Privacy-Preserving Authenticated Key Exchange: Stronger Privacy and Generic Constructions

Sebastian Ramacher, Daniel Slamanig(✉), and Andreas Weninger

AIT Austrian Institute of Technology, Vienna, Austria
{sebastian.ramacher,daniel.slamanig,andreas.weninger}@ait.ac.at

Abstract. Authenticated key-exchange (AKE) protocols are an important class of protocols that allow two parties to establish a common session key over an insecure channel such as the Internet to then protect their communication. They are widely deployed in security protocols such as TLS, IPsec and SSH. Besides the confidentiality of the communicated data, an orthogonal but increasingly important goal is the protection of the confidentiality of the identities of the involved parties (aka privacy). For instance, the Encrypted Client Hello (ECH) mechanism for TLS 1.3 has been designed for exactly this reason. Recently, a series of works (Zhao CCS'16, Arfaoui et al. PoPETS'19, Schäge et al. PKC'20) studied privacy guarantees of (existing) AKE protocols by integrating privacy into AKE models. We observe that these so called privacy-preserving AKE (PPAKE) models are typically strongly tailored to the specific setting, i.e., concrete protocols they investigate. Moreover, the privacy guarantees in these models might be too weak (or even are non-existent) when facing active adversaries.

In this work we set the goal to provide a single PPAKE model that captures privacy guarantees against different types of attacks, thereby covering previously proposed notions as well as so far not achieved privacy guarantees. In doing so, we obtain different "degrees" of privacy within a single model, which, in its strongest forms also capture privacy guarantees against powerful active adversaries. We then proceed to investigate (generic) constructions of AKE protocols that provide strong privacy guarantees in our PPAKE model. This includes classical Diffie-Hellman type protocols as well as protocols based on generic building blocks, thus covering post-quantum instantiations.

1 Introduction

Authenticated key exchange (AKE) protocols are among the most important cryptographic building blocks to enable secure communication over insecure networks. Essentially, an AKE allows two parties A and B, in possession of long term key pairs $(\mathsf{pk}_A, \mathsf{sk}_A)$ and $(\mathsf{pk}_B, \mathsf{sk}_B)$ respectively, to authenticate each other and securely establish a common session key. Security should thereby even hold in the presence of active attackers, which may intercept, read, alter, replay, or

E. Bertino et al. (Eds.): ESORICS 2021, LNCS 12973, pp. 676–696, 2021.
https://doi.org/10.1007/978-3-030-88428-4_33

drop any message transmitted between these parties. Moreover, in state-of-the-art protocols one requires security of the session key (i.e., confidentiality) of past interactions of A and B, even if attackers are able to compromise the long term secrets sk_A and sk_B. This is typically denoted as perfect forward secrecy (PFS). In current real world applications, such AKE protocols typically rely on the Diffie-Hellman (DH) protocol and digital signatures and are widely deployed in security protocols such as TLS, IPsec and SSH. The emerging threat of the feasibility of powerful quantum computers additionally revived the interest in AKE protocols that do not rely on DH key exchange, but instead are based generically on public key encryption (PKE) or key encapsulation mechanisms (KEMs) [16,18,31].

Privacy in AKE. While confidentiality of communicated data is the prime target for a security protocol, another important property is the confidentiality of the identities of the parties involved in the AKE. We will call this goal of hiding the identities from external parties privacy.[1] Schäge et al. [30] recently coined the term privacy-preserving authenticated key exchange (PPAKE) for AKE protocols with such privacy guarantees. While the study of PPAKE is an interesting subject on its own right, we currently can observe an increasing interest in such features in real world protocols. For instance, TLS 1.3 [27] aims to protect the identities of the server and client by encrypting messages as soon as possible during the authentication and in particular hiding the certificate sent by the server. Besides, many other protocols such as QUIC, IPsec IKE, SSH and certain patterns of the Noise protocol framework [26] aim to protect identity-related information such as identities, public keys or digital signatures. This is usually done by running an anonymous DH handshake where the derived keying material is then used to encrypt all subsequent messages (essentially the SIGMA-R template [21]). Moreover, the recent proposal of Encrypted Client Hello (ECH) mechanism for TLS encrypts the initial client message (the ClientHello) [28] with the aim of hiding the target domain for a given connection from attackers listening on the network. We also want to note that over the years various protocols have been designed to provide some intuitive identity protection measures, such as SKEME [20] or the SIGMA-I and SIGMA-R variants of the SIGMA protocol family [21]. The work of Schäge et al. [30], for instance, formally analyzes the privacy guarantees of SIGMA-R as used in IKEv2 within IPSec.

Relevance of PPAKE in Practice. From the above mentioned protocols that try to conceal identifying information, in particular encrypted Server Name Indication (ESNI) and its successor ECH have demonstrated its usefulness in practice. Especially when considering network censorship, ESNI/ECH can help to thwart censorship [7]. Consequently, all ESNI protected TLS connections have been blocked in China.[2] In general, one can observe a push towards an Internet

[1] We note that key-exchange protocols that hide the identity of one party even from the peer in the key exchange (e.g., as in [13,24]) are outside the scope of this work.

[2] https://www.zdnet.com/article/china-is-now-blocking-all-encrypted-https-traffic-using-tls-1-3-and-esni/.

infrastructure that reduces the amount of identifiable information. DNS over HTTPS/TLS [15,17] for instance helps in hiding identifying information associated to a connection from an adversary listening to public network traffic.

While in the above cases typically only one party, i.e., the server, is authenticated, with the Internet-of-Things (IoT) [14,29] or FIDO2 [3,12] we see an adoption of mutually authenticated AKE protocols and interest towards identity privacy. For instance, Wu et al. [33] study protocols for private service discovery and private mutual authentication in both the IoT and the mobile landscape (with a case study on Apple AirDrop). Similarly, many VPN implementations also offer the ability to configure certificate-based client authentications during the initial handshake which is also the only option in WireGuard [10,11] to establish connections.

Previous Work on PPAKE. To the best of our knowledge, the first work that specifically addresses privacy in key agreement is by Aiello et al. [1]. Informally, their privacy property wants to achieve that protocols must not reveal the identity of a participant to any unauthorized party, including an active attacker that attempts to act as the peer. They concretely propose two protocols, where one protects the identity of the initiator from an active attacker and the second one that of the responder. However, we note that this privacy property is neither modeled nor rigorously analyzed. Another informal discussion of how to achieve "identity concealment" by encrypting the identities was even earlier mentioned by Canetti and Krawczyk in [6]. Later Zhao in [34] introduced the notion of identity-concealed authenticated key exchange (CAKE), which enforces the notion of forward identity-privacy (which we simply call forward privacy) and some form of man-in-the-middle (MITM) privacy for completed sessions.

Recently, Schäge et al. [30] provided a PPAKE model, which similarly to Zhao [34] incorporates forward privacy and some form of MITM privacy for completed sessions, but considers a different setting. In their model, the identity of any two communicating parties are known (so it is visible who communicates with whom), but each party has two additional identities associated to it and it should be hard to figure out which identities the parties are using. Consequently, this model is tailored to a specific setting, e.g., where one server hosts multiple virtual machines or services and these identities need to be protected. Schäge et al. then use their model to analyze the privacy of the IKEv2 protocol [19]. Also recently Arfaoui et al. [2] investigate privacy in TLS 1.3 including session resumption. They capture a weaker notion of privacy than what is required by forward privacy, as they do not allow any corruptions. Their model also only considers uni-lateral authentication and models privacy as a separate property using the concept of a virtual identifier known from privacy analysis of RFID protocols (cf. [30] for a discussion why this is not desirable). Interestingly, none of the previously proposed formal models (including [34]) considers strong active adversaries against the privacy of the AKE protocols. To be more precise, while they actually allow active attacks, they only allow the adversaries to attack accepted sessions. And for any reasonable AKE, this essentially boils down to passive attacks (we will discuss this in more detail in Sect. 2).

