On Optimal Tightness for Key Exchange with Full Forward Secrecy via Key Confirmation

Kai Gellert[1], Kristian Gjøsteen[2], Håkon Jacobsen[3,4], and Tibor Jager[1]

[1] University of Wuppertal, {kai.gellert,jager}@uni-wuppertal.de
[2] Norwegian University of Science and Technology, kristian.gjosteen@ntnu.no
[3] Thales Norway
[4] University of Oslo, hakon.jacobsen@its.uio.no

Abstract. A standard paradigm for building key exchange protocols with *full* forward secrecy (and *explicit* authentication) is to add key confirmation messages to an underlying protocol having only *weak* forward secrecy (and *implicit* authentication). Somewhat surprisingly, we show through an impossibility result that this simple trick must nevertheless incur a linear tightness loss in the number of parties for many natural protocols. This includes Krawczyk's HMQV protocol (CRYPTO 2005) and the protocol of Cohn-Gordon et al. (CRYPTO 2019).

Cohn-Gordon et al. gave a very efficient underlying protocol with *weak* forward secrecy having a linear security loss, and showed that this is optimal for certain reductions. However, they also claimed that *full* forward secrecy could be achieved by adding key confirmation messages, and *without any additional loss*. Our impossibility result disproves this claim, showing that their approach, in fact, has an overall *quadratic* loss.

Motivated by this predicament we seek to restore the original linear loss claim of Cohn-Gordon et al. by using a different proof strategy. Specifically, we start by lowering the goal for the underlying protocol with weak forward secrecy, to a *selective* security notion where the adversary must commit to a long-term key it cannot reveal. This allows a *tight* reduction rather than a linear loss reduction. Next, we show that the protocol can be upgraded to full forward secrecy using key confirmation messages with a linear tightness loss, even when starting from the weaker selective security notion. Thus, our approach yields an *overall* tightness loss for the fully forward-secret protocol that is only linear, as originally claimed. Finally, we confirm that the underlying protocol of Cohn-Gordon et al. can indeed be proven selectively secure, tightly.

1 Introduction

A security reduction is said to be *tight* if it preserves the security of the object being reduced to. The benefit of a tight reduction is that it allows to closely

This work has been supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, grant agreement 802823.

relate the security of a complex object to a simpler and, hopefully, more easy to analyze object. Moreover, a tight reduction allows cryptographic schemes to be instantiated with optimal parameters in a theoretically sound way. Unfortunately, for key exchange protocols the security reductions have historically been inordinately *non-tight*. But recent works have started to address these deficiencies either by doing more careful analyses of existing protocols, or by proposing new protocols more suitable for tighter reductions. Typically the approach has either been to use digital signatures with tight multi-user security [2,16,25,17], or signature-less protocols based on some variant of Diffie-Hellman [10,19]. While the latter tend to be more efficient than the signature-based approaches in practice, the comparison isn't completely fair since the signature-based protocols provide *full* forward secrecy and *explicit* authentication (AKE) while the protocols based on Diffie-Hellman ones only give *weak* forward secrecy and *implicit* authentication.

Forward secrecy, authentication and key confirmation. *Full* forward secrecy refers to the ability of a protocol to provide security of session keys even if the long-term secret key of a communicating party is leaked [7]. It is considered an essential standard security goal for modern key exchange protocols. *Weak* forward secrecy achieves this property under the assumption that the adversary does not actively interfere with the protocol messages in the sessions it attacks. The notions of weak and full forward secrecy are intimately connected to authentication [4,8,13].

Explicit authentication guarantees that the intended communication partner is indeed "online" and has actively participated in the protocol. Implicit authentication guarantees that only the intended peer will be able to derive the same session key. However, it does not guarantee that the peer actually has participated in the protocol. Typically, for key exchange protocols having only implicit authentication, an adversary can efficiently impersonate Alice towards Bob, in the sense that it sends messages on behalf of Alice and such that Bob derives a session key that he believes is suitable for communicating with Alice, but the adversary will still not be able to distinguish this session key from a random one. The question then becomes: how can we upgrade from weak forward secrecy and implicit authentication to full forward secrecy and explicit authentication, while still maintaining tightness and efficiency?

A very natural idea is to have the participants send *key confirmation* messages derived from the session key. This solution is simple, efficient, and has already been treated in multiple works [4,20,27,15,10,13]. Key confirmation ensures that a second protocol participant has indeed computed the same session key and thus turns an implicitly authenticated protocol into an explicitly authenticated protocol [13]. However, key confirmation also serves a dual role as a simple tool for upgrading weak forward secrecy to full forward secrecy. The HMQV-C [20] protocol is a notable example of this usage. In this paper we use key confirmation in both senses above.

Finally, while implicit (resp. explicit) authentication often corresponds to weak (resp. full) forward secrecy, we note that these are in fact separate notions. Protocols having explicit authentication but no forward secrecy are common (see

for instance the AKEP1 protocol in [5, Fig. 2]). Examples of protocols achieving full forward security without explicit authentication are given in [8, Protocol 4] and [12, Fig. 3].

Does key confirmation preserve tightness? Suppose Π is an arbitrary key exchange protocol providing weak forward secrecy and implicit authentication. The protocol participants exchange a session key using Π and from this derive key confirmation messages as well as a new session key using a pseudorandom function (PRF). They then exchange and verify the confirmation messages before outputting the new session key. Call this extended protocol Π^+. Intuitively, protocol Π^+ should achieve explicit authentication via a *tight* reduction to the implicit authentication of protocol Π as well as the multi-user security of the PRF. Indeed, this is the claim of Theorem 6 in [10]. Unfortunately, this claim turns out to be wrong. In fact, as we will show, for certain natural protocols, such as the protocol from Cohn-Gordon et al. [10] and HMQV [20], adding key confirmation messages like this or in any other deterministic way must *necessarily* lose a factor of U, where U is the number parties in the protocol. Hence, it seems that the notions of weak forward secrecy and implicit authentication are too weak to be *tightly* upgraded to full forward secrecy and explicit authentication by simply adding key confirmation messages. Interestingly and surprisingly, we will however also argue that an even weaker *selective* security notion is sufficient to obtain security with the same linear loss, which provides a new approach to obtain full forward secrecy with optimal linear tightness loss.

The flaw in Cohn-Gordon et al. [10]. The high-level idea of the security reduction from protocol Π^+ to protocol Π in [10] is as follows. The reduction uses Test or Reveal queries to get the session keys from Π and uses these to simulate the key confirmation messages of protocol Π^+. However, the reduction must decide which session keys it will reveal and which it will issue a Test query for. The trivial standard strategy would be to guess which session the adversary will test, but this cannot be deployed in [10] as it would immediately incur a linear security loss in the number of users *times* the number of sessions per user. Instead, the reduction proceeds as follows: once a session has reached an accepting state in the underlying AKE protocol Π, the reduction will base its decision on which query to use on the *current* freshness of the session. If the session is not fresh, it will issue a Reveal query. If the session is fresh, it will issue a Test query.

The problem with this strategy is that the freshness notion is with respect to protocol Π, which is only guaranteeing *weak* forward secrecy. Unfortunately this notion is too weak (i.e., too restrictive) to accommodate the reduction. More specifically, in a weak forward secrecy model the adversary is forbidden from both being active in a Test session and revealing the long-term secret of its peer. This is due to a classic attack described by [4] and [20] (see [8] and [12] for further discussions).

In this attack the adversary $\mathcal{A}$ impersonates Alice towards Bob by creating the DH share g^x on her behalf. Once Bob receives this message he creates its own DH share g^y and accepts in protocol Π. Since $\mathcal{A}$ has not (yet) revealed the

long-term key of Alice, Bob is at this point still fresh in protocol Π according to weak forward secrecy. Consequently, the reduction will issue a Test query in order to simulate its key confirmation message. However, if $\mathcal{A}$ now reveals the long-term key of Alice, then Bob will no longer be fresh (in protocol Π). At this point the reduction is stuck. This means that the reduction in Theorem 6 of Cohn-Gordon et al. [10] does not work. This issue has been confirmed by the authors of [10].

1.1 Our contributions

While the reduction of [10] does not work, can the result nevertheless be salvaged? On the one hand, we show that a tight reduction from full forward secrecy and explicit authentication to weak forward secrecy and implicit security is impossible for a large class of compilers and protocols of interest for practical applications. In particular, this includes the common key confirmation message compiler discussed above and the key exchange protocol of [10]. We prove this using a meta-reduction described in more detail below.

On the other hand, by considering what the actual *end goal* of [10] is, we can in fact recover the intended result by a rearranging of arguments. That is, the end goal is to create an as efficient as possible key exchange protocol having full forward secrecy and explicit authentication, with optimal tightness. Here, tightness is with respect to the lowest level building block of the protocol. In the case of [10] this is the strong Diffie-Hellman (stDH) assumption [1]. It was shown in [10] that a large class of DH-based implicitly authenticated key exchange protocols must lose a factor of U when reducing to stDH, where U is the number of parties. If the reduction from Π^+ to Π had been tight, as mistakenly claimed in [10], then the overall result would have been a protocol Π^+ with full forward secrecy and an optimal tightness loss of U to the stDH assumption. However, in light of our impossibility result, the best one can hope for using this approach is a loss of U^2, since there is a tightness loss of U going from Π^+ to Π and a tightness loss of U going from Π to stDH.

But this begs the question: if we know from the beginning that we at least have to lose a factor of U, is there some other way of structuring our arguments in order to avoid a quadratic loss? The solution is to first reduce the security of protocol Π^+ to an even *weaker* notion of implicit security for protocol Π, taking the "hit" of U here. Then, we show that this weaker notion for Π can be reduced further to stDH but now *tightly*. Thus, overall we obtain a modular reduction from Π^+ down to stDH losing only a factor of U.

What is this weaker notion for Π? It is a type of *selective security* game where the adversary needs to commit to a single party it will not reveal the long-term key of. This is related to the selective security notion from [21], but differs in two important ways. First, the requirement that one long-term key must stay *unrevealed*—rather than simply being involved in some event—makes the two notions technically incomparable (see Remark 1). Second, in [21] the adversary commits to *both* parties *and their sessions* involved in the event. This incurs a quadratic security loss, making it unsuitable for our purposes.

In summary, our main results are:

- We give a generic impossibility result showing that no security proof for adding key confirmation to a weakly forward-secret key exchange protocol can avoid a loss factor of U (Section 6).
- We provide an optimal security proof (i.e., with a linear loss in U) for adding key confirmation, which reduces to a weaker security notion for the underlying key exchange protocol (Section 4). This weaker notion allows us to avoid a tightness loss when proving the underlying protocol secure.
- Finally, we give a tight proof of the CCGJJ protocol [10] under the weaker notion, showing that the overall strategy achieves the end goal (Section 5).

One important consequence of our work is that future key exchange protocols having only weak forward secrecy can now be designed towards the goal of selective key secrecy, not full key secrecy. As shown by the analysis of CCGJJ [10], this may simplify proofs significantly.

Basic idea of the impossibility result. Our impossibility result shows essentially that if one constructs a protocol Π^+ from an underlying implicitly authenticated protocol Π by extending Π with two additional key confirmation messages, and if the security analysis of Π^+ includes a reduction $\mathcal{R}$ to the security of Π, then $\mathcal{R}$ loses a factor which is at least linear in the number U of parties. The basic idea of the argument is as follows.

We first define a (hypothetical) adversary $\mathcal{A}$, which proceeds in four steps:

1. First $\mathcal{A}$ receives the public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_U$ of all parties.
2. Then it interacts with $\mathcal{R}$ to create a session $s_{i,j}$ of protocol Π^+ for every pair of parties i, j. In all of these sessions the protocol is executed until $\mathcal{R}$ outputs the first of the two key confirmation messages.
   Note that $\mathcal{R}$ may simulate messages of Π^+ that correspond to messages of the underlying protocol Π by relaying these messages to its own security experiment. However, $\mathcal{R}$ also has to simulate the first key confirmation messages, which depend on the session key k of protocol Π.
3. Finally, $\mathcal{A}$ reveals the long-term secret keys of all but one party, receiving sk_i for all $i \in \{1, \ldots, U\} \setminus \{i^*\}$, where i^* is chosen at random by $\mathcal{A}$. Then $\mathcal{A}$ uses these secret keys to verify all key confirmation messages received from $\mathcal{R}$ for all sessions $s_{i,j}$ with $i \neq i^*$. If at least one of these key confirmation messages is invalid, then $\mathcal{A}$ terminates.
4. If all key confirmation messages are correct, then $\mathcal{A}$ breaks the security of Π^+ in a target session $s_{i^*,j}$ for some j.

