Signed Diffie-Hellman Key Exchange with Tight Security

Jiaxin Pan(✉), Chen Qian, and Magnus Ringerud

Department of Mathematical Sciences,
NTNU – Norwegian University of Science and Technology, Trondheim, Norway
{jiaxin.pan,chen.qian,magnus.ringerud}@ntnu.no

Abstract. We propose the first tight security proof for the ordinary two-message signed Diffie-Hellman key exchange protocol in the random oracle model. Our proof is based on the strong computational Diffie-Hellman assumption and the multi-user security of a digital signature scheme. With our security proof, the signed DH protocol can be deployed with optimal parameters, independent of the number of users or sessions, without the need to compensate any security loss. We abstract our approach with a new notion called verifiable key exchange.

In contrast to a known tight three-message variant of the signed Diffie-Hellman protocol (Gjøsteen and Jager, CRYPTO 2018), we do not require any modification to the original protocol, and our tightness result is proven in the "Single-Bit-Guess" model which we known can be tightly composed with symmetric cryptographic primitives to establish a secure channel.

Keywords: Authenticated key exchange · Signed Diffie-Hellman · Tight security

1 Introduction

Authenticated key exchange (AKE) protocols are protocols where two users can securely share a session key in the presence of active adversaries. Beyond passively observing, adversaries against an AKE protocol can modify messages and adaptively corrupt users' long-term keys or the established session key between users. Hence, it is very challenging to construct a secure AKE protocol.

The signed Diffie-Hellman (DH) key exchange protocol is a classical AKE protocol. It is a two-message (namely, two message-moves or one-round) protocol and can be viewed as a generic method to transform a passively secure Diffie-Hellman key exchange protocol [14] into a secure AKE protocol using digital signatures. Figure 1 visualizes the protocol. The origin of signed DH is unclear to us, but its idea has been used in and serves as a solid foundation for many well-known AKE protocols, including the Station-to-Station protocol [15], IKE protocol [19], the one in TLS 1.3 [32], and many others [7,18,22,23,25].

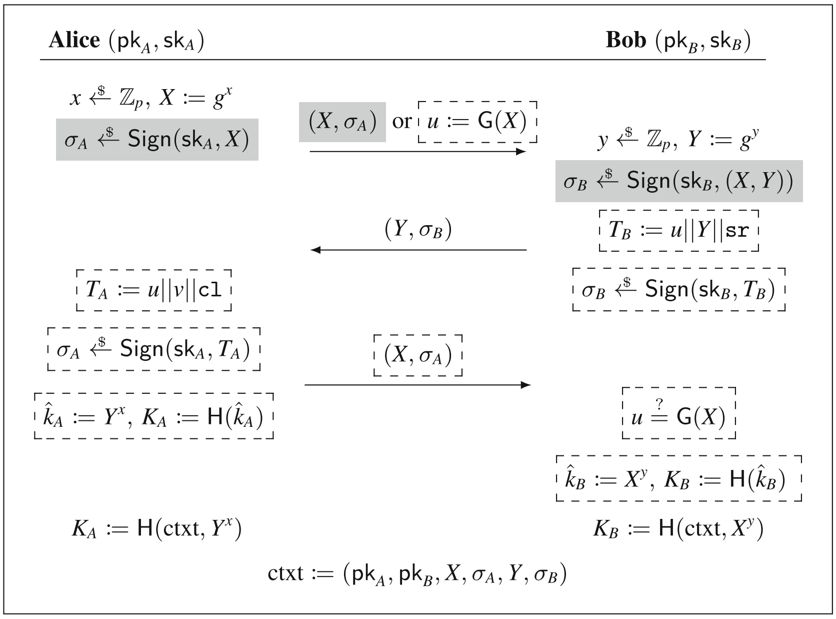
Alice $(\mathsf{pk}_A, \mathsf{sk}_A)$
Bob $(\mathsf{pk}_B, \mathsf{sk}_B)$
$x \xleftarrow{\$} \mathbb{Z}_p,\ X := g^x$
$\sigma_A \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}_A, X)$
(X, σ_A) or $u := \mathsf{G}(X)$
$y \xleftarrow{\$} \mathbb{Z}_p,\ Y := g^y$
$\sigma_B \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}_B, (X, Y))$
$T_B := u||Y||\mathtt{sr}$
(Y, σ_B)
$\sigma_B \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}_B, T_B)$
$T_A := u||v||\mathtt{cl}$
$\sigma_A \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}_A, T_A)$
(X, σ_A)
$\hat{k}_A := Y^x,\ K_A := \mathsf{H}(\hat{k}_A)$
$u \overset{?}{=} \mathsf{G}(X)$
$\hat{k}_B := X^y,\ K_B := \mathsf{H}(\hat{k}_B)$
$K_A := \mathsf{H}(\text{ctxt}, Y^x)$
$K_B := \mathsf{H}(\text{ctxt}, X^y)$
$\text{ctxt} := (\mathsf{pk}_A, \mathsf{pk}_B, X, \sigma_A, Y, \sigma_B)$

Fig. 1. Our signed Diffie-Hellman key exchange protocol and the tight variant of Gjøsteen and Jager [18]. The functions H and G are hash functions. Operations marked with a gray box are for our signed DH protocol, and dashed boxes are for Gjøsteen and Jager's. Operations without a box are performed by both protocols. All signatures are verified upon arrival with the corresponding messages, and the protocol aborts if any verification fails.

Tight Security. Security of a cryptographic scheme is usually proven by constructing a reduction. Asymptotically, a reduction reduces any efficient adversary $\mathcal{A}$ against the scheme into an adversary $\mathcal{R}$ against the underlying computational problem. Concretely, a reduction provides a security bound for the scheme, $\varepsilon_{\mathcal{A}} \leq \ell \cdot \varepsilon_{\mathcal{R}}$, where $\varepsilon_{\mathcal{A}}$ is the success probability of $\mathcal{A}$ and $\varepsilon_{\mathcal{R}}$ is that of $\mathcal{R}$. We say a reduction is *tight* if ℓ is a small constant and the running time of $\mathcal{A}$ is approximately the same as that of $\mathcal{R}$. For the same scheme, it is more desirable to have a tight security proof than a non-tight one, since a tight security proof enables implementations without the need to compensate a security loss with increased parameters.

Multi-challenge Security for AKE. An adversary against an AKE protocol has full control of the communication channel and, additionally, it can adaptively corrupt users' long-term keys and reveal session keys. The goal of an adversary is to distinguish between a (non-revealed) session key and a random bit-string of the same length, which is captured by the Test query. We follow the Bellare-Rogaway (BR) model [5] to capture these capabilities, but formalize it with the game-based style of [21]. Instead of weak perfect forward secrecy, our model captures the (full) perfect forward secrecy.

Unlike the BR model, our model captures multi-challenge security, where an adversary can make T many Test queries which are answered with a single random bit. This is a standard and well-established multi-challenge notion, and [21] called it "Single-Bit-Guess" (SBG) security. Another multi-challenge notion is the "Multi-Bit-Guess" (MBG) security where each Test query is answered with a different random bit. Although several tightly secure AKE protocols [2,18,28,35] are proven in the MBG model, we stress that the SBG model is well-established and allows tight composition of the AKE with symmetric cryptographic primitives, which is not the case for the non-standard MBG model. Thus, the SBG multi-challenge model is more desirable than the MBG model. More details about this have been provided by Jager et al.[21, Introduction] and Cohn-Gordon et al. [10, Section 3].

The Non-tight Security of Signed DH. Many existing security proofs of signed DH-like protocols [7,22,23] lose a quadratic factor, $O(\mu^2 S^2)$, where μ and S are the maximum numbers of users and sessions. In the SBG model with T many Test queries, these proofs also lose an additional multiplicative factor T.

At CRYPTO 2018, Gjøsteen and Jager [18] proposed a tightly secure variant of it by introducing an additional message move into the ordinary signed DH protocol. They showed that if the signature scheme is tightly secure in the multi-user setting then their protocol is tightly secure. They required the underlying signature scheme to be strongly unforgeable against adaptive Corruption and Chosen-Message Attacks (StCorrCMA) which is a notion in the multi-user setting and an adversary can adaptively corrupt some of the honest users to see their secret keys. Moreover, they constructed a tightly multi-user secure signature scheme based on the Decisional Diffie-Hellman (DDH) assumption in the random oracle model [4]. Combining these two results, they gave a practical three message fully tight AKE. We note that their tight security is proven in the less desirable MBG model, and, to the best of our knowledge, the MBG security can only non-tightly imply the SBG security [21]. Due to the "commitment problem", the additional message is crucial for the tightness of their protocol. In particular, the "commitment problem" seems to be the reason why most security proofs for AKEs are non-tight.

1.1 Our Contribution

In this paper, we propose a new tight security proof of the ordinary two-message signed Diffie-Hellman key exchange protocol in the random oracle model. More precisely, we prove the security of the signed DH protocol *tightly* based on the multi-user security of the underlying signature scheme in the random oracle model. Our proof improves upon the work of Gjøsteen and Jager [18] in the sense that we do not require any modification to the signed DH protocol and our tight multi-challenge security is in the SBG model. This implies that our analysis supports the optimal implementation of the ordinary signed DH protocol with theoretically sound security in a meaningful model.

Our technique is a new approach to resolve the “commitment problem”. At the core of it is a new notion called *verifiable key exchange protocols*. We first briefly recall the “commitment problem” and give an overview of our approach.

Technical Difficulty: The “commitment problem”. As explained in [18], this problem is the reason why almost all proofs of classical AKE protocols are non-tight. In a security proof of an AKE protocol, the reduction needs to embed a hard problem instance into the protocol messages of Test sessions so that in the end the reduction can extract a solution to the hard problem from the adversary $\mathcal{A}$. After the instance is embedded, $\mathcal{A}$ has not committed itself to which sessions it will query to Test yet, and, for instance, $\mathcal{A}$ can ask the reduction for Reveal queries on sessions with a problem instance embedded to get the corresponding session keys. At this point, the reduction cannot respond to these Reveal queries. A natural way to resolve this is to guess which sessions $\mathcal{A}$ will query Test on, and to embed a hard problem instance in those sessions only. However, this introduces an extremely large security loss. To resolve this “commitment problem”, a tight reduction should be able to answer both Test and Reveal for every session without any guessing. Gjøsteen and Jager achieved this for the signed DH by adding an additional message.

In this paper, we show that this additional message is not necessary for tight security.

Our Approach: Verifiable Key Exchange. In this work we, for simplicity, use the signed Diffie-Hellman protocol based on the plain Diffie-Hellman protocol [14] (as described in Fig. 1) to explain our approach. In the technical part, we abstract and present our idea with a new notion called verifiable key exchange protocols. Our approach is motivated by the two-message non-tight AKE in [10].

Let $\mathbb{G} := \langle g \rangle$ be a cyclic group of prime-order p where the computational Diffie-Hellman (CDH) problem is hard. Let (g^α, g^β) (where $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p$) be an instance of the CDH problem. By its random self-reducibility, we can efficiently randomize it to multiple independent instances $(g^{\alpha_i}, g^{\beta_i})$, and, given a $g^{\alpha_i \beta_i}$, we can extract the solution $g^{\alpha\beta}$.

For preparation, we assume that a Test session does not contain any forgeries. This can be tightly justified by the $\mathsf{StCorrCMA}$ security of the underlying signature scheme which can be implemented tightly by the recent scheme in [12].

After that, our reduction embeds the randomized instance $(g^{\alpha_i}, g^{\beta_i})$ into each session. Now it seems we can answer neither Test nor Reveal queries: The answer has the form $K := \mathsf{H}(\text{ctxt}, g^{xy})$, but the term g^{xy} cannot be computed by the reduction, since g^x is from either adversary $\mathcal{A}$ or the CDH problem challenge. However, our reduction can answer this by simulating the random oracle H. More precisely, we answer Test and Reveal queries with a random K, and we carefully program the random oracle H so that adversary $\mathcal{A}$ cannot detect this change. To achieve this, when we receive a random oracle query $\mathsf{H}(\text{ctxt}, Z)$, we answer it consistently if the secret element Z corresponds to the context ctxt and ctxt belongs to one of the Test or Reveal queries. This check can be efficiently done by using the strong DH oracle [1].

The approach described above can be abstract by a notion called verifiable key exchange (VKE) protocols. Roughly speaking, a VKE protocol is firstly passively secure, namely, a passive observer cannot compute the secret session key. Additionally, a VKE allows an adversary to check whether a session key belongs to some honestly generated session, and to forward honestly generated transcripts in a different order to create non-matching sessions. This VKE notion gives rise to a tight security proof of the signed DH protocol. We believe this is of independent interest.

On the Strong CDH Assumption. Our techniques require the Strong CDH assumption [1] for the security of our VKE protocol. We refer to [11, Appendix B] for a detailed analysis of this assumption in the Generic Group Model (GGM). Without using the GGM, we can use the twinning technique [9] to remove this strong assumption and base the VKE security tightly on the (standard) CDH assumption. This approach will double the number of group elements. Alternatively, we can use the group of signed Quadratic Residues (QR) [20] to instantiate our VKE protocol, and then the VKE security is tightly based on the factoring assumption (by [20, Theorem 2]).

