard, then he cannot impersonate that party to a session. Note also that because the messages are authenticated, the parties do not need to perform group element checks.

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 [15, 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 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.\(^1\)

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 A. We first state the main security result.

**Theorem 1.** Under the 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 7, when \( KDF \) is modeled as a random oracle. The adversary \( E \)’s advantage is bounded by

\[
Adv_{E}^\Pi(k) \leq \frac{(q_s + q_{ro})^2}{2^k} + \frac{q_s^2 + 2q_s}{p} + 2Nq_sAdv_{M}^{Sig}(k) + 2q_s^2 q_{ro}Adv_{CDH}^{CDH}(k) + \frac{q_s}{2^k},
\]

where \( N \) is an upper bound on the number of parties and \( q_s, q_{ro} \) are upper bounds on the number of initiated 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

\(^1\) 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.
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 after its completion. 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 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 unfresh) and break the CDH with respect to a corresponding instance. The hardness of the 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 9.

*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 $\Sigma_{\hat{I}}^{\hat{A}, \hat{C}}$ where $\hat{C} \neq \hat{B}$. The simulator activates the oracle $\Sigma_{\hat{I}}^{\hat{A}, \hat{C}}$ and gets as response the message $m = (\hat{A}, X, \text{Sign}_{sk_{\hat{A}}}(X))$ at time $\tau_1$. 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 the signature $\sigma$ on $Y$ with respect to the public key of $\hat{B}$ is valid, and if the verification is 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 $\Sigma_{\hat{I}}^{\hat{A}, \hat{C}}$ where $\hat{C} \neq \hat{B}$. The simulator $S_R$ first activates the oracle $\Sigma_{\hat{I}}^{\hat{A}, \hat{C}}$ and gets as response the message $m = (\hat{A}, X, \text{Sign}_{sk_{\hat{A}}}(X))$ at time $\tau_1$. 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 the verification of the signature
σ on Y with respect to the public key of 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

Our main concern is to achieve maximum security in one round. However, our protocol is surprisingly efficient with respect to the properties that it achieves.

**Computational complexity.** A run of the protocol requires for each party two exponentiations (one for the ephemeral public key and one for the session key), 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.

Alternatively, the “NAXOS trick” [17] 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 security notion is a strengthening of extended-CK [17]. 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 our model.

Deniable authentication was first introduced by Dwork et al. [10] using the simulation paradigm. Deniability of key-exchange protocols has been formalized by Di Raimondo et al. [21]. 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 [12]. 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 computations (checking that the received value is of prime order) and more communication (for the zero-knowledge proofs). The YAK protocol also provides features for scenarios in which there is no one-to-one mapping between communication nodes and owners of private keys. Our protocol can be adapted to cater for this case by including the unique user identifiers (SignerID) of the parties in the input to the key derivation function.

The modified-Okamoto-Tanaka (mOT) protocol by Gennaro, Krawczyk and Rabin [11] 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.

In comparison to implicitly authenticated protocols, our use of signatures allows us to omit the group element check and the half-exponentiation. Because the signature generation can be performed off-line, the on-line computational costs of our protocol are similar to those of MQV: instead of MQV’s on-line group element check and half-exponentiation, our protocol requires an on-line signature verification. Furthermore, unlike the one-round (two-message) versions of (H)MQV, our protocol provides full perfect forward secrecy.
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 currently the only one-round key exchange protocol that satisfies this strong security notion, which implies both extended-CK security and perfect forward secrecy.

Additionally, because the signed messages are authenticated, we no longer need to consider adversary-generated terms, unlike MQV-style protocols. This removes the need for group element checks and mitigates the risk of exploitation of particular values. As a concrete example, consider the case in which weaknesses in the key computation or the underlying cryptographic primitives are found for some particular values. In our protocol, such weaknesses cannot be exploited by arbitrary parties unless the signature scheme is broken.

In terms of efficiency, our protocol is slightly more expensive than protocols that offer only weak perfect forward secrecy. In comparison to MQV, the additional cost of generating and verifying the signatures is mitigated by dropping the group element check and not requiring MQV's half-exponentiation. 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.

