Highly Efficient Key Exchange Protocols with Optimal Tightness

Katriel Cohn-Gordon[1], Cas Cremers[2],
Kristian Gjøsteen[3], Håkon Jacobsen[4], Tibor Jager[5]*

[1] Independent Scholar, me@katriel.co.uk
[2] CISPA Helmholtz Center for Information Security, cremers@cispa.saarland
[3] NTNU - Norwegian University of Science and Technology,
Trondheim, Norway, kristian.gjosteen@ntnu.no
[4] McMaster University, jacobseh@mcmaster.ca
[5] Paderborn University, Paderborn, Germany, tibor.jager@upb.de

Abstract. In this paper we give nearly-tight reductions for modern implicitly authenticated Diffie-Hellman protocols in the style of the Signal and Noise protocols, which are extremely simple and efficient. Unlike previous approaches, the combination of nearly-tight proofs and efficient protocols enables the first real-world instantiations for which the parameters can be chosen in a theoretically sound manner.
Our reductions have only a linear loss in the number of users, implying that our protocols are more efficient than the state of the art when instantiated with theoretically sound parameters. We also prove that our security proofs are optimal: a linear loss in the number of users is unavoidable for our protocols for a large and natural class of reductions.

1 Introduction

Key exchange protocols serve as a building block for almost all secure communication today. However, deploying a key exchange protocol requires implementors to carefully choose concrete values for several parameters, such as group and key sizes, which we here abstract into a single security parameter n. But how should n be selected? An answer is to select it based on formal reductionist arguments in the style of concrete security [7]. These arguments relate the security parameter n of a protocol to the security parameter $f(n)$ of an assumed-hard problem, such that breaking the protocol with parameter n would lead to an attack on the hard problem with parameter $f(n)$. We say a protocol is deployed in a *theoretically sound* way if n is chosen such that the underlying problem is "hard enough" with parameter $f(n)$.

Unfortunately, for most deployed protocols the parameters are actually *not* chosen in a theoretically sound way. This means that the formal security arguments are in reality vacuous since $f(n)$ is too small for the underlying problem to

* Supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, grant agreement 802823, and the Deutsche Forschungsgemeinschaft (DFG), project number 265919409.

be hard. For example, existing security proofs for TLS [11, 22, 27] have a security loss which is quadratic in the total number of sessions, but the parameters chosen in practice does not account for this. If one aims for "128-bit security", and assumes 2^{30} users and up to 2^{30} sessions per user (very plausible for TLS), then a theoretically sound choice of parameters would have to provide at least "248-bit security". In the particular case of the algebraic groups used for Diffie-Hellman (DH) in TLS, this would require a group of order $|\mathbb{G}| \approx 2^{496}$ instead of the common 128-bit-secure choice of $|\mathbb{G}| \approx 2^{256}$. But larger parameters typically leads to worse performance so this is not done in practice. Thus, for TLS as actually used, the proofs do not provide any meaningful security guarantees since they relate the hardness of breaking TLS to a DH instance which is too easy to solve.

It would be desirable if protocols could be instantiated in a theoretically sound way without sacrificing efficiency. This has led to the study of so-called *tight security*, in which one aims to construct proofs such that the gap between n and $f(n)$ is as small as possible. While there have been several recent advances in this field [3, 19], typically they trade tighter proofs for the use of more complex primitives and constructions—which themselves require more or larger keys. This leads to the perhaps counter-intuitive observation that the resulting protocols have a tighter security proof, but are substantially less efficient in practice. For example, the recent protocol of Gjøsteen and Jager [19] has a constant security loss, meaning that an attack on their protocol leads to an attack on decisional DH with essentially the same parameter. However, it is a signed DH protocol, and thus must be instantiated with a tightly-secure signature scheme. The solution used by Gjøsteen and Jager [19] requires a total of 17 exponentiations which can negate the efficiency savings from using a smaller group. In some sense they overshoot their target: they achieve tightness without reaching the actual goal of *efficient* theoretically sound deployment in practice.

In this work we will instead aim between the two extremes of real-world protocols on the one end having very non-tight proofs, and the more theoretical protocols on the other having fully tight proofs, focusing instead on the actual end-goal of achieving efficient theoretically sound deployments in practice. Our constructions fall into the class of implicitly authenticated DH protocols, which often are more efficient than signed DH variants, and can additionally offer various forms of deniability. Implicitly authenticated key exchange protocols have been studied extensively in the literature, and in the past few years have also started to see deployments in the real world. Perhaps the most well-known example is the Signal protocol [38], which encrypts messages for WhatsApp's 1.5 billion users. Another example is the Noise protocol framework [36], whose so-called IK pattern powers the new Linux kernel VPN Wireguard [16]. Similar protocols in the literature include KEA+ [30] and UM [24].

We will give a security proof for a simple instance of this class, very close to Signal's basic design. In and of itself this isn't particularly noteworthy. What *is* noteworthy, however, is the *tightness* of the proof. Unlike any other proof for a protocol as simple and efficient as ours, our proof only incurs a security loss which is *linear* in the number of users μ and *constant* in the number of sessions

per user ℓ. This is in stark contrast to most other key exchange proofs that are typically *quadratic* in at least one of these parameters, and most of the time quadratic even in their product $\mu\ell$.

Our contributions. Our contributions revolve around three protocols which all aim for high practical efficiency when instantiated with theoretically sound parameters. The first protocol, which we call Π, is a simple and clean implicitly authenticated DH protocol very close to Signal, Noise-KK, KEA+ and UM, and provides weak forward secrecy. In protocol Π users exchange a single group element and perform four group exponentiations to establish a session key. Protocol Π—specified precisely in Section 4—aims for maximal efficiency under the strong DH assumption.

The other two protocols, which can be seen as variants of protocol Π, are designed to avoid the strong DH assumption of Π. The first protocol, which we call Π_{Twin}, adapts the "twinning" technique of Cash et al. [13] to protocol Π, and needs four more exponentiations. The second, which we call Π_{Com}, additionally adapts the "commitment" technique of Gjøsteen and Jager [19], and only needs two more exponentiations than protocol Π. On the other hand, it requires one more round of communication. Both Π_{Twin} and Π_{Com} are slightly more costly than protocol Π, but in return require only the standard CDH and DDH assumptions.

Common to all our protocols is that they are simple and conventional, with no heavyweight cryptographic machinery. They exchange ephemeral keys and derive a session key from the combination of static-ephemeral, ephemeral-static and ephemeral-ephemeral DH values via a hash function H. In our proofs H will be a random oracle.

Our first core contribution is thus to give new reductions for all these protocols with a linear loss $L = O(\mu)$ in the random oracle model. This is better than almost all known AKE protocols. As we will see, even though the loss is not constant, our protocols are so efficient that they perform better than both fully-tight protocols as well as the most efficient non-tight AKEs[6]. In contrast to previous works, our proofs enable theoretically sound deployment of conventional protocols while maintaining high efficiency.

Our second core contribution is to show that the $O(\mu)$ tightness loss is essentially optimal for the protocols considered in this paper, at least for "simple" reductions. A "simple" reduction runs a *single* copy of the adversary only *once*. To the best of our knowledge, all known security reductions for AKE protocols are either of this type or use the forking lemma (which of necessity leads to a non-tight proof). Hence, to give a tighter security proof, one would have to develop a completely new approach to prove security.

The lower-bound proof will be based on the *meta-reduction* techniques described by Bader et al. [4]. However, these techniques are only able to handle tight reductions from *non-interactive* assumptions, while our first protocol is based on the *interactive* strong DH assumption. Therefore we develop a new

[6] When instantiated with theoretically sound parameters under reasonable assumptions on μ and ℓ in modern deployment settings.

variant of the approach, which makes it possible to also handle the strong DH assumption.

Finally, we prove that our protocols can be enhanced to also provide explicit entity authentication by adding key-confirmation messages, while still providing tight security guarantees. To do so, we generalise a theorem of Yang [41] in two ways: we apply it to n-message protocols for $n > 2$, and we give a tight reduction to the multi-user versions of the underlying primitives.

To summarise:

1. We give three protocols with linear-loss security reductions, making them faster than both fully-tight protocols and the most efficient non-tight ones when instantiated in a theoretically sound manner for reasonable numbers of users and sessions.
2. We prove optimality of linear loss for our protocols under "simple" reductions.
3. We tightly extend our protocols with key confirmation messages to provide explicit entity authentication.

Related work. We briefly touch upon some other protocols with non-quadratic security loss. KEA+ [30] achieves $L = O(\mu\ell)$ under the Gap-DH assumption, and where the reduction for pairing-friendly curves takes $O(t \log t)$ time. However, for non-pairing-friendly curves the reduction takes $O(t^2)$ time. Moreover, KEA+ also does not achieve weak forward secrecy in a modern model: only one side's long term key can be corrupted.

The first AKE protocols with L *independent* of μ and ℓ were described by Bader et al. [3] at TCC 2015. They describe two protocols, one with constant security loss $L = O(1)$ and another with loss $L = O(\kappa)$ linear in the security parameter. Both protocols make use of rather heavy cryptographic building blocks, such as tree-based signature schemes, Groth-Sahai proofs [20], and cryptographic pairings, and are therefore not very efficient.

As already mentioned, Gjøsteen and Jager [19] recently described a more practical protocol, which essentially is a three-message variant of "signed Diffie-Hellman". Even though their protocol uses a rather complex signature scheme to achieve tightness (a single key exchange requires 17 exponentiations and the exchange of in total 16 group elements/exponents), when instantiated with theoretically sound parameters it turns out to be more efficient than even plain signed DH with ECDSA, at least for large-scale deployments. Unlike [3], the security analysis in [19] is in the random oracle model [8] since the paper aims at maximal practical efficiency.

2 Background

In this section we recap some background and standard definitions. Let $\mathbb{G}$ be a cyclic group of prime order p with generator g.

Diffie-Hellman Problems. The computational and decisional Diffie-Hellman problems are natural problems related to breaking the Diffie-Hellman protocol.

Definition 1. *Consider the following experiment involving an adversary $\mathcal{A}$. The experiment samples $x, y \xleftarrow{\$} \mathbb{Z}_p$ and starts $\mathcal{A}(g^x, g^y)$. The advantage of $\mathcal{A}$ in solving the* computational Diffie-Hellman *problem is defined as*

$$\mathrm{Adv}^{\mathsf{CDH}}_{\mathbb{G},g}(\mathcal{A}) := \Pr\left[\mathcal{A}(g^x, g^y) = g^{xy}\right]$$

Definition 2. *Consider the following experiment involving an adversary $\mathcal{A}$. The experiment samples $x, y, z \xleftarrow{\$} \mathbb{Z}_p$ and tosses a coin $\hat{b} \xleftarrow{\$} \{0,1\}$. If $\hat{b} = 1$ then it sets $Z := g^{xy}$, while if $\hat{b} = 0$ then it sets $Z = g^z$. We define the advantage of $\mathcal{A}$ in solving the* decisional Diffie-Hellman *problem as*

$$\mathrm{Adv}^{\mathsf{DDH}}_{\mathbb{G},g}(\mathcal{A}) := \left|\Pr\left[\mathcal{A}(g^x, g^y, Z) = \hat{b}\right] - 1/2\right|$$

Let $\mathsf{DDH}(g^x, g^y, g^z)$ be an oracle that returns 1 if and only if $xy = z$. The *gap* Diffie-Hellman problem asks to solve the computational Diffie-Hellman problem, given access to the oracle $\mathsf{DDH}(\cdot,\cdot,\cdot)$. The *strong* Diffie-Hellman problem is related to the gap Diffie-Hellman problem, except that the adversary now gets a less capable oracle where the first input is fixed, i.e., $\mathsf{stDH}_x(\cdot,\cdot) = \mathsf{DDH}(g^x,\cdot,\cdot)$.

