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

Sebastian Ramacher[1], Daniel Slamanig[1], and Andreas Weninger[2]

[1] AIT Austrian Institute of Technology, Vienna, Austria
firstname.lastname@ait.ac.at
[2] TU Wien, Vienna, Austria
firstname.lastname@tuwien.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

* This is the full version of a paper which appears in Computer Security - ESORICS 2021, Lecture Notes in Computer Science, Springer. The proceedings version is available online at: https://doi.org/10.1007/978-3-030-88428-4_33.

active attackers, which may intercept, read, alter, replay, or 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) [HKSU20, HNS+21, SSW20].

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.[3] Schäge et al. [SSL20] 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 [Res18] 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 [Per17] 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 [Kra03]). Moreover, the recent proposal of Encrypted Client Hello (ECH) mechanism for TLS encrypts the initial client message (the ClientHello) [ROSW20] 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 [Kra96] or the SIGMA-I and SIGMA-R variants of the SIGMA protocol family [Kra03]. The work of Schäge et al. [SSL20], 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 [CGH19]. Consequently, all ESNI protected TLS connections have been blocked in China.[4] In general, one can observe a push towards an Internet infrastructure that reduces the amount of identifiable information. DNS over HTTPS/TLS [HM18, HZH+16] for in-

[3] 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 [ØS07, GSU13]) are outside the scope of this work.

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

stance 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) [dSGdCR+15, GHSS15] or FIDO2 [BBCW21, FLJ+17] we see an adoption of mutually authenticated AKE protocols and interest towards identity privacy. For instance, Wu et al. [WTSB16] 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 [Don17, DP18] 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. [ABB+04]. 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 [CK02]. Later Zhao in [Zha16] 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. [SSL20] provided a PPAKE model, which similarly to the one by Zhao [Zha16] 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 [KHN+14]. Also recently Arfaoui et al. [ABF+19] 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. [SSL20] for a discussion why this is not desirable). Interestingly, none of the previously proposed formal models (including [Zha16]) 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 Section 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 [CCG+19]. It is more gen-

eral than the recent PPAKE by Schäge et al. [SSL20] and among variants of privacy notions known from previous works [Zha16, SSL20] 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. [SSL20] and Zhao [Zha16], 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.

Follow-up work. In a recent work by Lyu et al. [LLHG22], the authors introduce the concept of robustness in PPAKE and present generic construction without the need to rely on random oracles. The authors also present a strategy to attack our four move protocol Π^4_{PKE}. In particular, the attack is targeted at forward privacy for identities of both the initiator and the responder when the responder is corrupted. Here it is important to note, that this strategy assumes broadcasting channels and that every potential receiver answers to every received message which is different from our model. Furthermore, our model considers forward privacy for past sessions (without adversarial interference) on compromise of long-term keys, whereas the adversaries in the attack of Lyu et al. needs to actively modify the session by responding to the first message of the initiator and thereby has knowledge of the ephemeral keys. Thus, the setting is outside of our security model and not considered in our paper.

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 [SSL20] for PPAKE, the assumptions on the network are stronger than those required by network anonymization protocols like Tor [DMS04]. 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 [SSL20], 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 [Zha16, ABF+19, SSL20], 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 [Zha16, SSL20]. We call it forward privacy and it can be seen

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

	completed session	incomplete session
passive $\mathcal{A}$	forward privacy $(\checkmark\checkmark)$	—
active $\mathcal{A}$	completed-session privacy $(\checkmark)$	(weak) 2-way MITM privacy $(\times)$ strong 2-way MITM privacy $(\checkmark)$

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.