Our Contribution. Subsequently, we briefly summarize our contributions:

- We revisit privacy in context of AKE and introduce a comprehensive PPAKE model building upon and extending the recent AKE model in [8]. It is more general than the recent PPAKE by Schäge et al. [30] and among variants of privacy notions known from previous works [30,34] supports stronger notions against active adversaries.
- The main contribution of this work is that we deal with *incomplete session attacks*, i.e., active MITM adversaries that learn the identity of one party but are unable to then complete the protocol run. This is typically due to the inability to authenticate themselves, which is caused by a lack of secret key material. The models and protocols of Schäge et al. [30] and Zhao [34], as noted by the authors, do not prevent such attacks. In each case the adversary can create the first message(s) of either the initiator or the responder without having access to the user's long-term secret key. This is due to the fact that the first messages only serve the purpose of exchanging ephemeral randomness, e.g., via an anonymous DH key exchange. Then the other side will authenticate itself, allowing the adversary to trivially learn the identity. We stress that this attack can be done by any MITM adversary without corrupting any user.
- We present generic constructions of PPAKE protocols with strong privacy guarantees. Our constructions rely on standard primitives such as public-key encryption or key-encapsulation mechanisms, signature schemes and unauthenticated two-move key exchange protocols. Thus, our constructions can be instantiated with post-quantum secure building blocks. In contrast, previous works exclusively focused on DH based protocols.

2 On Modeling Privacy in AKE

There are different privacy properties that are considered to be relevant, some of which that can and others that cannot be covered within PPAKE. In this section we discuss these issues, highlight aspects that have not been considered so far in PPAKE models and present a comprehensive overview of the different privacy properties and their relations.

2.1 What Can(not) Be Handled by PPAKE

Identity-related information such as client specific identifiers, public keys (certificates in particular) and digital signatures can be used by an adversary to break privacy. All these information are available on the layer of the AKE protocol, but there are clearly other network dependent information outside the AKE layer and our model, e.g., network addresses such as IP or MAC addresses, that allow adversaries to break privacy. Consequently, as discussed in [30] for PPAKE, the assumptions on the network are stronger than those required by network anonymization protocols like Tor [9]. Latter implement an overlay network and provide privacy against an adversary who controls large parts of the underlying network (i.e., the Internet) but not the complete network, as well as parts of the overlay network (e.g., Tor) itself.

PPAKE considers an adversary that is weaker and in particular assumes an active MITM attacker that controls a large, but well-defined part of the network. Consequently, one omits the consideration of network identifiers like IP or MAC addresses in PPAKE. This firstly allows to make the model simpler and independent of any network technology and topology. Secondly, as argued in [30], by using trustworthy proxies at the entry points of the adversary controlled network the usefulness of these information to an adversary can be significantly reduced. Nevertheless, we argue that even in case of absence of such proxies PPAKE still provides a meaningful countermeasure to large scale privacy attacks. In particular, it is easily possible to record identity-related information such as certificates (which can simply be parsed locally) on the AKE layer. Consequently, compared to basing the analysis on network address information, which might be additionally complicated by Network Address Translation (NAT), this is much more efficient and easily leads to a unique identification of the entities.

While it is clear that fully hiding all identity information is not possible in practice, privacy can only be lost. Consequently, guaranteeing an adequate level of privacy via PPAKE is a first step to reduce privacy risks.

2.2 Privacy Goals in PPAKE

Now we are going to discuss privacy goals relevant to PPAKE and distill a set of privacy properties from that. Unlike previous works [2,30,34], which basically design PPAKE models in a way that they allow to analyze a specific AKE protocol (family) such as used in TLS 1.3 or IKEv2, in this work we ask what are desirable properties and how to design PPAKE protocols providing strong privacy guarantees. In doing so we do not consider a single privacy notion (as done in previous work), but propose a set of privacy notions that allow to cover properties relevant to diverse use-cases.

Roughly, we can classify privacy attacks in either *passive* or *active* attacks and whether we either consider only *completed sessions* or we allow even *incomplete sessions* to be the target of an attack. Thereby, a passive adversary only behaves passive during the session establishment but can corrupt parties after the session-establishment. Note that for incomplete sessions, a purely passive adversary is not reasonable and is thus not considered. Active adversaries and incomplete sessions are however reasonable, i.e., actively trying to identify peers that are establishing a session which might already provide a sufficient amount of compromising information. Nevertheless, such notions have not been considered in previous models. See Table 1 for an overview.

Passive Adversaries. We start with a property that is implicitly covered by the privacy notion in previous PPAKE [30,34]. We call it forward privacy and it can be seen as the privacy analogue of forward secrecy. Namely, it requires that for any completed session even if an adversary can later on corrupt the long term secrets of all parties, the identities of the actual parties that were involved in the session are not revealed.

Table 1. Type of adversary $\mathcal{A}$ and state of the attacked session. ($\times$) denotes no corruption; (✓) denotes corruption of all but the users in the target session (i.e., the session to be attacked); (✓✓) denotes corruption of *all* users. Corruption always refers to the long-term secrets.

	Completed session	Incomplete session
Passive $\mathcal{A}$	Forward privacy (✓✓)	—
Active $\mathcal{A}$	Completed-session privacy (✓)	(Weak) 2-way MITM privacy ($\times$) Strong 2-way MITM privacy (✓)

For instance, the signed DH protocol does not provide any privacy, but one could imagine to add public-key encryption (PKE), i.e., party A sends g^x in plain but the value $\mathsf{Sig}_{\mathsf{sk}_A}(g^x \| id_B)$ is encrypted under the public key of B and vice versa (this pattern is similar to what is done to achieve identity protection in SKEME [20]). This will conceal the identities from any eavesdropper as long as no corruptions happen. If, however, the long-term secret keys of A and B corresponding to their PKE public keys are leaked, their identities are clearly revealed from a recorded transcript. The same holds for other protocols such as KEA or KEA+ [22] when in addition all messages are encrypted with a PKE.

Note that with such a fix (that unfortunately does not give forward privacy), the initiator, besides needing to know the responders identity, would also needs to know its public key. However, we want to stress that this is a quite reasonable assumption as in many scenarios the public keys can already be deployed on the devices or can be fetched from key repositories. Clearly, in the latter case it is not advisable to do this immediately before running the AKE as this yields another channel that leaks privacy relevant information. But in many real world settings, e.g., Encrypted Client Hello (ECH) in TLS 1.3, responder's public keys are assumed to be fetched out-of-band.

Active Adversaries. First, we note that several works [1,2,21] state that active adversaries against privacy are hard to handle:

"...it is not possible for a protocol to protect both the initiator and the responder against an active attacker; one of the participants must always go first." [1]

However, this statement seems to implicitly assume that the parties do not know public keys of the other parties beforehand or have no means to detect whether the public keys are revoked. Recent PPAKE models [30,34] indeed achieve privacy against active adversaries, though in only a limited setting. In particular, they consider active man-in-the-middle (MITM) adversaries but restrict them to completed sessions and thus requiring the involved entities have not been corrupted, i.e., the respective long term secret keys are not compromised/revoked.