References

A Security Proof

In this appendix we provide detailed proofs of Lemma 1 and Theorem 1. We first recall the Difference Lemma introduced by Shoup in [22]. 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 \( \neg F \)).

**Lemma 2 (Difference Lemma [22]).** Let \( A, B, F \) be events defined on some probability space, and suppose that \( A \land F^c \leftrightarrow B \land F^c \). Then

\[
|P(A) - P(B)| \leq P(F).
\]

**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 $s \neq t$ if and only if $(s_{\text{actor}} \neq t_{\text{peer}} \lor s_{\text{peer}} \neq t_{\text{actor}} \lor s_{\text{send}} \neq t_{\text{recv}} \lor s_{\text{recv}} \neq t_{\text{send}} \lor s_{\text{role}} = t_{\text{role}})$. 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**: $t_{\text{role}} = s_{\text{role}} = i$. Then we must have that $Z_s = Z_t$ so that $K_s = K_t$.

**Case 2**: $t_{\text{role}} = s_{\text{role}} = r$. 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 $\tau_m$ and session $s$ receives a message at time $\tau_n$ with $m > n$. Then session $t$ establishes the session-key first which involves the computation of the group element $(Q_t * pk_{t_{\text{peer}}})(z_t + sk_{t_{\text{actor}}})$ (where $z_t$ denotes the discrete logarithm of $Z_t = g^{z_t}$ and $pk_{t_{\text{peer}}}, sk_{t_{\text{actor}}}$ denote the public key of $t_{\text{peer}}$ and secret key of $t_{\text{actor}}$, respectively). Necessary conditions to have $K_s = K_t$ are that $t_{\text{actor}} = s_{\text{actor}}$ and $t_{\text{peer}} = s_{\text{peer}}$. This implies that $Z_s = Z_t$.

Note that we do not need to consider the case where $s_{\text{role}} = i$ and $t_{\text{role}} = r$. To get $K_s = K_t$, it must hold that $s_{\text{actor}} = t_{\text{peer}}$ and $s_{\text{peer}} = t_{\text{actor}}$ (by the ordering of the identities as input to the KDF according to their roles and 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 $s_{\text{send}} = t_{\text{recv}}$ and $t_{\text{send}} = s_{\text{recv}}$. Hence, the sessions $s$ and $t$ are matching (that is, $s \sim t$).

**Theorem 3.** Under the 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 7, 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

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

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

It is straightforward to verify the first condition of Definition 7, i.e., that matching sessions compute the same key. We show next that the second condition of Definition 7 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 $KDF : \{0, 1\}^* \rightarrow \{0, 1\}^k$ is modeled as a random oracle. Let $N, q_s, q_{ro}$ be upper bounds on the number of activated parties, initiated 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

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

Hence, by the Difference Lemma,

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

and therefore

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

**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

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

The Difference Lemma yields that

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

So

$$\alpha_1 = |2P(S_1) - 1| = 2|P(S_1) - P(S_2) + P(S_2) - 1/2| \leq 2(|P(S_1) - P(S_2)| + |P(S_2) - 1/2|) \leq \frac{(q_s)^2}{p} + \alpha_2.$$
Game 3 [Transition based on a small failure event] Let Event $U$ be the session-specific failure event that there exists a session $t$ such that the long-term public key of $t_{actor}$ 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 $t_{actor}$ implies learning the ephemeral secret key used in session $t$, because they are identical, without explicitly issuing an ephemeral-key query against session $t$ 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 $P(U) = \frac{q_s}{p}$.

The Difference Lemma yields that

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

So

$$\alpha_2 = |2P(S_2) - 1| = 2|P(S_2) - P(S_3) + P(S_3) - 1/2|$$

$$\leq 2(|P(S_2) - P(S_3)| + |P(S_3) - 1/2|)$$

$$\leq \frac{2q_s}{p} + \alpha_3.$$

Game 4 [Transition based on a large failure event (see [9], [7])] Before the adversary $E$ starts the attack game, the challenger chooses a random value $m \in \mathbb{R}\{1, 2, ..., q_s\}$. 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 [9]. 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 outputs a random bit (so that $P(S_4|T) = 1/2$). We have,

$$P(S_4) = P(S_4|T)P(T) + P(S_4|T^c)P(T^c)$$

$$= \frac{1}{2}P(T) + P(S_3)P(T^c)$$

$$= P(T^c)(P(S_3) - \frac{1}{2}) + \frac{1}{2}.$$ 

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 $pk_P$ of some party $P$ occurs when adversary $E$ issues a send$(s^*, (P, h, \sigma))$ query such that