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 [Kra96]). 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+ [LM06] 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 [Kra03, ABB+04, ABF+19] 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."* [ABB+04]

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 [Zha16, SSL20] 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 [SSL20] representing a variant of the SIGMA protocol family [Kra03] covering protocols such as TLS 1.3, QUIC, IPsec IKE or SSH. In particular, the SIGMA-R protocol that is designed to provide

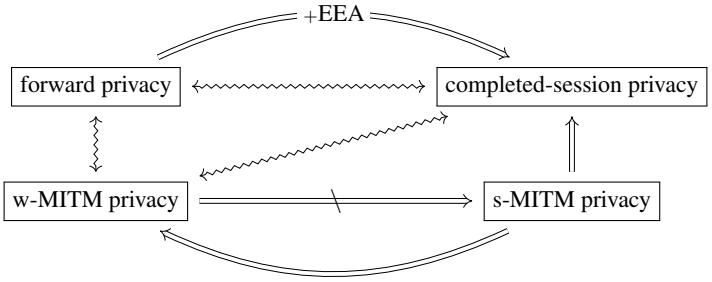


Fig. 1. Overview of implications and separations between privacy notions. $\leftrightsquigarrow$ denotes two incomparable properties and EEA denotes explicit entity authentication.

receiver identity protection is investigated. This template uses an 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 [SSL20] 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 [SSL20]. Also the model and the protocols by Zhao [Zha16] 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 [Zha16].

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 Figure 1 we provide an overview of the privacy notions captured in this paper and how they relate to each other (cf. Section 3.2 for a formal treatment). We note that completed-session privacy essentially reflects the privacy notions proposed by Zhao [Zha16] as well as Schäge et al. [SSL20].

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 [SSL20] 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 [CCG+19] which we extend to cover privacy features. Like [CCG+19], our model accounts for key impersonation (KCI) security and weak forward secrecy and we use their notion of origin-oracle partnering. We note that [CCG+19] avoid no-match attacks [LS17] as their concrete protocol's messages only contain group elements and deterministic functions of them. We consider the generic countermeasure from [LS17] 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[5] and to the public keys of all other parties, and maintains a list of internal state variables that are described in the following:

- Pid_i^s ("peer id") stores the identity of the intended communication partner. 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.
- $\Psi_i^s \in \{\emptyset, \mathsf{Accept}, \mathsf{Reject}\}$ indicates whether π_i^s has successfully completed the protocol execution and "accepted" the resulting key.

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

– 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 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. [CCG+19]).

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 $\mathsf{sent}_\mathsf{j}^\mathsf{t} = \mathsf{recv}_\mathsf{i}^\mathsf{s}$.*

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., $\mathsf{Pid}_\mathsf{i}^\mathsf{s} = j$ and $\mathsf{Pid}_\mathsf{j}^\mathsf{t} = i$; 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:
($\mathbf{START}$: 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 $\mathsf{Pid}_\mathsf{i}^\mathsf{s} := j$ and $\mathsf{role}_i^s := \mathsf{role}$. If π_i^s is started in the initiator role (role $=$ 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.[6]

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

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) TestKeyIndist: The tested oracle remains fresh (cf. Definition 3).
   (b) Test-w-MITMPriv: No oracle is ever corrupted.
   (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 $\pi_{i|j}^1$ has a partner oracle π_k^r at the end of the game. Furthermore no oracle besides π_k^r may be instructed to start a protocol run with intended partner $P_{i|j}$.
   (e) TestCompletedSessionPriv: The returned oracle $\pi_{i|j}^1$'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_k^r that has matching conversations) was never queried.

Furthermore no oracle besides π_k^r 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_i^s is fresh if*

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

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*

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

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 Section 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. $\mathsf{Pid}_{\mathsf{i}}^{\mathsf{s}} = j$ *and party* j *is not corrupted.*
3. *There is no oracle* π_j^t *having:*
   (a) *matching conversations to* π_i^s *and*
   (b) $\mathsf{Pid}_{\mathsf{j}}^{\mathsf{t}} = i$ *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 Figure 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^2_{PKE} (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 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 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 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 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. TestCompletedSessionPriv is intended to represent the privacy notions of the literature, specifically Schäge et al. [SSL20] and Zhao [Zha16]. 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 [SSL20], 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 [SSL20] always model two identities per party, in our model every party only has a single identity.[7]

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

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 [BCF+13], 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.