Real-World Impacts. As mentioned earlier, the signed DH protocol serves as a solid foundation for many real-world protocols, including the one in TLS 1.3 [32], IKE [19], and the Station-to-Station [15] protocols. We believe our approach can naturally be extended to tighten the security proofs of these protocols. In particular, our notion of VKE protocols can abstract some crucial steps in a recent tight proof of TLS 1.3 [11].

Another practical benefit of our tight security proof is that, even if we implement the underlying signature with a standardized, non-tight scheme (such as Ed25519 [8] or RSA-PKCS #1 v1.5 [31]), our implementation does not need to lose the additional factor that is linear in the number of sessions. In today's Internet, there can be easily 2^{60} sessions per year.

1.2 Protocol Comparison

We compare the instantiation of signed DH according to our tight proof with the existing explicitly authenticated key exchange protocols in Fig. 2. For complete tightness, all these protocols require tight multi-user security of their underlying signature scheme. We implement the signature scheme in all protocols with the recent efficient scheme from Diemert et al. [12] whose signatures contain 3 $\mathbb{Z}_p$ elements, and whose security is based on the DDH assumption. The implementation of TLS is according to the recent tight proofs in [11, 13], and we instantiate the underlying signature scheme with the same DDH-based scheme from [12].

We note that the non-tight protocol from Cohn-Gorden et al. [10], whose security loss is linear in the number of users, has better communication efficiency $(2, 0, 0)$. However, its security is weaker than all protocols listed in Fig. 2, since their protocol is only implicitly authenticated and achieves weak perfect forward secrecy.

Protocol	**Comm.** $(\mathbb{G}, \{0,1\}^\lambda, \mathbb{Z}_p)$	**#Msg.**	**Assumption**	**Auth.**	**Model**	**State Reveal**	**Security loss**
TLS* [11,13]	$(2,4,6)$	3	StCDH + DDH	expl.	SBG	no	$O(1)$
GJ [18]	$(2,1,6)$	3	DDH	expl.	MBG	no	$O(1)$
LLGW [28]	$(3,0,6)$	2	DDH	expl.	MBG	no	$O(1)$
JKRS [21]	$(5,1,3)$	2	DDH	expl.	SBG	yes	$O(1)$
This work	$(2,0,6)$	2	StCDH + DDH	expl.	SBG	no	$O(1)$

Fig. 2. Comparison of AKE protocols. We denote **Comm.** as the communication complexity of the protocols in terms of the number of group elements, hashes and $\mathbb{Z}_p$ elements (which is due to the use of the signature scheme in [12]). The column **Model** lists the AKE security model and distinguishes between multi-bit guessing (MBG) and the single-bit-guessing (SBG) security.

We detail the comparison with JKRS [21]. Using the DDH-based signature scheme in [12], the communication complexity of our signed DH protocol is $(2, 0, 6)$, while that of JKRS is $(5, 1, 3)$. We suppose the efficiency of our protocol is comparable to JKRS.

Our main weakness is that our security model is weaker that of JKRS. Namely, ours does not allow adversaries to corrupt any internal secret state. We highlight that our proof does not inherently rely on any decisional assumption. In particular, if there is a tightly multi-user secure signature scheme based on only search assumptions, our proof directly gives a tightly secure AKE based on search assumptions only, which is not the case for [21].

Open Problems. We do not know of any tightly multi-user secure signature schemes with corruptions based on a search assumption, and the schemes in [30] based on search assumptions do not allow any corruption. It is therefore insufficient for our purpose, and we leave constructing a tightly secure AKE based purely on search assumptions as an open problem.

2 Preliminaries

For $n \in \mathbb{N}$, let $[n] = \{1, \ldots, n\}$. For a finite set $\mathcal{S}$, we denote the sampling of a uniform random element x by $x \overset{\$}{\leftarrow} \mathcal{S}$. By $[\![B]\!]$ we denote the bit that is 1 if the evaluation of the Boolean statement B is **true** and 0 otherwise.

Algorithms. For an algorithm $\mathcal{A}$ which takes x as input, we denote its computation by $y \leftarrow \mathcal{A}(x)$ if $\mathcal{A}$ is deterministic, and $y \overset{\$}{\leftarrow} \mathcal{A}(x)$ if $\mathcal{A}$ is probabilistic. We assume all the algorithms (including adversaries) in this paper to be probabilistic unless we state it. We denote an algorithm $\mathcal{A}$ with access to an oracle O by $\mathcal{A}^{\mathrm{O}}$.

Games. We use code-based games [6] to present our definitions and proofs. We implicitly assume all Boolean flags to be initialized to 0 (**false**), numerical variables to 0, sets to $\varnothing$ and strings to $\perp$. We make the convention that a procedure terminates once it has returned an output. $G^{\mathcal{A}} \Rightarrow b$ denotes the final (Boolean) output b of game G running adversary $\mathcal{A}$, and if $b = 1$ we say $\mathcal{A}$ wins G. The randomness in $\Pr[G^{\mathcal{A}} \Rightarrow 1]$ is over all the random coins in game G. Within a procedure, "**abort** " means that we terminate the run of an adversary $\mathcal{A}$.

Digital signatures. We recall the syntax and security of a digital signature scheme. Let par be some system parameters shared among all participants.

Definition 1 (Digital Signature). *A digital signature scheme* $\mathsf{SIG} := (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ *is defined as follows.*

- *The key generation algorithm* $\mathsf{Gen}(\mathsf{par})$ *returns a public key and a secret key* $(\mathsf{pk}, \mathsf{sk})$. *We assume that* pk *implicitly defines a message space* $\mathcal{M}$ *and a signature space* Σ.
- *The signing algorithm* $\mathsf{Sign}(\mathsf{sk}, m \in \mathcal{M})$ *returns a signature* $\sigma \in \Sigma$ *on* m.
- *The deterministic verification algorithm* $\mathsf{Ver}(\mathsf{pk}, m, \sigma)$ *returns 1 (accept) or 0 (reject).*

SIG *is perfectly correct, if for all* $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{Gen}(\mathsf{par})$ *and all messages* $m \in \mathcal{M}$, $\mathsf{Ver}(\mathsf{pk}, m, \mathsf{Sign}(\mathsf{sk}, m)) = 1$.

In addition, we say that SIG *has* α *bits of (public) key min-entropy if an honestly generated public key* pk *is chosen from a distribution with at least* α *bits min-entropy. Formally, for all bit-strings* pk' *we have* $\Pr[\mathsf{pk} = \mathsf{pk}' : (\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Gen}(\mathsf{par})] \leq 2^{-\alpha}$.

Definition 2 ($\mathsf{StCorrCMA}$ **Security** [12,18]**).** *A digital signature scheme* SIG *is* $(t, \varepsilon, \mu, Q_s, Q_{\mathrm{COR}})$-$\mathsf{StCorrCMA}$ *secure* (Strong unforgeability against Corruption and Chosen Message Attacks)*, if for all adversaries* $\mathcal{A}$ *running in time at most* t*, interacting with* μ *users, making at most* Q_s *queries to the signing oracle* Sign*, and at most* Q_{COR} $(Q_{\mathrm{COR}} < \mu)$ *queries to the corruption oracle* Corr *as in Fig. 3, we have*

$$\Pr[\mathsf{StCorrCMA}^{\mathcal{A}} \Rightarrow 1] \leq \varepsilon.$$

```
GAME StCorrCMA:
  for $i \in [\mu]$: $(\mathsf{pk}_i, \mathsf{sk}_i) \xleftarrow{\$} \mathsf{Gen}(\mathsf{par})$
  $(i^*, m^*, \sigma^*) \xleftarrow{\$} \mathcal{A}^{\mathrm{O}}(\{\mathsf{pk}_i\}_{i \in [\mu]})$
  return $[\![\mathsf{Ver}(\mathsf{pk}_{i^*}, m^*, \sigma^*)]\!]$
      $\wedge [\![(i^*, m^*, \sigma^*) \notin \mathcal{L}_{\mathcal{S}}]\!] \wedge [\![i^* \notin \mathcal{L}_{\mathcal{C}}]\!]$

SIGN$(i, m)$:
  $\sigma := \mathsf{Sign}(\mathsf{sk}_i, m)$
  $\mathcal{L}_{\mathcal{S}} := \mathcal{L}_{\mathcal{S}} \cup \{(i, m, \sigma)\}$
  return $\sigma$

CORR$(i)$:
  $\mathcal{L}_{\mathcal{C}} := \mathcal{L}_{\mathcal{C}} \cup \{i\}$
  return $\mathsf{sk}_i$
```

Fig. 3. $\mathsf{StCorrCMA}$ security game for a signature scheme SIG. $\mathcal{A}$ has access to the oracles $\mathrm{O} := \{\textsc{Sign}, \textsc{Corr}\}$.

Security in the Random Oracle Model. A common approach to analyze the security of signature schemes that involve a hash function is to use the random oracle model [4] where hash queries are answered by an oracle H, where H is defined as follows: On input x, it first checks whether $\mathsf{H}(x)$ has previously been defined. If so, it returns $\mathsf{H}(x)$. Otherwise, it sets $\mathsf{H}(x)$ to a uniformly random value in the range of H and then returns $\mathsf{H}(x)$. We parameterize the maximum number of hash queries in our security notions. For instance, we define $(t, \varepsilon, \mu, Q_s, Q_{\text{COR}}, Q_{\mathsf{H}})$-$\mathsf{StCorrCMA}$ as security against any adversary that makes at most Q_{H} queries to H in the $\mathsf{StCorrCMA}$ game. Furthermore, we make the standard convention that any random oracle query that is asked as a result of a query to the signing oracle in the $\mathsf{StCorrCMA}$ game is also counted as a query to the random oracle. This implies that $Q_s \leq Q_{\mathsf{H}}$.

Signature Schemes. The tight security of our authenticated key exchange (AKE) protocols are established based on the StCorrCMA security of the underlying signature schemes. To obtain a completely tight AKE, we use the recent signature scheme from [12] to implement our protocols.

By adapting the non-tight proof in [17], the standard unforgeability against chosen-message attacks (UF-CMA) notion for signature schemes implies the StCorrCMA security of the same scheme non-tightly (with security loss μ). Thus, many widely used signature schemes (such as the Schnorr [33], Ed25519 [8] and RSA-PKCS #1 v1.5 [31] signature schemes) are non-tightly StCorrCMA secure. We do not know any better reductions for these schemes. We leave proving the StCorrCMA security of these schemes without losing a linear factor of μ as a future direction. However, our tight proof for the signed DH protocol strongly indicates that the aforementioned non-tight reduction is optimal for these practical schemes. This is because if we can prove these schemes tightly secure, we can combine them with our tight proof to obtain a tightly secure AKE with unique and verifiable private keys, which may contradict the impossibility result from [10].

For the Schnorr signature, we analyze its StCorrCMA security in the generic group model (GGM) [29,34]. We recall the Schnorr signature scheme below and show the GGM bound of its StCorrCMA security in Theorem 1.

Let $\mathsf{par} = (p, g, \mathbb{G})$, where $\mathbb{G} = \langle g \rangle$ is a cyclic group of prime order p with a hard discrete logarithm problem. Let $\mathsf{G} : \{0,1\}^* \to \mathbb{Z}_p$ be a hash function. Schnorr's signature scheme, $\mathsf{Schnorr} := (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$, is defined as follows:

```
Gen(par):
  $x \xleftarrow{\$} \mathbb{Z}_p$
  $X := g^x$
  $\mathsf{pk} := X$
  $\mathsf{sk} := x$
  return $(\mathsf{pk}, \mathsf{sk})$
```

```
Sign(sk, m):
  parse $x =: \mathsf{sk}$
  $r \xleftarrow{\$} \mathbb{Z}_p$;   $R := g^r$
  $h := \mathsf{G}(\mathsf{pk}, R, m)$
  $s := r + x \cdot h$
  return $(h, s)$
```

```
Ver(pk, m, σ):
  parse $(h, s) =: \sigma$
  parse $X =: \mathsf{pk}$
  $R = g^s \cdot X^{-h}$
  return $[\![\mathsf{G}(R, m) = h]\!]$
```

Theorem 1 (StCorrCMA **Security of** Schnorr **in the GGM).** *Schnorr's signature* SIG *is* $(t, \varepsilon, \mu, Q_s, Q_{\text{COR}}, Q_{\mathsf{G}})$-StCorrCMA-*secure in the* GGM *and in the programmable random oracle model, where*

$$\varepsilon \leq \frac{(Q_{\mathbb{G}} + \mu + 1)^2}{2p} + \frac{(\mu - Q_{\text{COR}})}{p} + \frac{Q_{\mathsf{G}} Q_s + 1}{p}, \quad \text{and} \quad t' \approx t.$$

Here, $Q_{\mathbb{G}}$ is the number of group operations queried by the adversary.