Definition 3. *Consider the following experiment involving an adversary $\mathcal{A}$. The experiment samples $x, y \xleftarrow{\$} \mathbb{Z}_p$ and starts $\mathcal{A}^{\mathsf{stDH}_x(\cdot,\cdot)}(g^x, g^y)$. The advantage of $\mathcal{A}$ in solving the* strong Diffie-Hellman *problem is defined as*

$$\mathrm{Adv}^{\mathsf{stDH}}_{\mathbb{G},g}(\mathcal{A}) := \Pr\left[\mathcal{A}^{\mathsf{stDH}_x(\cdot,\cdot)}(g^x, g^y) = g^{xy}\right].$$

One may wonder to which extent the number of oracle queries to the strong DH oracle affects the concrete security of this assumption. That is, how does the security of strong DH degrade with the number of queries to the stDH oracle? We are not aware of any concrete attacks that exploit the oracle to solve the CDH problem more efficiently than other algorithms for CDH. In particular, in many elliptic curves with practical bilinear pairings it is reasonable to assume hardness of CDH, even though the bilinear pairing is a much stronger tool than a strong DH oracle.

A crucial technique in any tight proof using Diffie-Hellman problems is rerandomisation [6], where a single Diffie-Hellman problem instance can be turned into many, in such a way that an answer to any one of them can be turned into an answer to the original instance. We will use this technique in our proofs.

The Strong Twin Diffie-Hellman Problem. The strong twin Diffie-Hellman problem was introduced by Cash, Kiltz, and Shoup [13] at EUROCRYPT 2008. It is closely related to the standard computational Diffie-Hellman (CDH) problem, except that it "twins" certain group elements, in order to enable an efficient "trapdoor-DDH" test that makes it possible to simulate a strong-CDH oracle. This makes it possible to show that the twin-DH problem is *equivalent* to the standard CDH problem. Let $\mathsf{twinDH}_{x_0,x_1}(Y, Z_0, Z_1)$ be an oracle which returns 1 if and only if $\mathsf{DDH}(g^{x_0}, Y, Z_0) = 1$ *and* $\mathsf{DDH}(g^{x_1}, Y, Z_1) = 1$.

Definition 4. *Consider the following experiment involving an adversary $\mathcal{A}$. The experiment samples $x_0, x_1, y \xleftarrow{\$} \mathbb{Z}_p$ and starts $\mathcal{A}^{\mathsf{twinDH}_{x_0,x_1}(\cdot,\cdot,\cdot)}(g^{x_0}, g^{x_1}, g^y)$. The advantage of $\mathcal{A}$ in solving the* strong twin Diffie-Hellman *problem is defined as*

$$\mathrm{Adv}^{\mathsf{2\text{-}CDH}}_{\mathbb{G},g}(\mathcal{A}) := \Pr\left[\mathcal{A}^{\mathsf{twinDH}_{x_0,x_1}(\cdot,\cdot,\cdot)}(g^{x_0}, g^{x_1}, g^y) = (g^{x_0 y}, g^{x_1 y})\right]$$

The following theorem was proven by Cash, Kiltz, and Shoup [13, Theorem 3].

Theorem 1. *Let $\mathcal{A}$ be a strong twin DH adversary that makes at most Q queries to oracle $\mathcal{O}$ and runs in time $t_{\mathcal{A}}$. Then one can construct a DH adversary $\mathcal{B}$ that runs in time $t_{\mathcal{A}} \approx t_{\mathcal{B}}$ such that*

$$\mathrm{Adv}^{\mathsf{2\text{-}CDH}}_{\mathbb{G},g}(\mathcal{A}) \leq \mathrm{Adv}^{\mathsf{CDH}}_{\mathbb{G},g}(\mathcal{B}) + Q/p.$$

3 AKE Security Model

In this section we define our game-based key exchange security model. It is based on the real-or-random ("RoR") security definition of Abdalla, Fouque, and Pointcheval [2], and incorporates the extension of Abdalla, Benhamouda, and MacKenzie [1] to capture forward secrecy. The central feature of the RoR-model is that the adversary can make *many* Test-queries, and that all queries are answered with a "real" or "random" key based on the *same* random bit $\hat{b}$.

We prefer to work in a RoR-model because it automatically lends itself to tight composition with protocols that use the session keys of the key exchange protocol. For security models where there is only a single Test-query, or where each Test-query is answered based on an individual random bit [3, 19], such a composition is not automatically tight.

Although we mainly consider key exchange protocols with *implicit* authentication in this paper, we show in Section 8 how they can easily be upgraded to also have *explicit* authentication by adding key-confirmation messages to the protocol. The advantage of working in the RoR-model is that it allows us to do this transformation tightly.

Execution Environment. We consider μ parties $1, \ldots, \mu$. Each party i is represented by a set of ℓ oracles, $\{\pi_i^1, \ldots, \pi_i^\ell\}$, where each oracle corresponds to a session, i.e., a single execution of a protocol role, and where $\ell \in \mathbb{N}$ is the maximum number of protocol sessions per party. Each oracle is equipped with a randomness tape containing random bits, but is otherwise deterministic. Each oracle π_i^s has access to the long-term key pair (sk_i, pk_i) of party i and to the public keys of all other parties, and maintains a list of internal state variables that are described in the following:

- Pid_i^s ("peer id") stores the identity of the intended communication partner.
- $\Psi_i^s \in \{\emptyset, \mathtt{accept}, \mathtt{reject}\}$ indicates whether oracle π_i^s has successfully completed the protocol execution and "accepted" the resulting key.
- k_i^s stores the session key computed by π_i^s.

– sent_i^s contains the list of messages sent by π_i^s in chronological order.
– recv_i^s contains the list of messages received by π_i^s in chronological order.
– $\mathsf{role}_i^s \in \{\emptyset, \texttt{init}, \texttt{resp}\}$ indicates π_i^s's role during the protocol execution.

For each oracle π_i^s these variables are all initialized to the empty string $\emptyset$. The computed session key is assigned to the variable k_i^s if and only if π_i^s reaches the `accept` state, that is, we have $k_i^s \neq \emptyset \iff \Psi_i^s = \texttt{accept}$.

Partnering. To define when two oracles are supposed to derive the same session key we use a variant of matching conversations. In addition to agreement on their message transcripts, they should also agree upon each other's identities and have compatible roles (one being the initiator the other the responder). We remark that our protocol messages consist only of group elements and deterministic functions of them. This means that they are not vulnerable to the "no-match" attacks of Li and Schäge [32].

Definition 5 (Origin-oracle). *An oracle π_j^t is an* origin-oracle *for an oracle π_i^s if $\Psi_j^t \neq \emptyset$, $\Psi_i^s = \texttt{accept}$, and the messages sent by π_j^t equal the messages received by π_i^s, i.e., if $\mathsf{sent}_j^t = \mathsf{recv}_i^s$.*

Definition 6 (Partner oracles). *We say that two oracles π_i^s and π_j^t are* partners *if (1) each is an origin-oracle for the other; (2) each one's identity is the other one's peer identity, i.e., $\mathsf{Pid}_i^s = j$ and $\mathsf{Pid}_j^t = i$; and (3) they do not have the same role, i.e., $\mathsf{role}_i^s \neq \mathsf{role}_j^t$.*

Attacker Model. The adversary $\mathcal{A}$ interacts with the oracles through queries. It is assumed to have full control over the communication network, modeled by a Send query which allows it to send arbitrary messages to any oracle. The adversary is also granted a number of additional queries that model the fact that various secrets might get lost or leaked. The queries are described in detail below.

– $\mathsf{Send}(i, s, j, m)$: This query allows $\mathcal{A}$ to send any message m of its choice to oracle π_i^s on behalf of party P_j. The oracle will respond according to the protocol specification and depending on its internal state. For starting a role there are additional actions:
[Initiator] If $(\mathsf{Pid}_i^s, \Psi_i^s) = (\emptyset, \emptyset)$ and $m = \emptyset$, then this means that $\mathcal{A}$ requests π_i^s to start the initiator role with peer P_j. In this case, π_i^s will set $\mathsf{Pid}_i^s := j$ and $\mathsf{role}_i^s := \texttt{init}$.
[Responder] If $(\mathsf{Pid}_i^s, \Psi_i^s) = (\emptyset, \emptyset)$ and $m \neq \emptyset$, then this means that $\mathcal{A}$ requests π_i^s to start the responder role with peer P_j with first message m. In this case, π_i^s will set $\mathsf{Pid}_i^s := j$ and $\mathsf{role}_i^s := \texttt{resp}$.
– $\mathsf{RevLTK}(i)$: For $i \leq \mu$, this query allows the adversary to learn the long-term private key sk_i of user i. After the query i is said to be *corrupted*, and all oracles $\pi_i^1, \ldots, \pi_i^\ell$ now respond with $\perp$ to all queries.
– $\mathsf{RegisterLTK}(i, pk_i)$: For $i > \mu$, this query allows the adversary to register a new party i with public key pk_i. We do not require that the adversary knows the corresponding private key. After the query the pair (i, pk_i) is distributed to all other parties. Parties registered by $\mathsf{RegisterLTK}$ are corrupted by definition.

– $\mathsf{RevSessKey}(i, s)$: This query allows the adversary to learn the session key derived by an oracle. That is, query $\mathsf{RevSessKey}(i, s)$ returns the contents of k_i^s. Recall that we have $k_i^s \neq \emptyset$ if and only if $\Psi_i^s = \texttt{accept}$. After this query π_i^s is said to be *revealed*.

Note that unlike, e.g., [10, 12], we do not allow the adversary to learn the sessions' ephemeral randomness.

Security experiment. To define the security of a key exchange protocol we want to evaluate the attacker's knowledge of the session keys. Formally, we have an AKE security game, played between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, where the adversary can issue the queries defined above. Additionally, it is given access to a special Test query, which, depending on a secret bit $\hat{b}$ chosen by the challenger, either returns real or random keys. The goal of the adversary is to guess $\hat{b}$.

– $\mathsf{Test}(i, s)$: If $\Psi_i^s \neq \texttt{accept}$, return $\perp$. Else, return $k_{\hat{b}}$, where $k_0 = k_s^i$ and $k_1 \xleftarrow{\$} \mathcal{K}$ is a random key. If a Test query is repeated in the case $b = 1$, the same random key is returned. After the query, oracle π_i^s is said to be *tested*.

The adversary can issue many Test queries, to different oracles, but all are answered using the *same* bit $\hat{b}$.

The AKE security game, denoted $G_\Pi(\mu, \ell)$, is parameterized by the protocol Π and two numbers μ (the number of honest parties) and ℓ (the maximum number of protocol executions per party), and is run as follows.