$\mathcal{A}$ is a valid adversary that breaks Π^+ in the security experiment with maximal advantage. The choice of i^* is perfectly hidden from the reduction until Step 3 of $\mathcal{A}$, as all queries in Step 2 are independent of i^*. Note in particular that we can trivially simulate $\mathcal{A}$, if $\mathcal{R}$ outputs at least one invalid key confirmation message for any session $s_{i,j}$, $i \neq i^*$. Furthermore, note that we can always simulate the first three steps of $\mathcal{A}$ efficiently.

We will essentially argue that the reduction $\mathcal{R}$ is only able to simulate all key confirmation messages of sessions $s_{i,j}$ of parties $i \neq i^*$ properly, if it asks its security experiment to reveal the corresponding session keys $k_{i,j}$. However, at the same time $\mathcal{R}$ must not ask its security experiment to reveal the session key $k_{i^*,j}$ of the target session $s_{i^*,j}$, as otherwise it cannot leverage $\mathcal{A}$ to break the security of this session. Hence, the reduction faces the challenge that it has to "predict" i^* already in Step 2 of the adversary, in order to make sure that the key confirmation messages are simulated correctly, but still $\mathcal{A}$ can be leveraged to break the security of Π. Since i^* is chosen uniformly from $\{1, \ldots, U\}$, this yields a linear loss in U.

We stress that this sketch of the impossibility result is simplified, the actual formal result is more involved and subtle. For instance, in Section 6.1 we formulate precise conditions on which classes of reductions, protocols Π, and which constructions of Π^+ are covered by the impossibility result. These will cover the construction from [10] but also many other natural constructions.

The common way of arguing that a reduction $\mathcal{R}$ does not "need" $\mathcal{A}$ in certain cases is to perform a meta-reduction where $\mathcal{A}$ can efficiently be simulated in these cases. Normally, the standard approach of meta-reductions used in many prior works, such as [18,23,3], is to *rewind* $\mathcal{R}$ in order to be able to simulate $\mathcal{A}$ properly. Unfortunately, these results are usually only able to rule out reductions to *non-interactive* hardness assumptions. In contrast, the assumption that Π is secure is *interactive*. By rewinding $\mathcal{R}$ and running it multiple times with different queries from the "snapshot" state, we might cause $\mathcal{R}$ to make a sequence of queries that is not allowed in the key exchange security experiment of Π, such as revealing and then testing the same session s. Hence, we need to find another argument that avoids rewinding.

The main technical novelty of this result is that it consists of a combination of several different meta-reductions that enable us to argue that the reduction indeed "commits" itself to one particular choice of i^* after simulating the key confirmation messages in Step 2 of $\mathcal{A}$. The proof consists of several meta-reductions that do not perform rewinding and only simulate the first two or three steps of $\mathcal{A}$ (which can be done efficiently), in combination with a careful argument showing that this is indeed sufficient.

2 Definitions

The formalism and definitions we use to model key exchange protocols are adapted from de Saint Guilhelm et al. [13]. Unlike the traditional Bellare–Rogaway [5,6,4] and (e)CK models [9,22], security in this model is not formulated as a single all-in-one game that implicitly captures all the properties a protocol should have. Instead, security is split into many smaller definitions that each captures a single "atomic" security property. This leads to a slight increase in the number of definitions, as well as the number proofs one have to carry out in order to establish a protocol as "secure". On the other hand, the advantage of

this approach is that each definition/property is much simpler and focused, and many of the proofs will similarly also be very simple.

2.1 Syntax

A *key exchange protocol* is a tuple of algorithms $(\mathsf{KeyGen}, \mathsf{Init}, \mathsf{Run})$ where KeyGen is the long-term key generation algorithm; Init creates a *session state* at party i having intended peer j and role role, and returns this session's initial message (empty if a responder role); and Run takes as input a session state st and a message m and outputs an updated state st' and response message m'.

Session state. A *session state* st consists of the following variables.

- $\mathsf{accept} \in \{\mathsf{true}, \mathsf{false}, \perp\}$ – indicates the status of the key exchange run; initialized to $\perp$ and indicates a running, non-completed, session.
- $\mathsf{key} \in \{0,1\}^* \cup \{\perp\}$ – the local session key derived during the key exchange run; set once $\mathsf{accept} = \mathsf{true}$.
- $\mathsf{role} \in \{\mathsf{init}, \mathsf{resp}\}$ – the role of the session in the key exchange run.
- party – the party identity to which this session belongs.
- peer – the party identity of the intended peer for this key exchange run.
- sk – the secret long-term key of the party this session belongs to.
- pk – the public long-term key of the intended peer of the session.
- $\mathsf{transcript}$ – the (ordered) transcript of all messages sent and received by session s. We use $\mathsf{transcript}^-$ to denote the transcript minus the last message.
- aux – auxiliary protocol specific state, such as internal randomness and ephemeral values.

Security experiment We shall use the generic formal experiment $\mathbf{Exp}^{\mathsf{Pred}}_{\Pi,U}(\mathcal{A})$ given in Fig. 1 to define the various security properties of a key exchange protocol (see Section 3). The experiment is parameterized on a *security predicate* Pred that captures the security property being modeled. The experiment uses a number of counters, variables and collections for bookkeeping purposes.

- $\mathsf{query\_ctr}$ – incremented for each query made by the adversary. Used to order events in time; needed to define (full) forward secrecy.
- $\mathsf{session\_ctr}$ – incremented for each new session created. Each session state is associated with a unique session number which functions as an administrative label for that session (state). The session number is also given to the adversary which can use it as an opaque handle to refer to a given session in its queries. We use the notation "$s.\mathsf{x}$" to refer to the variable x of the session state identified by the administrative session number s. Note that the adversary cannot "dereference" a session number in order to obtain internal variables of the session state.
- $\mathsf{Accepted}$, Tested, $\mathsf{Revealed}$, $\mathsf{RevealedLTK}$ – associative arrays that records when a session accepted, was tested, or when its session or long-term key was revealed.

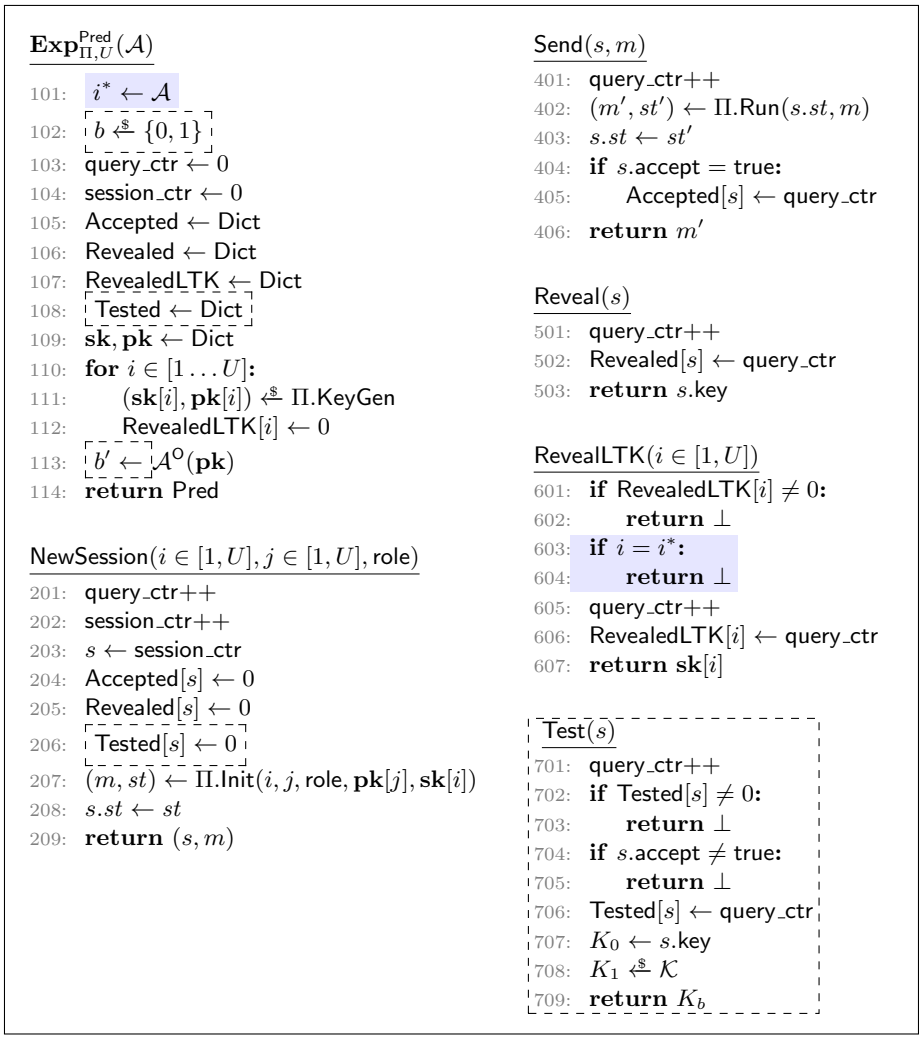

Fig. 1: Generic experiment parameterized on winning predicate Pred, where $\mathcal{A}$ can make the queries in $\mathsf{O} = \{\mathsf{NewSession}, \mathsf{Send}, \mathsf{Reveal}, \mathsf{RevealLTK}, \boxed{\mathsf{Test}}\}$. Code in dashed boxes is only for the key secrecy game; code in filled boxes is only for the selective key secrecy game. The notation $s.st \leftarrow st'$ means to assign all the variables in st' to the corresponding variables associated with session s. Dict defines an associative array.

Common predicates. It will be useful to introduce a number of predicates on the security experiment.

Definition 1 (Origin sessions). *A (possibly non-accepted) session s' is an* origin-session *for an accepted session s if predicate $\mathsf{Orig}(s, s')$ holds true, where*

$$\mathsf{Orig}(s, s') \iff s'.\mathsf{transcript} \in \{s.\mathsf{transcript}, s.\mathsf{transcript}^{-}\}. \tag{1}$$

Definition 2 (Partnering). *Two sessions s, s' are* partners *if they have matching conversations; that is, if the predicate $\mathsf{Partner}(s, s')$ holds true, where*

$$\mathsf{Partner}(s, s') \iff s.\mathsf{transcript} = s'.\mathsf{transcript}. \tag{2}$$

Like [13] we do not require partners to agree upon each other's identities. This is an authentication property which will be covered by other definitions in Section 3. Unlike [13] we use matching conversations instead of abstract session identifiers as our partnering mechanism. This is mainly done for the sake of concreteness and is not a fundamental difference, although certain well-known pitfalls need to be avoided when using matching conversations [24].

Definition 3 (SameKey). *The predicate $\mathsf{SameKey}(s, s')$ holds true if the sessions both have established a session key and they are equal, that is*

$$\mathsf{SameKey}(s, s') \iff [s.\mathsf{key} = s'.\mathsf{key} \neq \bot]. \tag{3}$$

Definition 4 (Authentication fresh). *A session is* authentication fresh *if the long-term key of its intended peer has not been revealed, that is:*

$$\mathsf{aFresh}(s) \iff \mathsf{RevealedLTK}[s.\mathsf{peer}] = 0. \tag{4}$$

Finally, we define freshness predicates used for the key secrecy games. These come in two flavors: *weak forward secrecy* and *full forward secrecy* [4]. Common to both is that the adversary cannot reveal the session key of a tested session or its partner. The difference is how long-term key leakage is handled. For weak forward secrecy the adversary is forbidden from revealing the long-term key of a session's peer if it was actively interfering in the protocol run of the session (indicated by the lack of an origin-session for the session in question). For full forward secrecy this restriction is lifted, provided the leak happened *after* the session in question accepted.