The proof of Theorem 1 is following the approach in [3,24]: We first define an algebraic interactive assumption, $\mathsf{CorrIDLOG}$, which is tightly equivalent to the $\mathsf{StCorrCMA}$ security of $\mathsf{Schnorr}$, and then we analyze the hardness of $\mathsf{CorrIDLOG}$ in the GGM. $\mathsf{CorrIDLOG}$ stands for Interactive Discrete Logarithm with Corruption. It is motivated by the IDLOG (Interactive Discrete Logarithm) assumption in [24]. $\mathsf{CorrIDLOG}$ is a stronger assumption than IDLOG in the sense that it allows an adversary to corrupt the secret exponents of some public keys. Due to space limit, we leave the detailed proof of Theorem 1 in our full version.

3 Security Model for Two-Message Authenticated Key Exchange

In this section, we use the security model in [21] to define the security of two-message authenticated key exchange protocols. This section is almost verbatim to Sect. 4 of [21]. We highlight the difference we make for our protocol: Since our protocols do not have security against (ephemeral) state reveal attacks (as in the extended Canetti-Krawczyk (eCK) model [26]), we do not consider state reveals in our model.

A two-message key exchange protocol $\mathsf{AKE} := (\mathsf{Gen}_{\mathsf{AKE}}, \mathsf{Init}_{\mathrm{I}}, \mathsf{Der}_{\mathsf{R}}, \mathsf{Der}_{\mathrm{I}})$ consists of four algorithms which are executed interactively by two parties as shown in Fig. 4. We denote the party which initiates the session by P_i and the party which responds to the session by P_r. The key generation algorithm $\mathsf{Gen}_{\mathsf{AKE}}$ outputs a key pair $(\mathsf{pk}, \mathsf{sk})$ for one party. The initialization algorithm $\mathsf{Init}_{\mathrm{I}}$ inputs the initiator's long-term secret key sk_i and the responder's long-term public key pk_r, and outputs a message m_i and a state st. The responder's derivation algorithm $\mathsf{Der}_{\mathsf{R}}$ takes as input the responder's long-term secret key, the initiator's public key pk_i and a message m_i. It computes a message m_r and a session key K. The initiator's derivation algorithm $\mathsf{Der}_{\mathrm{I}}$ inputs the initiator's long term key sk_i, the responder's long term public key pk_r, the responder's message m_r and the state st. Note that the responder is not required to save any internal state information besides the session key K.

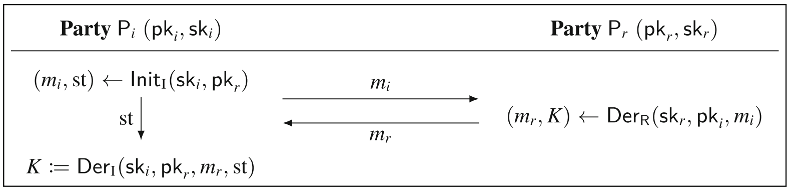
Party P_i $(\mathsf{pk}_i, \mathsf{sk}_i)$
Party P_r $(\mathsf{pk}_r, \mathsf{sk}_r)$
$(m_i, \mathrm{st}) \leftarrow \mathsf{Init}_{\mathrm{I}}(\mathsf{sk}_i, \mathsf{pk}_r)$
m_i
st
$(m_r, K) \leftarrow \mathsf{Der}_{\mathsf{R}}(\mathsf{sk}_r, \mathsf{pk}_i, m_i)$
m_r
$K := \mathsf{Der}_{\mathrm{I}}(\mathsf{sk}_i, \mathsf{pk}_r, m_r, \mathrm{st})$

Fig. 4. Running an authenticated key exchange protocol between two parties.

We give a security game written in pseudocode. We define a model for *explicit authenticated* protocols achieving (full) forward secrecy instead of weak forward secrecy. Namely, an adversary in our model can be active and corrupt the user who owns the Test session sID*, and the only restriction is that if there is no matching session to sID*, then the peer of sID* must not be corrupted before the session finishes.

Here explicit authentication means entity authentication in the sense that a party can explicitly confirm that he is talking to the actual owner of the recipient's public key. The key confirmation property is only implicit [16], where a party is assured that the other identified party can compute the same session key. The game IND-FS is given in Figs. 5 and 6.

```
GAME IND-FS
for $n \in [\mu]$
    $(\mathsf{pk}_n, \mathsf{sk}_n) \leftarrow \mathsf{Gen}_{\mathsf{AKE}}$
$b \xleftarrow{\$} \{0,1\}$
$b' \leftarrow \mathcal{A}^{\mathrm{O}}(\mathsf{pk}_1, \cdots, \mathsf{pk}_\mu)$
for $\mathrm{sID}^* \in \mathcal{S}$
    if $\textsc{Fresh}(\mathrm{sID}^*) = $ false
        return $b$                    //session not fresh
    if $\textsc{Valid}(\mathrm{sID}^*) = $ false
        return $b$                    //no valid attack
return $[\![b = b']\!]$

$\textsc{Session}_{\mathrm{R}}((i,r) \in [\mu]^2, m_i)$
$\mathrm{cnt_S}$ ++
$\mathrm{sID} := \mathrm{cnt_S}$
$(\mathrm{init}[\mathrm{sID}], \mathrm{resp}[\mathrm{sID}]) := (i, r)$
$\mathrm{type}[\mathrm{sID}] :=$ "Re"
$\mathrm{peerCorrupted}[\mathrm{sID}] := \mathrm{corrupted}[i]$
$(m_r, K) \leftarrow \mathsf{Der_R}(\mathsf{sk}_r, \mathsf{pk}_i, m_i)$
$(I[\mathrm{sID}], R[\mathrm{sID}], \mathrm{sKey}[\mathrm{sID}]) := (m_i, m_r, K)$
return $(\mathrm{sID}, m_r)$

$\textsc{Test}(\mathrm{sID})$
if $\mathrm{sID} \in \mathcal{S}$ return $\perp$          //already tested
if $\mathrm{sKey}[\mathrm{sID}] = \perp$ return $\perp$
$\mathcal{S} := \mathcal{S} \cup \{\mathrm{sID}\}$
$K_0^* := \mathrm{sKey}[\mathrm{sID}]$
$K_1^* \xleftarrow{\$} \mathcal{K}$
return $K_b^*$
```

```
$\textsc{Session}_{\mathrm{I}}((i,r) \in [\mu]^2)$
$\mathrm{cnt_S}$ ++
$\mathrm{sID} := \mathrm{cnt_S}$
$(\mathrm{init}[\mathrm{sID}], \mathrm{resp}[\mathrm{sID}]) := (i, r)$
$\mathrm{type}[\mathrm{sID}] :=$ "In"
$(m_i, \mathrm{st}) \leftarrow \mathsf{Init_I}(\mathsf{sk}_i, \mathsf{pk}_r)$
$(I[\mathrm{sID}], \mathrm{state}[\mathrm{sID}]) := (m_i, \mathrm{st})$
return $(\mathrm{sID}, m_i)$

$\textsc{Der}_{\mathrm{I}}(\mathrm{sID} \in [\mathrm{cnt_S}], m_r)$
if $\mathrm{sKey}[\mathrm{sID}] \neq \perp$ or $\mathrm{type}[\mathrm{sID}] \neq$ "In"
    return $\perp$                    //no re-use
$(i, r) := (\mathrm{init}[\mathrm{sID}], \mathrm{resp}[\mathrm{sID}])$
$\mathrm{st} := \mathrm{state}[\mathrm{sID}]$
$\mathrm{peerCorrupted}[\mathrm{sID}] := \mathrm{corrupted}[r]$
$K := \mathsf{Der_I}(\mathsf{sk}_i, \mathsf{pk}_r, m_r, \mathrm{st})$
$(R[\mathrm{sID}], \mathrm{sKey}[\mathrm{sID}]) := (m_r, K)$
return $\epsilon$

$\textsc{Reveal}(\mathrm{sID})$
$\mathrm{revealed}[\mathrm{sID}] :=$ true
return $\mathrm{sKey}[\mathrm{sID}]$

$\textsc{Corr}(n \in [\mu])$
$\mathrm{corrupted}[n] :=$ true
return $\mathsf{sk}_n$
```

Fig. 5. Game IND-FS for AKE. $\mathcal{A}$ has access to oracles $\mathrm{O} := \{\textsc{Session}_{\mathrm{I}}, \textsc{Session}_{\mathrm{R}}, \textsc{Der}_{\mathrm{I}}, \textsc{Reveal}, \textsc{Corr}, \textsc{Test}\}$. Helper procedures Fresh and Valid are defined in Fig. 6. If there exists any test session which is neither fresh nor valid, the game will return b.

Execution Environment. We consider μ parties $\mathsf{P}_1, \ldots, \mathsf{P}_\mu$ with long-term key pairs $(\mathsf{pk}_n, \mathsf{sk}_n)$, $n \in [\mu]$. Each session between two parties has a unique identification number sID and variables which are defined relative to sID:

– $\mathrm{init}[\mathrm{sID}] \in [\mu]$ denotes the initiator of the session.
– $\mathrm{resp}[\mathrm{sID}] \in [\mu]$ denotes the responder of the session.

```
FRESH(sID*)
00 $(i^*, r^*) := (\mathrm{init}[\mathrm{sID}^*], \mathrm{resp}[\mathrm{sID}^*])$
01 $\mathfrak{M}(\mathrm{sID}^*) := \{\mathrm{sID} \mid (\mathrm{init}[\mathrm{sID}], \mathrm{resp}[\mathrm{sID}]) = (i^*, r^*) \wedge (I[\mathrm{sID}], R[\mathrm{sID}]) =$
       $(I[\mathrm{sID}^*], R[\mathrm{sID}^*]) \wedge \mathrm{type}[\mathrm{sID}] \neq \mathrm{type}[\mathrm{sID}^*]\}$   //matching sessions
02 if $\mathrm{revealed}[\mathrm{sID}^*]$ or $(\exists \mathrm{sID} \in \mathfrak{M}(\mathrm{sID}^*) : \mathrm{revealed}[\mathrm{sID}] = \mathbf{true})$
03   return false   //$\mathcal{A}$ trivially learned the test session's key
04 if $\exists \mathrm{sID} \in \mathfrak{M}(\mathrm{sID}^*)$ s. t. $\mathrm{sID} \in \mathcal{S}$
05   return false   //$\mathcal{A}$ also tested a matching session
06 return true

VALID(sID*)
07 $(i^*, r^*) := (\mathrm{init}[\mathrm{sID}^*], \mathrm{resp}[\mathrm{sID}^*])$
08 $\mathfrak{M}(\mathrm{sID}^*) := \{\mathrm{sID} \mid (\mathrm{init}[\mathrm{sID}], \mathrm{resp}[\mathrm{sID}]) = (i^*, r^*) \wedge (I[\mathrm{sID}], R[\mathrm{sID}]) =$
       $(I[\mathrm{sID}^*], R[\mathrm{sID}^*]) \wedge \mathrm{type}[\mathrm{sID}] \neq \mathrm{type}[\mathrm{sID}^*]\}$   //matching sessions
09 for attack $\in$ Table 1
10   if attack = true return true
11 return false
```

Fig. 6. Helper procedures FRESH and VALID for game IND-FS defined in Fig. 5. Procedure FRESH checks if the adversary performed some trivial attack. In procedure VALID, each attack is evaluated by the set of variables shown in Table 1 and checks if an allowed attack was performed. If the values of the variables are set as in the corresponding row, the attack was performed, i.e. attack = **true**, and thus the session is valid.

– type[sID] $\in$ {"In", "Re"} denotes the session's view, i.e. whether the initiator or the responder computes the session key.
– I[sID] denotes the message that was computed by the initiator.
– R[sID] denotes the message that was computed by the responder.
– state[sID] denotes the (secret) state information, i.e. ephemeral secret keys.
– sKey[sID] denotes the session key.

To establish a session between two parties, the adversary is given access to oracles SESSION$_\mathrm{I}$ and SESSION$_\mathsf{R}$, where the first one starts a session of type "In" and the second one of type "Re". The SESSION$_\mathsf{R}$ oracle also runs the Der_R algorithm to compute it's session key and complete the session, as it has access to all the required variables. In order to complete the initiator's session, the oracle DER$_\mathrm{I}$ has to be queried.

Following [21], we do not allow the adversary to register adversarially controlled parties by providing long-term public keys, as the registered keys would be treated no differently than regular corrupted keys. If we would include the key registration oracle, then our proof requires a stronger notion of signature schemes in the sense that our signature challenger can generate the system parameters with some trapdoor. With the trapdoor, the challenger can simulate a valid signature under the adversarially registered public keys. This is the case for the Schnorr signature and the tight scheme in [12], since they are honest-verifier zero-knowledge and the aforementioned property can be achieved by programming the random oracles. However, for readability, we treat the registered keys as corrupted keys.