1. $\mathcal{C}$ begins by drawing a random bit $\hat{b} \xleftarrow{\$} \{0, 1\}$, then generates μ long-term key pairs $\big\{(sk_i, pk_i) \;\big|\; i \in [1, \ldots, \mu]\big\}$, and initializes the collection of oracles $\big\{\pi_i^s \;\big|\; i \in [1, \ldots, \mu], s \in [1, \ldots, \ell]\big\}$.
2. $\mathcal{C}$ now runs $\mathcal{A}$, providing all the public keys $pk_1, \ldots, pk_\mu$ as input. During its execution, $\mathcal{A}$ may adaptively issue Send, RevLTK, $\mathsf{RevSessKey}$, $\mathsf{RegisterLTK}$ and Test queries any number of times and in arbitrary order. The only requirement is that all tested oracles remain *fresh* throughout the game (see Definition 7 below). Otherwise, the game aborts and outputs a random bit.
3. The game ends when $\mathcal{A}$ terminates with output b', representing its guess of $\hat{b}$. If not all test oracles are fresh, the security game outputs a random bit. If all test oracles are fresh and $b' = \hat{b}$, it outputs 1. Otherwise, it outputs 0.

Definition 7 (Freshness). *An oracle π_i^s is* fresh*, written $\mathsf{fresh}(i, s)$, if:*

(i) $\mathsf{RevSessKey}(i, s)$ *has not been issued,*
(ii) no query $\mathsf{Test}(j, t)$ *or* $\mathsf{RevSessKey}(j, t)$ *has been issued, where π_j^t is a partner of π_i^s, and*
(iii) Pid_i^s *was:*
  (a) not corrupted before π_i^s accepted if π_i^s has an origin-oracle, and
  (b) not corrupted at all if π_i^s has no origin-oracle.

Definition 8 (Winning events). *We define the following three winning events on game $G_\Pi(\mu, \ell)$.*

(i) Event $\mathsf{break}_{\mathrm{Sound}}$ *occurs if there exist two partner oracles* π_i^s *and* π_j^t *with* $k_i^s \neq k_j^t$*. In other words, there are two partner oracles which compute different session keys.*

(ii) Event $\mathsf{break}_{\mathrm{Unique}}$ *occurs if for some oracle* π_i^s *there exist distinct oracles* π_j^t *and* $\pi_{j'}^{t'}$ *such that* π_i^s *is a partner oracle to* both π_j^t *and* $\pi_{j'}^{t'}$*. In other words, there exists an oracle with more than one partner oracle.*

(iii) Let $\mathsf{guess}_{\mathrm{KE}}$ *be the output of game* $G_\Pi(\mu, \ell)$*. We define* $\mathsf{break}_{\mathrm{KE}}$ *to be the event* $\mathsf{guess}_{\mathrm{KE}} = 1$*.*

Definition 9 (AKE Security). *An attacker* $\mathcal{A}$ *breaks the security of protocol* Π*, if at least one of* $\mathsf{break}_{\mathrm{Sound}}$*,* $\mathsf{break}_{\mathrm{Unique}}$*, or* $\mathsf{break}_{\mathrm{KE}}$ *occurs in* $G_\Pi(\mu, \ell)$*. The* advantage *of the adversary* $\mathcal{A}$ *against AKE security of* Π *is*

$$\mathrm{Adv}_\Pi^{\mathsf{AKE}}(\mathcal{A}) = \max\left\{\Pr\left[\mathsf{break}_{\mathrm{Sound}}\right], \Pr\left[\mathsf{break}_{\mathrm{Unique}}\right], \left|\Pr\left[\mathsf{break}_{\mathrm{KE}}\right] - 1/2\right|\right\}.$$

We say that $\mathcal{A}$ $(\epsilon_{\mathcal{A}}, t, \mu, \ell)$-breaks Π *if its running time is* t *and* $\mathrm{Adv}_\Pi^{\mathsf{AKE}}(\mathcal{A}) \geq \epsilon_{\mathcal{A}}$*. The running time of* $\mathcal{A}$ *includes the running time of the security experiment (see [19, Remark 1]).*

Security properties. The core aspects of the security properties in our model are captured by the $\mathsf{break}_{\mathrm{KE}}$ event, combined with the adversary's capabilities and the restrictions imposed on them through the freshness predicate.

The freshness clauses (i) and (ii) imply that we only exclude the reveal of session keys for tested oracles as well as their partners. This encodes both (a) key independence if the revealed key is different from the session key: knowing some keys must not enable computing other keys, as well as (b) implicitly ensuring agreement on the involved parties, since sessions that compute the same session key but disagree on the parties would not be partnered, and reveal the Test session's key.

Our freshness clause (iii) encodes *weak forward secrecy*: the adversary can learn the peer's long-term key after the tested oracle accepted, but only if it has been passive in the run of the oracle [26]. Another property captured by our model is resistance to *key-compromise impersonation* attacks. Recall that KCI attacks are those where the adversary uses a party A's own private long-term key to impersonate other users towards A. This is (implicitly) encoded by the absence of any adversary restrictions on learning the private long-term key of a test-oracle itself. Additionally, the $\mathsf{break}_{\mathrm{Unique}}$ event captures the resistance to replay attacks. The $\mathsf{break}_{\mathrm{Sound}}$ event ensures that two parties that execute the protocol together in the absence of an attacker (or at least a passive one), compute the same session key.

Some recent protocols also offer *post-compromise security*, in which the communication partner π_j^t may be corrupted before π_i^s has accepted. However, in this work we consider only stateless protocols, which cannot achieve this goal [14].

4 Protocol Π

Protocol Π, defined in Fig. 1, uses a mix of static-ephemeral and ephemeral-ephemeral Diffie-Hellman key exchanges to get a protocol that is extremely efficient in terms of communications as well as computational effort required. Specifically, the two protocol participants exchange ephemeral Diffie-Hellman shares g^r and g^s for random r, s, and then compute a session key from three Diffie-Hellman shared secrets (static-ephemeral, ephemeral-static, ephemeral-ephemeral) as well as identities and a transcript. Note that this is very close to the Noise-KK pattern [36].

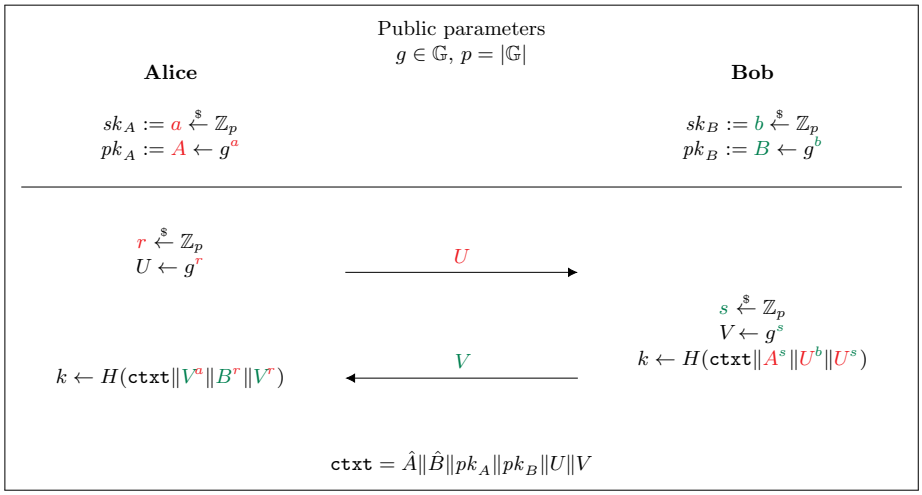

Fig. 1. Protocol Π. The session key is derived from the combination of the parties' static-ephemeral, ephemeral-static, and ephemeral-ephemeral DH values.

Theorem 2. *Consider the protocol* Π *defined in Fig. 1 where* H *is modeled as a random oracle. Let* $\mathcal{A}$ *be an adversary against the AKE security of* Π*. Then there exist adversaries* $\mathcal{B}_1$*,* $\mathcal{B}_2$ *and* $\mathcal{B}_3$ *against strong Diffie-Hellman such that*

$$\mathrm{Adv}^{\mathsf{AKE}}_{\Pi}(\mathcal{A}) \leq \mu \cdot \mathrm{Adv}^{\mathsf{stDH}}_{\mathbb{G},g}(\mathcal{B}_1) + \mathrm{Adv}^{\mathsf{stDH}}_{\mathbb{G},g}(\mathcal{B}_2) + \mu \cdot \mathrm{Adv}^{\mathsf{stDH}}_{\mathbb{G},g}(\mathcal{B}_3) + \frac{\mu\ell^2}{p}.$$

The strong Diffie-Hellman adversaries all run in essentially the same time as $\mathcal{A}$*, and make at most as many queries to their strong DH-oracle as* $\mathcal{A}$ *makes to its hash oracle* H*.*

The proof of the theorem is structured as a sequence of games running variations on the security experiment, with the first game identical to the experiment. We bound the difference in the probability of the event that the experiment

outputs 1 in each game. As a side effect, along the way we also get a bound on $\mathsf{break}_{\mathrm{Unique}}$. Then we argue that the probability that the experiment outputs 1 is 1/2 in the final game, which gives us a bound on $\mathsf{break}_{\mathrm{KE}}$. Since the scheme has perfect correctness, the theorem follows.

To achieve this result in the final game, we shall have our oracles choose session keys at random, without reference to secret keys or messages. Obviously, we have to ensure consistency with what the adversary can learn. This means that we have to make sure that partnered oracles both choose the same key (Game 2); that keys the adversary should be able to compute on his own are the same as chosen by the oracle (Game 2), and that corruptions of long-term keys that enable the adversary to compute session keys on his own return results consistent with previous RevSessKey-queries (Game 3 and 5).

The general technique we use is to have our oracles refrain from computing the input to the key derivation hash oracle, but instead check to see if the adversary somehow computes it. The idea is that computing the hash input is hard to simulate in the strong Diffie-Hellman game, but checking if someone else has computed the hash input is easy using the strong DH oracle provided.

We call an oracle *honest* (at some point) if the user it belongs to has not yet been corrupted (at that point). There are five types of oracles that we will have to deal with in separate ways, and the first four are essentially fresh oracles:

- (I) initiator oracles whose response message comes from a responder oracle, which has the same `ctxt` (i.e., they agree on the message transcript and participant identities and public keys) and which is honest when the response is received;
- (II) other initiator oracles whose intended peer is honest until the oracle accepts;
- (III) responder oracles whose initial message comes from an initiator, which has the same `ctxt` up to the responder message (thus agreeing on the first message and participant identities and public keys) and which is honest when the response is received;
- (IV) other responder oracles whose intended peer is honest until the oracle accepts; and
- (V) oracles whose intended peer is corrupted.

Note that at the time an initiator oracle starts, we cannot know if it will be of type I or II. However, we will know what type it is when it is time to compute the oracle's session key. We also remark that types I and III correspond to case (iii)a in the definition of freshness. Types II and IV correspond to case (iii)b.

In the following, let S_j denote the event that the experiment in Game j outputs 1.

Game 0. Our starting point Game 0 is the security experiment defining AKE security. We have that

$$\Pr\left[\mathsf{break}_{\mathrm{KE}}\right] = \Pr[S_0]. \tag{1}$$

We begin with an administrative step to avoid pathologies where honest players choose the same random nonces.