To illustrate this, we consider a template analyzed in [30] representing a variant of the SIGMA protocol family [21] covering protocols such as TLS 1.3, QUIC, IPsec IKE or SSH. In particular, the SIGMA-R protocol that is designed to provide receiver identity protection is investigated. This template uses an

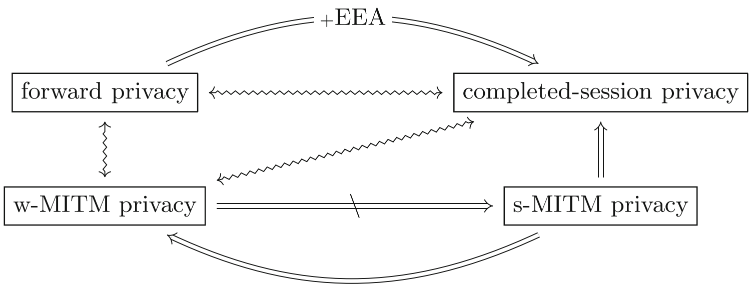


Fig. 1. Overview of implications and separations between privacy notions. $\leftrightsquigarrow$ denotes two incomparable properties and EEA denotes explicit entity authentication.

anonymous DH key exchange, i.e., party A sends g^x for ephemeral x and party B responds with g^y for ephemeral y. Subsequently, parties authenticate using digital signatures, where these authentication messages are encrypted using a symmetric key derived from the shared secret g^{xy}. This protocol can provide privacy against active MITM attackers, but only if the session completes (requiring that the involved entities are not corrupted). So this only can happen "after the fact". Nevertheless, it is easy to see that for incomplete sessions there is no privacy guarantee as the initiator "goes first" and thus anyone can identify the initiator. Schäge et al. in [30] explicitly discuss this limitation of their model which allows to always reveal the server identity in TLS or QUIC or the client identity in IPsec IKE and mention that *"It is therefore conceivable to formalize a stronger property for the secrecy of identities selected by the responder which does not rely on session acceptance."* Indeed, this is a setting we want to cover with our privacy notion. Consequently, we will formalize adequate properties for privacy against active adversaries even if sessions are not completed.

Summarizing, such a stronger notion cannot work in the PPAKE model by Schäge et al. in [30]. Also the model and the protocols by Zhao [34] do not consider adversaries that do not need to know the long-term secret key to perform the attack (and thus only consider completed sessions). Note there are simple attack strategies against these protocols that do not require the attacker to obtain any long-term keys or otherwise compromise any party and can be performed by anyone. But we stress that such attacks are outside the model of [34].

Previous PPAKE models only achieve the notion of active MITM attacks against completed sessions (which implicitly covers forward privacy), but not against incomplete sessions. In order to also capture such attacks, we introduce the notion of MITM privacy in two flavors. The first and easier to achieve variant allows adversaries to also attack incomplete sessions but require that no user corruption happens. The second and stronger notion removes this requirement and also allows corruption of users (clearly with exception of the attacked ones). Looking ahead, to achieve MITM privacy requires that even in case of failure protocol messages that look like real protocol messages needs to be send.

Whether this notion is meaningful consequently depends on the context of the use of the protocol and might not be meaningful if used within some higher level protocols where the required behavior cannot be realized.

In Fig. 1 we provide an overview of the privacy notions captured in this paper and how they relate to each other (cf. Sect. 3.2 for a formal treatment). We note that completed-session privacy essentially reflects the privacy notions proposed by Zhao [34] as well as Schäge et al. [30].

Initiator and Responder Privacy. Another aspect, which typically depends on the structure of the protocol as well as the application, is whether privacy only holds for either the initiator or the responder or both of them. For instance, in the most common TLS application scenario clients do not authenticate and thus, unless client authentication is used, only responder privacy is important. Schäge et al. in [30] model privacy in a way that the adversary can explicitly trigger (via a bit) whether to attack the initiator or the responder. In our model, we also consider both aspects simultaneously (which we denote as 2-way privacy), but the adversary controls whom to attack by means of how it engages with the respective oracles. We discuss how to restrict the adversary in our model to model either initiator or responder privacy in the next section.

3 Our PPAKE Model

3.1 Security Model

Our formal security model builds upon the model in [8] which we extend to cover privacy features. Like [8], our model accounts for key impersonation (KCI) security and weak forward secrecy and we use their notion of origin-oracle partnering. We note that [8] avoid no-match attacks [23] as their concrete protocol's messages only contain group elements and deterministic functions of them. We consider the generic countermeasure from [23] by including all exchanged messages (the context) in the final key derivation.

Execution Environment. We consider μ parties $1, \ldots, \mu$. Each party P_i is represented by a set of oracles, $\{\pi_i^1, \ldots, \pi_i^\ell\}$, where each oracle corresponds to a session, i.e., a single execution of a protocol role, and where $\ell \in \mathbb{N}$ is the maximum number of protocol sessions per party. Each oracle π_i^s is equipped with a randomness tape r_i^s containing random bits, but is otherwise deterministic. Each oracle π_i^s has access to the long-term key pair $(\mathsf{sk}_i, \mathsf{pk}_i)$ of party P_i[3] and to the public keys of all other parties, and maintains a list of internal state variables that are described in the following:

– $\mathsf{Pid}_\mathsf{i}^\mathsf{s}$ ("peer id") stores the identity of the intended communication partner. We assume the initiator of a protocol to know who she contacts, hence for the initiator this value is set immediately. Due to the nature of PPAKE the responder might not immediately know the identity of the initiator, hence for the responder this value is initialized to $\perp$ and only set once he receives a message containing the initiator's identity.

[3] This might contain various private and public keys for signatures and encryption.

- $\Psi_i^s \in \{\emptyset, \mathsf{Accept}, \mathsf{Reject}\}$ indicates whether π_i^s has successfully completed the protocol execution and "accepted" the resulting key.
- k_i^s stores the session key computed by π_i^s
- $\mathsf{role}_i^s \in \{\emptyset, \mathsf{Initiator}, \mathsf{Responder}\}$ indicates π_i^s's role during the protocol execution.

For each oracle π_i^s these variables are initialized to the empty string $\emptyset$. The computed session key is assigned to the variable k_i^s if and only if π_i^s reaches the Accept state, that is we have $k_i^s \neq \emptyset \Leftrightarrow \Psi_i^s = \mathsf{Accept}$. Furthermore the environment maintains three initially empty lists $\mathsf{L}_{\mathsf{corr}}$, $\mathsf{L}_{\mathsf{Send}}$ and $\mathsf{L}_{\mathsf{SessKey}}$ of all corrupted parties, sent messages and session keys respectively.

Partnering. We use the following partnering definitions (cf. [8]).