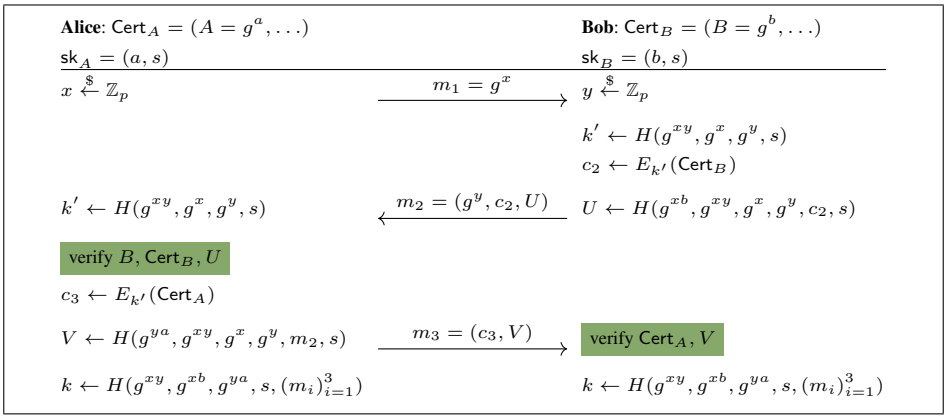
MITM Cross Tunnel Attack. We now discuss a generic MITM attack on privacy that does not require the corruption of any party, dubbed *MITM cross tunnel attack*. The goal of the attack is to de-anonymize a party that acts as a responder in the protocol. Specifically, the attack targets MITM privacy (both 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, but with randomly sampled messages and the sessions are internally marked as invalid. Our protocols in Section 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. [TV11]), 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



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

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.*

For the proof of this theorem we refer to Appendix B.1. Similar to the protocols from Wu et al. [WTSB16], 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. [WTSB16] 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 Section 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 Section 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.

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. [BBDP01, PS09]). Moreover, to obtain forward privacy the PKE ciphertext needs to be encrypted using the key from the anonymous two-move KE.[8]

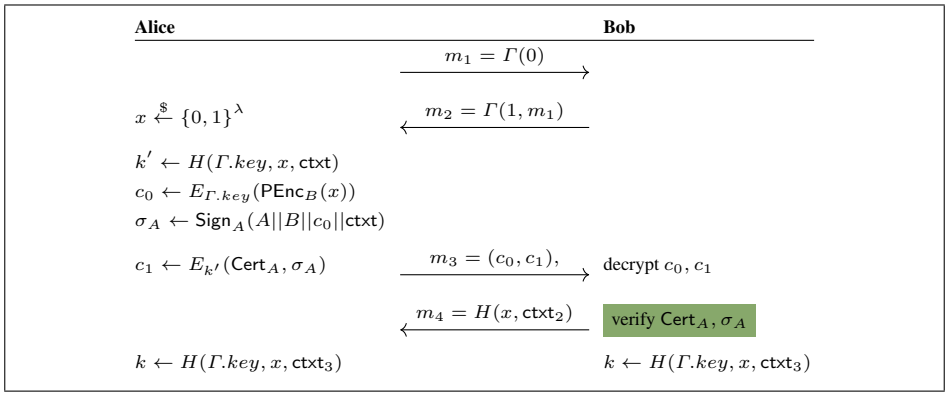
Theorem 3. *If KE Γ is unauthenticated and secure, the PKE PKE is PKE-IND-CCA- and PKE-IK-CCA-secure, symmetric encryption scheme Ω is SE-LH-IND-CCA-secure, and the signature scheme Σ is EUF-CMA-secure, then Π^4_{PKE} provides explicit entity authentication, is secure, strongly MITM private and forward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

The proofs are provided in Appendix B.2.

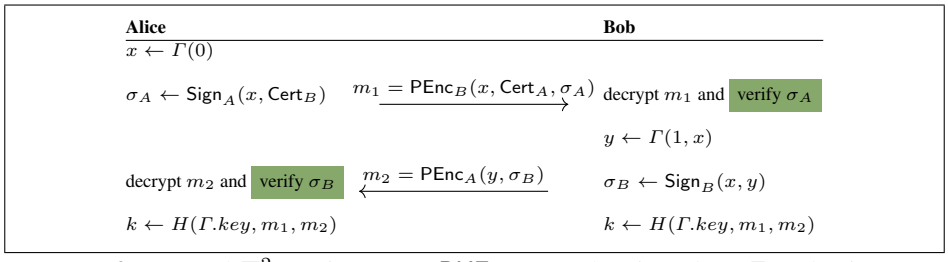
4.3 Two-Move PPAKE Protocol Without Forward Privacy

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

[8] 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.