Game 1. In this game, we abort if two initiator oracles or two responder oracles ever arrive at the same `ctxt`. The probability of this happening can be upper-bounded by the probability of two oracles for the same peer choosing the same random exponents, and we get that

$$|\Pr[S_1] - \Pr[S_0]| \leq \frac{\mu\ell^2}{p}. \tag{2}$$

We also note that the event in this game that corresponds to $\mathsf{break}_{\mathrm{Unique}}$ cannot happen in this game. It follows that

$$\Pr[\mathsf{break}_{\mathrm{Unique}}] \leq \frac{\mu\ell^2}{p}. \tag{3}$$

4.1 Preparing Oracles

Our goal in this game is to change every oracle so that it no longer computes the input to the key derivation hash H, but instead checks if the adversary computes this input and adapts accordingly. This is essential for later games, since it allows us to replace every use of the secret key with queries to a strong DH oracle.

Game 2. In this game, we modify how our oracles determine their session keys. Note that at the point in time where an initiator oracle determines its session key, we know its type exactly.

A type III, IV or V responder oracle with $\texttt{ctxt} = \hat{i}\|\hat{j}\|pk_i\|pk_j\|U\|V$, secret key b and random exponent s does the following to determine its session key k: First, it checks to see if any oracle queries $\hat{i}\|\hat{j}\|pk_i\|pk_j\|U\|V\|W_1\|W_2\|W_3$ have been made satisfying

$$W_1 = pk_i^s \qquad W_2 = U^b \qquad W_3 = U^s. \tag{4}$$

If any such query is found k is set to the corresponding hash value. Otherwise, the session key is chosen at random. And if such a hash query happens later, the hash value is set to the chosen session key.

A type I initiator oracle will simply use the key from the corresponding responder oracle.

A type II or V initiator oracle with $\texttt{ctxt} = \hat{i}\|\hat{j}\|pk_i\|pk_j\|U\|V$, secret key a and random exponent r does the following to determine its session key k: First, it checks to see if any oracle queries $\hat{i}\|\hat{j}\|pk_i\|pk_j\|U\|V\|W_1\|W_2\|W_3$ have been made satisfying

$$W_1 = V^a \qquad W_2 = pk_j^r \qquad W_3 = V^r. \tag{5}$$

If any such query is found, k is set to the corresponding hash value. Otherwise, the session key is chosen at random. And if such a hash query happens later, the hash value is set to the chosen session key.

The only potential change in this game is at which point in time the key derivation hash oracle value is first defined, which is unobservable. It follows that

$$\Pr[S_2] = \Pr[S_1]. \tag{6}$$

4.2 Type IV Responder Oracles

Game 3. In this game type IV oracles choose their session key at random, but do not modify the hash oracle unless the intended peer is corrupted. If the adversary corrupts the intended peer i of a type IV oracle running as user j with secret key b, random exponent s and chosen key k, then from that point in time, any query of the form

$$\hat{i}\|\hat{j}\|pk_i\|pk_j\|U\|V\|pk_i^s\|U^b\|U^s$$

to the key derivation hash oracle H will result in the hash value k.

Unless one of these queries happen before user i is corrupted, the only change is at which point in time the key derivation hash oracle value is first defined, which is unobservable. Let F be the event that a query as above happens before the corresponding long-term key is corrupted. Then

$$|\Pr[S_3] - \Pr[S_2]| \leq \Pr[F].$$

Let F_i be the same event as F, but with the intended peer being user i. We then have that $\Pr[F] = \sum_i \Pr[F_i]$.

Next, consider the event E_i which is that for some type IV oracle as above, any query of the form

$$\hat{i}\|\hat{j}\|pk_i\|pk_j\|U\|V\|W_1\|W_2\|W_3 \qquad\qquad W_1 = pk_i^s = V^a \tag{7}$$

to the key derivation hash oracle H happens before user i is corrupted. Then $\Pr[F_i] \leq \Pr[E_i]$.

We shall now bound the probability of the event E_i by constructing an adversary against strong Diffie-Hellman. This adversary will embed its DH challenge in some user i's public key and type IV oracle responses for oracles whose intended peer is user i, and recover the solution to its DH challenge from the hash query in event E_i.

Strong Diffie-Hellman adversary $\mathcal{B}_1$. The algorithm $\mathcal{B}_1$ takes as input a DH challenge $(X, Y) = (g^x, g^y)$ and outputs a group element Z. It has access to a strong Diffie-Hellman oracle $\mathsf{stDH}_x(\cdot,\cdot)$.

Reduction $\mathcal{B}_1$ runs Game 2 with the following changes: it chooses i uniformly at random and sets user i's public key to $pk_i = X$ (and thus implicitly sets i's private key to the unknown value x). For type IV oracles whose intended peer is user i, $\mathcal{B}_1$ sets $V = Y \cdot g^{\rho_0}$, with ρ_0 random. If the adversary corrupts user i, the reduction $\mathcal{B}_1$ aborts. (For other users, the reduction simply returns the secret key, as in Game 2.)

We need to recognise hash queries of the form (4) and (5) that involve user i, as well as queries of the form (7). For (4), where user i acts in the responder role, we know the oracle's random exponent s, so we only need to recognise if W_2 is U raised to user i's secret key, which can be done by checking if $\mathsf{stDH}_x(U, W_2) = 1$.

For (5), where user i is the initiator, we know the oracle's random exponent r, so we only need to recognise if W_1 is V raised to user i's secret key, which can be done by checking if $\mathsf{stDH}_x(V, W_1) = 1$.

Finally, for (7), we need to recognise if a group element W_1 is V raised to user i's secret key, which can be done by checking if $\mathsf{stDH}_x(V, W_1) = 1$. When we recognise a query of the form (7), since we know that $V = Y \cdot g^{\rho_0}$, we output

$$Z = W_1 X^{-\rho_0} = V^x X^{-\rho_0} = Y^x g^{\rho_0 x} g^{-x\rho_0} = Y^x.$$

In other words, our adversary $\mathcal{B}_1$ succeeds whenever E_i would happen in Game 2. Furthermore, E_i in Game 2 can only happen before user i is corrupted, so whenever E_i would happen in Game 2, $\mathcal{B}_1$ would not have aborted.

We get that

$$\mathrm{Adv}^{\mathsf{stDH}}_{\mathbb{G},g}(\mathcal{B}_1) \geq \frac{1}{\mu} \sum_i \Pr[E_i] \geq \frac{1}{\mu} \sum_i \Pr[F_i] = \frac{1}{\mu} \Pr[F],$$

from which it follows that

$$|\Pr[S_3] - \Pr[S_2]| \leq \Pr[F] \leq \mu \cdot \mathrm{Adv}^{\mathsf{stDH}}_{\mathbb{G},g}(\mathcal{B}_1). \tag{8}$$

4.3 Type III Responder Oracles

Game 4. In this game type III responder oracles choose their session key at random, and do not modify the key derivation hash oracle.

Consider a type III responder oracle for user j with secret key b, random exponent s and intended peer i, who has secret key a. Unless the adversary ever makes a hash query of the form

$$\hat{i} \| \hat{j} \| pk_i \| pk_j \| U \| V \| W_1 \| W_2 \| W_3 \qquad\qquad W_3 = U^s, \tag{9}$$

this change is unobservable. Call this event F. We thus have

$$|\Pr[S_4] - \Pr[S_3]| \leq \Pr[F]. \tag{10}$$

We shall bound the probability of F by constructing an adversary against strong Diffie-Hellman. This adversary will embed its challenge in type I or II initiator oracles' message, as well as in type III responder oracles' message. It will recover the solution to its DH challenge from the hash query in event F.

Strong Diffie-Hellman adversary $\mathcal{B}_2$. The algorithm $\mathcal{B}_2$ takes as input a DH challenge $(X, Y) = (g^x, g^y)$ and outputs a group element Z. It has access to a strong DH-oracle $\mathsf{stDH}_x(\cdot, \cdot)$.

Our reduction $\mathcal{B}_2$ runs Game 3 with the following changes: for type I and II initiator oracles (we cannot distinguish these at this point in time), it computes $U = X \cdot g^{\rho_0}$, with ρ_0 random. For type III responder oracles, it computes $V = Y \cdot g^{\rho_1}$, with ρ_1 random. Note that in this game, the reduction knows all static secret keys, so user corruption is handled exactly as in Game 3.

We need to recognise hash queries of the form (5) for type II initiator oracles, as well as queries of the form (9) for type III oracles. Although we do not know the oracle's random exponents, we do know their secret keys. This means that we only

need to recognise if W_3 is V raised to $\log_g U = x + \rho_0$. Of course, if $W_3 = V^{x+\rho_0}$, then $W_3 V^{-\rho_0} = V^x$, which we can detect by checking if $\mathsf{stDH}_x(V, W_3 V^{-\rho_0}) = 1$. If this is the case for a query of the form (9), then we output

$$Z = W_3 \cdot V^{-\rho_0} \cdot X^{-\rho_1} = V^x \cdot X^{-\rho_1} = g^{yx+\rho_1 x} g^{-x\rho_1} = Y^x$$

as the solution to the DH challenge. In other words, $\mathcal{B}_2$ succeeds whenever F would happen in Game 3, hence

$$|\Pr[S_4] - \Pr[S_3]| \leq \Pr[F] \leq \mathrm{Adv}^{\mathsf{stDH}}_{\mathbb{G},g}(\mathcal{B}_2). \tag{11}$$

Note that we do not stop the simulation in the case we detect a hash query of the form (5) for a type II initiator oracle, because in this case the responder message V does not contain the embedded DH challenge.

4.4 Type II Initiator Oracles

Game 5. In this game type II initiator oracles choose their session key at random, but do not modify the hash oracle unless the intended peer is corrupted. If the adversary corrupts the intended peer j of a type II oracle running as user i with secret key a, random exponent r and chosen key k, then from that point in time, any query of the form

$$\hat{i}\|\hat{j}\|pk_i\|pk_j\|U\|V\|V^a\|pk_j^r\|V^r$$

to the key derivation hash oracle H will result in the hash value k.

Unless one of these queries happen before the user j is corrupted, the only change is at which point in time the key derivation hash oracle value is first defined, which is unobservable. Let F be the event that a query as above happens before the corresponding long-term key is corrupted. Then

$$|\Pr[S_5] - \Pr[S_4]| \leq \Pr[F].$$

Let F_j be the same event as F, but with the intended peer being user j. We then have that $\Pr[F] = \sum_j \Pr[F_j]$.

Next, consider the event E_j which is that for some type II oracle as above, any query of the form

$$\hat{i}\|\hat{j}\|pk_i\|pk_j\|U\|V\|W_1\|W_2\|W_3 \qquad\qquad W_2 = pk_j^r = U^b \tag{12}$$

to the key derivation hash oracle H happens before user j is corrupted. Then $\Pr[F_j] \leq \Pr[E_j]$.

We shall now bound the probability of the event E_j by constructing an adversary against strong Diffie-Hellman. This adversary will embed its DH challenge in some user j's public key and type II oracle messages for oracles whose intended peer is user j, and recover the solution to its DH challenge from the hash query in event E_j.

Strong Diffie-Hellman adversary $\mathcal{B}_3$. The algorithm $\mathcal{B}_3$ takes as input a DH challenge $(X, Y) = (g^x, g^y)$ and outputs a group element Z. It has access to a strong DH-oracle $\mathsf{stDH}_x(\cdot, \cdot)$.