Definition 5 (Session key freshness). *Let $s.\mathsf{peer} = j$. The $\mathsf{kFreshWFS}(s)$ (resp. $\mathsf{kFreshFFS}(s)$) predicate hold if:*

$$\mathsf{Revealed}[s] = 0 \tag{5}$$

$$\forall s' :: \mathsf{Partner}(s, s') \implies \mathsf{Revealed}[s'] = 0 \wedge \mathsf{Tested}[s'] = 0 \tag{6}$$

$$\text{(wFS)} \quad \{s' \mid \mathsf{Orig}(s, s')\} = \emptyset \implies \mathsf{aFresh}(s) \tag{7}$$

$$\text{(fFS)} \quad \{s' \mid \mathsf{Orig}(s, s')\} = \emptyset \implies \mathsf{aFresh}(s) \vee (\mathsf{RevealedLTK}[j] > \mathsf{Accepted}[s]) \tag{8}$$

3 Protocol security properties

This section defines the security properties a secure key exchange protocol ought to have. The breakdown follows that of [13] and consists of: soundness properties (match and key-match soundness); various authentication properties (implicit/explicit key and entity authentication); and session key secrecy. An application will typically require all of these properties. Refer to [13] for further discussion and background.

3.1 Match soundness

Match soundness is primarily a sanity check on the choice of partnering mechanism. Namely, partnered sessions should derive the same session key (9); and sessions will at most have one partner (10).

Definition 6 (Match soundness). *The* Match *predicate evaluates to 1 iff* $\forall s, s', s''$*:*

$$\mathsf{Partner}(s, s') \implies \mathsf{SameKey}(s, s') \tag{9}$$

$$(\mathsf{Partner}(s, s') \wedge \mathsf{Partner}(s, s'')) \implies s' = s'' \tag{10}$$

The match soundness advantage *of an adversary* $\mathcal{A}$ *is*

$$\mathbf{Adv}^{\mathsf{Match}}_{\Pi,U}(\mathcal{A}) \stackrel{\text{def}}{=} \Pr[\mathbf{Exp}^{\mathsf{Match}}_{\Pi,U}(\mathcal{A}) \Rightarrow 1] \tag{11}$$

3.2 Key-match soundness

Key-match soundness (KMSound) is basically the converse of Match soundness. While Match soundness says (among other things) that partners should have equal session keys, KMSound says that sessions having equal session keys should be partners.

Definition 7 (Key-match soundness). *The* KMSound *predicate evaluates to 1 if and only if*

$$\forall s :: (\mathsf{aFresh}(s) \wedge s.\mathsf{accept}) \implies \forall s' :: (\mathsf{SameKey}(s, s') \implies \mathsf{Partner}(s, s')) \tag{12}$$

The key-match soundness advantage *of an adversary* $\mathcal{A}$ *is*

$$\mathbf{Adv}^{\mathsf{KMSound}}_{\Pi,U}(\mathcal{A}) \stackrel{\text{def}}{=} \Pr[\mathbf{Exp}^{\mathsf{KMSound}}_{\Pi,U}(\mathcal{A}) \Rightarrow 1]. \tag{13}$$

3.3 Implicit key authentication

Implicit key authentication stipulates that two sessions that derive the same session key should agree upon *whom* they are sharing this key with.

Definition 8 (Implicit key authentication). *The* iKeyAuth *predicate evaluates to 1 if and only if*

$$\forall s :: s.\mathsf{accept} \implies \forall s' :: (\mathsf{SameKey}(s, s') \implies s.\mathsf{peer} = s'.\mathsf{party})$$

The implicit key authentication advantage *of an adversary* $\mathcal{A}$ *is*

$$\mathbf{Adv}^{\mathsf{iKeyAuth}}_{\Pi,U}(\mathcal{A}) \stackrel{\text{def}}{=} \Pr[\mathbf{Exp}^{\mathsf{iKeyAuth}}_{\Pi,U}(\mathcal{A}) \Rightarrow 1]. \tag{14}$$

3.4 Explicit key authentication

Explicit key authentication stipulates that any two sessions that derive the same session key should agree upon whom they are sharing this key with (as for implicit key authentication), and as long as the session is authentication fresh some other session deriving the same session key should exist.

Obviously, the session that sends the last message can never guarantee that this message arrives at its destination, which means that this session can only achieve the notion of *almost-full* key authentication, namely that an origin session should exist and any origin session that has derived a session key has derived the same key. A session that receives the last message, however, can guarantee that another session exists that has derived the same key, and thereby achieve *full* key authentication.

Let $\mathcal{L}_{\mathsf{rcv}}$ denote the collection of all sessions that *receives* the last message of the protocol, and let $\mathcal{L}_{\mathsf{send}}$ denote the collection of all sessions that *sends* the last message of the protocol.

Definition 9 (Explicit key authentication). *The* fexKeyAuth *predicate (resp.* afexKeyAuth *predicate) evaluates to 1 if and only if*

$$\forall s \in \mathcal{L}_{\mathsf{rcv}}\,(\text{resp. } \mathcal{L}_{\mathsf{send}}) :: s.\mathsf{accept} \implies \forall s' :: (\mathsf{SameKey}(s,s') \Rightarrow s.\mathsf{peer} = s'.\mathsf{party}) \quad (15)$$

$$\wedge$$

$$\text{(full)} \qquad \mathsf{aFresh}(s) \Rightarrow \exists s' :: \mathsf{SameKey}(s,s')$$

$$\text{(almost-full)} \qquad \mathsf{aFresh}(s) \Rightarrow \exists s' :: \Big(\mathsf{Orig}(s,s') \wedge [s'.\mathsf{key} \neq \bot \implies \mathsf{SameKey}(s,s')]\Big)$$

The full (resp. almost-full) explicit key authentication advantage *of* $\mathcal{A}$ *is*

$$\mathbf{Adv}^{\mathsf{fexKeyAuth}}_{\Pi,U}(\mathcal{A}) \stackrel{\text{def}}{=} \Pr[\mathbf{Exp}^{\mathsf{fexKeyAuth}}_{\Pi,U}(\mathcal{A}) \Rightarrow 1] \quad (16)$$

$$\mathbf{Adv}^{\mathsf{afexKeyAuth}}_{\Pi,U}(\mathcal{A}) \stackrel{\text{def}}{=} \Pr[\mathbf{Exp}^{\mathsf{afexKeyAuth}}_{\Pi,U}(\mathcal{A}) \Rightarrow 1] \quad (17)$$

3.5 Explicit entity authentication

Explicit *entity* authentication is almost identical to explicit *key* authentication, the only difference being that the former is based on the Partner predicate while the latter is based on the SameKey predicate. Basically, explicit key authentication says that if a session with an honest peer accepts then there *is* some other session holding the same session key, while explicit entity authentication says that if a session with an honest peer accepts then it *has* a partner session.

Explicit key authentication and explicit entity authentication are closely related, as shown in [13] and further expounded in the full version.

Definition 10 (Explicit entity authentication). *The* fexEntAuth *predicate (resp.* afexEntAuth *predicate) evaluates to 1 if and only if*

$$\forall s \in \mathcal{L}_{\mathsf{rcv}}\,(\text{resp. } \mathcal{L}_{\mathsf{send}}) :: s.\mathsf{accept} \implies \forall s' ::(\mathsf{Partner}(s,s') \implies s.\mathsf{peer} = s'.\mathsf{party})$$

$$\wedge$$

$$\text{(full)} \qquad \mathsf{aFresh}(s) \implies \exists s' :: \mathsf{Partner}(s,s')$$

$$\text{(almost-full)} \quad \mathsf{aFresh}(s) \implies \exists s' :: \Big(\mathsf{Orig}(s,s') \wedge [s'.\mathsf{accept} \implies \mathsf{Partner}(s,s')]\Big)$$

The full (resp. almost-full) explicit entity authentication advantage *of* $\mathcal{A}$ *is*

$$\mathbf{Adv}^{\mathsf{fexEntAuth}}_{\Pi,U}(\mathcal{A}) \stackrel{\text{def}}{=} \Pr[\mathbf{Exp}^{\mathsf{fexEntAuth}}_{\Pi,U}(\mathcal{A}) \Rightarrow 1] \tag{18}$$

$$\mathbf{Adv}^{\mathsf{afexEntAuth}}_{\Pi,U}(\mathcal{A}) \stackrel{\text{def}}{=} \Pr[\mathbf{Exp}^{\mathsf{afexEntAuth}}_{\Pi,U}(\mathcal{A}) \Rightarrow 1] \tag{19}$$

3.6 Key secrecy

Key secrecy is defined as usual with the adversary using a Test query to get the real session key or a random key of a session. Note that the adversary may make multiple test queries, and they all share the same challenge bit, so that either all Test queries return real session keys, or all Test queries return random (and independently) sampled keys. Our experiment does not prevent the adversary from making Test queries for sessions that are not key fresh, so we need to account for this in the definition of advantage (called the *penalty-style* in [26]).

Definition 11 (Key secrecy). *If* $\forall s \in \mathsf{Tested} :: \mathsf{kFreshWFS}(s) = \mathsf{true}$ *(resp.* $\mathsf{kFreshFFS}(s) = \mathsf{true}$*), the* KeySecWFS *(resp.* KeySecFFS *) predicate returns 1 if and only if* $b' = b$*. Else it returns* b*. The* weak (resp. full) forward key secrecy advantage *of an adversary* $\mathcal{A}$ *is*

$$\mathbf{Adv}^{\mathsf{KeySecWFS}}_{\Pi,U}(\mathcal{A}) \stackrel{\text{def}}{=} 2 \cdot \Pr[\mathbf{Exp}^{\mathsf{KeySecWFS}}_{\Pi,U}(\mathcal{A}) \Rightarrow 1] - 1 \tag{20}$$

$$\mathbf{Adv}^{\mathsf{KeySecFFS}}_{\Pi,U}(\mathcal{A}) \stackrel{\text{def}}{=} 2 \cdot \Pr[\mathbf{Exp}^{\mathsf{KeySecFFS}}_{\Pi,U}(\mathcal{A}) \Rightarrow 1] - 1 \tag{21}$$

Selective key secrecy. The *selective* key secrecy experiment is defined over the experiment given in Fig. 1, where now the code inside the blue boxes is included. In the selective security experiment the adversary has to commit to one party it will not reveal the long-term key of throughout the game.

Definition 12 (Selective key secrecy). *If* $\forall s \in \mathsf{Tested} :: \mathsf{kFreshWFS}(s) = \mathsf{true}$*, the* S-KeySecWFS *predicate returns 1 if and only if* $b' = b$*. Else it returns* b*. The* selective key secrecy advantage *of an adversary* $\mathcal{A}$ *is*

$$\mathbf{Adv}^{\mathsf{S\text{-}KeySecWFS}}_{\Pi,U}(\mathcal{A}) \stackrel{\text{def}}{=} 2 \cdot \Pr[\mathbf{Exp}^{\mathsf{S\text{-}KeySecWFS}}_{\Pi,U}(\mathcal{A}) \Rightarrow 1] - 1. \tag{22}$$

Remark 1. Ordinary key secrecy does not reduce trivially to selective key secrecy with a U tightness loss as one might expect. Specifically, for an adversary that starts by revealing *all* long-term keys a reduction to selective key secrecy will not be able to simulate the one key it committed to. This makes our selective security notion incomparable to the selective notion of [21].

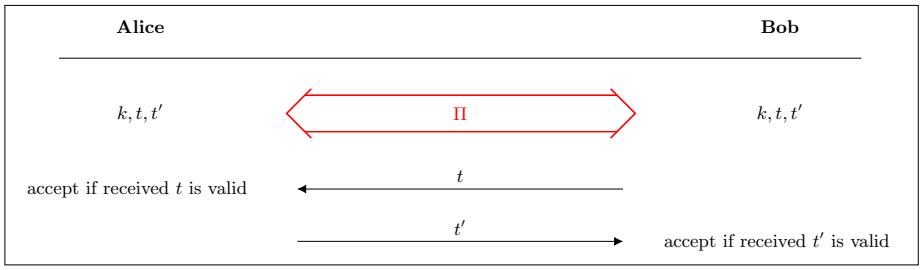

Fig. 2: Protocol Π^+ obtained by extending protocol Π with key confirmation tags. All session variables in Π^+ are inherited from Π, except for accept which is defined as shown. The session sending the last message in protocol Π sends tag t, and the session receiving the last message in protocol Π sends tag t'.

4 The security of adding key confirmation

Let Π denote an arbitrary key exchange protocol, and let Π^+ denote the protocol that extends Π by adding key confirmation messages from each side as illustrated in Fig. 2. Conventionally, the key confirmation messages are derived from the session key of Π using a PRF (and possibly a MAC) but in order to simplify the later analysis we assume that Π produces session keys of the form (k, t, t') directly. Protocol Π^+ is then derived from Π simply by defining its session key to be k, and the key confirmation tags to be t and t'. Using this trick we can relate the security of protocol Π^+ purely to the security of protocol Π without having to rely on PRFs or MACs.