Finally, the adversary has access to oracles Corr and Reveal to obtain secret information. We use the following boolean values to keep track of which queries the adversary made:

– corrupted$[n]$ denotes whether the long-term secret key of party P_n was given to the adversary.
– revealed[sID] denotes whether the session key was given to the adversary.
– peerCorrupted[sID] denotes whether the peer of the session was corrupted and its long-term key was given to the adversary at the time the session key is computed, which is important for forward security.

The adversary can forward messages between sessions or modify them. By that, we can define the relationship between two sessions:

– **Matching Session:** Two sessions sID and sID$'$ *match* if the same parties are involved (init[sID] = init[sID$'$] and resp[sID] = resp[sID$'$]), the messages sent and received are the same (I[sID] = I[sID$'$] and R[sID] = R[sID$'$]) and they are of different types (type[sID] $\neq$ type[sID$'$]).

Our protocols use signatures to preserve integrity so that any successful no-match attacks described in [27] will lead to a signature forgery and thus can be excluded.

Finally, the adversary is given access to oracle Test, which can be queried multiple times and which will return either the session key of the specified session or a uniformly random key. We use one bit b for all test queries, and store test sessions in a set $\mathcal{S}$. The adversary can obtain information on the interactions between two parties by querying the long-term secret keys and the session key. However, for each test session, we require that the adversary does not issue queries such that the session key can be trivially computed. We define the properties of freshness and validity which all test sessions have to satisfy:

– **Freshness:** A (test) session is called *fresh* if the session key was not revealed. Furthermore, if there exists a matching session, we require that this session's key is not revealed and that this session is not also a test session.
– **Validity:** A (test) session is called *valid* if it is fresh and the adversary performed any attack which is defined in the security model. We capture this with attack Table 1.

Attack Tables. We define validity of different attack strategies. All attacks are defined using variables to indicate which queries the adversary may (not) make. We consider three dimensions:

– whether the test session is on the initiator's (type[sID*] = "In") or the responder's side (type[sID*] = "Re"),
– all combinations of long-term secret key reveals, taking into account when a corruption happened (corrupted and peerCorrupted variables),
– whether the adversary acted passively (matching session) or actively (no matching session).

Table 1. Distilled table of attacks for adversaries against explicitly authenticated two-message protocols without ephemeral state reveals. An attack is regarded as an AND conjunction of variables with specified values as shown in the each line, where "–" means that this variable can take arbitrary value and **F** means "false".

	$\mathcal{A}$ gets (Initiator, Responder)	corrupted[i^*]	corrupted[r^*]	peerCorrupted[sID*]	type[sID*]	$\lvert\mathfrak{M}(\mathrm{sID}^*)\rvert$
0.	**multiple matching sessions**	–	–	–	–	>1
1.+2.	**(long-term, long-term)**	–	–	–	–	1
5.+6.	**(long-term, long-term)**	–	–	**F**	–	0

This way, we capture all kind of combinations which are possible. From the 6 attacks in total presented in Table 2, two are trivial wins for the adversary and can thus be excluded:

– Attack (3.)+(4.): If there is no matching session, and the peer is corrupted, the adversary will trivially win, as he can forge a signature on any message of his choice, and then compute the session key.

Other attacks covered in our model capture *forward secrecy* (FS) and *key compromise impersonation* (KCI) attacks. An attack was performed if the variables are set to the corresponding values in the table.

However, if the protocol does not use appropriate randomness, it should not be considered secure. Thus, if the adversary is able to create more than one matching session to a test session, he may also run a trivial attack. We model this in row (0.) of Table 2.

Note that we do not include reflection attacks, where the adversary makes a party run the protocol with himself. For the $\mathsf{KE}_{\mathsf{DH}}$ protocol, we could include these and create an additional reduction to the square Diffie-Hellman assumption (given g^x, to compute g^{x^2}), but for simplicity of our presentation we will not consider reflection attacks in this paper.

How to read the tables. As an example, we choose row (5.) of Table 2. Then, if the test session is an initiating session (namely, type[sID*] = "In"), the responder is not corrupted when the key is computed, and there does not exist a matching session (namely, $|\mathfrak{M}(\mathrm{sID}^*)| = 0$), this row will evaluate to true. In this scenario, the adversary is allowed to query both long-term secret keys. Note that row (6.) denotes a similar attack against a responder session. Since the session's type does not change the queries the adversary is allowed to make in this case, we merge these rows in Table 1. For the same reason, we also merge lines (1.) and (2.).

Table 2. Full table of attacks for adversaries against explicitly authenticated two-message protocols. The trivial attacks where the session's peer is corrupted when the key is derived, and the corresponding variables are set to **T**, are marked with gray. The $\perp$ symbol indicates that the adversary cannot query anything from this party, as he already possesses the long-term key.

	$\mathcal{A}$ gets (Initiator, Responder)	corrupted[i^*]	corrupted[r^*]	peerCorrupted[sID*]	type[sID*]	$\lvert\mathfrak{M}(\text{sID}^*)\rvert$
0.	**multiple matching sessions**	–	–	–	–	>1
1.	**(long-term, long-term)**	–	–	–	"In"	1
2.	**(long-term, long-term)**	–	–	–	"Re"	1
3.	**(long-term, $\perp$)**	–	**T**	**T**	"In"	0
4.	**($\perp$, long-term)**	**T**	–	**T**	"Re"	0
5.	**(long-term, long-term)**	–	–	**F**	"In"	0
6.	**(long-term, long-term)**	–	–	**F**	"Re"	0

The purpose of these tables are to make our proofs precise, by listing all the possible attacks. We note that while in our case it would have been possible to simply write out the attacks, the number of possible combinations get too large if state-reveals are considered. As we adopt our model from [21], which does include state-reveals, we stuck to their notation.

For all test sessions, at least one attack has to evaluate to true. Then, the adversary wins if he distinguishes the session keys from uniformly random keys which he obtains through queries to the TEST oracle.

Definition 3 (Key Indistinguishability of AKE). *We define game* IND-FS *as in Figs. 5 and 6. A protocol* AKE *is* $(t, \varepsilon, \mu, S, T, Q_{\text{COR}})$-IND-FS-*secure if for all adversaries* $\mathcal{A}$ *attacking the protocol in time* t *with* μ *users,* S *sessions,* T *test queries and* Q_{COR} *corruptions, we have*

$$\left|\Pr[\mathsf{IND\text{-}FS}^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}\right| \leq \varepsilon.$$

Note that if there exists a session which is neither fresh nor valid, the game outputs the bit b, which implies that $\Pr[\mathsf{IND\text{-}FS}^{\mathcal{A}} \Rightarrow 1] = 1/2$, giving the adversary an advantage equal to 0. This captures that an adversary will not gain any advantage by performing a trivial attack.

4 Verifiable Key Exchange Protocols

A key exchange protocol $\mathsf{KE} := (\mathsf{Init}_{\mathrm{I}}, \mathsf{Der}_{\mathsf{R}}, \mathsf{Der}_{\mathrm{I}})$ can be run between two (unauthenticated) parties i and r, and can be visualized as in Fig. 4, but with differences where (1): parties does not hold any public key or private key, and (2): public and private keys in algorithms $\mathsf{Init}_{\mathrm{I}}, \mathsf{Der}_{\mathsf{R}}, \mathsf{Der}_{\mathrm{I}}$ are replaced with the corresponding users' (public) identities.

The standard signed Diffie-Hellman (DH) protocol can be viewed as a generic way to transform a passively secure key exchange protocol to an actively secure AKE protocol using digital signatures. Our tight transformation does not modify the construction of the signed DH protocol, but requires a security notion (i.e. One-Wayness against Honest and key Verification attacks, or OW-HV) that is (slightly) stronger than passive security: Namely, in addition to passive attacks, an adversary is allowed to check if a key corresponds to some honestly generated transcripts and to forward transcripts in a different order to create non-matching sessions. Here we require that all the involved transcripts must be honestly generated by the security game and not by the adversary. This is formally defined by Definition 4 with security game OW-HV as in Fig. 7.

```
GAME OW-HV
$(\mathrm{sID}^*, K^*) \xleftarrow{\$} \mathcal{A}^{\mathrm{O}}(\mu)$
if $\mathrm{sID}^* > \mathrm{cnt}_{\mathrm{S}}$
    return 0
else
    return $\textsc{KVer}(\mathrm{sID}^*, K^*)$

$\textsc{KVer}(\mathrm{sID}, K)$
return $[\![\mathrm{sKey}[\mathrm{sID}] = K]\!]$

$\textsc{Der}_{\mathrm{I}}(\mathrm{sID}, Y)$
if $\mathrm{sKey}[\mathrm{sID}] \neq \perp$ or $\mathrm{type}[\mathrm{sID}] \neq$ "In"
    return $\perp$
if $\forall \mathrm{sID}' \in [\mathrm{cnt}_{\mathrm{S}}] : R[\mathrm{sID}'] \neq Y$
    return $\perp$                    // $Y$ is not honest
$(i, r) := (\mathrm{init}[\mathrm{sID}], \mathrm{resp}[\mathrm{sID}])$
$\mathrm{st} := \mathrm{state}[\mathrm{sID}]$
$K := \mathsf{Der}_{\mathrm{I}}(i, r, Y, \mathrm{st})$
$(R[\mathrm{sID}], \mathrm{sKey}[\mathrm{sID}]) := (Y, K)$
return $\epsilon$

$\textsc{Session}_{\mathrm{I}}((i, r) \in [\mu]^2)$          // $i \neq r$
$\mathrm{cnt}_{\mathrm{S}}$ ++
$\mathrm{sID} := \mathrm{cnt}_{\mathrm{S}}$
$(\mathrm{init}[\mathrm{sID}], \mathrm{resp}[\mathrm{sID}]) := (i, r)$
$\mathrm{type}[\mathrm{sID}] :=$ "In"
$(X, \mathrm{st}) \xleftarrow{\$} \mathsf{Init}_{\mathrm{I}}(i, r)$
$(I[\mathrm{sID}], \mathrm{state}[\mathrm{sID}]) := (X, \mathrm{st})$
return $(\mathrm{sID}, X)$

$\textsc{Session}_{\mathsf{R}}((i, r) \in [\mu]^2, X)$          // $i \neq r$
if $\forall \mathrm{sID} \in [\mathrm{cnt}_{\mathrm{S}}] : I[\mathrm{sID}] \neq X$
    return $\perp$                    // $X$ is not honest
$\mathrm{cnt}_{\mathrm{S}}$ ++
$\mathrm{sID}' := \mathrm{cnt}_{\mathrm{S}}$
$(\mathrm{init}[\mathrm{sID}'], \mathrm{resp}[\mathrm{sID}']) := (i, r)$
$\mathrm{type}[\mathrm{sID}'] :=$ "Re"
$I[\mathrm{sID}'] := X$
$(Y, K') \xleftarrow{\$} \mathsf{Der}_{\mathsf{R}}(r, i, X)$
$R[\mathrm{sID}'] := Y$
$\mathrm{sKey}[\mathrm{sID}'] := K'$
return $(\mathrm{sID}', Y)$
```

Fig. 7. Game OW-HV for KE. $\mathcal{A}$ has access to oracles $\mathrm{O} := \{\textsc{Session}_{\mathrm{I}}, \textsc{Session}_{\mathsf{R}}, \textsc{Der}_{\mathrm{I}}, \textsc{KVer}\}$.

Definition 4 (One-Wayness against Honest and key Verification attacks (OW-HV)). *A key exchange protocol* KE *is* $(t, \varepsilon, \mu, S, Q_V)$-OW-HV *secure, where* μ *is the number of users,* S *is the number of sessions and* Q_V

is the number of calls to KVer*, if for all adversaries* $\mathcal{A}$ *attacking the protocol in time at most* t*, we have*

$$\Pr[\mathsf{OW\text{-}HV}^{\mathcal{A}} \Rightarrow 1] \leq \varepsilon.$$

We require that a key exchange protocol KE has α *bits of min-entropy*, i.e. that for all messages m' we have $\Pr[m = m'] \leq 2^{-\alpha}$, where m is output by either $\mathsf{Init}_{\mathrm{I}}$ or $\mathsf{Der}_{\mathsf{R}}$.

4.1 Example: Plain Diffie-Hellman Protocol

We show that the plain Diffie-Hellman (DH) protocol over prime-order group [14] is a $\mathsf{OW\text{-}HV}$-secure key exchange under the strong computational DH (StCDH) assumption [1]. We use our syntax to recall the original DH protocol $\mathsf{KE}_{\mathsf{DH}}$ in Fig. 8.

Let $\mathsf{par} = (p, g, \mathbb{G})$ be a set of system parameters, where $\mathbb{G} := \langle g \rangle$ is a cyclic group of prime order p.