Our reduction $\mathcal{B}_3$ runs Game 4 with the following changes: It chooses j uniformly at random and sets user j's public key to $pk_j = X$ (and thus implicitly sets j's private key to the unknown value $b = x$). For type I and II initiator oracles whose intended peer is user j, $\mathcal{B}_3$ sets $U = Y \cdot g^{\rho_0}$, with ρ_0 random. If the adversary corrupts user j, the reduction $\mathcal{B}_3$ aborts. (For other users, the reduction simply returns the secret key, as in Game 4.)

We need to recognise hash queries of the form (4) and (5) that involve user j, as well as queries of the form (12). For (4), where user j is the responder, we know the oracle's random exponent s, so we only need to recognise if W_2 is U raised to user j's secret key, which can be done by checking if $\mathsf{stDH}_x(U, W_2) = 1$. For (5), where user j is the initiator, we know the oracle's random exponent r, so we only need to recognise if W_1 is V raised to user j's secret key, which can be done by checking if $\mathsf{stDH}_x(V, W_1) = 1$. Finally, for (12), we need to recognise if a group element W_2 is U raised to user j's secret key, which can be done by checking if $\mathsf{stDH}_x(U, W_2) = 1$.

When we recognise a query of the form (12), meaning that $W_2 = U^x$ where know that $U = Y \cdot g^{\rho_0}$, then we output

$$Z = W_2 X^{-\rho_0} = U^x X^{-\rho_0} = Y^x g^{\rho_0 x} g^{-x\rho_0} = Y^x.$$

In other words, our adversary $\mathcal{B}_3$ succeeds whenever E_j would happen in Game 4. Furthermore, E_j in Game 4 can only happen before user j is corrupted, so whenever E_j would happen in Game 4, $\mathcal{B}_3$ would not have aborted. We get that

$$\mathrm{Adv}^{\mathsf{stDH}}_{\mathbb{G},g}(\mathcal{B}_3) \geq \frac{1}{\mu} \sum_j \Pr[E_j] \geq \frac{1}{\mu} \sum_j \Pr[F_j] = \frac{1}{\mu} \Pr[F],$$

from which it follows that

$$|\Pr[S_5] - \Pr[S_4]| \leq \Pr[F] \leq \mu \cdot \mathrm{Adv}^{\mathsf{stDH}}_{\mathbb{G},g}(\mathcal{B}_3). \tag{13}$$

4.5 Summary

Note that in Game 5, every session key is chosen at random independent of every key and sent message.

For type V oracles, the key derivation oracle is immediately programmed so that the session key is available to the adversary. But type V oracles are never fresh and therefore never subject to a Test query.

For type II and IV oracles, the key derivation hash oracle is programmed to make the session key available to the adversary only after the intended peer is corrupted. But if the intended peer is corrupted, a type II or IV oracle will become non-fresh, hence no Test query can be made to it.

For type I and III oracles, the key derivation hash oracle will never make the session key available to the adversary.

This means that for any oracle subject to a Test query, the session key is and will remain independent of every key and sent message. Which means that the adversary cannot distinguish the session key from a random key. It follows that

$$\Pr[S_5] = \frac{1}{2}. \tag{14}$$

Furthermore, (3) from Game 1 gives us $\Pr[\mathsf{break}_{\mathrm{Unique}}] \leq \mu\ell^2/p$. Because of perfect correctness $\Pr[\mathsf{break}_{\mathrm{Sound}}] = 0$. It is now easy to see that Theorem 2 follows from the construction of $\mathcal{B}_1$, $\mathcal{B}_2$ and $\mathcal{B}_3$ as well as equations (1), (2), (6), (8), (11), (13) and (14).

5 Avoiding the Strong Diffie-Hellman Assumption

The proof of Π relies on the strong Diffie-Hellman assumption, which is an interactive assumption. A natural goal is to look for a protocol whose proof relies on standard non-interactive assumptions. In this section we present two protocols that solve this problem. Both can be seen as different modifications of Π.

5.1 Protocol $\Pi_{\mathbf{Twin}}$

The first protocol, which we call Π_{Twin}, applies the twinning technique of [13] to the different DH values in Π. This requires some additional exponentiations over protocol Π, as well as the need to transmit one extra group element. The details are given in Fig. 2: instead of sending a single Diffie-Hellman share, the protocol initiator samples and sends two ephemeral shares, and both shares are used in the key derivation. This duplication allows us to reduce to twin Diffie-Hellman.

Theorem 3. *Consider the protocol Π_{Twin} defined in Fig. 2 where H is modeled as a random oracle. Let $\mathcal{A}$ be an adversary against the AKE security of Π_{Twin}. Then there exists adversaries $\mathcal{B}_1$, $\mathcal{B}_2$ and $\mathcal{B}_3$ against twin Diffie-Hellman such that*

$$\mathrm{Adv}^{\mathsf{AKE}}_{\Pi_{\mathrm{Twin}}}(\mathcal{A}) \leq \mu \cdot \mathrm{Adv}^{\mathsf{2\text{-}CDH}}_{\mathbb{G},g}(\mathcal{B}_1) + \mathrm{Adv}^{\mathsf{2\text{-}CDH}}_{\mathbb{G},g}(\mathcal{B}_2) + \mu \cdot \mathrm{Adv}^{\mathsf{2\text{-}CDH}}_{\mathbb{G},g}(\mathcal{B}_3) + \frac{\mu\ell^2}{p}.$$

The adversaries all run in essentially the same time as $\mathcal{A}$ and make at most as many queries to their twin DH oracle as $\mathcal{A}$ makes to its hash oracle H.

The proof is given in the full version. Note that by Theorem 1, we can tightly replace the twin Diffie-Hellman terms in the theorem statement by ordinary computational Diffie-Hellman terms.

5.2 Protocol $\Pi_{\mathbf{Com}}$

The second protocol, which we call Π_{Com}, again uses the twinning technique of [13], but this time only applied to the static DH values in Π. This provides tight

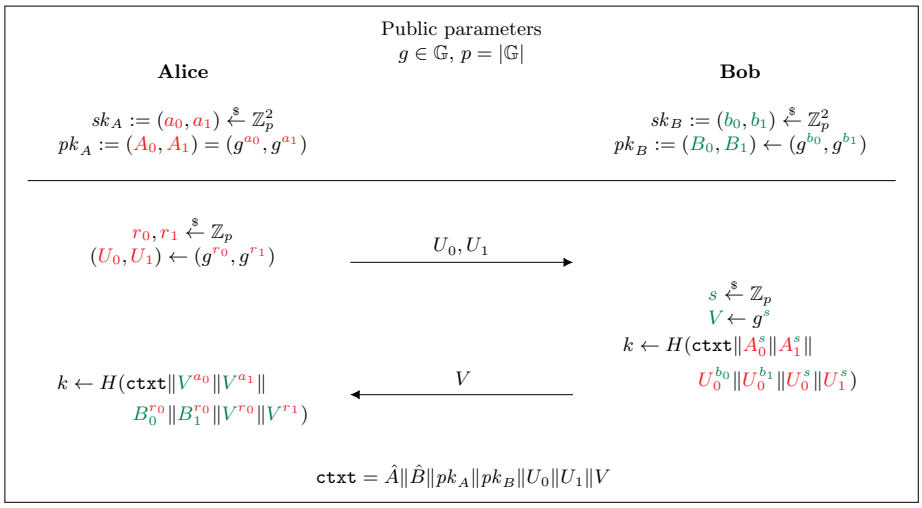

Fig. 2. Protocol Π_{Twin}. It is obtained from protocol Π by applying the twinning trick of [13] to the DH values.

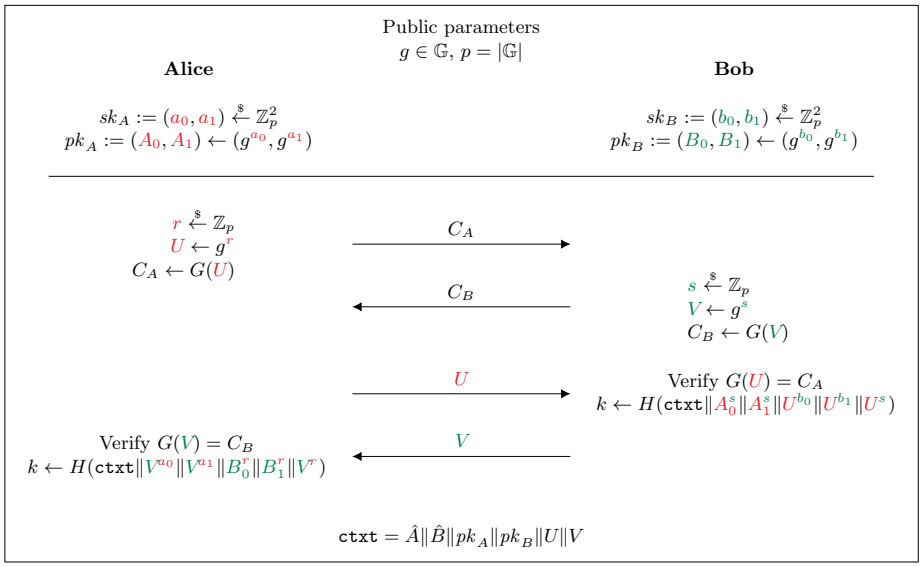

Fig. 3. Protocol Π_{Com}. It is obtained from protocol Π by applying the twinning trick of [13] to the static DH values and the commitment trick of [19] to the ephemeral DH values.

implicit authentication. However, instead of also twinning the ephemeral DH values we use a variant of the commitment trick of [19]. This reduces the number of exponentiations compared to Π_{Twin}, but adds another round of communication. Also, we need to rely on the Decision Diffie-Hellman assumption instead of computational Diffie-Hellman. The details are given in Fig. 3. The proof of the following theorem is given in the full version.

Theorem 4. *Consider the protocol Π_{Com} defined in Fig. 3 where H and G are modeled as random oracles. Let $\mathcal{A}$ be an adversary against the AKE security of Π_{Com}. Then there exists adversaries $\mathcal{B}_1$ and $\mathcal{B}_3$ against computational Diffie-Hellman and an adversary $\mathcal{B}_2$ against Decision Diffie-Hellman such that*

$$\mathrm{Adv}^{\mathsf{AKE}}_{\Pi_{\mathrm{Twin}}}(\mathcal{A}) \leq \mu \cdot \mathrm{Adv}^{\mathsf{CDH}}_{\mathbb{G},g}(\mathcal{B}_1) + \mathrm{Adv}^{\mathsf{DDH}}_{\mathbb{G},g}(\mathcal{B}_2) + \mu \cdot \mathrm{Adv}^{\mathsf{CDH}}_{\mathbb{G},g}(\mathcal{B}_3) + \frac{\mu\ell^2(1+2t)}{p}.$$

The adversaries all run in essentially the same time t as $\mathcal{A}$ and make at most as many queries to their twin DH oracle as $\mathcal{A}$ makes to its hash oracle H.