Unfortunately, defining Π^+ in terms of the key triple output by Π introduces one technicality. We will often want to make an assertion of the form "if s and s' have equal keys in protocol Π^+ (meaning k), then they also have equal keys in protocol Π (meaning (k, t, t'))". While this assertion easily follows in practice—for instance if (k, t, t') is derived from the session transcript using a function for which getting a collision just in k is unlikely, such as an extendable-output function or a random oracle—in the generality we have presented Π and Π^+ above the assertion does not automatically follow. Thus, to cleanly state and prove our generic results we introduce the implication "equal $k \implies$ equal (k, t, t')" as an explicit security property.

To this end, let $\mathrm{prefix} : \{0,1\}^* \to \{0,1\}^*$ be a function that returns a prefix of a particular length (left unspecified) from its argument and define

$$\mathsf{SamePrefix}(s, s') \iff [s.\mathsf{key}, s'.\mathsf{key} \neq \bot \wedge \mathrm{prefix}(s.\mathsf{key}) = \mathrm{prefix}(s'.\mathsf{key})]. \quad (23)$$

Definition 13 (Same prefix security). *The* PreEqAllEq *predicate evaluates to* 1 *if and only if*

$$\forall s \in \mathcal{L}_{\mathsf{rcv}} :: s.\mathsf{accept} \implies \forall s' :: (\mathsf{SamePrefix}(s, s') \implies \mathsf{SameKey}(s, s')). \quad (24)$$

The same prefix advantage *of* $\mathcal{A}$ *is*

$$\mathbf{Adv}^{\mathsf{PreEqAllEq}}_{\Pi,U}(\mathcal{A}) \stackrel{\mathrm{def}}{=} \Pr[\mathbf{Exp}^{\mathsf{PreEqAllEq}}_{\Pi,U}(\mathcal{A}) \Rightarrow 1]. \quad (25)$$

We now state the first main theorem of the paper: a protocol with weak forward secrecy can be upgraded to full forward secrecy by adding key confirmation messages, and, moreover, this upgrade can be achieved with a linear security loss in the number of parties. The second main theorem of the paper is that this linear security loss is unavoidable for a larger class of compilers (see Section 6).

Theorem 1. *Let $\mathcal{A}$ be an adversary against key secrecy for Π^+. Then there exist adversaries $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_6$, all with about the same runtime as $\mathcal{A}$, such that*

$$\begin{aligned}\mathbf{Adv}^{\mathsf{KeySecFFS}}_{\Pi^+,U}(\mathcal{A}) \leq\ & 4 \cdot U \cdot \mathbf{Adv}^{\mathsf{S\text{-}KeySecWFS}}_{\Pi,U}(\mathcal{B}_1) + 8 \cdot \mathbf{Adv}^{\mathsf{iKeyAuth}}_{\Pi,U}(\mathcal{B}_2) + \frac{4US}{2^{\mathsf{taglen}}} \\ & + 12 \cdot \mathbf{Adv}^{\mathsf{Match}}_{\Pi,U}(\mathcal{B}_3) + 4 \cdot \mathbf{Adv}^{\mathsf{KMSound}}_{\Pi,U}(\mathcal{B}_4) + 12 \cdot \mathbf{Adv}^{\mathsf{PreEqAllEq}}_{\Pi,U}(\mathcal{B}_5) \\ & + 4 \cdot \mathbf{Adv}^{\mathsf{KeySecWFS}}_{\Pi,U}(\mathcal{B}_6),\end{aligned}$$

where taglen *is the length of the key confirmation tags used by Π^+ and S is the number of sessions.*

The proof of Theorem 1 is given in the full version. At a high level the proof consists of two parts: one where all accepting sessions with peers whose long-term keys are unrevealed have an origin session, and one where they don't. In the first case full forward key secrecy of protocol Π^+ reduces straightforwardly to the weak forward key secrecy of protocol Π. The main challenge is to deal with the second case, namely to prove that protocol Π^+ achieves explicit entity authentication. In fact, the main technical tool for this is to prove that Π^+ achieves explicit *key* authentication, which is where we use the *selective* key secrecy notion. The proof of explicit key authentication is the focus of Section 4.1.

4.1 Implicit to explicit key authentication

In this section, we establish that explicit key authentication can be based on *selective* key secrecy, implicit key authentication, and same prefix security. This is a key technical result needed to restore the tight security of the explicitly authenticated protocol of [10]. The use of selective security may have further applications in constructing highly efficient explicitly authenticated key exchange protocols with full forward secrecy in the future.

Lemma 1. *Let $\mathcal{A}$ be an adversary against full explicit key authentication for Π^+. Then there exists an adversary $\mathcal{B}_2$ against selective key secrecy and an adversary $\mathcal{B}_1$ against implicit key authentication and same prefix security, both with the same runtime as $\mathcal{A}$, such that*

$$\mathbf{Adv}^{\mathsf{fexKeyAuth}}_{\Pi^+,U}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{iKeyAuth}}_{\Pi,U}(\mathcal{B}_1) + \mathbf{Adv}^{\mathsf{PreEqAllEq}}_{\Pi,U}(\mathcal{B}_1) + U \cdot \mathbf{Adv}^{\mathsf{S\text{-}KeySecWFS}}_{\Pi,U}(\mathcal{B}_2) + \frac{US}{2^{\mathsf{taglen}}},$$

where taglen *is the length of the key confirmation tags used by Π^+ and S is the number of sessions.*

We need to deal with two cases. The first case considers attacks on explicit authentication that result from breaking implicit authentication of the underlying protocol Π. This case does not incur a tightness loss.

The second case considers attacks on explicit authentication that rely on breaking the weak forward secrecy of the underlying protocol Π. The important point is that in order to break explicit authentication, the partner long-term key must be unrevealed at the point in time where authentication is broken. This means that the session will be fresh at the time authentication is broken, which means that we can deduce the challenge bit at the point in time where authentication is broken. Any subsequent reveal of the partner long-term key can therefore be ignored.

Proof. The proof is structured as a sequence of games. Let Win_{G_i} denote the event that $\mathcal{A}$ wins in Game i. Winning in this case means that full explicit key authentication in (15) from Definition 9 does not hold.

Game 0. This is the original game for protocol Π^+. We have that

$$\mathbf{Adv}^{\mathsf{fexKeyAuth}}_{\Pi^+,U}(\mathcal{A}) = \Pr[\mathsf{Win}_{G_0}]. \tag{26}$$

Game 1. We modify the game so that if (15) does not hold for the Π part of a session of Π^+, then that session never accepts. Let Except_{G_1} be the event that this happens.

It is immediate that until Except_{G_1} happens, Game 1 proceeds exactly as Game 0, so

$$|\Pr[\mathsf{Win}_{G_1}] - \Pr[\mathsf{Win}_{G_0}]| \leq \Pr[\mathsf{Except}_{G_1}]. \tag{27}$$

We create an adversary $\mathcal{B}_1$ against implicit key authentication for Π that runs a copy of $\mathcal{A}$ and uses its experiment to run the Π part of Π^+. When a session of Π outputs a session key, $\mathcal{B}_1$ reveals the session key and uses that to simulate sending and receiving the key confirmation messages. Let $\mathsf{Win}_{\mathcal{B}_1}$ denote the probability that $\mathcal{B}_1$ wins.

It is immediate that $\mathcal{B}_1$ and its experiment together simulate the experiment in Game 0 perfectly with respect to the copy of $\mathcal{A}$ run by $\mathcal{B}_1$. Since SameKey for Π^+ implies SamePrefix for Π, if (15) does not hold for Π^+ in an execution, either it will not hold when we consider the game as an execution of Π, or PreEqAllEq will not hold when we consider the game as an execution of Π. In other words,

$$\Pr[\mathsf{Except}_{G_1}] \leq \Pr[\mathsf{Win}^{\mathsf{iKeyAuth}}_{\mathcal{B}_1}] + \Pr[\mathsf{Win}^{\mathsf{PreEqAllEq}}_{\mathcal{B}_1}]. \tag{28}$$

Game 2. We modify the game by sampling $j \in \{1, 2, \ldots, U\}$ at the start. Let Win'_{G_2} be the event that Win_{G_2} happens and one session for which authentication is broken has the jth key as its peer's public key. Clearly,

$$\Pr[\mathsf{Win}'_{G_2}] \geq \frac{1}{U}\Pr[\mathsf{Win}_{G_2}] = \frac{1}{U}\Pr[\mathsf{Win}_{G_1}]. \tag{29}$$

Game 3. We modify the game so that if (15) holds for a session of Π^+ that has the jth key as its peer public key but it has no origin session, then that session samples random tags to use for the Π^+ part of the protocol, instead of the tags output by Π.

It is immediate that

$$\Pr[\mathsf{Win}'_{G_3}] \leq \frac{S}{2^{\mathsf{taglen}}}. \tag{30}$$

We create an adversary $\mathcal{B}_2$ against selective key secrecy for Π that runs a copy of $\mathcal{A}$ and uses its experiment to run the Π part of Π^+, simulating the sending and receiving of key confirmation messages as modified in Game 2, further modified as follows:

- At the start, $\mathcal{B}_2$ selects an integer $j \in \{1, 2, \ldots, U\}$.
- When a session of Π, using the ith key as its peer key, outputs a session key, (15) holds for the session and it has no origin session, then:
  - If $i \neq j$, then $\mathcal{B}_2$ reveals the session key of the session and uses that key to simulate the Π^+ part of the session.
  - If $i = j$, then $\mathcal{B}_2$ tests the Π instance and uses that key to simulate the Π^+ part of the session.
- If $\mathcal{A}$ reveals the jth long-term key, $\mathcal{B}_2$ outputs 0 and stops.

If $\mathcal{A}$ breaks authentication for a session with the jth key as its peer key, $\mathcal{B}_2$ outputs 1, otherwise $\mathcal{B}_2$ outputs 0.

Let $\mathsf{Win}'_{\mathcal{B}_2,b}$ denote the event that $\mathcal{B}_2$ outputs 1, when its experiment has the secret bit b. We have that

$$\mathbf{Adv}^{\mathsf{S\text{-}KeySecWFS}}_{\Pi,U}(\mathcal{B}_2) = |\Pr[\mathsf{Win}_{\mathcal{B}_2,0}] - \Pr[\mathsf{Win}_{\mathcal{B}_2,1}]|. \tag{31}$$

If the experiment's secret bit $b = 0$, then $\mathcal{B}_2$ perfectly simulates Game 2 with respect to the Win'_{G_2} event, since the only observable difference is that $\mathcal{B}_2$ terminates when Win'_{G_2} no longer can occur (when the jth long-term key is revealed), so

$$\Pr[\mathsf{Win}_{\mathcal{B}_2,0}] = \Pr[\mathsf{Win}'_{G_2}]. \tag{32}$$

If the experiment's secret bit $b = 1$, then $\mathcal{B}_2$ perfectly simulates Game 3 with respect to the Win'_{G_3} event, again because of termination, so

$$\Pr[\mathsf{Win}_{\mathcal{B}_2,1}] = \Pr[\mathsf{Win}'_{G_3}]. \tag{33}$$

The claim follows from (26)–(33). □

The same argument proves the similar statement:

Lemma 2. *Let $\mathcal{A}$ be an adversary against almost full explicit key authentication for Π^+. Then there exists an adversary $\mathcal{B}_2$ against selective key secrecy and an adversary $\mathcal{B}_1$ against implicit key authentication and same prefix security, both with the same runtime as $\mathcal{A}$, such that*

$$\mathbf{Adv}^{\mathsf{afexKeyAuth}}_{\Pi^+,U}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{iKeyAuth}}_{\Pi,U}(\mathcal{B}_1) + \mathbf{Adv}^{\mathsf{PreEqAllEq}}_{\Pi,U}(\mathcal{B}_1) + U \cdot \mathbf{Adv}^{\mathsf{S\text{-}KeySecWFS}}_{\Pi,U}(\mathcal{B}_2) + \frac{US}{2^{\mathsf{taglen}}},$$

where taglen *is the length of the key confirmation tags used by Π^+.*

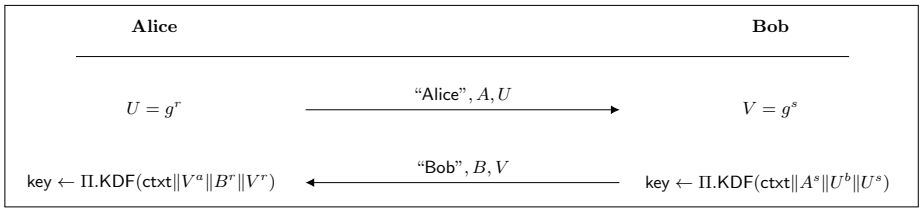

Fig. 3: The CCGJJ protocol from [10] for prime-ordered group G with generator g. Alice has secret key a, public key A; Bob has secret key b, public key B. Their context ctxt contains their names, their public keys and the two messages U and V. We include names and public keys in the messages; in practice they may be communicated in other ways.