Definition 5 (Strong CDH Assumption). *The strong CDH (*StCDH*) assumption is said to be* $(t, \varepsilon, Q_{\mathrm{DH}})$*-hard in* $\mathsf{par} = (p, g, \mathbb{G})$*, if for all adversaries* $\mathcal{A}$ *running in time at most* t *and making at most* Q_{DH} *queries to the DH predicate oracle* DH_a*, we have:*

$$\Pr\left[Z = B^a \;\middle|\; \begin{array}{l} a, b \xleftarrow{\$} \mathbb{Z}_p;\ A := g^a\ B := g^b \\ Z \xleftarrow{\$} \mathcal{A}^{\mathrm{DH}_a}(A, B) \end{array}\right] \leq \varepsilon,$$

where the DH predicate oracle $\mathrm{DH}_a(C, D)$ *outputs 1 if* $D = C^a$ *and 0 otherwise.*

```
$\mathsf{Init}_{\mathrm{I}}(i, r)$:
    $\mathrm{st} := x \xleftarrow{\$} \mathbb{Z}_p$
    $X := g^x$
    return $(X, \mathrm{st})$

$\mathsf{Der}_{\mathsf{R}}(r, i, X \in \mathbb{G})$
    $y \xleftarrow{\$} \mathbb{Z}_p$
    $Y := g^y$
    $K := X^y$
    return $(Y, K)$

$\mathsf{Der}_{\mathrm{I}}(i, r, Y \in \mathbb{G}, \mathrm{st} \in \mathbb{Z}_p)$
    $K := Y^{\mathrm{st}}$
    return $K$
```

Fig. 8. The Diffie-Hellman key exchange protocol, $\mathsf{KE}_{\mathsf{DH}}$, in our syntax definition.

Lemma 1. *Let* $\mathsf{KE}_{\mathsf{DH}}$ *be the DH key exchange protocol as in Fig. 8. Then* $\mathsf{KE}_{\mathsf{DH}}$ *has* $\alpha = \log_2 p$ *bits of min-entropy, and for every adversary* $\mathcal{A}$ *that breaks the* $(t, \varepsilon, \mu, S, Q_V)$*-*$\mathsf{OW\text{-}HV}$*-security of* $\mathsf{KE}_{\mathsf{DH}}$*, there is an adversary* $\mathcal{B}$ *that breaks the* $(t', \varepsilon', Q_{\mathrm{DH}})$*-*$\mathsf{StCDH}$ *assumption with*

$$\varepsilon' = \varepsilon, \quad t' \approx t, \quad \textit{and} \quad Q_{\mathrm{DH}} = Q_V + 1. \tag{1}$$

Proof. The min-entropy assertion is straightforward, as the DH protocol generates messages by drawing exponents $x, y \xleftarrow{\$} \mathbb{Z}_p$ uniformly as random.

We prove the rest of the lemma by constructing a reduction $\mathcal{B}$ which inputs the StCDH challenge (A, B) and is given access to the decisional oracle DH_a. $\mathcal{B}$ simulates the OW-HV security game for the adversary $\mathcal{A}$, namely, answers $\mathcal{A}$'s oracle access as in Fig. 9. More precisely, $\mathcal{B}$ uses the random self-reducibility of StCDH to simulate the whole security game, instead of using the Init_I and Der_R algorithms. The most relevant codes are highlighted with **bold** line numbers.

```
$\mathcal{B}^{\mathrm{DH}_a}(A, B)$
01 $(\mathrm{sID}^*, K^*) \xleftarrow{\$} \mathcal{A}^{\mathrm{O}}(\mu)$
02 if $\mathrm{sID}^* > \mathrm{cnt}_\mathrm{S}$ or $\mathrm{KVer}(\mathrm{sID}^*, K^*) = 0$
03     return 0
04 else
05     $(X, Y) := (I[\mathrm{sID}^*], R[\mathrm{sID}^*])$
06     fetch $\mathrm{sID}_1$ s.t. $\mathrm{type}[\mathrm{sID}_1] =$ "In" and $I[\mathrm{sID}_1] = X$
07     fetch $\mathrm{sID}_1$ s.t. $\mathrm{type}[\mathrm{sID}_2] =$ "Re" and $R[\mathrm{sID}_2] = Y$
08     $Z := K^*/(Y^{\alpha[\mathrm{sID}_1]} \cdot A^{\alpha[\mathrm{sID}_2]})$
09     return $[\![Z \in \mathsf{Win}_{\mathsf{StCDH}}]\!]$        //break StCDH

$\mathrm{KVer}(\mathrm{sID}, K)$
10 $(X, Y) := (I[\mathrm{sID}], R[\mathrm{sID}])$
11 fetch $\mathrm{sID}_1$ s.t. $\mathrm{type}[\mathrm{sID}_1] =$ "In" and $I[\mathrm{sID}_1] = X$
12 fetch $\mathrm{sID}_1$ s.t. $\mathrm{type}[\mathrm{sID}_2] =$ "Re" and $R[\mathrm{sID}_2] = Y$
13 if $\mathrm{sID}_1 = \perp$ or $\mathrm{sID}_2 = \perp$
14     return $\perp$
15 return $\mathrm{DH}_a(Y, K/Y^{\alpha[\mathrm{sID}_1]})$

$\mathrm{Der}_\mathrm{I}(\mathrm{sID}, Y)$
16 if $\mathrm{sKey}[\mathrm{sID}] \neq \perp$ or $\mathrm{type}[\mathrm{sID}] \neq$ "In"
17     return $\perp$
18 if $\forall \mathrm{sID}' \in [\mathrm{cnt}_\mathrm{S}] : R[\mathrm{sID}'] \neq Y$
19     return $\perp$        //Y is not honest
20 return $\epsilon$

$\mathrm{Session}_\mathrm{I}((i, r) \in [\mu]^2)$        //$i \neq r$
21 $\mathrm{cnt}_\mathrm{S}$ ++
22 $\mathrm{sID} := \mathrm{cnt}_\mathrm{S}$
23 $(\mathrm{init}[\mathrm{sID}], \mathrm{resp}[\mathrm{sID}]) := (i, r)$
24 $\mathrm{type}[\mathrm{sID}] :=$ "In"
25 $\alpha[\mathrm{sID}] \xleftarrow{\$} \mathbb{Z}_p$
26 $X := A \cdot g^{\alpha[\mathrm{sID}]}$
27 $(I[\mathrm{sID}], \mathrm{state}[\mathrm{sID}]) := (X, \perp)$
28 return $(\mathrm{sID}, X)$

$\mathrm{Session}_\mathsf{R}((i, r) \in [\mu]^2, X)$        //$i \neq r$
29 if $\forall \mathrm{sID} \in [\mathrm{cnt}_\mathrm{S}] : I[\mathrm{sID}] \neq X$
30     return $\perp$        //X is not honest
31 $\mathrm{cnt}_\mathrm{S}$ ++
32 $\mathrm{sID}' := \mathrm{cnt}_\mathrm{S}$
33 $(\mathrm{init}[\mathrm{sID}'], \mathrm{resp}[\mathrm{sID}']) := (i, r)$
34 $\mathrm{type}[\mathrm{sID}'] :=$ "Re"
35 $I[\mathrm{sID}'] := X$
36 $\alpha[\mathrm{sID}'] \xleftarrow{\$} \mathbb{Z}_p$
37 $Y := B \cdot g^{\alpha[\mathrm{sID}']}$
38 $R[\mathrm{sID}'] := Y$
39 return $(\mathrm{sID}', Y)$
```

Fig. 9. Reduction $\mathcal{B}$ that breaks the StCDH assumption and simulates the OW-HV game for $\mathcal{A}$, when $A = g^a$ and $B = g^b$ for some unknown a and b.

We show that $\mathcal{B}$ simulates the OW-HV game for $\mathcal{A}$ perfectly:

– Since X generated in line 26 and Y generated in line 37 are uniformly random, the outputs of $\mathrm{Session}_\mathrm{I}$ and $\mathrm{Session}_\mathsf{R}$ are distributed as in the real protocol. Note that the output of Der_I does not get modified.
– For $\mathrm{KVer}(\mathrm{sID}, K)$, if K is a valid key that corresponds to session sID, then there must exist sessions sID_1 and sID_2 such that $\mathrm{type}[\mathrm{sID}_1] =$ "In" (defined in line 24) and $\mathrm{type}[\mathrm{sID}_2] =$ "Re" (defined in line 34) and

$$K = (B \cdot g^{\alpha[\mathrm{sID}_2]})^{(a+\alpha[\mathrm{sID}_1])} = Y^a \cdot Y^{\alpha[\mathrm{sID}_1]}. \quad (2)$$

where $I[\mathrm{sID}] = I[\mathrm{sID}_1] = A \cdot g^{\alpha[\mathrm{sID}_1]}$ (defined in line 26) and $R[\mathrm{sID}] = R[\mathrm{sID}_2] = Y := B \cdot g^{\alpha[\mathrm{sID}_2]}$ (defined in line 37). Thus, the output of $\textsc{KVer}(\mathrm{sID}, K)$ is the same as that of $\textsc{Dh}_a(Y, K/Y^{\alpha[\mathrm{sID}_1]})$.

Finally, $\mathcal{A}$ returns $\mathrm{sID}^* \in [\mathrm{cnt_S}]$ and a key K^*. If $\mathcal{A}$ wins, then $\textsc{KVer}(\mathrm{sID}^*, K^*) = 1$ which means that there exists sessions sID_1 and sID_2 such that $\mathrm{type}[\mathrm{sID}_1] =$ "In", $\mathrm{type}[\mathrm{sID}_2] =$ "Re" and

$$K^* = g^{(a+\alpha[\mathrm{sID}_1])(b+\alpha[\mathrm{sID}_2])} = g^{ab} \cdot A^{\alpha[\mathrm{sID}_2]} \cdot B^{\alpha[\mathrm{sID}_1]} g^{\alpha[\mathrm{sID}_1]\alpha[\mathrm{sID}_2]} = g^{ab} \cdot A^{\alpha[\mathrm{sID}_2]} \cdot Y^{\alpha[\mathrm{sID}_1]},$$

where $Y = R[\mathrm{sID}_2] = B \cdot g^{\alpha[\mathrm{sID}_2]}$. This means $\mathcal{B}$ breaks the StCDH with $g^{ab} = K^*/(Y^{\alpha[\mathrm{sID}_1]} \cdot A^{\alpha[\mathrm{sID}_2]})$ as in line 08, if $\mathcal{A}$ break the $\mathsf{OW\text{-}HV}$ of $\mathsf{KE_{DH}}$. Hence, $\varepsilon = \varepsilon'$. The running time of $\mathcal{B}$ is the running time of $\mathcal{A}$ plus one exponentiation for every call to $\textsc{Session}_\mathrm{I}$ and $\textsc{Session}_\mathsf{R}$, so we get $t \approx t'$. The number of calls to $\textsc{Dh}_a$ is the number of calls to $\textsc{KVer}$, plus one additional call to verify the adversary's forgery, and hence $Q_{\mathrm{DH}} = Q_V + 1$.

Group of Signed Quadratic Residues. Our construction of a key exchange protocol in Fig. 8 is based on the StCDH assumption over a prime order group. Alternatively, we can instantiate our VKE portocol in a group of signed quadratic residues $\mathbb{QR}_N^+$ [20]. As the StCDH assumption in $\mathbb{QR}_N^+$ groups is tightly implied by the factoring assumption (by [20, Theorem 2]), our VKE protocol is secure based on the classical factoring assumption.

5 Signed Diffie-Hellman, Revisited

Following the definition in Sect. 3, we want to construct a $\mathsf{IND\text{-}FS}$-secure authenticated key exchange protocol $\mathsf{AKE} = (\mathsf{Gen_{AKE}}, \mathsf{Init_I}, \mathsf{Der_I}, \mathsf{Der_R})$ by combining a $\mathsf{StCorrCMA}$-secure signature scheme $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$, a $\mathsf{OW\text{-}HV}$-secure key exchange protocol $\mathsf{KE} = (\mathsf{Init}'_\mathrm{I}, \mathsf{Der}'_\mathrm{I}, \mathsf{Der_R}')$, and a random oracle H. The construction is given in Fig. 10, and follow the execution order from Fig. 4.

```
Gen_AKE(par):
01 $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Gen}(\mathsf{par})$
02 return $(\mathsf{pk}, \mathsf{sk})$

Der_R(sk_r, pk_i, X, σ_i)
03 if $\mathsf{Ver}(\mathsf{pk}_i, X, \sigma_i) = 0$
04     return $\bot$
05 $(Y, K^*) \leftarrow \mathsf{Der_R}'(r, i, X)$
06 $\sigma_r \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}_r, (X, Y))$
07 $\mathrm{ctxt} := (\mathsf{pk}_i, \mathsf{pk}_r, X, \sigma_i, Y, \sigma_r)$
08 $K := \mathsf{H}(\mathrm{ctxt}, K^*)$
09 return $((Y, \sigma_r), K)$

Init_I(sk_i, pk_r):
10 $(X, \mathrm{st}) \xleftarrow{\$} \mathsf{Init}'_\mathrm{I}(i, r)$
11 $\sigma_i \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}_i, X)$
12 return $(X, \mathrm{st}, \sigma_i)$

Der_I(sk_i, pk_r, Y, σ_r, st)
13 if $\mathsf{Ver}(\mathsf{pk}_r, (X, Y), \sigma_r) = 0$
14     return $\bot$
15 $K^* := \mathsf{Der}'_\mathrm{I}(i, r, Y, \mathrm{st})$
16 $\mathrm{ctxt} := (\mathsf{pk}_i, \mathsf{pk}_r, X, \sigma_i, Y, \sigma_r)$
17 $K := \mathsf{H}(\mathrm{ctxt}, K^*)$
18 return $K$
```

Fig. 10. Generic construction of AKE from SIG, KE and a random oracle H.