6 Efficiency Analysis

In this section we argue that our protocols are more efficient than other comparable[7] protocols in the literature when instantiated with theoretically sound parameter choices. There are two reasons for this. First, the most efficient key protocols do not have tight proofs. Hence, for theoretically sound deployment they must use larger parameters to compensate for the proof's security loss, which directly translates into more expensive operations. The result is that although some protocols require fewer operations than ours (typically group exponentiations), the increase in computational cost *per operation* dominates whatever advantage they might have over our protocols in terms of *number of operations*.

Second, the few known key exchange protocols which *do* have tight proofs, require a large number of operations or heavy cryptographic machinery. Thus, even though they can use small parameters, such as the P-256 elliptic curve, here the sheer number of operations dominates their advantage over our protocols.

To illustrate the first point in more detail, here are some examples of very efficient key exchange protocols having non-tight security proofs: UM [33], KEA+ [30], HMQV [26], CMQV [39], $\mathcal{TS}1/2/3$ [24], Kudla-Paterson [28], and NAXOS [29]. Typically, these proofs have a tightness loss between $L = O(\mu\ell)$ and $L = O(\mu^2\ell^2)$ as illustrated for a few of the protocols in Table 1.

Suppose we now want to compare the efficiency of the protocols Π, Π_{Twin}, Π_{Com} and HMQV, aiming for around 110-bits of security. Following Gjøsteen and Jager [19], let us imagine two different scenarios: a small-to-medium-scale setting with $\mu = 2^{16}$ users and $\ell = 2^{16}$ sessions per user, and a large-scale setting with $\mu = 2^{32}$ users and $\ell = 2^{32}$ sessions per user. To instantiate the protocols in a theoretically sound manner we need to select a group large enough so that the

[7] Comparing protocols is complex, and we return to this at the end of this section.

Table 1. The number of group exponentiations in our protocols compared to other protocols in the literature. All protocols are one-round except Π_{Com}, which has two rounds of communication. All security proofs are in the random oracle model. The security loss is in terms of the number of users (μ), the number of protocol instances per user (ℓ), and reduction's running time (t).

Protocol	**#Exponentiations**	**Assumption**	**Security loss** $O(\cdot)$
HMQV [26]	2.5	CDH	$\mu^2\ell^2$
NAXOS [29]	3	Gap-DH	$\mu^2\ell^2$
UM [33]	3	Gap-DH	$\mu^2\ell^2$
Kudla-Paterson [28]	3	Gap-DH	$\mu^2\ell$
KEA+ [30]	3	Gap-DH	$\mu\ell^\dagger$
Π (Fig. 1)	4	Strong-DH	μ
Π_{Twin} (Fig. 2)	8/7	CDH	μ
Π_{Com} (Fig. 3)	6	DDH	μ
GJ [19]	17	DDH	1

† Only when using pairing-friendly curves; otherwise $L = O(\mu\ell t)$.

Table 2. OpenSSL Benchmark Results for NIST Curves [19, Table 1].

Curve	**Exp. / Sec.**	**Time / Exp.**
NIST P-256	476.9	2.1 ms
NIST P-384	179.7	5.6 ms
NIST P-521	62.0	16.1 ms

underlying DH-assumptions are still hard even when accounting for the security loss. For simplicity, we only consider selecting among elliptic curve groups based on the NIST curves P-256, P-384, and P-521, and assume that the CDH, DDH, and Gap-DH problems are equally hard in each group.

HMQV. Supposing HMQV has a tightness loss of $L \approx \mu^2\ell^2$, this translates into a loss of 2^{64} in the small-to-medium-scale setting, and a loss of 2^{128} in the large-scale setting. To compensate we have to increase the group size by a factor of $L^2 \approx 2^{128}$ and $L^2 \approx 2^{256}$, respectively. With a target of 110-bit security, this means that we have to instantiate HMQV with curve P-384 and P-521, respectively.

Π, Π_{Twin}, Π_{Com}. Our protocols' security proofs have a tightness loss of $L \approx \mu$, which translates into 2^{16} in the small-to-medium-scale setting and 2^{32} in the large-scale setting. In the first setting P-256 is still sufficient for 110-bit security, but in the later setting P-384 must be used instead.

We can now compare these instantiations by multiplying the number of exponentiations required with the cost of an exponentiation in the relevant group.

For the latter values we use the OpenSSL benchmark numbers from Gjøsteen and Jager [19] (reproduced in Table 2). Calculating the numbers we get:

	HMQV	Π	Π_{Twin}	Π_{Com}
S-M	$2.5 \times 5.6 = 14$	$4 \times 2.1 = 8.4$	$8 \times 2.1 = 16.8$	$6 \times 2.1 = 12.6$
L	$2.5 \times 16.1 = 40.3$	$4 \times 5.6 = 22.4$	$8 \times 5.6 = 44.8$	$6 \times 5.6 = 33.6$

Observe that Π is more efficient than HMQV in both the small-to-medium-scale setting as well as in the large-scale setting despite needing more exponentiations. This is because it can soundly use smaller curves than HMQV due to the relative tightness of its reduction. Protocol Π_{Twin} is about as efficient as HMQV in both settings, while Π_{Com} lies somewhere in between Π and Π_{Twin}, but since it requires one extra round of communication a direct comparison is more difficult. Of course, the main reason to prefer Π_{Twin} and Π_{Com} over Π is the reliance on the weaker CDH and DDH assumptions rather than strong DH. A complicating factor in comparing with HMQV is the difference in security properties and security models (see the end of this section).

To illustrate the second point mentioned above—that our protocols are also more efficient than protocols with fully tight proofs—we also compute the numbers for the recent protocol of Gjøsteen and Jager (GJ) which is currently the most efficient key exchange protocol with a fully tight proof. Since GJ can use P-256 independent of the number of users and sessions its cost is $17 \times 2.1 = 35.7$ in both the small-to-medium scale setting as well as the large-scale setting. Nevertheless, we observe that the large number of exponentiations in GJ dominates its tightness advantage in realistic settings.

Thus, absent a fully tight proof, our protocols hit a proverbial "sweet spot" between security loss and computational complexity: they can be instantiated soundly on relatively small curves using only a few exponentiations.

Communication complexity. For completeness we also briefly mention communication complexity. Since in most implicitly-authenticated DH-based protocols each user only sends one or two group elements, there is in practice little difference between Π, Π_{Twin}, and Π_{Com}, and protocols like HMQV when it comes to communication cost. Especially if elliptic curve groups are used.

This is in contrast to the fully tight signature-based GJ protocol, which in total needs to exchange two group elements for the Diffie-Hellman key exchange, two signatures (each consisting of a random 256-bit exponent, two group elements, and four 256-bit exponents), and one hash value. Altogether, this gives a total of ≈ 545 bytes communicated when instantiated for a security level of, say, 128 bits [19, Section 5]. In comparison, Π, Π_{Twin}, and Π_{Com} would only need to exchange around 160 to 224 bytes for the same security level. This assumes curve P-384 and includes the addition of two 256-bit key-confirmation messages to provide explicit entity authentication in order to make the comparison with the GJ protocol fair.

On the (im)possibility of fairly comparing protocols. Our protocols are the first implicitly authenticated key exchange protocols that were designed to provide efficient deployment in a theoretically sound manner. This implies that we must compare their efficiency with other protocols with slightly different goals. In Table 1 we included protocols with closely related goals and similar structure, but not aiming for exactly the same target.

One example of such a different goal is that NAXOS was designed to be proven in the eCK model, which also allows the reveal of the randomness of the tested session, similar to HMQV. Our protocols, like TLS 1.3, currently do not offer this property. We conjecture that the NAXOS transformation could be directly applied to our protocols to obtain eCK-secure protocols without adding exponentiations, but it is currently unclear if this could be done with a *tight* proof, and hence we leave this to future work.

7 Optimality of our Security Proofs

In this section we will show that the tightness loss of $L = O(\mu)$ in Theorem 2, Theorem 3 and Theorem 4 is essentially optimal—at least for "simple" reductions. Basically, a "simple" reduction runs a *single* copy of the adversary only *once*. To the best of our knowledge, all known security reductions for AKE protocols are either of this type or use the forking lemma. For example, the original reduction for HMQV uses the forking lemma and thus is very non-tight, but does not fall under our lower bound. In contrast, the HMQV reduction by Barthe et al. [5] is simple and thus our lower bound applies. Hence, in order to give a tighter security proof, one would have to develop a completely new approach to prove security for such protocols.

Tightness bounds for different cryptographic primitives were given in [4, 15, 17, 18, 21, 23, 25, 31, 35, 37, 40], for instance. Bader et al. [4] describe a generic framework that makes it possible to derive tightness lower bounds for many different primitives. However, these techniques are only able to consider tight reductions from *non-interactive* assumptions, while our first protocol is based on the *interactive* strong Diffie-Hellman assumption. Morgan and Pass [34] showed how to additionally capture *bounded-round* interactive assumptions, but the strong Diffie-Hellman assumption does not bound the number of possible oracle queries, so we cannot use their approach directly.

Therefore we develop a new variant of the approach of Bader et al. [4], which makes it possible to capture interactive assumptions with an unbounded number of oracle queries, such as strong Diffie-Hellman assumption. For clarity and simplicity, we formulate this specifically for the class of assumptions and protocols that we consider, but we discuss possible extensions below.

Considered class of protocols. In the following we consider protocols where public keys are group elements of the form $pk = g^x$ and the corresponding secret key is $sk = x$. We denote the class of all protocols with this property with Π_{DH}. Note that this class contains, in particular, NAXOS [29], KEA+ [30], and HMQV [26].

Remark 1. One can generalize our results to *unique and verifiable* secret keys, which essentially requires that for each value pk there exists only one unique matching secret key sk, and that there exists an efficiently computable relation R such that $R(pk, sk) = 1$ if and only if (pk, sk) is a valid key pair. Following Bader et al. [4], one can generalize this further to so-called *efficiently re-randomizable* keys. We are not aware of concrete examples of protocols that would require this generality, and thus omit it here. All protocols considered in the present paper and the vast majority of high-efficiency protocols in the literature have keys of the form $(pk, sk) = (g^x, x)$, so we leave such extensions for future work.

Why does GJ18 not contradict our lower bound? As mentioned in Remark 1, our bound applies to protocols with *unique and verifiable* secret keys. In contrast, the protocol of Gjøsteen and Jager [19] constructs a tightly-secure digital signature scheme based on OR-proofs, where secret keys are *not* unique. As explained in [19, Section 1.1], these non-unique secret keys seem inherently necessary to achieve fully-tight security.

Simple reductions from (strong) Diffie-Hellman. Intuitively, a *simple* reduction $\mathcal{R} = \mathcal{R}^{\mathcal{O}}$ from (strong) CDH takes as input a CDH instance (g^x, g^y) and may query an oracle $\mathcal{O}$ that, on input Y, Z, returns 1 if and only if $Y^x = Z$ (cf. Definition 3). More formally:

Definition 10. *A* simple *reduction $\mathcal{R}$ interacts with an adversary $\mathcal{A}$ as follows.*

1. *$\mathcal{R}$ receives as input a CDH instance (g^x, g^y).*
2. *It generates μ public keys and starts $\mathcal{A}(pk_1, \ldots, pk_\mu)$. $\mathcal{R}$ provides $\mathcal{A}$ with access to all queries provided in the security model described in Section 3.*
3. *$\mathcal{R}$ outputs a value h.*

*We say that $\mathcal{R}$ is a $(t_{\mathcal{R}}, \epsilon_{\mathcal{R}}, \epsilon_{\mathcal{A}})$-*reduction*, if it runs in time at most $t_{\mathcal{R}}$ and for any adversary $\mathcal{A}$ with $\epsilon_{\mathcal{A}} = \mathrm{Adv}^{\mathsf{AKE}}_{\Pi}(\mathcal{A})$ holds that*

$$\Pr\left[h = g^{xy}\right] \geq \epsilon_{\mathcal{R}}.$$

We say that $\mathcal{R} = \mathcal{R}^{\mathcal{O}}$ is a reduction from the strong *CDH problem if it makes at least one query to its oracle $\mathcal{O}$, and a reduction from the CDH problem if not.*

Remark 2. The formalization in this section very specifically considers the computational problems CDH and sCDH, as concrete examples of reasonable hardness assumptions that a typical security proof for the protocols considered in this work may be based on. We will later discuss how our results can be extended to other interactive and non-interactive problems.