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 $\mathsf{ctxt}_3 = \mathsf{ctxt}_2 \| m_4$.



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.

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.

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 of this theorem, we refer to Appendix B.3.

5 Discussion and Future Work

In Table 2, we present an overview of the protocols presented in Section 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 [WTSB16]). 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}.

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

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 IKE v2 in [SSL20], 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). The work of the third author was done in the course of his master thesis at AIT Austrian Institute of Technology.

References

ABB+04. William Aiello, Steven M. Bellovin, Matt Blaze, Ran Canetti, John Ioannidis, Angelos D. Keromytis, and Omer Reingold. Just fast keying: Key agreement in a hostile internet. *ACM Trans. Inf. Syst. Secur.*, 7(2):242–273, 2004.

ABF+19. Ghada Arfaoui, Xavier Bultel, Pierre-Alain Fouque, Adina Nedelcu, and Cristina Onete. The privacy of the TLS 1.3 protocol. *PoPETs*, 2019(4):190–210, October 2019.

BBCW21. Manuel Barbosa, Alexandra Boldyreva, Shan Chen, and Bogdan Warinschi. Provable security analysis of FIDO2. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part III*, volume 12827 of *LNCS*, pages 125–156, Virtual Event, August 2021. Springer, Heidelberg.

BBDP01. Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 566–582. Springer, Heidelberg, December 2001.

BBM00. Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 259–274. Springer, Heidelberg, May 2000.

BCF+13. Colin Boyd, Cas Cremers, Michele Feltz, Kenneth G. Paterson, Bertram Poettering, and Douglas Stebila. ASICS: Authenticated key exchange security incorporating certification systems. In Jason Crampton, Sushil Jajodia, and Keith Mayes, editors, *ESORICS 2013*, volume 8134 of *LNCS*, pages 381–399. Springer, Heidelberg, September 2013.

BFGJ17. Jacqueline Brendel, Marc Fischlin, Felix Günther, and Christian Janson. PRF-ODH: Relations, instantiations, and impossibility results. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 651–681. Springer, Heidelberg, August 2017.

CCG+19. Katriel Cohn-Gordon, Cas Cremers, Kristian Gjøsteen, Håkon Jacobsen, and Tibor Jager. Highly efficient key exchange protocols with optimal tightness. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 767–797. Springer, Heidelberg, August 2019.

CGH19. Zimo Chai, Amirhossein Ghafari, and Amir Houmansadr. On the importance of encrypted-sni (ESNI) to censorship circumvention. In *FOCI @ USENIX*. USENIX Association, 2019.

CK02. Ran Canetti and Hugo Krawczyk. Security analysis of IKE's signature-based key-exchange protocol. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 143–161. Springer, Heidelberg, August 2002. https://eprint.iacr.org/2002/120/.

DMS04. Roger Dingledine, Nick Mathewson, and Paul F. Syverson. Tor: The second-generation onion router. In Matt Blaze, editor, *USENIX Security 2004*, pages 303–320. USENIX Association, August 2004.

Don17. Jason A. Donenfeld. WireGuard: Next generation kernel network tunnel. In *NDSS 2017*. The Internet Society, February / March 2017.

DP18. Benjamin Dowling and Kenneth G. Paterson. A cryptographic analysis of the WireGuard protocol. In Bart Preneel and Frederik Vercauteren, editors, *ACNS 18*, volume 10892 of *LNCS*, pages 3–21. Springer, Heidelberg, July 2018.

dSGdCR+15. Glederson Lessa dos Santos, Vinicius Tavares Guimaraes, Guilherme da Cunha Rodrigues, Lisandro Zambenedetti Granville, and Liane Margarida Rockenbach Tarouco. A dtls-based security architecture for the internet of things. In *ISCC*, pages 809–815. IEEE, 2015.

FLJ+17. Kai Fan, Hui Li, Wei Jiang, Chengsheng Xiao, and Yintang Yang. U2F based secure mutual authentication protocol for mobile payment. In *ACM TUR-C*, pages 27:1–27:6. ACM, 2017.