We now prove that this construction is in fact a secure AKE protocol.

Theorem 2. *For every adversary $\mathcal{A}$ that breaks the $(t, \varepsilon, \mu, ST, Q_{\mathsf{H}},, Q_{\mathrm{COR}})$-*$\mathsf{IND}$-$\mathsf{FS}$*-security of a protocol* AKE *constructed as in Fig. 10, we can construct an adversary $\mathcal{B}$ against the $(t', \varepsilon', \mu, Q_s, Q'_{\mathrm{COR}})$-*$\mathsf{StCorrCMA}$*-security of a signature scheme* SIG *with α bits of key min-entropy, and an adversary $\mathcal{C}$ against the $(t'', \varepsilon'', \mu, S', Q_V)$-*$\mathsf{OW}$-$\mathsf{HV}$ *security of a key exchange protocol* KE *with β bits of min-entropy, such that*

$$\begin{aligned} \varepsilon &\le 2\varepsilon' + \frac{\varepsilon''}{2} + \frac{\mu^2}{2^{\alpha+1}} + \frac{S^2}{2^{\beta+1}} \\ t' &\approx t, \quad Q_s \le S, \quad Q'_{\mathrm{COR}} = Q_{\mathrm{COR}} \\ t'' &\approx t, \quad S' = S, \quad Q_V \le Q_{\mathsf{H}}. \end{aligned}$$

Proof. We will prove this by using the following hybrid games, which are illustrated in Fig. 11.

Game G_0: This is the IND-FS security game for the protocol AKE. We assume that all long term keys, and all messages output by $\mathsf{Init}_{\mathrm{I}}$ and $\mathsf{Der}_{\mathsf{R}}$ are distinct. If a collision happens, the game aborts. To bound the probability of this happening, we use that SIG has α bits of key min-entropy, and KE has β bits of min-entropy. We can upper bound the probability of a collision happening in the keys as $\mu^2/2^{\alpha+1}$ for μ parties, and the probability of a collision happening in the messages as $S^2/2^{\beta+1}$ for S sessions, as each session computes one message. Thus we have

$$\Pr[\mathsf{IND\text{-}FS}^{\mathcal{A}} \Rightarrow 1] = \Pr[G_0^{\mathcal{A}} \Rightarrow 1] + \frac{\mu^2}{2^{\alpha+1}} + \frac{S^2}{2^{\beta+1}}. \quad (3)$$

Game G_1: In this game, when the oracles $\textsc{Der}_{\mathrm{I}}$ and $\textsc{Session}_{\mathsf{R}}$ try to derive a session key, they will abort if the input message does not correspond to a previously sent message, and the corresponding signature is valid *w.r.t.* an uncorrupted party (namely, $\mathcal{A}$ generates the message itself).

This is the preparation step for reducing an IND-FS adversary of AKE to an OW-HV adversary of KE. Note that in this game we do not exclude all the non-matching $\textsc{Test}$ sessions, but it is already enough for the "IND-FS-to-OW-HV" reduction. For instance, $\mathcal{A}$ can still force some responder session to be non-matching by reusing some of the previous initiator messages to query $\textsc{Session}_{\mathsf{R}}$, and then $\mathcal{A}$ uses the non-matching responder session to query $\textsc{Test}$.

The only way to distinguish G_0 and G_1 is to trigger the new abort event in either line 19 (i.e. $\mathsf{AbortDer}_{\mathsf{R}}$) or line 39 (i.e. $\mathsf{AbortDer}_{\mathsf{I}}$) of Fig. 11. We define the event $\mathsf{AbortDer} := \mathsf{AbortDer}_{\mathsf{I}} \vee \mathsf{AbortDer}_{\mathsf{R}}$ and have that

$$\left|\Pr[G_0^{\mathcal{A}} \Rightarrow 1] - \Pr[G_1^{\mathcal{A}} \Rightarrow 1]\right| \le \Pr[\mathsf{AbortDer}].$$

To bound this probability, we construct an adversary $\mathcal{B}$ against the $(t', \varepsilon', \mu, Q_s, Q'_{\mathrm{COR}})$-$\mathsf{StCorrCMA}$-security of SIG in Fig. 12.

We note that $\mathsf{AbortDer}$ is **true** only if $\mathcal{A}$ performs attacks $5+6$ in Table 1 which may lead to a session without any matching session. If $\mathsf{AbortDer} =$ **true**

```
GAMES $G_0$-$G_2$
01 $\mathsf{cnt_S} := 0$                                  //session counter
02 for $n \in [\mu]$
03     $(\mathsf{pk}_n, \mathsf{sk}_n) \xleftarrow{\$} \mathsf{Gen_{AKE}}$
04 $b \xleftarrow{\$} \{0,1\}$
05 $b' \xleftarrow{\$} \mathcal{A}^{\mathrm{O}}(\mathsf{pk}_1, \cdots, \mathsf{pk}_\mu)$
06 for $\mathrm{sID}^* \in \mathcal{S}$
07     if $\textsc{Fresh}(\mathrm{sID}^*) = \mathbf{false}$
08         return $b$
09     if $\textsc{Valid}(\mathrm{sID}^*) = \mathbf{false}$
10         return $b$
11 return $[\![b = b']\!]$

$\textsc{Session}_{\mathsf{R}}((i,r) \in [\mu]^2, (X, \sigma_i))$
12 $\mathsf{cnt_S}$ ++
13 $\mathrm{sID} := \mathsf{cnt_S}$
14 $(\mathrm{init}[\mathrm{sID}], \mathrm{resp}[\mathrm{sID}]) := (i, r)$
15 $\mathrm{type}[\mathrm{sID}] :=$ "Re"
16 $\mathrm{peerCorrupted}[\mathrm{sID}] := \mathrm{corrupted}[i]$
17 $((Y, \sigma_r), K) \xleftarrow{\$} \mathsf{Der_R}(\mathsf{sk}_r, \mathsf{pk}_i, (X, \sigma_i))$
18 if $\mathrm{peerCorrupted}[\mathrm{sID}] = \mathbf{false}$ and
     $\nexists \mathrm{sID}'$: $(\mathrm{init}[\mathrm{sID}'], \mathrm{type}[\mathrm{sID}'], I[\mathrm{sID}'])$
      $= (i, \text{"In"}, (X, \sigma_i))$                       //$G_{1\text{-}2}$
19     $\mathsf{AbortDer_R} := \mathbf{true}$                          //$G_{1\text{-}2}$
20     abort                                           //$G_{1\text{-}2}$
21 $(I[\mathrm{sID}], R[\mathrm{sID}]) := ((X, \sigma_i), (Y, \sigma_r))$
22 $\mathrm{sKey}[\mathrm{sID}] := K$
23 return $(\mathrm{sID}, (Y, \sigma_r))$
```

```
$\textsc{Session}_{\mathrm{I}}((i,r) \in [\mu]^2)$
24 $\mathsf{cnt_S}$ ++
25 $\mathrm{sID} := \mathsf{cnt_S}$
26 $(\mathrm{init}[\mathrm{sID}], \mathrm{resp}[\mathrm{sID}]) := (i, r)$
27 $\mathrm{type}[\mathrm{sID}] :=$ "In"
28 $(X, \mathrm{st}, \sigma_i) \xleftarrow{\$} \mathsf{Init_I}(\mathsf{sk}_i, \mathsf{pk}_r)$
29 $(I[\mathrm{sID}], \mathrm{state}[\mathrm{sID}]) := ((X, \sigma_i), \mathrm{st})$
30 return $(\mathrm{sID}, (X, \sigma_i))$

$\textsc{Der}_{\mathrm{I}}(\mathrm{sID}, (Y, \sigma_r))$
31 if $\mathrm{sKey}[\mathrm{sID}] \neq \perp$ or $\mathrm{type}[\mathrm{sID}] \neq$ "In"
32     return $\perp$                                   //no re-use
33 $(i, r) := (\mathrm{init}[\mathrm{sID}], \mathrm{resp}[\mathrm{sID}])$
34 $\mathrm{st} := \mathrm{state}[\mathrm{sID}]$
35 $\mathrm{peerCorrupted}[\mathrm{sID}] := \mathrm{corrupted}[r]$
36 $K := \mathsf{Der_I}(\mathsf{sk}_i, \mathsf{pk}_r, Y, \sigma_r, \mathrm{st})$
37 $(X, \sigma_i) := I[\mathrm{sID}]$
38 if $\mathrm{peerCorrupted}[\mathrm{sID}] = \mathbf{false}$ and
     $\nexists \mathrm{sID}'$: $(\mathrm{resp}[\mathrm{sID}'], \mathrm{type}[\mathrm{sID}'], I[\mathrm{sID}'], R[\mathrm{sID}'])$
      $= (r, \text{"Re"}, (X, \sigma_i), (Y, \sigma_r))$              //$G_{1\text{-}2}$
39     $\mathsf{AbortDer_I} := \mathbf{true}$                          //$G_{1\text{-}2}$
40     abort                                           //$G_{1\text{-}2}$
41 $(R[\mathrm{sID}], \mathrm{sKey}[\mathrm{sID}]) := ((Y, \sigma_r), K)$
42 return $\epsilon$

$\textsc{Test}(\mathrm{sID})$
43 if $\mathrm{sID} \in \mathcal{S}$ return $\perp$                   //already tested
44 if $\mathrm{sKey}[\mathrm{sID}] = \perp$ return $\perp$
45 $\mathcal{S} := \mathcal{S} \cup \{\mathrm{sID}\}$
46 $K_0^* := \mathrm{sKey}[\mathrm{sID}]$                              //$G_{0\text{-}1}$
47 $K_0^* \xleftarrow{\$} \mathcal{K}$                                    //$G_2$
48 $K_1^* \xleftarrow{\$} \mathcal{K}$
49 return $K_b^*$
```

Fig. 11. Games G_0-G_2. $\mathcal{A}$ has access to oracles $\mathrm{O} := \{\textsc{Session}_{\mathrm{I}}, \textsc{Session}_{\mathsf{R}}, \textsc{Der}_{\mathrm{I}}, \textsc{Reveal}, \textsc{Corr}, \textsc{Test}\}$, where $\textsc{Reveal}$ and $\textsc{Corr}$ are simulated as in the original $\mathsf{IND\text{-}FS}$ game in Fig. 5. Game G_0 implicitly assumes that there is no collision between long term keys or messages output by the experiment.

then Σ is defined in lines 26 and 42 of Fig. 12 and Σ is a valid $\mathsf{StCorrCMA}$ forge for SIG. We only show that for the case when $\mathsf{AbortDer_R} = \mathbf{true}$ here, and the argument is similar for the case when $\mathsf{AbortDer_I} = \mathbf{true}$. Given that $\mathsf{AbortDer_R}$ happens, we have that $\mathsf{Ver}(\mathsf{pk}_i, X, \sigma_i) = 1$ and peerCorrupted[sID] $= \mathbf{false}$. Due to the criteria in line 40, the pair (X, σ_i) has not been output by $\textsc{Session}_{\mathrm{I}}$ on input (i, r) for any r, and hence (i, X) has never been queried to the $\textsc{Sign}'$ oracle. Therefore, $\mathcal{B}$ aborts $\mathcal{A}$ in the $\mathsf{IND\text{-}FS}$ game and returns (i, X, σ_i) to the $\mathsf{StCorrCMA}$ challenger to win the $\mathsf{StCorrCMA}$ game. Therefore, we have