Theorem 5. *Let Π be an AKE protocol such that $\Pi \in \Pi_{\mathsf{DH}}$. Let $|\mathcal{K}|$ denote the size of the key space of Π. For any simple $(t_{\mathcal{R}}, \epsilon_{\mathcal{R}}, 1 - 1/|\mathcal{K}|)$-reduction $\mathcal{R}^{\mathcal{O}}$ from (strong) CDH to breaking Π in the sense of Definition 9 there exists an algorithm $\mathcal{M}^{\mathcal{O}}$, the meta-reduction, that solves the (strong) CDH problem in time $t_{\mathcal{M}}$ and with success probability $\epsilon_{\mathcal{M}}$ such that $t_{\mathcal{M}} \approx \mu \cdot t_{\mathcal{R}}$ and*

$$\epsilon_{\mathcal{M}} \geq \epsilon_{\mathcal{R}} - \frac{1}{\mu}.$$

Remark 3. Note that the lower bound $\epsilon_{\mathcal{M}} \geq \epsilon_{\mathcal{R}} - 1/\mu$ implies that the success probability $\epsilon_{\mathcal{R}}$ cannot significantly exceed $1/\mu$, as otherwise there exists an efficient algorithm $\mathcal{M}$ for a computationally hard problem. Note also that this implies that the reduction cannot be tight, as it "loses" a factor of at least $1/\mu$, even if the running time of $\mathcal{R}$ is not significantly larger than that of the adversary.

In the sequel we write $[\mu \setminus i]$ as a shorthand for $[1 \dots i-1, i+1 \dots \mu]$.

Proof. We describe a *meta-reduction* $\mathcal{M}$ that uses $\mathcal{R}$ as a subroutine to solve the (strong) CDH problem. Following Hofheinz et al. [21] and Bader et al. [4], we will first describe a *hypothetical* inefficient adversary $\mathcal{A}$. Then we explain how this adversary is efficiently simulated by $\mathcal{M}$. Finally, we bound the success probability of $\mathcal{M}$, which yields the claim.

Hypothetical adversary. The hypothetical adversary $\mathcal{A}$ proceeds as follows.

1. Given μ public keys $pk_1 = g^{x_1}, \dots, pk_\mu = g^{x_\mu}$, $\mathcal{A}$ samples a uniformly random index $j^* \xleftarrow{\$} [\mu]$. Then it queries $\mathsf{RevLTK}(i)$ for all $i \in [\mu \setminus j^*]$ to obtain all secret keys except for sk_{j^*}.
2. Next, $\mathcal{A}$ computes $sk_{j^*} = x_{j^*}$ from $pk_{j^*} = g^{x_{j^*}}$, e.g., by exhaustive search.[8]
3. Then $\mathcal{A}$ picks an arbitrary oracle, say π_s^1 for $s = (j^* + 1) \bmod \mu$, and executes the protocol with π_s^1, impersonating user j^*. That is, $\mathcal{A}$ proceeds exactly as in the protocol specification, but on behalf of user j^*. Note that $\mathcal{A}$ it is able to compute all messages and the resulting session key on behalf of user j^*, because it "knows" sk_{j^*}.
4. Finally, $\mathcal{A}$ asks $\mathsf{Test}(s, 1)$. Note that this is a valid Test-query, as $\mathcal{A}$ has never asked any $\mathsf{RevSessKey}$-query or $\mathsf{RevLTK}(j^*)$ to the peer j^* of oracle π_s^1. If the experiment returns the "real" key, then $\mathcal{A}$ outputs "1". Otherwise it outputs "0".

Note that $\mathcal{A}$ wins the security experiment with optimal success probability $1 - 1/|\mathcal{K}|$, where $|\mathcal{K}|$ is the size of the key space. The loss of $1/|\mathcal{K}|$ is due to the fact that the random key chosen by the Test-query may be equal to the actual session key.

Description of the meta-reduction. Meta-reduction $\mathcal{M}$ interacts with reduction $\mathcal{R}$ by simulating the hypothetical adversary $\mathcal{A}$ as follows.

1. $\mathcal{M}$ receives as input a CDH instance (g^x, g^y). It starts $\mathcal{R}$ on input (g^x, g^y).
2. Whenever $\mathcal{R}$ issues a query to oracle $\mathcal{O}$, $\mathcal{M}$ forwards it to its own oracle. Note that both oracles are equivalent, because $\mathcal{M}$ has simply forwarded the CDH instance.
3. When $\mathcal{R}$ outputs public keys $pk_1 = g^{x_1}, \dots, pk_\mu = g^{x_\mu}$ to $\mathcal{A}$, $\mathcal{M}$ makes a snapshot of the current state $st_{\mathcal{R}}$ of $\mathcal{R}$.
4. For $j \in [1 \dots \mu]$, $\mathcal{M}$ now proceeds as follows.

[8] Note that we are considering an inefficient adversary here. As usual for meta-reductions, we will later describe how $\mathcal{A}$ can be simulated *efficiently*.

(a) It lets $\mathcal{A}$ query $\mathsf{RevLTK}(i)$ for all $i \in [\mu \setminus j]$, in order to obtain all secret keys except for sk_j. Note that the reduction may or may not respond to all $\mathsf{RevLTK}(i)$ queries. For instance, $\mathcal{R}$ may abort for certain queries.
(b) Then it resets $\mathcal{R}$ to state $st_{\mathcal{R}}$.

5. Now $\mathcal{M}$ proceeds to simulate the hypothetical adversary. That is:
   (a) It picks a uniformly random index $j^* \xleftarrow{\$} [1 \dots \mu]$ and queries $\mathsf{RevLTK}(i)$ for all $i \in [\mu \setminus j^*]$.
   (b) Then it executes the protocol with π_s^1, impersonating user j^*. Note that this works only if $\mathcal{M}$ was able to obtain sk_{j^*} in Step (4).
   (c) Finally, $\mathcal{M}$ lets $\mathcal{A}$ ask $\mathsf{Test}(s, 1)$. If the experiment returns the "real" key, then $\mathcal{A}$ outputs "1". Otherwise it outputs "0".
6. If $\mathcal{R}$ outputs some value h throughout the experiment, then $\mathcal{M}$ outputs the same value.

Note that $\mathcal{M}$ provides a perfect simulation of the hypothetical adversary, provided that it "learns" sk_{j^*} in the loop in Step (4).

Analysis of the meta-reduction. $\mathcal{M}$ essentially runs reduction $\mathcal{R}$ at most μ times. Apart from that, it performs only minor additional operations, such that we have $t_{\mathcal{M}} \approx \mu \cdot t_{\mathcal{R}}$.

In order to analyse the success probability of $\mathcal{M}$, let us say that bad occurs, if j^* is the *only* index for which $\mathcal{R}$ did not abort in Step (4) of the meta-reduction. Note that in this case $\mathcal{M}$ learns all secret keys, *except* for sk_{j^*}, in which is the only case where the simulation of $\mathcal{A}$ in Step (5.b) fails. Since we may assume without loss of generality that the reduction $\mathcal{R}$ works for at least one index $j \in [\mu]$ and we chose $j^* \xleftarrow{\$} [\mu]$ uniformly random, we have

$$\Pr[\mathsf{bad}] \leq \frac{1}{\mu}.$$

Let $\mathsf{win}(\mathcal{R}, \mathcal{A})$ denote the event that $\mathcal{R}$ outputs $h = g^{xy}$ when interacting with $\mathcal{A}$, and $\mathsf{win}(\mathcal{R}, \mathcal{M})$ the corresponding event with $\mathcal{M}$. Since $\mathcal{M}$ simulates $\mathcal{A}$ perfectly unless bad occurs, we have

$$|\Pr[\mathsf{win}(\mathcal{R}, \mathcal{A})] - \Pr[\mathsf{win}(\mathcal{R}, \mathcal{M})]| \leq \Pr[\mathsf{bad}].$$

Furthermore, note that by definition we have $\epsilon_{\mathcal{R}} = \Pr[\mathsf{win}(\mathcal{R}, \mathcal{A})]$ and $\epsilon_{\mathcal{M}} = \Pr[\mathsf{win}(\mathcal{R}, \mathcal{M})]$. Hence we get $|\epsilon_{\mathcal{R}} - \epsilon_{\mathcal{M}}| \leq 1/\mu$, which in turn yields the lower bound $\epsilon_{\mathcal{M}} \geq \epsilon_{\mathcal{R}} - 1/\mu$.

Generalizations. The tightness lower bound proven above makes several very specific assumptions about the considered protocols, hardness assumptions, and security models. The main purpose of this is to keep the formalization and proof focused on the type of protocols that we are considering in this paper. However, a natural question is to which extent the results also apply to more general protocols, models, and assumptions, and whether and how the tightness bound can be evaded by tweaking the considered setting.

First of all, we consider only protocols where long-term secrets are of the form $(pk, sk) = (g^x, x)$. As already briefly discussed above, one can generalize this to other protocols, as long as the simulation of the hypothetical adversary by the meta-reduction is able to recover properly distributed secret keys. In particular, one can generalize to arbitrary *efficiently re-randomizable* long-term keys, as defined by Bader et al. [4]. Note that current AKE protocols with tight security proofs [3, 19] do *not* have efficiently rerandomizable keys, and therefore do not contradict our result.

In order to obtain a tighter security proof one may try to make different complexity assumptions. These can be either *non-interactive* (i.e., the reduction does not have access to an oracle $\mathcal{O}$, such as e.g. DDH), or stronger *interactive* assumptions. Let us first consider non-interactive assumptions. A very general class of such assumptions was defined abstractly in Bader et al. [4], and it is easy to verify that our proof works exactly the same way with such an abstract non-interactive assumption instead of CDH.

Some stronger assumptions may yield tight security proofs, but not all of them do. Consider for instance the *gap Diffie-Hellman* assumption, which is identical to strong Diffie-Hellman, except that the first input to the provided DDH-oracle is not fixed, but can be arbitrary. It is easy to verify that our proof also works for this assumption, in exactly the same way. More generally, our proof works immediately for any assumption for which the "winning condition" of the reduction is independent of the sequence of oracle queries issued by the reduction. An example of an interactive assumptions where this does *not* hold is the trivial interactive assumption that the protocol is secure (which, of course, immediately yields a tight security proof).