4.2 Additional security reductions

Lemma 1 is *the* key result needed to show that protocol Π^+ achieves explicit authentication from the implicitly authenticated protocol Π in a manner that only incurs a tightness loss of U. From this result all the other security properties defined in Section 3 follow in a straightforward and modular way. That is, in the full version we show that Π^+ has all the security properties from Section 3 via reductions to the same properties of protocol Π, in addition to same prefix security. Moreover, none of these reductions loses more than a factor of U (in fact, most of the reductions are fully tight, and those that are not only accrue the U term as a result of invoking Lemma 1).

5 The CCGJJ protocol

The CCGJJ protocol from Cohn-Gordon et al. [10] is a highly efficient implicitly authenticated key exchange protocol with optimal tightness. We use this protocol to illustrate our framework, which means that we need to prove various properties for the protocol.

We begin by proving that the protocol has the basic properties we want, in particular match soundness, key match soundness and the same prefix property.

Proposition 1. *Let $\mathcal{A}$ be an adversary against the CCGJJ protocol. Then*

$$\mathbf{Adv}^{\mathsf{Match}}_{CCGJJ,U}(\mathcal{A}) \leq \frac{S^2}{2^{|\mathsf{key}|}}, \quad \mathbf{Adv}^{\mathsf{KMSound}}_{CCGJJ,U}(\mathcal{A}) \leq \frac{S^2}{|G|} \quad \textit{and} \quad \mathbf{Adv}^{\mathsf{iKeyAuth}}_{CCGJJ,U}(\mathcal{A}) \leq \frac{S^2}{2^{|\mathsf{key}|}}.$$

Proposition 2. *Let $\mathcal{A}$ be an adversary against the CCGJJ protocol. Then*

$$\mathbf{Adv}^{\mathsf{PreEqAllEq}}_{CCGJJ,U}(\mathcal{A}) \leq \frac{S^2}{2^{|\mathsf{key}|-2\mathsf{taglen}}}.$$

Proposition 3. *Let $\mathcal{A}$ be an adversary against selective key secrecy for CCGJJ. Then there exists adversaries $\mathcal{B}_1$, $\mathcal{B}_2$ and $\mathcal{B}_3$ against strong Diffie-Hellman (in group G with generator g) with essentially the same runtime as $\mathcal{A}$ such that*

$$\mathbf{Adv}^{\mathsf{S\text{-}KeySecWFS}}_{CCGJJ,U}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{stDH}}_{G}(\mathcal{B}_1) + \mathbf{Adv}^{\mathsf{stDH}}_{G}(\mathcal{B}_2) + \mathbf{Adv}^{\mathsf{stDH}}_{G}(\mathcal{B}_3) + \frac{US^2}{|G|}$$

The proof closely follows the structure of the proof in [10], modelling $\Pi.\mathsf{KDF}$ as a random oracle.

6 Impossibility of tightly-secure explicit authentication via key confirmation

In this section we show that a large, natural, and widely-used class of compilers for turning implicitly authenticated protocols into explicitly authenticated protocols, inevitably must incur a linear security loss in the number of parties. The class includes the generic compiler from [10] which was incorrectly claimed to achieve tight security but also the MAC-based approach to turn HMQV into HMQV-C [20] (which does not give explicit security bounds) and the compiler by Yang [27] (which has a linear loss in the number of parties times sessions per party).

6.1 Requirements on Π and Π^+

We want to consider *generic* approaches that turn *any* implicitly authenticated key exchange protocol Π into an explicitly authenticated protocol Π^+. To this end, we will in the sequel focus on underlying protocols Π and constructions Π^+ that satisfy certain requirements that we will define in this section.

Messages of Π are independent of the secret key. We consider protocols Π where the messages sent by a party are *independent* of the long-term secret key of this party (and the long-term secret is only used during session key computation). More precisely, we consider n-message protocols $\Pi = (\mathsf{KeyGen}, \mathsf{Init}, \mathsf{Run})$, with two associated session key computation algorithms $\Pi.\mathsf{KDF}, \Pi.\mathsf{KDF}'$. Recall that Run is a state-dependent algorithm, i.e., the state "prescribes" which protocol message needs to be generated next. The algorithms are executed as follows.

1. The initiator samples randomness r_I uniformly from some space (depending on the protocol) and uses this to compute the first protocol message and session state as
$$(m_1, st_1) \leftarrow \Pi.\mathsf{Init}(I, R, \mathsf{init}, \mathsf{pk}_R, -).$$
Here pk_R is the public key of the intended communication partner (the responder). Note that I's secret long-term key sk_I is not used to produce m_1, st_1.

2. Upon receiving message m_1 the responder samples uniform randomness r_R and initializes a session state as $st'_2 \leftarrow \Pi.\mathsf{Init}(R, I, \mathsf{resp}, \mathsf{pk}_I, -)$, then computes the second protocol message and an updated state as

$$(m_2, st_2) \leftarrow \Pi.\mathsf{Run}(st'_2, m_1).$$

   Again, note that R's secret long-term key sk_R is not used to produce m_2, st_2.
3. If $n > 2$, then the parties keep exchanging messages until the protocol is finished. Hence, whenever a party with state st_{i-1} receives a message m_i with $i < n$, then it computes a message m_{i+1} and an updated state st_{i+1} as

$$(m_{i+1}, st_{i+1}) \leftarrow \Pi.\mathsf{Run}(st_{i-1}, m_i).$$

4. When a party receives the $(n-1)^{\text{th}}$ message m_{n-1}, then it computes the n^{th} message and updated state st_n as above, and—using the updated state and secret long-term key sk—additionally outputs a session key as

$$k \leftarrow \Pi.\mathsf{KDF}(st_n, \mathsf{sk}).$$

5. Similarly, when a party receives the n^{th} message m_n, it uses this, together with its current state st_{i-1} and secret long-term key sk, to derive

$$k \leftarrow \Pi.\mathsf{KDF}'(st_{n-1}, \mathsf{sk}, m_n).$$

The messages must be independent of the secret key because we will construct an efficient adversary which has to send messages on behalf of other parties without knowing their secret keys. This includes typical implicitly authenticated protocols, such as protocols where each party sends a group element g^x for random $x \leftarrow \mathbb{Z}_p$ and the long-term secret keys are only used during key derivation at the end of the protocol. Many typical implicitly authenticated high-efficiency protocols have messages that are independent of the secret key. This includes in particular the implicitly authenticated variant of the CCGJJ19 protocol [10] but also protocols such as HMQV [20].

One class of protocols which is *not* covered by this assumption are those based on digital signatures, such as the signed Diffie-Hellman protocol. Digitally signing messages is thus a way to circumvent our impossibility result. However, note that implicitly authenticated key exchange protocols, such as [10,20], typically avoid the use of digital signatures since they add a very significant overhead (w.r.t. computation and communication) to the protocol. This holds in particular when tightness is considered, where the most efficient signature schemes with fully tight multi-user security (in the random oracle model) [14] are significantly more expensive than corresponding schemes without tight security.

An extension to NAXOS-like protocols [22], where parties send a group element g^x where $x = H(sk, r)$ depends on the secret key of the sending party and some randomness r, as well as to reductions in the random oracle model, is discussed in Section 6.3.

Π^+ *adds key confirmation messages with canonical verification to* Π. In the sequel let Π be an (implicitly authenticated) n-message protocol as defined above. We consider $(n+1)$-messages protocols Π^+ defined in terms of Π, and two associated functions $\Pi^+.\mathsf{Conf}$ and $\Pi^+.\mathsf{KDF}$, where $\Pi^+.\mathsf{Conf}$ computes the *key confirmation messages* added to the protocol and $\Pi^+.\mathsf{KDF}$ may perform additional key derivation. The first three steps of an execution of protocol Π^+ are identical to Π as described above. The remaining steps proceed as follows (cf. Figure 4).

4. When a party receives the $(n-1)^{\text{th}}$ message m_{n-1}, then it computes the n^{th} message of Π and updated state st_n as above, and derives the session key of Π as an *intermediate* key $k \leftarrow \Pi.\mathsf{KDF}(st_n, \mathsf{sk})$
   Then it computes a *key confirmation* message $m_{n+1} \leftarrow \Pi^+.\mathsf{Conf}(k, T_n)$, depending on k and the transcript $T_n = (m_1, \ldots, m_n)$ of all protocol messages as sent and received by this party so far. The n^{th} message of Π^+ now consists of the tuple (m_n, m_{n+1}), that is, the n^{th} message of Π plus the key confirmation message.
5. When a party receives the n^{th} message of Π^+ (m_n, m_{n+1}), then it derives an *intermediate* key as $k \leftarrow \Pi.\mathsf{KDF}'(st_{n-1}, \mathsf{sk}, m_n)$ using its current state st_{n-1}, long-term secret key sk, and message m_n.
   Then it checks the first key confirmation message by computing $m'_{n+1} \leftarrow \Pi^+.\mathsf{Conf}(k, T_n)$ and setting $\mathsf{accept} \leftarrow \mathsf{true}$ if and only if $m_{n+1} = m'_{n+1}$.
   If $\mathsf{accept} = \mathsf{false}$, then it outputs $k' = \perp$. If $\mathsf{accept} = \mathsf{true}$, then it sends a second *key confirmation* message $m_{n+2} \leftarrow \Pi^+.\mathsf{Conf}(k, T_{n+1})$ and outputs $k' \leftarrow \Pi^+.\mathsf{KDF}(k, T_{n+2})$, where $T_{n+2} = (m_1, \ldots, m_{n+2})$.
6. Finally, when a party receives the $(n+1)^{\text{th}}$ message of Π^+, i.e., the second key confirmation message, then it computes $m'_{n+2} \leftarrow \Pi^+.\mathsf{Conf}(k, T_{n+1})$ and sets $\mathsf{accept} = \mathsf{true}$ if and only if $m_{n+2} = m'_{n+2}$. If $\mathsf{accept} = \mathsf{false}$, then it outputs $k' = \perp$. If $\mathsf{accept} = \mathsf{true}$, then it outputs $k' \leftarrow \Pi^+.\mathsf{KDF}(k, T_{n+2})$.

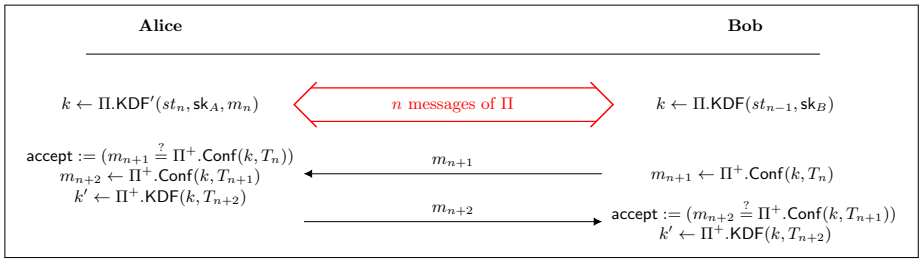

Fig. 4: Protocol Π^+ extending protocol Π with key-confirmation messages. In this concrete example Bob would have sent the last message m_n of Π.

Remark 2. We have defined Π^+ such that the first key confirmation message is sent along with the last message of Π, and then the second key confirmation

message is sent as a reply. This adds *one* extra message to Π, that is, Π^+ is an $(n+1)$-message protocol. Alternatively, we could have defined Π^+ so that the first key confirmation is sent as a *reply* to m_n. This would have added *two* messages to Π, making Π^+ an $(n+2)$-message protocol. We consider the former approach more natural, and is the approach used in CCGJJ19 [10] and HMQV-C [20]. The latter approach is used by Yang [27]. Even though we only treat the former variant here, our results apply equally to both variants.