$$\Pr[\mathsf{AbortDer_R}] \leq \varepsilon', \tag{4}$$

```
$\mathcal{B}^{\text{CORR}', \text{SIGN}'}(\mathsf{pk}_1, \dots, \mathsf{pk}_\mu)$
01 $b \xleftarrow{\$} \{0,1\}$
02 $b' \leftarrow \mathcal{A}^{\mathrm{O}}(\mathsf{pk}_1, \dots, \mathsf{pk}_\mu)$
03 for $\mathrm{sID}^* \in \mathcal{S}$
04    if $\text{FRESH}(\mathrm{sID}^*) = $ false
05       return $b$
06    if $\text{VALID}(\mathrm{sID}^*) = $ false
07       return $b$
08 return $[\![\Sigma \in \mathsf{Win}_{\mathsf{StCorrCMA}}]\!]$        //break StCorrCMA

$\text{SESSION}_{\text{I}}((i,r) \in [\mu]^2)$
09 $\mathrm{cnt_S}$ ++
10 $\mathrm{sID} := \mathrm{cnt_S}$
11 $(\mathrm{init}[\mathrm{sID}], \mathrm{resp}[\mathrm{sID}]) := (i, r)$
12 $\mathrm{type}[\mathrm{sID}] :=$ “In”
13 $(X, \mathrm{st}) \xleftarrow{\$} \mathsf{Init}'_{\mathrm{I}}(i, r)$
14 $\sigma_i \xleftarrow{\$} \text{SIGN}'(\mathsf{pk}_i, X)$
15 $(I[\mathrm{sID}], \mathrm{state}[\mathrm{sID}]) := ((X, \sigma_i), \mathrm{st})$
16 return $(\mathrm{sID}, (X, \sigma_i))$

$\text{DER}_{\text{I}}(\mathrm{sID}, (Y, \sigma_r))$
17 if $\mathrm{sKey}[\mathrm{sID}] \neq \perp$ or $\mathrm{type}[\mathrm{sID}] \neq$ “In”
18    return $\perp$                    //no re-use
19 $(i, r) := (\mathrm{init}[\mathrm{sID}], \mathrm{resp}[\mathrm{sID}])$
20 $\mathrm{st} := \mathrm{state}[\mathrm{sID}]$
21 $\mathrm{peerCorrupted}[\mathrm{sID}] := \mathrm{corrupted}[r]$
22 if $\mathsf{Ver}(\mathsf{pk}_r, (X, Y), \sigma_r) = 0$
23    return $\perp$
24 if $\mathrm{peerCorrupted}[\mathrm{sID}] = $ false and
     $\nexists\, \mathrm{sID}'\colon (\mathrm{resp}[\mathrm{sID}'], \mathrm{type}[\mathrm{sID}'], I[\mathrm{sID}'], R[\mathrm{sID}'])$
     $= (r, \text{“Re”}, (X, \sigma_i), (Y, \sigma_r))$
25    $\mathsf{AbortDer_I} :=$ true
26    $\Sigma := (r, (X, Y), \sigma_r)$        //valid forgery
27    abort
28 $K^* := \mathsf{Der}'_{\mathrm{I}}(i, r, Y, \mathrm{st})$
29 $\mathrm{ctxt} := (\mathsf{pk}_i, \mathsf{pk}_r, X, \sigma_i, Y, \sigma_r)$
30 $K := \mathsf{H}(\mathrm{ctxt}, K^*)$
31 $(R[\mathrm{sID}], \mathrm{sKey}[\mathrm{sID}]) := ((Y, \sigma_r), K)$
32 return $\epsilon$

$\text{SESSION}_{\text{R}}((i,r) \in [\mu]^2, (X, \sigma_i))$
33 $\mathrm{cnt_S}$ ++
34 $\mathrm{sID} := \mathrm{cnt_S}$
35 $(\mathrm{init}[\mathrm{sID}], \mathrm{resp}[\mathrm{sID}]) := (i, r)$
36 $\mathrm{type}[\mathrm{sID}] :=$ “Re”
37 $\mathrm{peerCorrupted}[\mathrm{sID}] := \mathrm{corrupted}[i]$
38 if $\mathsf{Ver}(\mathsf{pk}_i, X, \sigma_i) = 0$
39    return $\perp$
40 if $\mathrm{peerCorrupted}[\mathrm{sID}] = $ false and
     $\nexists\, \mathrm{sID}'\colon (\mathrm{init}[\mathrm{sID}'], \mathrm{type}[\mathrm{sID}'], I[\mathrm{sID}'])$
     $= (i, \text{“In”}, (X, \sigma_i))$
41    $\mathsf{AbortDer_R} :=$ true
42    $\Sigma := (i, X, \sigma_i)$              //valid forgery
43    abort
44 $(Y, K^*) \xleftarrow{\$} \mathsf{Der_R}'(r, i, X)$
45 $\sigma_r \xleftarrow{\$} \text{SIGN}'(\mathsf{pk}_r, (X, Y))$
46 $\mathrm{ctxt} := (\mathsf{pk}_i, \mathsf{pk}_r, X, \sigma_i, Y, \sigma_r)$
47 $K := \mathsf{H}(\mathrm{ctxt}, K^*)$
48 $(I[\mathrm{sID}], R[\mathrm{sID}]) := ((X, \sigma_i), (Y, \sigma_r))$
49 $\mathrm{sKey}[\mathrm{sID}] := K$
50 return $(\mathrm{sID}, (Y, \sigma_r))$

$\text{CORR}(n \in [\mu])$
51 $\mathrm{corrupted}[n] :=$ true
52 $\mathsf{sk}_n \leftarrow \text{CORR}'(n)$
53 return $\mathsf{sk}_n$

$\mathsf{H}(\mathsf{pk}_i, \mathsf{pk}_r, X, Y, K^*)$
54 $\mathrm{ctxt} := (\mathsf{pk}_i, \mathsf{pk}_r, X, Y)$
55 if $\mathsf{H}[\mathrm{ctxt}, K^*] = K$
56    return $K$
57 $K \xleftarrow{\$} \mathcal{K}$
58 $\mathsf{H}[\mathrm{ctxt}, K^*] := K$
59 return $K$
```

Fig. 12. Adversary $\mathcal{B}$ against the $(t', \varepsilon', \mu, Q_s, Q'_{\text{COR}})$-StCorrCMA-security of SIG. The StCorrCMA game provides oracles SIGN$'$, CORR$'$. The adversary $\mathcal{A}$ has access to oracles $\mathrm{O} := \{\text{SESSION}_{\text{I}}, \text{SESSION}_{\text{R}}, \text{DER}_{\text{I}}, \text{REVEAL}, \text{CORR}, \text{TEST}, \mathsf{H}\}$, where REVEAL and TEST remain the same as in Fig. 4. We highlight the most relevant codes with **bold** line numbers.

which implies that

$$\left|\Pr[G_0^{\mathcal{A}} \Rightarrow 1] - \Pr[G_1^{\mathcal{A}} \Rightarrow 1]\right| \leq \Pr[\mathsf{AbortDer_I}] + \Pr[\mathsf{AbortDer_R}] \leq 2\varepsilon'. \qquad (5)$$

The running time of $\mathcal{B}$ is the same as that of $\mathcal{A}$, plus the time used to run the key exchange algorithms $\mathsf{Init}'_{\mathrm{I}}$, $\mathsf{Der_R}'$, $\mathsf{Der}'_{\mathrm{I}}$ and the signature verification algorithm

Ver. This gives $t' \approx t$. For the number of signature queries we have $Q_s \leq S$, since $\textsc{Session}_{\mathsf{R}}$ can abort before it queries the signature oracle, and the adversary can reuse messages output by $\textsc{Session}_{\mathrm{I}}$. For the number of corruptions, we have $Q'_{\textsc{Cor}} = Q_{\textsc{Cor}}$.

$\textsc{Game}$ G_2: The $\textsc{Test}$ oracle always returns a uniformly random key, independent on the bit b.

Since we have excluded collisions in the messages output by the experiment, it is impossible to create two sessions of the same type that compute the same session key. Hence, an adversary must query the random oracle H on the correct input of a test session to detect the change between G_1 and G_2 (which is only in case $b = 0$). More precisely, we have $\Pr[G_2^{\mathcal{A}} \Rightarrow 1 \mid b = 1] = \Pr[G_1^{\mathcal{A}} \Rightarrow 1 \mid b = 1]$ and

$$\begin{aligned}\left|\Pr[G_2^{\mathcal{A}} \Rightarrow 1] - \Pr[G_1^{\mathcal{A}} \Rightarrow 1]\right| &= \frac{1}{2}\,\big|\Pr[G_2^{\mathcal{A}} \Rightarrow 1 \mid b = 0] + \Pr[G_2^{\mathcal{A}} \Rightarrow 1 \mid b = 1] \\ &\quad - \Pr[G_1^{\mathcal{A}} \Rightarrow 1 \mid b = 0] - \Pr[G_1^{\mathcal{A}} \Rightarrow 1 \mid b = 1]\big| \\ &= \frac{1}{2}\,\left|\Pr[G_2^{\mathcal{A}} \Rightarrow 1 \mid b = 0] - \Pr[G_1^{\mathcal{A}} \Rightarrow 1 \mid b = 0]\right|. \end{aligned} \tag{6}$$

To bound Eq. (6), we construct an adversary $\mathcal{C}$ to $(t'', \varepsilon'', \mu, S', Q_V)$-break the $\mathsf{OW\text{-}HV}$ security of KE. The input to $\mathcal{C}$ is the number of parties μ, and system parameters par. In addition, $\mathcal{C}$ has access to oracles $\textsc{Session}'_{\mathrm{I}}, \textsc{Session}'_{\mathsf{R}}, \textsc{Der}'_{\mathrm{I}}$ and $\textsc{KVer}$.

We firstly show that the outputs of $\textsc{Session}_{\mathrm{I}}$, $\textsc{Session}_{\mathsf{R}}$ and $\textsc{Der}_{\mathrm{I}}$ (simulated by $\mathcal{C}$) are distributed the same as in G_1. Due to the abort conditions introduced in G_1, for all sessions that has finished computing a key without making the game abort, their messages are honestly generated, although they may be in a different order and there are non-matching sessions. Hence, $\textsc{Session}_{\mathrm{I}}$, $\textsc{Session}_{\mathsf{R}}$ and $\textsc{Der}_{\mathrm{I}}$ can be perfectly simulated using $\textsc{Session}'_{\mathrm{I}}$, $\textsc{Session}'_{\mathsf{R}}$ and $\textsc{Der}'_{\mathrm{I}}$ of the $\mathsf{OW\text{-}HV}$ game and the signing key.

It is also easy to see that the random oracle H simulated by $\mathcal{C}$ has the same output distribution as in G_1. We stress that if line 66 is executed then adversary $\mathcal{A}$ may use the sID to distinguish G_2 and G_1 for $b = 0$, which is the only case for $\mathcal{A}$ to see the difference. At the same time, we obtain a valid attack $\Sigma := (\mathrm{sID}, K^*)$ for the $\mathsf{OW\text{-}HV}$ security. Thus, we have

$$\left|\Pr[G_2^{\mathcal{A}} \Rightarrow 1 \mid b = 0] - \Pr[G_1^{\mathcal{A}} \Rightarrow 1 \mid b = 0]\right| \leq \varepsilon''.$$