- $Vrfy_{pk_P}(h, \sigma) = \text{accept}$, i.e. $\sigma$ is a valid signature on $h$ with respect to $pk_P$,
- $(P, h, \sigma)$ has never been output by party $P$ in response to a send(. ) query (i.e. $h$ is a fresh message),
- the message $(P, h, \sigma)$ 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 $s^*_{recv} = s_{send}$ with $s_{actor}$ uncorrupted until the completion of the test-session $s^*$.

**Analysis of Game 5:** Note that the adversary is only allowed to issue a corrupt$(s^*_{peer})$ query after completion of the test-session.

**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. $S_4 \land F^c \iff S_5 \land F^c$). 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 $P$, its public key by $pk_P = pk$ and the corresponding secret key by $sk_P$. 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.\(^2\)

**input:** $(g^x$ for some $x \in \mathbb{Z}_p$) or 0 )

**if** test-session incomplete and $m \neq 0$ **then**

  **output** $Sign_{sk_P}(m)$

**else**

  **output** $sk_P$

**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 $z_{actor} \neq P$, then choose $x \in \mathbb{Z}_p$, compute $\sigma(g^x) = Sign_{sk_{z_{actor}}}(g^x)$ and return $(z_{actor}, g^x, \sigma(g^x))$ to $E$.

\(^2\) 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.
If $z_{actor} = P$, then choose $b \in \mathbb{Z}_p$ 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($s^*$, $(P,h,\sigma)$) (where $\sigma$ is a valid signature on $h$ with respect to $pk_P$) before the completion of the test-session $s^*$ and $(h,\sigma) \notin L$, then store $(h,\sigma)$ as a forgery.

4. When $E$ makes a query to the random oracle $H$, answer it as specified in the analysis of the subsequent game.

5. Corrupt, Session-Key-Reveal, Ephemeral-key and Send queries are answered in the appropriate way ($M$ knows the secret keys of the parties (the one of party $P$ may only be learned after completion of the test-session) and has chosen the ephemeral secret keys for all the sessions). A corrupt($P$) query can be answered by $M$ after the end of the test-session (query the oracle on message 0 and get as response the long-term secret key of $P$).

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 $P$ is bounded above by the probability that $M$ outputs a forgery multiplied by the number of parties, that is, $P(F) \leq N \text{Adv}_{M}^{\text{Sign}}(k)$.

**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.

**Analysis of Game 6:**

**Claim.** We have

$$|P(S_5) - P(S_6)| \leq P(Q) \leq q_{s_{\text{role}}} \text{Adv}_{C}^{\text{CDH}}(k),$$

where $Q$ denotes the event that at any point during its execution, adversary $E$ queries message $(s_{\text{actor}}^{s^*}, s_{\text{peer}}^{s^*}, Z, X)$ to the random oracle for $KDF$ (where $Z = (Y * pk_{s_{\text{peer}}}^{s^*})^x + sk_{\text{actor}}^{s^*}$ for $x = \text{DLOG}_g(X)$). 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. $S_5 \land Q^c \iff S_6 \land Q^c$). The Difference Lemma yields that $|P(S_5) - P(S_6)| \leq P(Q)$.

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

Suppose that $s_{\text{role}}^{s^*} = i$. We denote by $s$ the origin-session for the test-session $s^*$ (which exists in this game). There are four different scenarios to consider.