Definition 1 (Origin-oracle). *An oracle π_j^t is an origin-oracle for an oracle π_i^s if $\Psi_j^t \neq \emptyset$, $\Psi_i^s = \mathsf{Accept}$ and the messages sent by π_j^t equal the messages received by π_i^s, i.e., if $\mathsf{sent_j^t} = \mathsf{recv_i^s}$.*

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

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

- $\mathsf{Send}(i, s, m)$: This query allows $\mathcal{A}$ to send an arbitrary message m to oracle π_i^s. The oracle will respond according to the protocol specification and its current internal state. To start a new oracle, the message m takes the form:
  ($\mathtt{START} : \mathsf{role}, j$): If π_i^s was already initialized before, return $\perp$. Otherwise this initializes π_i^s in the role role, having party P_j as its intended peer. Thus, it sets $\mathsf{Pid_i^s} := j$ and $\mathsf{role}_i^s := \mathsf{role}$. If π_i^s is started in the initiator role ($\mathsf{role} = \mathsf{Initiator}$), then it outputs the first message of the protocol.
  All $\mathsf{Send}(i, s, m)$ calls are recorded in the list $\mathsf{L}_{\mathsf{Send}}$.
- $\mathsf{RevLTK}(i)$: For $i \leq \mu$, this query returns the long-term private key sk_i of party P_i. After this query, P_i and all its protocol instances π_i^s (for any s) are said to be *corrupted* and P_i is added to $\mathsf{L}_{\mathsf{corr}}$.
- $\mathsf{RegisterLTK}(i, \mathsf{pk}_i)$: For $i > \mu$, this query allows the adversary to register a new party P_i with the public key pk_i. The adversary is not required to know the corresponding private key. After the query, the pair (i, pk_i) is distributed to all other parties. Parties registered by $\mathsf{RegisterLTK}(i, \mathsf{pk}_i)$ (and their protocol instances) are corrupted by definition and are added to $\mathsf{L}_{\mathsf{corr}}$.

– $\mathsf{RevSessKey}(i, s)$: This query allows the adversary to learn the session key derived by an oracle. If $\Psi_i^s = \mathsf{Accept}$, return k_i^s. Otherwise return a random key k^* and add (π_i^s, k^*) to $\mathsf{L}_{\mathsf{SessKey}}$. After this query, π_i^s is said to be revealed. If this query is called for an oracle π_i^s, while there is an entry (π_j^t, k^*) in $\mathsf{L}_{\mathsf{SessKey}}$, so that π_i^s and π_j^t have matching conversations, then k^* is returned.[4]

Security. Formally, we have a security game, played between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, where $\mathcal{A}$ can issue the queries defined above. Additionally, it is given access to a special query $\mathsf{Test}(m)$, which, depending on a secret bit b chosen by the challenger, either returns real or random keys (for key indistinguishability) or an oracle to communicate with one of two specified parties in the sense of a left-or-right oracle for the privacy notions. The goal of the adversary is to guess the bit b. The adversary is only allowed to call $\mathsf{Test}(m)$ once and we distinguish the following two cases:

– Case $m = (\mathsf{TestKeyIndist}, i, s)$: If $\Psi_i^s \neq \mathsf{Accept}$, return $\perp$. Else, return k_b where $k_0 = k_i^s$ and $k_1 \xleftarrow{\$} \mathcal{K}$ is a random key. After this query, oracle π_i^s is said to be tested.
– Case $m = (Y, i, j)$, $Y \in \{\mathsf{Test\text{-}w\text{-}MITMPriv}, \mathsf{Test\text{-}s\text{-}MITMPriv}, \mathsf{TestForwardPriv}, \mathsf{TestCompletedSessionPriv}\}, i, j \leq \mu$: Create a new Party $P_{i|j}$ with identifier $i|j$. This party has all properties of P_i (if $b = 0$) or P_j (if $b = 1$), but no active sessions. The public key of $P_{i|j}$ is not announced to the adversary and the query $\mathsf{RevLTK}(i|j)$ always returns $\perp$. Furthermore create exactly one session $\pi_{i|j}^1$. Return the new handle $i|j$.

One-Way Privacy. The second case in $\mathsf{Test}(m)$ above models two-way privacy, i.e., we are considering that privacy needs to hold for the initiator and the responder. In case of one-way privacy, i.e., the privacy only holds either for the initiator or the responder (depending on the protocol), we need to restrict the adversary in a way such that the first message sent to $\pi_{i|j}^1$ via $\mathsf{Send}(i|j, 1, m)$ must be a `START` command. Analogously, we can model scenarios where we only consider privacy of the responder involved in a session.

Security Experiment. The experiment $\mathsf{Exp}^{\mathrm{X}}_{\mathrm{PPAKE},\mathcal{A}}$ is defined as follows.

1. Let μ be the number of parties in the game and ℓ the number of sessions per user. $\mathcal{C}$ begins by drawing a random bit $b \xleftarrow{\$} \{0, 1\}$ and generating key pairs $\{(\mathsf{sk}_i, \mathsf{pk}_i) \mid 1 \leq i \leq \mu\}$ as well as oracles $\{\pi_i^s \mid 1 \leq i \leq \mu, 1 \leq s \leq \ell\}$.
2. $\mathcal{C}$ now runs $\mathcal{A}$, providing all the public keys as input. During its execution, $\mathcal{A}$ may adaptively issue $\mathsf{Send}(i, s, m)$, $\mathsf{RevLTK}(i)$, $\mathsf{RevSessKey}(i, s)$ and $\mathsf{RegisterLTK}(i, \mathsf{pk}_i)$ queries any number of times and the $\mathsf{Test}(m)$ query once.
3. Depending on what argument Y the $\mathsf{Test}(m)$ oracle was called with, we require the corresponding property below to hold through the entire game.
   (a) $\mathsf{TestKeyIndist}$: The tested oracle remains fresh (cf. Definition 3).
   (b) $\mathsf{Test\text{-}w\text{-}MITMPriv}$: No oracle is ever corrupted.

[4] Note that the bookkeeping and consistent answers for matched sessions are required to avoid trivial distinguishers in case of cross tunnel attacks (cf. Sect. 3.3).

(c) Test-s-MITMPriv: P_i and P_j are never corrupted. Furthermore we require that $\mathsf{Pid}^1_{\mathsf{i|j}} = \perp$ or $\mathsf{Pid}^1_{\mathsf{i|j}} = k$ for some k, while P_k is never corrupted.

(d) TestForwardPriv: The returned oracle $\pi^1_{i|j}$ has a partner oracle π^r_k at the end of the game. Furthermore no oracle besides π^r_k may be instructed to start a protocol run with intended partner $P_{i|j}$.

(e) TestCompletedSessionPriv: The returned oracle $\pi^1_{i|j}$'s state is Accept at the end of the game. Let $k = \mathsf{Pid}^1_{\mathsf{i|j}}$. P_k are not corrupted, $\mathsf{RevSessKey}(i|j, 1)$ was never queried and $\mathsf{RevSessKey}(k, r)$ (for any π^r_k that has matching conversations) was never queried.
Furthermore no oracle besides π^r_k may be instructed to start a protocol run with intended partner $P_{i|j}$.

4. The game ends when $\mathcal{A}$ terminates with output b', representing the guess of the secret bit b. If $b' = b$, output 1. Otherwise output 0.

Definition 3 (Freshness). *An oracle π^s_i is fresh if*

1. $\mathsf{RevSessKey}(i, s)$ *has not been issued*
2. *no query* $\mathsf{RevSessKey}(j, t)$ *has been issued, where π^t_j is a partner of π^s_i.*
3. $\mathsf{Pid}^{\mathsf{s}}_{\mathsf{i}}$ *was:*
   (a) *not corrupted before π^s_i accepted if π^s_i has an origin-oracle, and*
   (b) *not corrupted at all if π^s_i has no origin-oracle.*

PPAKE Security. The above model can be parameterized by allowing or prohibiting the different types of $\mathsf{Test}(m)$ queries. This leads to the following:

Definition 4. *A key-exchange protocol Γ is called X for if for any PPT adversary $\mathcal{A}$ with access to the oracle* $\mathsf{Test}(m)$ *with queries of the form defined below, the advantage function*

$$\mathsf{Adv}^X_\Gamma(\lambda) := \left| \Pr\left[\mathsf{Exp}^X_{PPAKE,\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2} \right|$$

is negligible in λ, where

– $\mathcal{A}$ *queries* TestKeyIndist*: X = secure.*
– $\mathcal{A}$ *queries* Test-w-MITMPriv*: X = 2-way MITM private.*
– $\mathcal{A}$ *queries* Test-s-MITMPriv*: X = strongly 2-way MITM private.*
– $\mathcal{A}$ *queries* TestForwardPriv*: X = forward private.*
– $\mathcal{A}$ *queries* TestCompletedSessionPriv*: X = completed-session private.*