Definition 14. *Let* $\Pi^+.\mathsf{Conf}$ *be such that key confirmation messages are elements of* $\{0,1\}^\beta$ *We say that* $\Pi^+.\mathsf{Conf}$ *is* δ-entropy-preserving*, if for every* $m \in \{0,1\}^\beta$ *and every string* T' *holds that*

$$\Pr_{k \xleftarrow{\$} \mathcal{K}}\left[\Pi^+.\mathsf{Conf}(k,T') = m\right] \leq \delta$$

Security of Π^+ *from* Π *via a valid black-box reduction.* We want to consider generic constructions of a fully forward secret (and explicitly authenticated) protocol Π^+ based on the assumption that the underlying Π is a weakly forward secret (and implicitly authenticated) protocol, plus possibly some additional assumptions, e.g., on the primitives used to create the key confirmation messages, and so on. This excludes artificial constructions of Π^+ which run Π as a redundant subroutine but where security is achieved in a completely different way, so that Π is actually superfluous.

The most natural way to establish security of Π^+ based on the security of Π is to have a security analysis of Π^+ which includes (possibly among other arguments and reductions) at least one reduction $\mathcal{R}$ to the KeySecWFS security of Π. We will argue that such a reduction cannot be tight. The full security analysis of Π^+ might include further arguments, such as reductions to the security of primitives used in the key confirmation messages.

More precisely, we assume that the security proof of Π^+ includes a black-box reduction $\mathcal{R}$, which treats the adversary $\mathcal{A}$ as a black box by submitting inputs and receiving outputs from $\mathcal{A}$ as specified in the explicitly authenticated security model, and which is able to leverage any successful $\mathcal{A}$ (independently of how $\mathcal{A}$ works internally) to break the KeySecWFS security of Π. Reduction $\mathcal{R}$ has access to the KeySecWFS security experiment of protocol Π, and to an adversary $\mathcal{A}$ on the KeySecFFS security of Π^+. We require that for every party i of the Π^+ experiment, there exists a unique corresponding party i' in the Π security experiment, and that $\mathcal{R}$ relays all messages of Π between the adversary and its security experiment. Hence, for every session $s_{i,j}$ of Π^+ there exists a unique session $s'_{i,j}$ of Π. The additional key confirmation messages of Π^+ are simulated by $\mathcal{R}$ (in any arbitrary way).

We also assume that the reduction is always "valid", i.e., it never makes any trivially invalid queries in its KeySecWFS security experiment. For example asking $\mathsf{Reveal}(s)$ and $\mathsf{Test}(s)$ against the same session s. We assume that a reduction rather aborts instead of making invalid queries. Obviously, any reduction $\mathcal{R}'$ that does not satisfy this can be generically transformed into a reduction $\mathcal{R}$ that does, with essentially the same running time and advantage, by simply putting

a wrapper around $\mathcal{R}'$ that relays all queries and their replies but terminates $\mathcal{R}'$ when it asks the first trivially invalid query.

Π *has unique and efficiently verifiable secret keys.* We assume that the public key pk of every party running protocol Π *uniquely* determines a matching secret key sk, and that one can efficiently and perfectly verify that a given sk matches a given pk. Note that this also holds for many typical implicitly authenticated high-efficiency protocols, in particular the CCGJJ19 protocol [10], but also HMQV [20], NAXOS [22], and many more, where a public key is a group element y of a group of order p and the matching secret key is the unique $x \in \mathbb{Z}_p$ such that $g^x = y$.

It is known that it is generally difficult to reduce the security of a protocol Π with unique secret keys tightly to the hardness of some non-interactive complexity assumption, due to the general impossibility results of Bader *et al.* [3]. However, note that our impossibility result is not about the tightness of security proofs for Π and reductions to non-interactive hardness assumptions, but rather about the tightness of reducing the security of a protocol Π^+ with key confirmation to the security of some underlying protocol Π (which then may or may not have a tight reduction to some hardness assumption). Hence, it is independent of the question of whether Π has a tight security proof under some (non-interactive) hardness assumption or not. Note also that in order to rule out tight generic constructions of explicit authentication via key confirmation, it is sufficient to rule out such constructions for protocols with unique secret keys, as a generic construction should work in particular for protocols with unique keys.

6.2 Impossibility result

Theorem 2. *Let Π be an AKE protocol and let Π^+ be an AKE protocol constructed by extending Π with δ-entropy-preserving key confirmation. Let $\mathcal{R}$ be a reduction which converts any adversary $\mathcal{A}$ against the* KeySecFFS *security of Π^+ into an adversary $\mathcal{R}(\mathcal{A})$ against the implicitly authenticated* KeySecWFS*-security of Π, such that Π, Π^+, and $\mathcal{R}$ satisfy all requirements described in Section 6.1. Let ϵ denote an upper bound on the advantage $\mathbf{Adv}^{\mathsf{KeySecWFS}}_{\Pi,U}(\mathcal{B})$ of any efficient adversary $\mathcal{B}$ in the* KeySecWFS *security experiment for Π with U parties.*

There exists a hypothetical adversary $\mathcal{A}$ with $\mathbf{Adv}^{\mathsf{KeySecFFS}}_{\Pi^+,U}(\mathcal{A}) = 1 - 1/2^{|\mathsf{key}|}$ such that for every reduction $\mathcal{R} = \mathcal{R}(\mathcal{A})$ holds that

$$\mathbf{Adv}^{\mathsf{KeySecWFS}}_{\Pi,U}(\mathcal{R}(\mathcal{A})) \leq \frac{2}{U} + 5 \cdot \epsilon + \frac{1}{2^{|\mathsf{key}|-1}} + \delta$$

Interpretation of Theorem 2. If Π is secure (otherwise a reduction to the security of Π is meaningless anyway), then ϵ is negligible. Furthermore, we can assume that $\frac{1}{2^{|\mathsf{key}|-1}}$ and δ are negligibly small, too, as otherwise there are trivial attacks on Π^+. Hence, we obtain that the advantage of any reduction $\mathcal{R}$ must be negligibly close to $O(1/U)$. However, note that the adversary $\mathcal{A}$ against Π^+ in the above theorem has advantage ≈ 1. Hence, a reduction $\mathcal{R}$ cannot be tight, as it must have a linear loss in the number of parties U.

Theorem 2 is formulated in terms of key secrecy, as it considers reductions leveraging an adversary $\mathcal{A}$ breaking the full forward secrecy (KeySecFFS) of Π^+. However, the impossibility result could equally have been phrased in terms of entity authentication. To see this, note that while the proof of Theorem 2 describes a meta-reduction which simulates a hypothetical adversary $\mathcal{A}$ breaking key secrecy, in Step 4 of this hypothetical adversary $\mathcal{A}$ also breaks entity authentication. Therefore one can equally phrase the theorem in terms of entity authentication, the overall proof and arguments all remaining the same. Consequently, Theorem 6 of [10], which is technically a claim about entity authentication and not about key secrecy, cannot be correct by Theorem 2. This was confirmed by the authors of [10].

Proof. As common in proofs based on meta-reductions [18,23,3] we first describe an (inefficient) hypothetical adversary. Then we explain how this hypothetical adversary is leveraged to prove the result, by showing that one can efficiently simulate $\mathcal{A}$ for any reduction $\mathcal{R}$ that is "too tight", which yields a contradiction.

Description of the hypothetical adversary. Consider the following (hypothetical) adversary $\mathcal{A}$ for the explicitly authenticated security experiment.

1. $\mathcal{A}$ receives a dictionary $\mathbf{pk}$ containing all the public keys of all users. It picks $i^* \xleftarrow{\$} \{1, \ldots, U\}$ at random.
2. $\mathcal{A}$ initiates $U(U-1)$ protocol sessions as follows. For every party i it picks $U-1$ random values $r_{i,j}$ for all $j \neq i$, that is, one for every party different from party i. Then it uses the fact that protocol messages of Π are independent of the secret long-term key to execute a run of protocol Π^+ such that the experiment outputs the *first* key confirmation message on behalf of j. $\mathcal{A}$ does not respond with the second key confirmation message.
   More precisely, if Π is an n-message protocol and n is even, then $\mathcal{A}$ impersonates every party i as an initiator by querying

   $$s_{i,j} \leftarrow \mathsf{NewSession}(j, i, \mathsf{resp})$$

   to create a session id $s_{i,j}$. Note that in our notation the session $s_{i,j}$ refers to a session where the adversary impersonates party i towards a session of party j with session id $s_{i,j}$ in the experiment. Then $\mathcal{A}$ uses $r_{i,j}$ to compute

   $$(m_{i,j,1}, st_{i,j,1}) \leftarrow \Pi.\mathsf{Init}(i, j, \mathsf{init}, \mathsf{pk}_j, -)$$

   and queries $\mathsf{Send}(s_{i,j}, m_{i,j,1})$ in order to send $m_{i,j,1}$ to the experiment on behalf of party i.

   If Π is a two-message protocol, then the experiment will respond with the second message of Π and the key confirmation message on behalf of j. If Π has more than two messages (i.e., $n \in \{4, 6, 8, \ldots\}$), then $\mathcal{A}$ continues to simulate all further messages of Π using $st_{i,j,1}$ on behalf of i by appropriate Send queries, until the experiment outputs the first key confirmation message.

If Π is an n-message protocol for odd n, then $\mathcal{A}$ proceeds similarly, except that instead of sending the first protocol message, it queries $(m_{i,j,1}, s_{i,j}) \leftarrow \mathsf{NewSession}(j, i, \mathsf{init})$ in order to receive the first protocol message $m_{i,j,1}$ from j to i and a corresponding session id $s_{i,j}$.

Thus, in total $\mathcal{A}$ obtains $U(U-1)$ key confirmation messages from its security experiment. We will later consider reductions $\mathcal{R}$ simulating the experiment of Π^+ by relaying all messages of Π to the security experiment of Π. But since the key confirmation messages exist only in Π^+ and not in Π, $\mathcal{R}$ is forced to somehow simulate the key confirmation messages. However, we will argue that these key confirmation messages are difficult to simulate properly for any $\mathcal{R}$ without predicting the index i^* chosen by $\mathcal{A}$ in its next step.

3. So far all queries of $\mathcal{A}$ were independent of i^*. Now $\mathcal{A}$ reveals the long-term keys of all users except for i^*, by querying $\mathsf{RevealLTK}(i)$ for all $i \in \{1, \ldots, U\} \setminus \{i^*\}$. The adversary aborts if anything is wrong. More precisely:
   (a) It checks whether all secret keys returned by the experiment match the public keys and aborts if not. Here we use that Π has unique and efficiently verifiable secret keys.
   Intuitively, this forces a reduction $\mathcal{R}$ simulating the experiment to relay all $\mathsf{RevealLTK}$ queries to the security experiment of Π. Otherwise, we can leverage $\mathcal{R}$ to break Π due to the fact that it is able to output a valid, *non-revealed*, long-term secret key. We prove this below.
   (b) For all $i \in \{1, \ldots, U\} \setminus \{i^*\}$ and all $j \neq i$, $\mathcal{A}$ uses its randomness $r_{i,j}$ and sk_i to compute all *intermediate* session keys $k_{i,j}$ (i.e., the session key of protocol Π of the session between i and j). To this end, it computes
   $$k_{i,j} \leftarrow \Pi.\mathsf{KDF}'(st_{i,j,n-1}, \mathsf{sk}_i, m_{i,j,n})$$
   and then uses $k_{i,j}$ and the transcript $T_{i,j,n}$ of the first n protocol messages of the session between i and j to test whether the key confirmation message produced by the security experiment in this session indeed match the correct key confirmation message determined by $k_{i,j}$ and the protocol transcript $T_{i,j,n}$. This is done by computing
   $$m'_{i,j,n+1} \leftarrow \Pi^+.\mathsf{Conf}(k_{i,j}, T_{i,j,n})$$
   and checking whether $m_{i,j,n+1} = m'_{i,j,n+1}$. $\mathcal{A}$ aborts if any key confirmation message is incorrect.
   Intuitively, this forces a reduction $\mathcal{R}$ simulating the security experiment to produce correct key confirmation messages for all sessions where $\mathcal{R}$ simulates party j. Below we'll argue that this is difficult for a reduction without predicting the index i^*, which incurs a linear security loss.
4. This last step is the "hypothetical" part of the adversary. $\mathcal{A}$ computes the unique value sk_{i^*} that corresponds to the secret key of party i^*. We intentionally do not specify precisely how this is done, as a black-box reduction should be able to leverage any adversary that somehow accomplishes this in some way. Then $\mathcal{A}$ finishes the key exchange protocol on behalf of i^* with *any* party j, for instance for $j = 1$.