---

3 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).
We solve the CDH problem with probability

Algorithm $C$, which implies that

Analysis of scenario 1 and proof of Claim A 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 $s^*_\text{actor}$ and $s^*_\text{peer}$ after the completion of the test-session, 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

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

Proof. Suppose that $s^*_\text{role} = i$ (a similar proof works for the responder case). We solve the CDH problem with probability $\frac{1}{q_{ro}q_{sk}} P(Q)$ 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 $\{1, ..., q_{sk}\}$ which reflects the guess that the $n$-th activated session, say session $s$, is the origin-session for session $s^*$. (We have to consider the probability of correctly guessing which authentic message (sent during some session $s$) the adversary will forward to the test-session.) $C$ 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 $s^*_\text{actor}$ is initiated (by $E$), set the ephemeral public key of session $s^*$ to $X$ and answer the query with the message $(s^*_\text{actor}, X, \sigma(X)) = (s^*_\text{actor}, X, \text{Sign}_{s^*_\text{actor}}(X))$.
3. When session $s$ is activated (either by an incoming message or by a start request), set the ephemeral public key of session $s$ to $Y$ and answer the query with the message $(s^*_\text{actor}, Y, \sigma(Y)) = (s^*_\text{actor}, Y, \text{Sign}_{s^*_\text{actor}}(Y))$.
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 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 $(s^*_\text{actor}, s^*_\text{actor}, Z, X)$ and output $Z$.

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


We solve the CDH problem with probability $\frac{1}{C}$ which implies that Analysis of scenario $C$ correctly computes the CDH instance with probability at least signature on $Y$.

Proof. Suppose that $s_{\text{role}}^i = i$ (a similar proof works for the responder case). We solve the CDH problem with probability $\frac{1}{q_{\text{ro}}q_{\text{ro}}^*}P(Q)$ 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 $(X = g^x, B = g^b)$ 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 $s^j$. $C'$ chooses public keys for all parties except for party $s_{\text{actor}}^j$ and stores the associated secret keys. It sets the public key of party $s_{\text{peer}}^j$ to $B = g^b$. Additionally, it is given access to a signing oracle $O^{\text{Sign}}$ that on input an ephemeral public key $Y$ outputs the signature on $Y$ with respect to the public key of $s_{\text{peer}}^j$.

1. Run $E$ on input $1^k$ and the public keys for all of the $N$ parties.
2. When the test-session at $s_{\text{actor}}^j$ is initiated (by $E$), set the ephemeral public key of session $s^j$ to $X$ and answer the query with the message $(s_{\text{actor}}^j, X, \sigma(X)) = (s_{\text{actor}}^j, X, \text{Sign}_{s_{\text{actor}}^j}(X))$.
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 $(s_{\text{actor}}^j, Y, \sigma(Y)) = (s_{\text{actor}}^j, Y, \text{Sign}_{s_{\text{actor}}^j}(Y))$.
4. If $E$ issues a send query to session $z$ with $z_{\text{actor}}^j = s_{\text{peer}}^j$, answer it in the following way. Choose $b \in R \mathbb{Z}_p$ and query the signing oracle on message $g^b$ to get $\sigma(g^b)$. Return $(z_{\text{actor}}^j, g^b, \sigma(g^b))$ to $E$.
5. Store pairs of strings ($\ldots$) 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 $E$.
6. In case of the test-session query, return a random value.
7. Corrupt, Session-Key-Reveal, Ephemeral-key and Send queries are answered in the appropriate way ($C$ knows the secret keys of the parties, except for party $s_{\text{actor}}^j$, and has chosen the corresponding ephemeral secret keys except for the test-session).
8. At the end of $E$’s execution (after it has output its guess $b'$), let $x_1, \ldots, x_w$ (with $w \leq q_{\text{ro}}$) be the list of all oracle queries made by $E$. Choose a random $i$ for which $x_i$ is of the form $(s_{\text{actor}}^j, s_{\text{actor}}^j, Z, X)$ and output $Z$.

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

B UKS attacks in our model

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

Definition 10 (UKS attack). A key-exchange protocol $\Pi$ 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. $s'_{\text{peer}} \neq s_{\text{actor}}$ (where $s'_{\text{peer}}$ denotes the intended peer of session $s'$ and $s_{\text{actor}}$ 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 7, a key-exchange protocol is resilient against UKS attacks, as the following proposition shows.

Proposition 1. If a protocol $\Pi$ is secure with respect to Definition 7, then it is resilient against UKS attacks in the sense of Definition 10.

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 $s'_{\text{peer}} \neq s_{\text{actor}}$. Notice that session $s'$ is non-matching to session $s$ since $s'_{\text{peer}} \neq s_{\text{actor}}$, hence the query session-key($s'$) is allowed.