In the above definition, secure corresponds to having indistinguishable session keys, weak forward secrecy and security against key compromise impersonation (KCI). We now show how to integrate explicit entity authentication in our model, which allows to simplify the proofs of the protocols in Sect. 4. Therefore, we require the following:

Definition 5 (Matching Conversation). *Let Π be an N-message two-party protocol in which all messages are sent sequentially.*

– *If a session oracle π_i^s sent the last message of the protocol, then π_j^t is said to have matching conversations to π_i^s if the first $N-1$ messages of π_i^s's transcript agrees with the first $N-1$ messages of π_j^t's transcript.*
– *If a session oracle π_i^s received the last message of the protocol, then π_j^t is said to have matching conversations to π_i^s if all N messages of π_i^s's transcript agrees with π_j^t's transcript.*

We define implicit authentication through the fact that even a MITM adversary would not be able to derive the session key. This can be done in two moves. Explicit authentication is characterized by the fact that, additionally to providing implicit authentication, the protocol fails if a party does not possess a valid secret key, i.e., an active MITM adversary.

Definition 6 (Explicit entity authentication). *On game $\mathsf{PPAKE}_{\mathcal{A}}^{\text{2-way-priv}}$ define $\mathsf{break}_{\mathsf{EA}}$ to be the event that there exists an oracle π_i^s for which all the following conditions are satisfied.*

1. *π_i^s has accepted, that is, $\Psi_i^s = \mathsf{Accept}$.*
2. *$\mathsf{Pid_i^s} = j$ and party j is not corrupted.*
3. *There is no oracle π_j^t having:*
   (a) *matching conversations to π_i^s and*
   (b) *$\mathsf{Pid_j^t} = i$ and*
   (c) *$\mathsf{role}_j^t \neq \mathsf{role}_i^s$*

Definition 7. *A key-exchange protocol Γ has explicit authentication, if, for any PPT adversary $\mathcal{A}$, the event $\mathsf{break}_{\mathsf{EA}}$ (see Definition 6) occurs with at most $\mathsf{negl}(\lambda)$ probability.*

3.2 Relation Between Privacy Notions

Subsequently, we investigate the relations between the different privacy notions (as informally shown in Fig. 1).

Lemma 1. *Strong 2-way MITM privacy is strictly stronger than (weak) 2-way MITM privacy.*

Proof. This immediately follows from the tighter restrictions put on the attacker in the (weak) 2-way MITM privacy test. Furthermore, there are protocols that are (weak) 2-way MITM anonymous but not strongly 2-way MITM anonymous (see, e.g., Π_{ss} in Protocol 1). □

Lemma 2. *The 2-way MITM privacy notions are independent of forward privacy.*

Proof. Note that the privacy notions do not allow corruptions of the test oracle and forward privacy does not allow the attacker modify any sent messages (i.e. does not allow the attack to act as an active MITM). Π_{PKE}^2 (see Protocol 3) for instance is strongly 2-way MITM private (see Theorem 4) and hence also (weakly) 2-way MITM private, but it is not forward private as the identities are only encrypted using long term keys. On the other hand a protocol that runs the classic Diffie-Helman key exchange followed by transmitting their identities symmetrically encrypted would reach forward privacy, but no 2-way MITM privacy, as any MITM adversary could simply run the protocol. □

Completed-session privacy is implied by the other privacy notions: if a protocol is strong MITM private or has explicit authentication and is forward private, then it also provides completed session-privacy. The following lemma shows the implication starting from strong MITM privacy.

Lemma 3. *Let Γ be a PPAKE protocol. If Γ is strong MITM private, then it is completed-session private.*

Proof. Strong 2-way MITM privacy test puts less restrictions on the attacker.□

Finally, the following Theorem covers completed-session privacy from explicit authentication and forward privacy.

Theorem 1. *Let Γ be a PPAKE protocol. If Γ has explicit authentication and is forward private, then it is completed-session private.*

Proof. Assume for contradiction that some Γ has explicit authentication and is forward private, but is not completed-session private. This means a PPT-adversary $\mathcal{A}$ is able to call $\mathsf{TestCompletedSessionPriv}$ and not violate the imposed restrictions, while also correctly guessing the challenge bit b with non-negligible probability. Since Γ is forward private, the adversary violates a necessary restriction for calling $\mathsf{TestForwardPriv}$ while correctly guessing the challenge bit b. (Note that otherwise the exact same adversary $\mathcal{A}$ breaks forward privacy by simply using the argument $\mathsf{TestForwardPriv}$ instead).

It follows that after $\mathcal{A}$ is done, $\pi^1_{i|j}$ does not have a partner oracle with non-negligible probability. As per requirement of winning $\mathsf{TestCompletedSessionPriv}$, there is the oracle $\pi^1_{i|j}$ which has accepted and party P_k, where $k = \mathsf{Pid}^1_{\mathsf{i|j}}$, is not corrupted. Due to Γ providing explicit authentication, there is an oracle π^r_k s.t. π^r_k has matching conversations to $\pi^1_{i|j}$, $\mathsf{Pid}^\mathsf{r}_\mathsf{k} = i|j$ and $\mathsf{role}^r_k \neq \mathsf{role}^1_{i|j}$ (see Definition 6 detailing explicit entity authentication). Then $\mathcal{A}$ could simply not drop the last message (if it did before) thereby making $\pi^1_{i|j}$ and π^r_k have matching conversations to each other. This also makes $\pi^1_{i|j}$ and π^r_k be partnered to each other, without making it less likely for $\mathcal{A}$ to correctly guess the challenge bit b. Hence $\mathcal{A}$ is able to break forward privacy, which is a contradiction. □

3.3 Discussion and Limitations of Our PPAKE Model

Completed Session Privacy. $\mathsf{TestCompletedSessionPriv}$ is intended to represent the privacy notions of the literature, specifically Schäge et al. [30] and Zhao [34]. The only addition we made is the requirement that "no oracle besides π^r_k may be instructed to start a protocol run with intended partner $P_{i|j}$". This is a necessary addition since due to the nature of our model there are otherwise trivial attacks against a large class of protocols: First of all the adversary makes the test oracle complete its session without interfering and hence fulfills the experiment's requirements. It then corrupts both of the test oracle's possible identities. Finally it instructs a new oracle to initiate the protocol with the test oracle being the intended recipient, but answers all messages itself using the information obtained with the corruptions. If the imitator at any point uses the intended recipient's public key, e.g. for PKE, then the adversary learns the test oracle's identity.

This problem does not exist in the model of [30], since they let each initiator determine the identity of the test oracle (if configured correspondingly), instead of having the identity of the test oracle fixed throughout the entire experiment. We note that while [30] always model two identities per party, in our model every party only has a single identity.[5]

Revocation. In our model, corruptions are immediately publicly known. While this is an idealization, defending against secret corruptions is infeasible, since an adversary could perfectly impersonate the corrupted user.

As typically done in AKE, we do not formally cover revocation of long term keys in our model. There is previous work that explicitly models revocation for AKE protocols [5], but we want to avoid this added complexity since at this point we are not interested in the specifics of the respective revocation mechanism. Nevertheless, we note that for any revocation mechanism, the revocation status of a communication partner can only be checked after they revealed their identity. For this reason, we model strong MITM privacy so that the adversary can corrupt users as long as it does not openly identify itself as that user.