Finally, we note that our impossibility result holds also for many weaker or stronger AKE security models. We only require that the model allows for active attacks and provides a RevLTK query. Thus, the result immediately applies also to weaker models that, e.g., do not provide a RevSessKey-query or only a single Test-query, and trivially also for stronger models, such as eCK-style *ephemeral key reveals* [10, 12]. It remains an interesting open question whether stronger impossibility results (e.g., with quadratic lower bound) can be proven for such eCK-style definitions.

8 Adding Explicit Entity Authentication

In this section we describe how explicit entity authentication (EA) [9] can be added to our protocols by doing an additional key-confirmation step. Recall that EA is the aliveness property that fresh oracles are guaranteed to have a partner once they accept. Our construction is a generic compiler which transforms an arbitrary AKE protocol Π, secure according to Definition 9, into one that also provides EA. The details of the compiler are given in Fig. 4.

Specifically, protocol Π^+ begins by running protocol Π to obtain a session key k_Π. This key, which we henceforth call the *intermediate key* for protocol Π^+, is then used to derive two additional keys: k_a and k_m. The first key becomes the

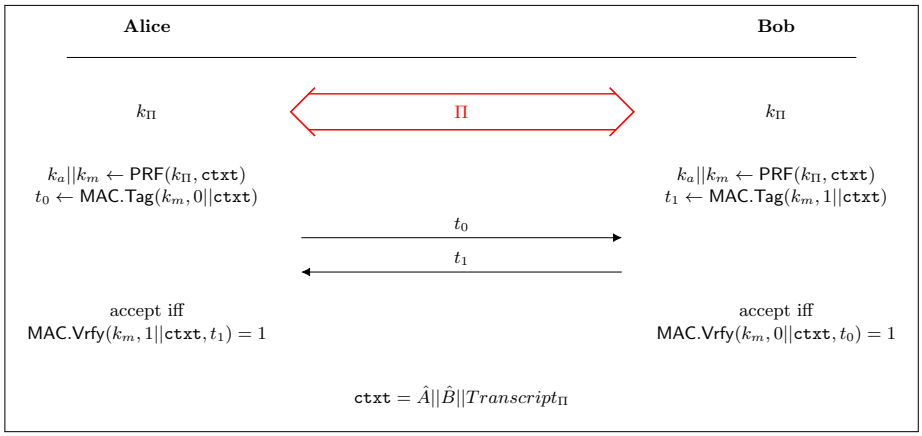

Fig. 4. Generic compiler from an AKE protocol Π with implicit authentication to a protocol Π^+ with explicit entity authentication.

final session key of protocol Π^+, while k_m is used to compute a key-confirmation message, i.e., a MAC, for each party. The EA property of Π^+ reduces to the AKE security of the initial protocol Π, the *multi-user* PRF security of the function used to derive k_a and k_m, as well as the *multi-user strong UF-CMA* (mu-SUF-CMA) security of the MAC scheme (see the full version for the formal definitions).

Theorem 6. *Let Π be an AKE protocol, let Π^+ be the protocol derived from Π as defined in Fig. 4, and let $\mathcal{A}$ be an adversary against the EA security of protocol Π^+. Then there exists adversaries $\mathcal{B}_1$, $\mathcal{B}_2$, $\mathcal{D}$, and $\mathcal{F}$, such that*

$$\mathrm{Adv}^{\mathsf{EA}}_{\Pi^+}(\mathcal{A}) \leq \mathrm{Adv}^{\mathsf{AKE}}_{\Pi}(\mathcal{B}_1) + 2 \cdot \mathrm{Adv}^{\mathsf{AKE}}_{\Pi}(\mathcal{B}_2) + \mathrm{Adv}^{\mathsf{mu\text{-}PRF}}_{\mathsf{PRF},\mu\ell}(\mathcal{D}) + \mathrm{Adv}^{\mathsf{mu\text{-}SUF\text{-}CMA}}_{\mathsf{MAC},\mu\ell}(\mathcal{F}),$$

where $\mu\ell$ is the number of sessions created by $\mathcal{A}$. The adversaries $\mathcal{B}_1$, $\mathcal{B}_2$, $\mathcal{D}$, and $\mathcal{F}$ all run in essentially the same time as $\mathcal{A}$.

Our result is basically a restatement of the theorem proved by Yang [41], but with two minor differences: (1) our result is stated for arbitrary protocols and not only two-message protocols, and (2) since we use the AKE-RoR model the proof is tighter and slightly simpler.

9 Conclusion

We showed that it is possible to achieve highly efficient AKE protocols that can be instantiated with theoretically sound parameters. Specifically, we gave protocol constructions that have only a linear tightness loss in the number of users, while using only a handful of exponentiations. Our constructions are at least as efficient as the best known AKE protocols in this setting. Perhaps surprisingly, our constructions only use standard building blocks as used by widely deployed protocols and are very similar to protocols like Noise-KK, and offer similar security guarantees.

While our proofs have a linear loss we have showed that this is actually unavoidable: any reduction from a protocol in our class to a wide class of hardness assumptions must lose a factor of at least μ. Thus, our reductions are optimal in this regard. Additionally, we proved that adding a key confirmation step tightly provides explicit authentication.

Taken together, these results demonstrate for the first time that AKE protocols can be instantiated in a theoretically sound way in real-world deployments without sacrificing performance.

Bibliography

[1] Abdalla, M., Benhamouda, F., MacKenzie, P.: Security of the J-PAKE password-authenticated key exchange protocol. In: 2015 IEEE Symposium on Security and Privacy. pp. 571–587. IEEE Computer Society Press (May 2015)

[2] Abdalla, M., Fouque, P.A., Pointcheval, D.: Password-based authenticated key exchange in the three-party setting. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 65–84. Springer, Heidelberg (Jan 2005)

[3] 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 (Mar 2015)

[4] Bader, C., Jager, T., Li, Y., Schäge, S.: On the impossibility of tight cryptographic reductions. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 273–304. Springer, Heidelberg (May 2016)

[5] Barthe, G., Crespo, J.M., Lakhnech, Y., Schmidt, B.: Mind the gap: Modular machine-checked proofs of one-round key exchange protocols. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 689–718. Springer, Heidelberg (Apr 2015)

[6] Bellare, M., Boldyreva, A., Micali, S.: Public-key encryption in a multi-user setting: Security proofs and improvements. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 259–274. Springer, Heidelberg (May 2000)

[7] Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: 38th FOCS. pp. 394–403. IEEE Computer Society Press (Oct 1997)

[8] 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 93. pp. 62–73. ACM Press (Nov 1993)

[9] Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO'93. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (Aug 1994)

[10] 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 (Mar / Apr 2015)

[11] Bhargavan, K., Fournet, C., Kohlweiss, M., Pironti, A., Strub, P.Y., Zanella Béguelin, S.: Proving the TLS handshake secure (as it is). In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part II. LNCS, vol. 8617, pp. 235–255. Springer, Heidelberg (Aug 2014)

[12] Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (May 2001)

[13] Cash, D., Kiltz, E., Shoup, V.: The twin Diffie-Hellman problem and applications. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 127–145. Springer, Heidelberg (Apr 2008)

[14] Cohn-Gordon, K., Cremers, C.J.F., Garratt, L.: On post-compromise security. In: IEEE 29th Computer Security Foundations Symposium, CSF 2016, Lisbon, Portugal, June 27 - July 1, 2016. pp. 164–178. IEEE Computer Society (2016), https://doi.org/10.1109/CSF.2016.19

[15] Coron, J.S.: Optimal security proofs for PSS and other signature schemes. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 272–287. Springer, Heidelberg (Apr / May 2002)

[16] Donenfeld, J.A.: WireGuard: Next generation kernel network tunnel. In: NDSS 2017. The Internet Society (Feb / Mar 2017)

[17] Fleischhacker, N., Jager, T., Schröder, D.: On tight security proofs for Schnorr signatures. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part I. LNCS, vol. 8873, pp. 512–531. Springer, Heidelberg (Dec 2014)

[18] Garg, S., Bhaskar, R., Lokam, S.V.: Improved bounds on security reductions for discrete log based signatures. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 93–107. Springer, Heidelberg (Aug 2008)

[19] 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, Heidelberg (Aug 2018)

[20] Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (Apr 2008)

[21] Hofheinz, D., Jager, T., Knapp, E.: Waters signatures with optimal security reduction. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 66–83. Springer, Heidelberg (May 2012)

[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 (Aug 2012)

[23] Jager, T., Stam, M., Stanley-Oakes, R., Warinschi, B.: Multi-key authenticated encryption with corruptions: Reductions are lossy. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part I. LNCS, vol. 10677, pp. 409–441. Springer, Heidelberg (Nov 2017)

[24] Jeong, I.R., Katz, J., Lee, D.H.: One-round protocols for two-party authenticated key exchange. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 04. LNCS, vol. 3089, pp. 220–232. Springer, Heidelberg (Jun 2004)

[25] Kakvi, S.A., Kiltz, E.: Optimal security proofs for full domain hash, revisited. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 537–553. Springer, Heidelberg (Apr 2012)
[26] Krawczyk, H.: HMQV: A high-performance secure Diffie-Hellman protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (Aug 2005)
[27] Krawczyk, H., Paterson, K.G., Wee, H.: On the security of the TLS protocol: A systematic analysis. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 429–448. Springer, Heidelberg (Aug 2013)
[28] Kudla, C., Paterson, K.G.: Modular security proofs for key agreement protocols. In: Roy, B.K. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 549–565. Springer, Heidelberg (Dec 2005)
[29] LaMacchia, B.A., 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 (Nov 2007)
[30] Lauter, K., Mityagin, A.: Security analysis of KEA authenticated key exchange protocol. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 378–394. Springer, Heidelberg (Apr 2006)
[31] Lewko, A.B., Waters, B.: Why proving HIBE systems secure is difficult. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 58–76. Springer, Heidelberg (May 2014)
[32] 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 (Oct / Nov 2017)
[33] Menezes, A., Ustaoglu, B.: Security arguments for the UM key agreement protocol in the NIST SP 800-56A standard. In: Abe, M., Gligor, V. (eds.) ASIACCS 08. pp. 261–270. ACM Press (Mar 2008)
[34] Morgan, A., Pass, R.: On the security loss of unique signatures. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018, Part I. LNCS, vol. 11239, pp. 507–536. Springer, Heidelberg (Nov 2018)
[35] Paillier, P., Vergnaud, D.: Discrete-log-based signatures may not be equivalent to discrete log. In: Roy, B.K. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 1–20. Springer, Heidelberg (Dec 2005)
[36] Perrin, T.: Noise protocol framework (2018), http://noiseprotocol.org
[37] Seurin, Y.: On the exact security of Schnorr-type signatures in the random oracle model. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 554–571. Springer, Heidelberg (Apr 2012)
[38] Signal Messenger: Technical information (2018), https://signal.org/docs
[39] Ustaoglu, B.: Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. Des. Codes Cryptography 46(3), 329–342 (2008)
[40] Wang, Y., Matsuda, T., Hanaoka, G., Tanaka, K.: Memory lower bounds of reductions revisited. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part I. LNCS, vol. 10820, pp. 61–90. Springer, Heidelberg (Apr / May 2018)
[41] Yang, Z.: Modelling simultaneous mutual authentication for authenticated key exchange. In: FPS. Lecture Notes in Computer Science, vol. 8352, pp. 46–62. Springer (2013)