To this end, it computes the *intermediate* key from Π as

$$k_{i^*,j} \leftarrow \Pi.\mathsf{KDF}'(st_{i^*,j,n-1}, \mathsf{sk}_{i^*}, m_{i^*,j,n})$$

and sets $\mathsf{accept} \leftarrow \mathsf{true}$ if and only if $m_{i^*,j,n+1} = \Pi^+.\mathsf{Conf}(k_{i^*,j}, T_{i^*,j,n+1})$, where $T_{i^*,j,n+1}$ is the transcript of all protocol messages of the session between i^* and j as observed by $\mathcal{A}$. If $\mathsf{accept} = \mathsf{false}$, then $\mathcal{A}$ aborts.
If $\mathsf{accept} = \mathsf{true}$, then $\mathcal{A}$ derives and sends the second key confirmation message of Π^+.[5] Since $s_{i^*,j}$ has not been revealed and no partner has been revealed, corrupted, or tested, it is eligible for a Test query.
Now $\mathcal{A}$ asks $\mathsf{Test}(s_{i^*,j})$ and receives back a key k', which is either the "real" session key or a random key. $\mathcal{A}$ then computes $k \leftarrow \Pi^+.\mathsf{KDF}(k, T_{i^*,j,n+2})$, where $T_{i^*,j,n+2}$ is the transcript of all protocol messages of the session between i^* and j as observed by $\mathcal{A}$. It outputs 1 if $k = k'$ and 0 otherwise.

Note that $\mathcal{A}$ is a correct (hypothetical) adversary against the explicitly authenticated KeySecFFS security of Π^+ with

$$\mathbf{Adv}^{\mathsf{KeySecFFS}}_{\Pi^+,U}(\mathcal{A}) = 1 - 1/2^{|\mathsf{key}|}$$

The term $1/2^{|\mathsf{key}|}$ is the probability that a random key k' equals the "real" session key k "by accident", which is the only case where $\mathcal{A}$ answers incorrectly. Note that this is the best possible advantage that an adversary that asks only a single Test query can achieve in the security experiment. Hence, any black-box reduction $\mathcal{R}$ that works for any correct adversary should work in particular for $\mathcal{A}$. As common in proofs based on meta-reductions, such as [11,18,23,3] our hypothetical adversary is not efficient but we will show how it can be efficiently *simulated* if the reduction $\mathcal{R}$ is tight. This yields that either the reduction must be non-tight, or the underlying hardness assumption (that Π is secure in an implicitly authenticated sense) must be wrong. Since we assume that Π is secure (as otherwise any reduction to the security of Π is meaningless and trivial, anyway), we conclude that $\mathcal{R}$ must be non-tight.

Analysis of $\mathcal{R}$. Consider the following sequence of games, where we denote with X_i the advantage of $\mathcal{R}(\mathcal{A})$ in the KeySecWFS security experiment of Π in Game i. Proofs for the lemmas can be found in the full version.

Game 0. As described in Section 6.1, we consider reductions $\mathcal{R}$ which have access to the KeySec security experiment of protocol Π, and to an adversary $\mathcal{A}$ on the *explicitly authenticated* KeySec security of Π^+. This game consists of an execution of any such reduction $\mathcal{R}$ with our hypothetical adversary. We have

$$X_0 = \mathbf{Adv}^{\mathsf{KeySecWFS}}_{\Pi,U}(\mathcal{R}(\mathcal{A}))$$

[5] This step already breaks entity authentication of Π^+ since the long-term key of $s_{i^*,j}$'s peer has not been revealed. However, since we focus on forward security here we let $\mathcal{A}$ continue.

Recall from Section 6.1 that for every party i of the Π^+ experiment, there exists a unique corresponding party i' in the Π security experiment. Recall also that $\mathcal{R}$ relays all messages of Π between the adversary and its security experiment, such that for every session $s_{i,j}$ of Π^+ there exists a unique session $s'_{i,j}$ of Π, and that we assume that $\mathcal{R}$ does not make any invalid queries in its security experiment that make $\mathcal{R}$ trivially "lose" the KeySec security experiment.

Game 1. Recall that according to Definition 1 the origin predicate evaluates to $\mathsf{Orig}(s, s') = \mathsf{true}$ for two sessions s, s', if the transcript of s' is equal to or a prefix of the transcript of s. Recall also that session key freshness (Definition 5) allows to reveal the long-term key of s's peer if there exists s' with $\mathsf{Orig}(s, s') = \mathsf{true}$.

Note that $\mathcal{A}$ impersonates one communication partner of every session created with $\mathcal{R}$ using independent randomness, and that $\mathcal{R}$ relays all protocol messages of Π between its own experiment and $\mathcal{A}$. Hence, intuitively it should be unlikely that there exist any two sessions s, s' of Π such that $\mathsf{Orig}(s, s') = \mathsf{true}$, and thus long-term key reveals of peers should not be allowed (as this would enable a trivial attack, where $\mathcal{A}$ sends a message on behalf of a party *and* it knows the corresponding secret key, such that it can trivially compute the session key and break the KeySec security). However, it might happen by coincidence, e.g., if $\mathcal{A}$ and the security experiment of Π happen to choose the same randomness.

The security of Π implies that the probability of this to happen is negligibly small. In Game 1 we ensure that indeed there are no two sessions s, s' such that $\mathsf{Orig}(s, s') = \mathsf{true}$ holds "by accident". Note that all sessions of Π are created in Step 2 of $\mathcal{A}$, therefore it suffices to consider the experiment until the end of Step 2 of $\mathcal{A}$, that is, before $\mathcal{A}$ asks the first RevealLTK query in Step 3.

Game 1 is identical to Game 0, except that we raise event WrongOrigin and abort, if at the end of Step 2 of the adversary there exist any two sessions s, s' such that $\mathsf{Orig}(s, s') = \mathsf{true}$. We have

$$|X_1 - X_0| \leq \Pr[\mathsf{WrongOrigin}]$$

Hence, after Game 1 we are guaranteed that there are no two sessions s, s' such that $\mathsf{Orig}(s, s') = \mathsf{true}$, and thus $\mathcal{R}$ is not allowed to reveal the long-term key of a test session's peer.

Lemma 3. *There exists an efficient adversary $\mathcal{B}_1$ against the key secrecy of Π with*

$$\mathbf{Adv}^{\mathsf{KeySecWFS}}_{\Pi,U}(\mathcal{B}_1) \geq \Pr[\mathsf{WrongOrigin}]$$

Game 2. This game is identical to Game 1, except that we abort if the event $\mathsf{NotAllRevLTK}(i)$ happens, where $\mathsf{NotAllRevLTK}(i)$ is the event that $\mathcal{R}$ has not asked $\mathsf{RevealLTK}(i)$ for all $i \neq i^*$ before $\mathcal{A}$ reaches Step 4. In other words, if $\mathsf{NotAllRevLTK}(i)$ happens, then when $\mathcal{A}$ reaches Step 4 there exists an index $i \neq i^*$ such that $\mathcal{R}$ has never queried $\mathsf{RevealLTK}(i)$. We have

$$|X_2 - X_1| \leq \Pr[\mathsf{NotAllRevLTK}(i)]$$

Hence, from Game 2 on we are guaranteed that every reduction $\mathcal{R}$ must satisfy all the following properties simultaneously throughout its execution:

– Throughout its execution, $\mathcal{R}$ eventually asks $\mathsf{RevealLTK}(i)$ for all $i \neq i^*$, as otherwise event $\mathsf{NotAllRevLTK}(i)$ occurs.
– $\mathcal{R}$ does *not* query $\mathsf{RevealLTK}(i^*)$, because this would mean that $\mathcal{R}$ learns *all* parties' long-term keys, which would make the reduction trivially invalid, since it could not issue a Test query to any session started by our $\mathcal{A}$.
– $\mathcal{R}$ does *not* query $\mathsf{Test}(s_{i,j})$ for any session $s_{i,j}$ with $i \neq i^*$, as for none of the sessions $s_{i,j}$ there exists an origin session and therefore this would violate session key freshness (Definition 5), and so the reduction would be trivially invalid.

Lemma 4. *There exists an efficient adversary $\mathcal{B}_2$ against the key secrecy of Π with*

$$\mathbf{Adv}^{\mathsf{KeySecWFS}}_{\Pi,U}(\mathcal{B}_2) \geq \Pr\left[\mathsf{NotAllRevLTK}(i)\right] - 1/2^{|\mathsf{key}|}.$$

Game 3. This game is identical to Game 2, except that we abort if the event $\mathsf{IncorrectConf}$ happens, where $\mathsf{IncorrectConf}$ is the event that $\mathcal{R}$ outputs at least one key confirmation message such that $\mathcal{A}$ aborts in Step 3. Note that $\mathsf{IncorrectConf}$ occurs, if there exists any session $s_{i,j}$ with $i \neq i^*$ where $\mathcal{R}$ outputs a key confirmation message $m_{i,j,n+1}$ such that

$$m_{i,j,n+1} \neq \Pi^+.\mathsf{Conf}(k_{i,j}, T_{i,j,n})$$

where $k_{i,j} \leftarrow \Pi.\mathsf{KDF}'(st_{i,j,n-1}, \mathsf{sk}_i, m_{i,j,n})$ and $st_{i,j,n-1}$ is the session state determined by sk_i and transcript $T_{i,j,n}$. We have

$$|X_3 - X_2| \leq \Pr\left[\mathsf{IncorrectConf}\right]$$

Hence, from Game 3 on we are guarenteed that $\mathcal{R}$ outputs correct key confirmation messages for all $s_{i,j}$ with $i \neq i^*$.

Lemma 5. *There exists an efficient adversary $\mathcal{B}_3$ against the key secrecy of Π with*

$$\mathbf{Adv}^{\mathsf{KeySecWFS}}_{\Pi,U}(\mathcal{B}_3) \geq \Pr\left[\mathsf{IncorrectConf}\right]$$

Observe that the decision whether $\mathcal{A}$ aborts is already made at the end of Step 2 of the adversary, where all queries of $\mathcal{A}$ are independent of i^*, and in particular before $\mathcal{A}$ asks any $\mathsf{RevealLTK}$ query in Step 3. Intuitively, this forces $\mathcal{R}$ to output correct key confirmation messages for all sessions $s_{i,j}$ with $i \neq i^*$.

Game 4. This game is identical to Game 3, except that we abort if $\mathcal{R}$ queries $\mathsf{Test}(s_{i^*,j})$ for any session of party j with i^*, where i^* is the index chosen by $\mathcal{A}$, before Step 2 of $\mathcal{A}$ ends, that is, before $\mathcal{A}$ makes the first $\mathsf{RevealLTK}$ query.

Recall that we have already established in Game 2 that $\mathcal{R}$ cannot make a $\mathsf{Test}(s_{i,j})$ query for any session $s_{i,j}$ with $i \neq i^*$ throughout the game. Hence, if $\mathcal{R}$ makes a $\mathsf{Test}(s_{i^*,j})$ query already in Step 2 of $\mathcal{A}$, then it correctly predicts the random i^*. Since all queries of $\mathcal{A}$ are independent of i^* until the beginning of Step 3, we have

$$|X_4 - X_3| \leq \frac{1}{U}$$

Game 5. Now we additionally abort if $\mathcal{R}$ queries $\mathsf{Reveal}(s_{i^*,j})$ for *all* sessions with party i^*, where i^* is the index chosen by $\mathcal{A}$, before Step 2 of $\mathcal{A}$ ends, that is, before $\mathcal{A}$ makes the first $\mathsf{RevealLTK}$ query in its Step 3. We claim that

$$|X_5 - X_4| = 0.$$

Note that none of the sessions created by $\mathcal{R}$ with its security experiment has an origin-session due to the construction of $\mathcal{A}$ and since we have ruled out the event $\mathsf{WrongOrigin}$ in Game 1. Recall also that $\mathcal{R}$ must eventually ask $\mathsf{RevealLTK}(i)$ for all $i \neq i^*$, as established in Game 2. Hence, $\mathcal{R}$ must not ask a $\mathsf{Test}(s_{i,j})$ query for any session $s_{i,j}$ with $i \neq i^*$, as this would violate the $\mathsf{aFresh}(s_{i,j})$ predicate of the $\mathsf{kFreshWFS}(s_{i,j})$ definition. However, if $\mathcal{R}$ would now also query $\mathsf{Reveal}(s_{i^*,j})$ for *all* sessions of party i^*, then $\mathcal{R}$ would not be allowed to query $\mathsf{Test}(s_{i,j})$ for any session $s_{i,j}$ with $i = i^*$, as this would violate the $\mathsf{kFreshWFS}(s_{i,j})$ definition.