MITM Cross Tunnel Attack. We now discuss a generic MITM attack on privacy that does not require the corruption of any party, dubbed *MITM cross tunnel attack*. The goal of the attack is to de-anonymize a party that acts as a responder in the protocol. Specifically, the attack targets MITM privacy (both weak and strong). Let the responder be called $P_{i|j}$ and $\pi^1_{i|j}$ its corresponding session. Assume π^r_k is trying to communicate with $\pi^1_{i|j}$, but the adversary is a MITM in that communication channel. Assume at the same time, the adversary is MITM on another channel, where it knows that some π^z_y is trying to communicate with π^s_i. The adversary now relays all messages of π^z_y (of the second channel) to $\pi^1_{i|j}$ (of the first channel) and vice versa. Clearly, if either party produces an error or otherwise noticeably changes its behavior (e.g. by initiating the protocol again), then the adversary knows that $\pi^1_{i|j}$ cannot be the intended partner of π^z_y. Therefore $P_{i|j}$ must be P_j.

Defining protocols such that the parties – from an eavesdropper's view – do not behave noticeably different on errors (e.g. a party cannot decrypt a received ciphertext) prevents this attack as well as trivial distinguishers in case a party is revoked. Specifically, protocols need to continue similar to a normal execution,

[5] Clearly, one could however group parties to generate virtual parties with more identities in our model though.

but with randomly sampled messages and the sessions are internally marked as invalid. Our protocols in Sect. 4 are designed to counter these attacks. As noted before, fully preventing this attack in practice is only possible if higher level protocols do not reveal the session status, e.g. by restarting the AKE protocol.

4 Constructing PPAKE with Strong Privacy

In this section we discuss generic construction methodologies to achieve weak and strong MITM privacy, respectively. While not made explicit, all protocols are assumed to behave indistinguishable to real executions (from an eavesdropper's view) even if some verification (indicated using 🟩 boxes) fails, i.e., either a random bitstring or encryption of a random message is returned. Also, we assume that communication partners check the revocation status of the respective peers prior to engaging in a session initiation. All used encryption schemes are required to be length-hiding (cf. [32]), which we make explicit in the theorems.

User Certification and PKIs. In our protocols Cert_A indicates a certificate that binds the identity of A to the long term public key(s). We assume all users have their keys certified by some certification authority (CA) and that there is a mechanism in place for checking the revocation status of certificates. All these features are typically realized via a public-key infrastructure (PKI), i.e., PKIX. As already mentioned, we do not make such a mechanism explicit in our model.

4.1 Achieving Weak MITM Private PPAKE Using Shared Secrets

For the first protocol, we assume all honest parties belong to the same group and have a shared secret s only known to the members of the group. In terms of our model, the shared secret s is part of the secret keys and can hence be compromised by corrupting any party. With this shared secret, we can preserve anonymity against an active MITM adversary, that does not have access to s. But compromise of s does not endanger the usual key indistinguishability. The idea is to derive all session keys by additionally including this shared secret. So, even an active MITM attacker will be unable to use its knowledge of its share of the ephemeral keys due to the lack of knowledge of s. The scheme extending anonymous Diffie-Hellman with a shared secret and encrypted transfer of the peer's certificates is presented in Protocol 1. Similar to the protocols we discuss later, this protocol can also be rewritten in terms of any unauthenticated KE replacing the ephemeral DH shares and a signature scheme replacing the long term keys. We can show the following:

Theorem 2. *If the Oracle Diffie-Hellman (ODH) assumption holds and symmetric encryption scheme Ω is SE-LH-IND-CCA-secure, then Π_{ss} in Protocol 1 provides explicit entity authentication, is secure, weakly 2-way MITM private and forward private.*

$$
\begin{array}{lcl}
\textbf{Alice}\text{: } \mathsf{Cert}_A = (A = g^a, \ldots) & & \textbf{Bob}\text{: } \mathsf{Cert}_B = (B = g^b, \ldots) \\
\mathsf{sk}_A = (a, s) & & \mathsf{sk}_B = (b, s) \\
\hline
x \xleftarrow{\$} \mathbb{Z}_p & \xrightarrow{\quad m_1 = g^x \quad} & y \xleftarrow{\$} \mathbb{Z}_p \\
 & & k' \leftarrow H(g^{xy}, g^x, g^y, s) \\
 & & c_2 \leftarrow E_{k'}(\mathsf{Cert}_B) \\
k' \leftarrow H(g^{xy}, g^x, g^y, s) & \xleftarrow{\quad m_2 = (g^y, c_2, U) \quad} & U \leftarrow H(g^{xb}, g^{xy}, g^x, g^y, c_2, s) \\
\boxed{\text{verify } B, \mathsf{Cert}_B, U} & & \\
c_3 \leftarrow E_{k'}(\mathsf{Cert}_A) & & \\
V \leftarrow H(g^{ya}, g^{xy}, g^x, g^y, m_2, s) & \xrightarrow{\quad m_3 = (c_3, V) \quad} & \boxed{\text{verify } \mathsf{Cert}_A, V} \\
k \leftarrow H(g^{xy}, g^{xb}, g^{ya}, s, (m_i)_{i=1}^3) & & k \leftarrow H(g^{xy}, g^{xb}, g^{ya}, s, (m_i)_{i=1}^3)
\end{array}
$$

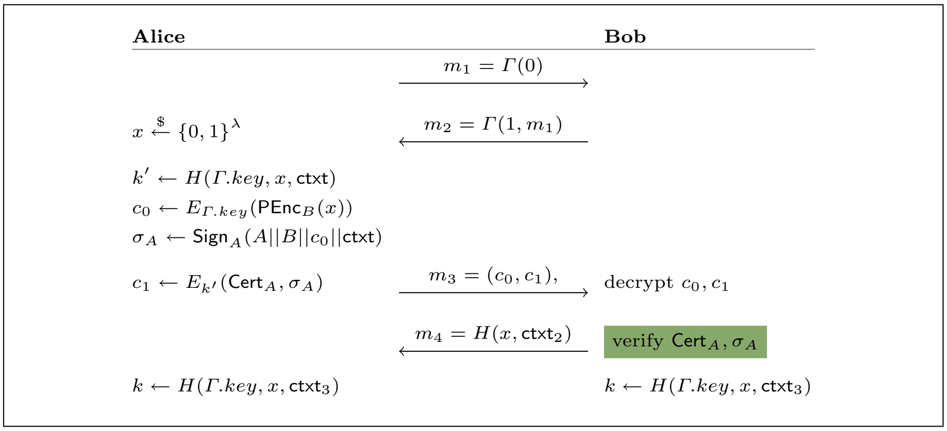
Protocol 1: Protocol Π_{ss} with shared secret s, using symmetric encryption $\Omega = (E, D)$.

For the proof we refer to the full version. Similar to the protocols from Wu et al. [33], the protocol in Protocol 1 is useful for managed groups. While their approach based on prefix encryption (PE) built from identity-based encryption (IBE) is more expressive, only their second protocol is able to provide weak MITM privacy. Our protocol highlights that weak MITM privacy can be obtained using less heavy tools than IBE. Note that Wu et al. [33] also require a trusted party (e.g., the CA) to generate and hand out secret keys to users. So this can be regarded as being similar to having a shared secret as in our approach.