As before, the running time of $\mathcal{C}$ is that of $\mathcal{A}$, plus generating and verifying signatures, and we have $t'' \approx t$. Furthermore, $S' = S$, as the counter for the $\mathsf{OW\text{-}HV}$ game increases once for every call to $\textsc{Session}_{\mathrm{I}}$ and $\textsc{Session}_{\mathsf{R}}$.

```
$\mathcal{C}^{O'}(\mu)$
01 for $n \in [\mu]$
02     $(\mathsf{pk}_n, \mathsf{sk}_n) \xleftarrow{\$} \mathsf{Gen}(\mathsf{par})$
03 $b \xleftarrow{\$} \{0,1\}$
04 $b' \leftarrow \mathcal{A}^{O}(\mathsf{pk}_1, \dots, \mathsf{pk}_\mu)$
05 for $\mathrm{sID}^* \in \mathcal{S}$
06     if $\textsc{Fresh}(\mathrm{sID}^*) = $ false
07         return $b$
08     if $\textsc{Valid}(\mathrm{sID}^*) = $ false
09         return $b$
10 return $[\![\Sigma \in \mathsf{Win}_{\mathsf{OW\text{-}HV}}]\!]$

$\textsc{Session}_{\mathrm{I}}((i,r) \in [\mu]^2)$
11 $(\mathrm{sID}, X) \xleftarrow{\$} \textsc{Session}_{\mathrm{I}}'(i,r)$
12 $\mathrm{cnt}_{\mathrm{S}}{+}{+}$
13 $(\mathrm{init}[\mathrm{sID}], \mathrm{resp}[\mathrm{sID}]) := (i,r)$
14 $\mathrm{type}[\mathrm{sID}] :=$ "In"
15 $\sigma_i \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}_i, X)$
16 $I[\mathrm{sID}] := (X, \sigma_i)$
17 return $(\mathrm{sID}, (X, \sigma_i))$

$\textsc{Der}_{\mathrm{I}}(\mathrm{sID}, (Y, \sigma_r))$
18 if $\mathrm{sKey}[\mathrm{sID}] \neq \perp$ or $\mathrm{type}[\mathrm{sID}] \neq$ "In"
19     return $\perp$                                  //no re-use
20 $(i,r) := (\mathrm{init}[\mathrm{sID}], \mathrm{resp}[\mathrm{sID}])$
21 $\mathrm{peerCorrupted}[\mathrm{sID}] := \mathrm{corrupted}[r]$
22 $(X, \sigma_i) := I[\mathrm{sID}]$
23 if $\mathsf{Ver}(\mathsf{pk}_r, (X,Y), \sigma_r) = 0$
24     return $\perp$
25 if $\mathrm{peerCorrupted}[\mathrm{sID}] = $ false and
   $\nexists \mathrm{sID}'\colon (\mathrm{resp}[\mathrm{sID}'], \mathrm{type}[\mathrm{sID}'], I[\mathrm{sID}'], R[\mathrm{sID}'])$
    $= (r, \text{"Re"}, (X, \sigma_i), (Y, \sigma_r))$
26     abort
27 $\mathrm{ctxt} := (\mathsf{pk}_i, \mathsf{pk}_r, X, \sigma_i, Y, \sigma_r)$
28 $\textsc{Der}_{\mathrm{I}}'(\mathrm{sID}, Y)$
29 if $\exists K^* : \mathsf{H}[\mathrm{ctxt}, K^*, 1] = K$
30     $\mathrm{sKey}[\mathrm{sID}] := K$
31 elseif $\mathsf{H}[\mathrm{ctxt}, \perp, \perp] = K$
32     $\mathrm{sKey}[\mathrm{sID}] := K$
33 else $K \xleftarrow{\$} \mathcal{K}$
34     $\mathsf{H}[\mathrm{ctxt}, \perp, \perp] := K$
35     $\mathrm{sKey}[\mathrm{sID}] := K$
36 $R[\mathrm{sID}] := (Y, \sigma_r)$
37 return $\epsilon$

$\textsc{Session}_{\mathsf{R}}((i,r) \in [\mu]^2, (X, \sigma_i))$
38 if $\mathsf{Ver}(\mathsf{pk}_i, X, \sigma_i) = 0$
39     return $\perp$
40 $(\mathrm{sID}, Y) \xleftarrow{\$} \textsc{Session}_{\mathsf{R}}'(i, r, X)$
41 $\mathrm{cnt}_{\mathrm{S}}{+}{+}$
42 $\mathrm{peerCorrupted}[\mathrm{sID}] := \mathrm{corrupted}[i]$
43 if $\mathrm{peerCorrupted}[\mathrm{sID}] = $ false and
   $\nexists \mathrm{sID}'\colon (\mathrm{init}[\mathrm{sID}'], \mathrm{type}[\mathrm{sID}'], I[\mathrm{sID}'])$
    $= (i, \text{"In"}, (X, \sigma_i))$
44     abort
45 $(\mathrm{init}[\mathrm{sID}], \mathrm{resp}[\mathrm{sID}]) := (i,r)$
46 $\mathrm{type}[\mathrm{sID}] :=$ "Re"
47 $I[\mathrm{sID}] := (X, \sigma_i)$
48 $\sigma_r \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}_r, (X, Y))$
49 $R[\mathrm{sID}] := (Y, \sigma_r)$
50 $\mathrm{ctxt} := (\mathsf{pk}_i, \mathsf{pk}_r, X, \sigma_i, Y, \sigma_r)$
51 if $\exists K^* : \mathsf{H}[\mathrm{ctxt}, K^*, 1] = K$
52     $\mathrm{sKey}[\mathrm{sID}] := K$
53 elseif $\mathsf{H}[\mathrm{ctxt}, \perp, \perp] = K$
54     $\mathrm{sKey}[\mathrm{sID}] := K$
55 else $K \xleftarrow{\$} \mathcal{K}$
56     $\mathsf{H}[\mathrm{ctxt}, \perp, \perp] := K$
57     $\mathrm{sKey}[\mathrm{sID}] := K$
58 return $(Y, \sigma_r)$

$\mathsf{H}(\mathsf{pk}_i, \mathsf{pk}_r, X, \sigma_i, Y, \sigma_r, K^*)$
59 $\mathrm{ctxt} := (\mathsf{pk}_i, \mathsf{pk}_r, X, \sigma_i, Y, \sigma_r)$
60 if $\mathsf{H}[\mathrm{ctxt}, K^*, \cdot] = K$
61     return $K$
62 $h := \perp$
63 if $\mathsf{H}[\mathrm{ctxt}, \perp, \perp] = K$ and $\exists \mathrm{sID}$:
       $(I[\mathrm{sID}], R[\mathrm{sID}]) = ((X, \sigma_i), (Y, \sigma_r))$
64     $\textsc{Der}_{\mathrm{I}}'(\mathrm{sID}, Y)$
65     if $\textsc{KVer}(\mathrm{sID}, K^*) = 1$
66         $\Sigma := (\mathrm{sID}, K^*)$  //attack for OW-HV

67         replace $(\perp, \perp)$ in $\mathsf{H}[\mathrm{ctxt}, \perp, \perp]$
               with $(K^*, 1)$
68         return $K$
69     else $h := 0$
70 $K \xleftarrow{\$} \mathcal{K}$
71 $\mathsf{H}[\mathrm{ctxt}, K^*, h] := K$
72 return $K$
```

Fig. 13. Reduction $\mathcal{C}$ against the $(t'', \varepsilon'', \mu, S', Q_V)$-OW-HV-security of KE. The OW-HV game provides oracles $\mathrm{O}' := \{\textsc{Session}_{\mathrm{I}}', \textsc{Session}_{\mathrm{R}}', \textsc{Der}_{\mathrm{I}}', \textsc{KVer}\}$. The adversary $\mathcal{A}$ has access to oracles $\mathrm{O} := \{\textsc{Session}_{\mathrm{I}}, \textsc{Session}_{\mathrm{R}}, \textsc{Der}_{\mathrm{I}}, \textsc{Reveal}, \textsc{Corr}, \textsc{Test}, \mathsf{H}\}$, where Reveal, Corr and Test are defined as in G_2 of Fig. 11. We highlight the most relevant codes with **bold** line numbers. The center dot '$\cdot$' in this figure means arbitrary value.

At last, for game G_2 we have $\Pr[G_2^{\mathcal{A}} \Rightarrow 1] = \frac{1}{2}$, as the response from the TEST oracle is independent of the bit b. Summing up all the equations, we obtain

$$\begin{aligned}
\varepsilon &\leq \left|\Pr[\mathsf{IND\text{-}FS}^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}\right| \\
&= \left|\Pr[G_0^{\mathcal{A}} \Rightarrow 1] + \frac{\mu^2}{2^{\alpha+1}} + \frac{S^2}{2^{\beta+1}} - \Pr[G_2^{\mathcal{A}} \Rightarrow 1]\right| \\
&= \left|\Pr[G_0^{\mathcal{A}} \Rightarrow 1] - \Pr[G_1^{\mathcal{A}} \Rightarrow 1] + \Pr[G_1^{\mathcal{A}} \Rightarrow 1] - \Pr[G_2^{\mathcal{A}} \Rightarrow 1] + \frac{\mu^2}{2^{\alpha+1}} + \frac{S^2}{2^{\beta+1}}\right| \\
&\leq \left|\Pr[G_0^{\mathcal{A}} \Rightarrow 1] - \Pr[G_1^{\mathcal{A}} \Rightarrow 1]\right| + \left|\Pr[G_1^{\mathcal{A}} \Rightarrow 1] - \Pr[G_2^{\mathcal{A}} \Rightarrow 1]\right| + \frac{\mu^2}{2^{\alpha+1}} + \frac{S^2}{2^{\beta+1}} \\
&\leq 2\varepsilon' + \frac{\varepsilon''}{2} + \frac{\mu^2}{2^{\alpha+1}} + \frac{S^2}{2^{\beta+1}},
\end{aligned}$$

and $t' \approx t, \quad Q_s \leq S, \quad Q'_{\text{COR}} = Q_{\text{COR}}, \quad t'' \approx t, \quad S' = S, \quad Q_V \leq Q_{\mathsf{H}}$.

Acknowledgement. We thank the anonymous reviewers for their many insightful suggestions to improve our paper.

References

1. Abdalla, M., Bellare, M., Rogaway, P.: The oracle Diffie-Hellman assumptions and an analysis of DHIES. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 143–158. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45353-9_12
2. Bader, C., Hofheinz, D., Jager, T., Kiltz, E., Li, Y.: Tightly-secure authenticated key exchange. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part I. LNCS, vol. 9014, pp. 629–658. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46494-6_26
3. Bellare, M., Dai, W.: The multi-base discrete logarithm problem: tight reductions and non-rewinding proofs for Schnorr identification and signatures. Cryptology ePrint Archive, Report 2020/416 (2020). https://eprint.iacr.org/2020/416
4. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: Denning, D.E., Pyle, R., Ganesan, R., Sandhu, R.S., Ashby, V. (eds.) ACM CCS 1993, pp. 62–73. ACM Press, November 1993
5. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48329-2_21
6. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game playing proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer, Heidelberg (2006). https://doi.org/10.1007/11761679_25
7. Bergsma, F., Jager, T., Schwenk, J.: One-round key exchange with strong security: an efficient and generic construction in the standard model. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 477–494. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46447-2_21
8. Bernstein, D.J., Duif, N., Lange, T., Schwabe, P., Yang, B.-Y.: High-speed high-security signatures. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 124–142. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23951-9_9

9. Cash, D., Kiltz, E., Shoup, V.: The twin Diffie-Hellman problem and applications. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 127–145. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_8
10. Cohn-Gordon, K., Cremers, C., Gjøsteen, K., Jacobsen, H., Jager, T.: Highly efficient key exchange protocols with optimal tightness. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part III. LNCS, vol. 11694, pp. 767–797. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26954-8_25
11. Davis, H., Günther, F.: Tighter proofs for the SIGMA and TLS 1.3 key exchange protocols. ACNS 2021 (2021). https://eprint.iacr.org/2020/1029
12. Diemert, D., Gellert, K., Jager, T., Lyu, L.: More efficient digital signatures with tight multi-user security. In: PKC 2021 (2021). https://ia.cr/2021/235
13. Diemert, D., Jager, T.: On the tight security of TLS 1.3: theoretically-sound cryptographic parameters for real-world deployments. J. Cryptol. (2020). https://eprint.iacr.org/2020/726
14. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Trans. Inf. Theory **22**(6), 644–654 (1976)
15. Diffie, W., van Oorschot, P.C., Wiener, M.J.: Authentication and authenticated key exchanges. Des. Codes Crypt. **2**(2), 107–125 (1992)
16. Fischlin, M., Günther, F., Schmidt, B., Warinschi, B.: Key confirmation in key exchange: a formal treatment and implications for TLS 1.3. In: 2016 IEEE Symposium on Security and Privacy, pp. 452–469. IEEE Computer Society Press, May 2016
17. Galbraith, S.D., Malone-Lee, J., Smart, N.P.: Public key signatures in the multi-user setting. Inf. Process. Lett. **83**(5), 263–266 (2002). https://doi.org/10.1016/S0020-0190(01)00338-6
18. Gjøsteen, K., Jager, T.: Practical and tightly-secure digital signatures and authenticated key exchange. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 95–125. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96881-0_4
19. Harkins, D., Carrel, D.: The internet key exchange (IKE). RFC 2409 (1998). https://www.ietf.org/rfc/rfc2409.txt
20. Hofheinz, D., Kiltz, E.: The group of signed quadratic residues and applications. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 637–653. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03356-8_37
21. Jager, T., Kiltz, E., Riepel, D., Schäge, S.: Tightly-secure authenticated key exchange, revisited. In: EUROCRYPT 2021 (2021). https://ia.cr/2020/1279
22. Jager, T., Kohlar, F., Schäge, S., Schwenk, J.: On the security of TLS-DHE in the standard model. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 273–293. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_17
23. Jager, T., Kohlar, F., Schäge, S., Schwenk, J.: Authenticated confidential channel establishment and the security of TLS-DHE. J. Cryptol. **30**(4), 1276–1324 (2017)
24. Kiltz, E., Masny, D., Pan, J.: Optimal security proofs for signatures from identification schemes. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part II. LNCS, vol. 9815, pp. 33–61. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53008-5_2
25. Krawczyk, H.: SIGMA: the "SIGn-and-MAc" approach to authenticated Diffie-Hellman and its use in the IKE protocols. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 400–425. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_24

26. LaMacchia, B., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 1–16. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75670-5_1
27. Li, Y., Schäge, S.: No-match attacks and robust partnering definitions: defining trivial attacks for security protocols is not trivial. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017, pp. 1343–1360. ACM Press, October–November 2017
28. Liu, X., Liu, S., Gu, D., Weng, J.: Two-pass authenticated key exchange with explicit authentication and tight security. In: ASIACRYPT 2020 (2020). https://ia.cr/2020/1088
29. Maurer, U.: Abstract models of computation in cryptography (invited paper). In: Smart, N.P. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 1–12. Springer, Heidelberg (2005). https://doi.org/10.1007/11586821_1
30. Pan, J., Ringerud, M.: Signatures with tight multi-user security from search assumptions. In: Chen, L., Li, N., Liang, K., Schneider, S. (eds.) ESORICS 2020, Part II. LNCS, vol. 12309, pp. 485–504. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-59013-0_24
31. PKCS #1: RSA Cryptography Standard. RSA Data Security, Inc., June 1991
32. Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446 (Proposed Standard (2018). https://tools.ietf.org/html/rfc8446
33. Schnorr, C.P.: Efficient signature generation by smart cards. J. Cryptol. **4**(3), 161–174 (1991)
34. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-69053-0_18
35. Xiao, Y., Zhang, R., Ma, H.: Tightly secure two-pass authenticated key exchange protocol in the CK model. In: Jarecki, S. (ed.) CT-RSA 2020. LNCS, vol. 12006, pp. 171–198. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-40186-3_9