In this case, no fresh session would remain for which $\mathcal{R}$ could make a Test query. Hence, this would be an invalid sequence of queries, and since we assume that $\mathcal{R}$ does not make any such invalid queries this cannot happen.

Game 6. This game is identical to Game 5, except that we abort if the event $\mathsf{AllConfCorrect}$ happens, where $\mathsf{AllConfCorrect}$ occurs if $\mathcal{R}$ outputs *all* key confirmation messages correctly in Step 2. Specifically, $\mathsf{AllConfCorrect}$ occurs if

$$m_{i,j,n+1} = \Pi^+.\mathsf{Conf}(k_{i,j}, T_{i,j,n})$$

holds for *all* sessions $s_{i,j}$, and thus for all sessions $s_{i^*,j}$ for some j. We have

$$|X_6 - X_5| \leq \Pr\left[\mathsf{AllConfCorrect}\right]$$

Hence, from Game 6 on we are guarenteed that $\mathcal{R}$ outputs correct key confirmation messages for all $s_{i,j}$ with $i \neq i^*$ as otherwise we abort due to Game 5, but there must be at least one incorrect key confirmation message for a session $s_{i^*,j}$, as otherwise we abort due to Game 6.

Lemma 6. *There exists an efficient adversary $\mathcal{B}_6$ against the key secrecy of Π with*

$$\mathbf{Adv}^{\mathsf{KeySecWFS}}_{\Pi,U}(\mathcal{B}_6) \geq \frac{\Pr\left[\mathsf{AllConfCorrect}\right]}{2} - \frac{1}{2^{|\mathsf{key}|}} - \delta \tag{34}$$

Analysis of Game 6. Finally, we claim that

$$X_6 \leq \frac{1}{U}.$$

In order to see this, consider the state of $\mathcal{R}$ in Game 6 immediately after it has output all key confirmation messages, that is, after Step 2 and *before* Step 3 of $\mathcal{A}$. We have established that $\mathcal{R}$ must output correct key confirmation messages for all sessions $s_{i,j}$ with $i \neq i^*$, as otherwise event $\mathsf{IncorrectConf}$ occurs. However, it also must output at least one incorrect key confirmation messages in Step 2 of $\mathcal{A}$, as otherwise event $\mathsf{AllConfCorrect}$ occurs. However, before Step 3, all queries made by $\mathcal{A}$ are independent of i^*, so essentially $\mathcal{R}$ has to "predict" the uniform choice of $i^* \xleftarrow{\$} \{1, \ldots, U\}$, which happens with probability at most $1/U$. □

6.3 Discussion of extensions and generalizations

Extension to NAXOS-like protocols. One limitation of the impossibility result is that it requires that all protocol messages are independent of the long-term secret key. As already discussed, this holds for many implictly-authenticated, in particular those aiming at maximal efficiency. However, one interesting class of protocols that are unfortunately excluded are NAXOS-like protocols [22], where parties send a group element g^x where $x = H(sk, r)$ depends on the secret key of the sending party and some randomness r.

We expect that Theorem 2 can also be generalized to such protocols, though. Recall that our hypothetical adversary $\mathcal{A}$ establishes protocol sessions between all parties in its Step 2. However, since it did not yet reveal any parties' long-term keys at this step, it cannot compute $x = H(sk, r)$ for the real secret key sk of a party. Observe, though that in such NAXOS-like protocols a party receiving g^x is not able to verify that $x = H(sk, r)$, because the receiving party also doesn't know sk (and also not r). Even though a reduction $\mathcal{R}$ might somehow be able to check consistency of x (e.g., using the random oracle queries made by $\mathcal{A}$), it would still have to continue the key exchange protocol like the real security experiment. Hence, $\mathcal{A}$ could simply pick x at random and send g^x to $\mathcal{R}$, and $\mathcal{R}$ would have to continue the protocol and send a key confirmation message at the end, which leads to a similar security loss as in the proof of Theorem 2, with almost exactly the same argument that it essentially requires $\mathcal{R}$ to "predict" the index i^* chosen by $\mathcal{A}$ already in Step 2 of $\mathcal{A}$.

The reason why we did not consider this extension is because it seems that we would either have to consider specific protocols concretely, such as specifically NAXOS, which would reduce the generality of the result, or alternatively we would have had to define a general notion of "efficient simulatability" of secret-key-dependent protocol messages. We refrained from the former to obtain a general impossibility result which explains the core reason of the inherent security loss of standard ways to do key confirmation, and from the latter because the argument in the proof of Theorem 2 is already relatively complex due to the inherent complexity of key exchange security models, and we preferred an as-clean-as-possible and more rigorous argument over full generality.

Extension to key confirmation in the random oracle model. Note that the argument in the proof of Theorem 2, specifically the construction of adversary $\mathcal{B}_6$ in Lemma 6, uses the key confirmation m_{n+1} as a "test value" to check whether a given challenge key k is real or random. This exploits that the message m_{n+1} produced by $\mathcal{R}$ in the tested session is computed deterministically as

$$m_{n+1} = \Pi^+.\mathsf{Conf}(k', T_n)$$

from the real session key k' and the public transcript T_n of protocol messages. Note that this accurately models the approach to do key confirmation used and incorrectly claimed to be tightly secure in [10]. It also covers the new approach described in the present paper, which additionally leverages selective security, since both are in the standard-model, that is, without random oracles.

Given that most highly-efficient implicitly authenticated protocols are proven secure in the random oracle model, one might ask whether it is possible to give a tightly-secure construction of Π^+ from Π in the random oracle model. For instance, one could consider computing the key confirmation message as

$$m_{n+1} = H(k', T_n)$$

If H is modeled as a random oracle, then this could enable a reduction to avoid the "commitment" implied by the key confirmation messages that it has to simulate properly, by just sending random strings m_{n+1} that then might later be "explained" as proper hash values by the $\mathcal{R}$, if necessary.

We expect that this approach also fails and once again we have an inherent tightness loss. The reason for this is because the reduction would also have to simulate the random oracle H consistently. But in order to achieve this, $\mathcal{R}$ would have to be able to distinguish a random oracle query $H(k', T_n)$ using the "real" key (in which case it would have to return m_{n+1}) from a query $H(k'', T_n)$ using an independent string k''. Since k' is the session key of Π, the reduction would thus have to be able to distinguish session keys of Π from random, which should give rise to another attacker $\mathcal{B}$ on Π that proceeds as follows:

1. $\mathcal{B}$ is again a meta-reduction, which runs $\mathcal{R}$ as a subroutine, relays all queries between $\mathcal{R}$ and its security experiment, and simulates our hypothetical adversary $\mathcal{A}$ until the end of Step 2.
2. Then $\mathcal{B}$ picks an arbitrary random session $s_{i,j}$ and queries $\mathsf{Test}(s_{i,j})$ to the security experiment of Π, receving back a challenge key k.
3. Now $\mathcal{B}$ issues many random oracle queries of the form $H(k_i, T_n)$ to $\mathcal{R}$, where $k_1, ..., k_Q$ are chosen at random, but $k_\ell := k$ is defined as the challenge key k for some random index $\ell \xleftarrow{\$} \{1, \ldots, Q\}$.
4. Now either $\mathcal{R}$ is able to distinguish the query with a "real" key k from a "random" one. In this case, $\mathcal{B}$ can also distinguish the real key from a random one, by checking whether $m_{n+1} = H(k_\ell, T_n)$.
   Or $\mathcal{R}$ is not able to distinguish the query with a "real" key k from a "random" one. In this case, $\mathcal{R}$ will fail with probability $1 - 1/Q$, so that once again we can simulate $\mathcal{A}$ efficiently because it aborts if $\mathcal{R}$ fails.

The above proof idea is only a sketch, and we expect a rigorous proof to be significantly more complex and subtle. Therefore we chose to focus on the simpler and cleaner case of ruling out tight standard model constructions, as this is also what was claimed in [10] and is achieved in the present 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 (Apr 2001). https://doi.org/10.1007/3-540-45353-9_12 4
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 (Mar 2015). https://doi.org/10.1007/978-3-662-46494-6_26 2
3. 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). https://doi.org/10.1007/978-3-662-49896-5_10 6, 22, 23, 25
4. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (May 2000). https://doi.org/10.1007/3-540-45539-6_11 2, 3, 6, 9
5. 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). https://doi.org/10.1007/3-540-48329-2_21 3, 6
6. Bellare, M., Rogaway, P.: Provably secure session key distribution: The three party case. In: 27th ACM STOC. pp. 57–66. ACM Press (May / Jun 1995). https://doi.org/10.1145/225058.225084 6
7. Boyd, C., Gellert, K.: A Modern View on Forward Security. The Computer Journal **64**(4), 639–652 (08 2020). https://doi.org/10.1093/comjnl/bxaa104, https://doi.org/10.1093/comjnl/bxaa104 2
8. Boyd, C., González Nieto, J.M.: On forward secrecy in one-round key exchange. In: Chen, L. (ed.) 13th IMA International Conference on Cryptography and Coding. LNCS, vol. 7089, pp. 451–468. Springer, Heidelberg (Dec 2011) 2, 3
9. 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). https://doi.org/10.1007/3-540-44987-6_28 6
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, Heidelberg (Aug 2019). https://doi.org/10.1007/978-3-030-26954-8_25 2, 3, 4, 5, 6, 14, 17, 18, 19, 21, 22, 23, 29, 30
11. 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). https://doi.org/10.1007/3-540-46035-7_18 25
12. Cremers, C., Feltz, M.: Beyond eCK: perfect forward secrecy under actor compromise and ephemeral-key reveal. Des. Codes Cryptogr. **74**(1), 183–218 (2015), https://doi.org/10.1007/s10623-013-9852-1 3
13. de Saint Guilhem, C., Fischlin, M., Warinschi, B.: Authentication in key-exchange: Definitions, relations and composition. In: Jia, L., Küsters, R. (eds.) CSF 2020 Computer Security Foundations Symposium. pp. 288–303. IEEE Computer Society Press (2020). https://doi.org/10.1109/CSF49147.2020.00028 2, 6, 9, 10, 11

14. Diemert, D., Gellert, K., Jager, T., Lyu, L.: More efficient digital signatures with tight multi-user security. In: Garay, J. (ed.) PKC 2021, Part II. LNCS, vol. 12711, pp. 1–31. Springer, Heidelberg (May 2021). https://doi.org/10.1007/978-3-030-75248-4_1 19
15. 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). https://doi.org/10.1109/SP.2016.34 2
16. 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). https://doi.org/10.1007/978-3-319-96881-0_4 2
17. Han, S., Jager, T., Kiltz, E., Liu, S., Pan, J., Riepel, D., Schäge, S.: Authenticated key exchange and signatures with tight security in the standard model. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part IV. LNCS, vol. 12828, pp. 670–700. Springer, Heidelberg, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84259-8_23 2
18. 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). https://doi.org/10.1007/978-3-642-30057-8_5 6, 23, 25
19. Jager, T., Kiltz, E., Riepel, D., Schäge, S.: Tightly-secure authenticated key exchange, revisited. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part I. LNCS, vol. 12696, pp. 117–146. Springer, Heidelberg (Oct 2021). https://doi.org/10.1007/978-3-030-77870-5_5 2
20. 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). https://doi.org/10.1007/11535218_33 2, 3, 18, 19, 21, 22
21. 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). https://doi.org/10.1007/978-3-642-40041-4_24 4, 12
22. 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) 6, 19, 22, 29
23. 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). https://doi.org/10.1007/978-3-642-55220-5_4 6, 23, 25
24. 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). https://doi.org/10.1145/3133956.3134006 9
25. Pan, J., Qian, C., Ringerud, M.: Signed diffie-hellman key exchange with tight security. In: Paterson, K.G. (ed.) CT-RSA 2021. LNCS, vol. 12704, pp. 201–226. Springer, Heidelberg (May 2021). https://doi.org/10.1007/978-3-030-75539-3_9 2
26. Rogaway, P., Zhang, Y.: Simplifying game-based definitions - indistinguishability up to correctness and its application to stateful AE. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 3–32. Springer, Heidelberg (Aug 2018). https://doi.org/10.1007/978-3-319-96881-0_1 12

27. Yang, Z.: Modelling simultaneous mutual authentication for authenticated key exchange. In: FPS. Lecture Notes in Computer Science, vol. 8352, pp. 46–62. Springer (2013) 2, 18, 21