4.2 Generic Construction of Strongly MITM Private PPAKE

Next, we introduce a protocol that achieves MITM privacy, in this case even strong MITM privacy, without relying on a shared secret. For this protocol and the protocol in Sect. 4.3, we consider a setting where the public keys (certificates) of responders are known a priori. Therefore, the initiator has all the information including all public keys of the responder available. Note however, that the first message cannot contain the initiator's certificate. Otherwise, if the long-term key of the responder is compromised, privacy of the initiator cannot be guaranteed (a trade-off that we make in Sect. 4.3). So, authentication of the initiator can only be performed after establishing an initial session key.

Similar to Π_{ss} we run a two-move KE and let the initiator sample a nonce which takes over the role of the shared secret of Π_{ss}, i.e., the (temporary) session keys are derived from the nonces and the result of the two-move KE. However, we now encrypt the nonce under the receivers public key. Moreover, after the initial shared key has been computed, the initiator is able to send its certificate to the responder and authenticate itself using a signature (which is encrypted together with the senders certificate). The protocol is depicted in Protocol 2.


Alice
Bob
$m_1 = \Gamma(0)$
$x \xleftarrow{\$} \{0,1\}^\lambda$
$m_2 = \Gamma(1, m_1)$
$k' \leftarrow H(\Gamma.key, x, \mathsf{ctxt})$
$c_0 \leftarrow E_{\Gamma.key}(\mathsf{PEnc}_B(x))$
$\sigma_A \leftarrow \mathsf{Sign}_A(A||B||c_0||\mathsf{ctxt})$
$c_1 \leftarrow E_{k'}(\mathsf{Cert}_A, \sigma_A)$
$m_3 = (c_0, c_1),$
decrypt c_0, c_1
$m_4 = H(x, \mathsf{ctxt}_2)$
verify $\mathsf{Cert}_A, \sigma_A$
$k \leftarrow H(\Gamma.key, x, \mathsf{ctxt}_3)$
$k \leftarrow H(\Gamma.key, x, \mathsf{ctxt}_3)$

Protocol 2: Protocol Π^4_{PKE}, using an unauthenticated KE Γ, PKE $\mathsf{PKE} = (\mathsf{PEnc}, \mathsf{PDec})$, symmetric encryption $\Omega = (E, D)$, signature scheme $\Sigma = (\mathsf{Sign}, \mathsf{Verify})$, $\mathsf{ctxt} = m_1 \| m_2$, $\mathsf{ctxt}_2 = A \| B \| \mathsf{ctxt} \| m_3$, and $\mathsf{ctxt}_3 = \mathsf{ctxt}_2 \| m_4$.

We note that due to active attacks the PKE is required to provide key-privacy, i.e., be PKE-IK-CCA-secure. Otherwise, an active attacker may determine the senders identity purely by means of the PKE ciphertext. This additional requirement on the PKE is fulfilled by many natural schemes (cf. [4,25]). Moreover, to obtain forward privacy the PKE ciphertext needs to be encrypted using the key from the anonymous two-move KE.[6]

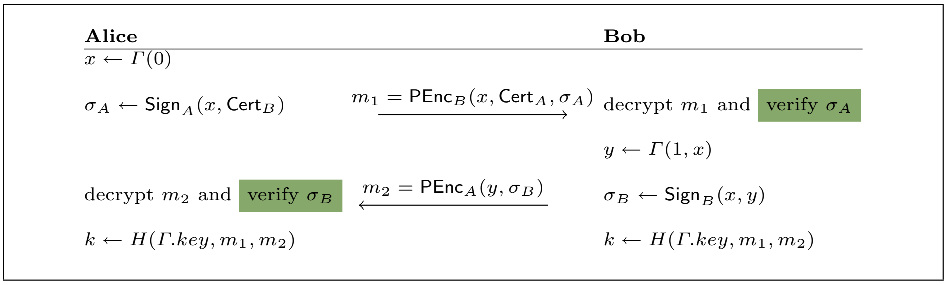
Theorem 3. *If KE Γ is unauthenticated and secure, the PKE* PKE *is PKE-IND-CCA- and PKE-IK-CCA-secure, symmetric encryption scheme Ω is SE-LH-IND-CCA-secure, and the signature scheme Σ is EUF-CMA-secure, then Π^4_{PKE} provides explicit entity authentication, is secure, strongly MITM private and forward private.*

For the proof we refer to the full version.

4.3 Two-Move PPAKE Protocol Without Forward Privacy

Finally, let us now present a two move variant of Π^4_{PKE}. Here, the initiator already includes the certificate in the first message and thus allows the responder to respond with a message encrypted with respect to the initiators public key and thus the protocol is authenticated after two moves. The resulting protocol, Π^2_{PKE}, is depicted in Protocol 3 and achieves strong MITM privacy, but obviously forward privacy can not be satisfied by this construction. In comparison to Π^4_{PKE}, the construction also requires the PKE to be length-hiding. Note though, when using anonymous DH as in Π_{ss} one can avoid the signatures.

[6] Otherwise an adversary obtaining all long-term PKE keys could simply try to test-decrypt. Omitting this countermeasure would require non-standard properties from the PKE, i.e.,. decryptions of ciphertexts under a key can also be decrypted with other keys and yield meaningful messages.


Alice
Bob
$x \leftarrow \Gamma(0)$
$\sigma_A \leftarrow \mathsf{Sign}_A(x, \mathsf{Cert}_B)$
$m_1 = \mathsf{PEnc}_B(x, \mathsf{Cert}_A, \sigma_A)$
decrypt m_1 and verify σ_A
$y \leftarrow \Gamma(1, x)$
decrypt m_2 and verify σ_B
$m_2 = \mathsf{PEnc}_A(y, \sigma_B)$
$\sigma_B \leftarrow \mathsf{Sign}_B(x, y)$
$k \leftarrow H(\Gamma.key, m_1, m_2)$
$k \leftarrow H(\Gamma.key, m_1, m_2)$

Protocol 3: Protocol Π^2_{PKE} using a PKE PKE, an unauthenticated KE Γ, and a signature scheme Σ. where Certs contain Σ and PKE public keys.

Theorem 4. *If KE Γ is secure, the PKE PKE is length-hiding, PKE-IND-CCA- and PKE-IK-CCA-secure, and the signature scheme Σ is EUF-CMA-secure, then Π^2_{PKE} provides explicit entity authentication, is secure, strongly MITM private and completed-session private.*

For the proof we refer to the full version.

5 Discussion and Future Work

In Table 2, we present an overview of the protocols presented in Sect. 4. All protocols provide completed-session privacy as well as weak MITM privacy, but for forward privacy and strong MITM privacy the picture looks different. Due the use of shared secret in Π_{ss}, strong MITM privacy does not hold. Yet this approach can be viewed as mitigation strategy for existing protocols to at least guarantee weak MITM privacy guarantees (e.g., for the IoT setting as targeted in [33]). For the PKE-based approach Π^4_{PKE} we require more than two moves to achieve forward privacy, but all other notions can already be achieved with the two move protocol Π^2_{PKE}.

Our motivation in this work was primarily to investigate the space of meaningful privacy notions and whether there are protocols that satisfy strong notions of privacy. An interesting question is the efficiency and privacy trade-off of concretely instantiated protocols as well as a strengthening of the model to support session state reveal queries. Currently only trivial ones would be supported and thus we decided to omit this feature. Another interesting direction, as done for

Table 2. Comparison of our protocols. "ss" denotes the requirement of a shared secret and "pk" the requirement to know the public key of the intended responder upfront.

	ss	pk	forward priv.	comp.-ses. priv.	w.-MITM	s.-MITM	# moves
Π_{ss}	✓	×	✓	✓	✓	×	3
Π^2_{PKE}	×	✓	×	✓	✓	✓	2
Π^4_{PKE}	×	✓	✓	✓	✓	✓	4

IKE v2 in [30], is to study which privacy properties deployed AKE protocols satisfy or how they can be modified in a way that they provide strong privacy guarantees.

Acknowledgements. This work was supported by the European Union's Horizon 2020 research and innovation programme under grant agreement n°826610 (Comp4Drones) and n°861696 (Labyrinth) and by the Austrian Science Fund (FWF) and netidee SCIENCE under grant agreement P31621-N38 (Profet).

References

1. Aiello, W., et al.: Just fast keying: key agreement in a hostile internet. ACM Trans. Inf. Syst. Secur. **7**(2), 242–273 (2004)
2. Arfaoui, G., Bultel, X., Fouque, P.A., Nedelcu, A., Onete, C.: The privacy of the TLS 1.3 protocol. PoPETs **2019**(4), 190–210 (2019). https://doi.org/10.2478/popets-2019-0065
3. Barbosa, M., Boldyreva, A., Chen, S., Warinschi, B.: Provable security analysis of FIDO2. Cryptology ePrint Archive, Report 2020/756 (2020). https://eprint.iacr.org/2020/756
4. Bellare, M., Boldyreva, A., Desai, A., Pointcheval, D.: Key-privacy in public-key encryption. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 566–582. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45682-1_33
5. Boyd, C., Cremers, C., Feltz, M., Paterson, K.G., Poettering, B., Stebila, D.: ASICS: authenticated key exchange security incorporating certification systems. In: Crampton, J., Jajodia, S., Mayes, K. (eds.) ESORICS 2013. LNCS, vol. 8134, pp. 381–399. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40203-6_22
6. Canetti, R., Krawczyk, H.: Security analysis of IKE's signature-based key-exchange protocol. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 143–161. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45708-9_10 https://eprint.iacr.org/2002/120/
7. Chai, Z., Ghafari, A., Houmansadr, A.: On the importance of encrypted-SNI (ESNI) to censorship circumvention. In: FOCI @ USENIX. USENIX Association (2019)
8. 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. LNCS, vol. 11694, pp. 767–797. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26954-8_25
9. Dingledine, R., Mathewson, N., Syverson, P.F.: Tor: the second-generation onion router. In: Blaze, M. (ed.) USENIX Security 2004, pp. 303–320. USENIX Association, August 2004
10. Donenfeld, J.A.: WireGuard: next generation kernel network tunnel. In: NDSS 2017. The Internet Society, Feb/Mar 2017
11. Dowling, B., Paterson, K.G.: A cryptographic analysis of the WireGuard protocol. In: Preneel, B., Vercauteren, F. (eds.) ACNS 2018. LNCS, vol. 10892, pp. 3–21. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93387-0_1
12. Fan, K., Li, H., Jiang, W., Xiao, C., Yang, Y.: U2F based secure mutual authentication protocol for mobile payment. In: ACM TUR-C, pp. 27:1–27:6. ACM (2017)

13. Goldberg, I., Stebila, D., Ustaoglu, B.: Anonymity and one-way authentication in key exchange protocols. Des. Codes Cryptogr. **67**(2), 245–269 (2013)
14. Gross, H., Hölbl, M., Slamanig, D., Spreitzer, R.: Privacy-aware authentication in the Internet of Things. In: Reiter, M., Naccache, D. (eds.) CANS 2015. LNCS, vol. 9476, pp. 32–39. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26823-1_3
15. Hoffman, P.E., McManus, P.: DNS queries over HTTPS (DoH). RFC **8484**, 1–21 (2018)
16. Hövelmanns, K., Kiltz, E., Schäge, S., Unruh, D.: Generic authenticated key exchange in the quantum random oracle model. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020. LNCS, vol. 12111, pp. 389–422. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45388-6_14
17. Hu, Z., Zhu, L., Heidemann, J.S., Mankin, A., Wessels, D., Hoffman, P.E.: Specification for DNS over transport layer security (TLS). RFC **7858**, 1–19 (2016)
18. Hülsing, A., Ning, K.C., Schwabe, P., Weber, F., Zimmermann, P.R.: Post-quantum WireGuard. Cryptology ePrint Archive, Report 2020/379 (2020). https://eprint.iacr.org/2020/379
19. Kaufman, C., Hoffman, P.E., Nir, Y., Eronen, P., Kivinen, T.: Internet key exchange protocol version 2 (IKEv2). RFC **7296**, 1–142 (2014)
20. Krawczyk, H.: SKEME: a versatile secure key exchange mechanism for internet. In: NDSS, pp. 114–127. IEEE (1996)
21. Krawczyk, H.: SIGMA: the 'SIGn-and-MAc' approach to authenticated Diffie-Hellman and its use in the IKE protocols. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 400–425. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_24
22. Lauter, K., Mityagin, A.: Security analysis of KEA authenticated key exchange protocol. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 378–394. Springer, Heidelberg (2006). https://doi.org/10.1007/11745853_25
23. 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
24. Øverlier, L., Syverson, P.: Improving efficiency and simplicity of Tor circuit establishment and hidden services. In: Borisov, N., Golle, P. (eds.) PET 2007. LNCS, vol. 4776, pp. 134–152. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75551-7_9
25. Paterson, K.G., Srinivasan, S.: Building key-private public-key encryption schemes. In: Boyd, C., González Nieto, J. (eds.) ACISP 2009. LNCS, vol. 5594, pp. 276–292. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02620-1_20
26. Perrin, T.: The noise protocol framework (2017). https://noiseprotocol.org
27. Rescorla, E.: The transport layer security (TLS) protocol version 1.3. RFC **8446**, 1–160 (2018)
28. Rescorla, E., Oku, K., Sullivan, N., Wood, C.A.: TLS encrypted client hello. Internet-Draft draft-ietf-tls-esni-07, Internet Engineering Task Force, June 2020. https://datatracker.ietf.org/doc/html/draft-ietf-tls-esni-07. Work in Progress
29. dos Santos, G.L., Guimaraes, V.T., da Cunha Rodrigues, G., Granville, L.Z., Tarouco, L.M.R.: A DTLS-based security architecture for the internet of things. In: ISCC, pp. 809–815. IEEE (2015)

30. Schäge, S., Schwenk, J., Lauer, S.: Privacy-preserving authenticated key exchange and the case of IKEv2. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020. LNCS, vol. 12111, pp. 567–596. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45388-6_20
31. Schwabe, P., Stebila, D., Wiggers, T.: Post-quantum TLS without handshake signatures. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) ACM CCS 2020, pp. 1461–1480. ACM Press, November 2020. https://doi.org/10.1145/3372297.3423350
32. Tezcan, C., Vaudenay, S.: On hiding a plaintext length by preencryption. In: Lopez, J., Tsudik, G. (eds.) ACNS 2011. LNCS, vol. 6715, pp. 345–358. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21554-4_20
33. Wu, D.J., Taly, A., Shankar, A., Boneh, D.: Privacy, discovery, and authentication for the Internet of Things. In: Askoxylakis, I., Ioannidis, S., Katsikas, S., Meadows, C. (eds.) ESORICS 2016. LNCS, vol. 9879, pp. 301–319. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45741-3_16
34. Zhao, Y.: Identity-concealed authenticated encryption and key exchange. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016, pp. 1464–1479. ACM Press, October 2016. https://doi.org/10.1145/2976749.2978350