GHSS15. Hannes Gross, Marko Hölbl, Daniel Slamanig, and Raphael Spreitzer. Privacy-aware authentication in the internet of things. In Michael Reiter and David Naccache, editors, *CANS 15*, LNCS, pages 32–39. Springer, Heidelberg, December 2015.

GSU13. Ian Goldberg, Douglas Stebila, and Berkant Ustaoglu. Anonymity and one-way authentication in key exchange protocols. *Des. Codes Cryptogr.*, 67(2):245–269, 2013.

HKSU20. Kathrin Hövelmanns, Eike Kiltz, Sven Schäge, and Dominique Unruh. Generic authenticated key exchange in the quantum random oracle model. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 389–422. Springer, Heidelberg, May 2020.

HM18. Paul E. Hoffman and Patrick McManus. DNS queries over HTTPS (doh). *RFC*, 8484:1–21, 2018.

HNS+21. Andreas Hülsing, Kai-Chun Ning, Peter Schwabe, Florian Weber, and Philip R. Zimmermann. Post-quantum WireGuard. In *2021 IEEE Symposium on Security and Privacy*, pages 304–321. IEEE Computer Society Press, May 2021.

HZH+16. Zi Hu, Liang Zhu, John S. Heidemann, Allison Mankin, Duane Wessels, and Paul E. Hoffman. Specification for DNS over transport layer security (TLS). *RFC*, 7858:1–19, 2016.

KHN+14. Charlie Kaufman, Paul E. Hoffman, Yoav Nir, Pasi Eronen, and Tero Kivinen. Internet key exchange protocol version 2 (ikev2). *RFC*, 7296:1–142, 2014.

Kra96. Hugo Krawczyk. SKEME: a versatile secure key exchange mechanism for internet. In James T. Ellis, B. Clifford Neuman, and David M. Balenson, editors, *NDSS'96*, pages 114–127. IEEE Computer Society, February 1996.

Kra03. Hugo Krawczyk. SIGMA: The "SIGn-and-MAc" approach to authenticated Diffie-Hellman and its use in the IKE protocols. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 400–425. Springer, Heidelberg, August 2003.

LLHG22. You Lyu, Shengli Liu, Shuai Han, and Dawu Gu. Privacy-preserving authenticated key exchange in the standard model. *IACR Cryptol. ePrint Arch.*, page 1217, 2022.

LM06. Kristin Lauter and Anton Mityagin. Security analysis of KEA authenticated key exchange protocol. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *PKC 2006*, volume 3958 of *LNCS*, pages 378–394. Springer, Heidelberg, April 2006.

LS17. Yong Li and Sven Schäge. No-match attacks and robust partnering definitions: Defining trivial attacks for security protocols is not trivial. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1343–1360. ACM Press, October / November 2017.

ØS07. Lasse Øverlier and Paul F. Syverson. Improving efficiency and simplicity of tor circuit establishment and hidden services. In Nikita Borisov and Philippe Golle, editors, *PET 2007*, volume 4776 of *LNCS*, pages 134–152. Springer, Heidelberg, June 2007.

Per17. Trevor Perrin. The noise protocol framework, 2017. https://noiseprotocol.org.

PS09. Kenneth G. Paterson and Sriramkrishnan Srinivasan. Building key-private public-key encryption schemes. In Colin Boyd and Juan Manuel González Nieto, editors, *ACISP 09*, volume 5594 of *LNCS*, pages 276–292. Springer, Heidelberg, July 2009.

Res18. Eric Rescorla. The transport layer security (TLS) protocol version 1.3. *RFC*, 8446:1–160, 2018.

ROSW20. Eric Rescorla, Kazuho Oku, Nick Sullivan, and Christopher A. Wood. TLS Encrypted Client Hello. Internet-Draft draft-ietf-tls-esni-07, Internet Engineering Task Force, June 2020. Work in Progress.

SSL20. Sven Schäge, Jörg Schwenk, and Sebastian Lauer. Privacy-preserving authenticated key exchange and the case of IKEv2. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 567–596. Springer, Heidelberg, May 2020.

SSW20. Peter Schwabe, Douglas Stebila, and Thom Wiggers. Post-quantum TLS without handshake signatures. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1461–1480. ACM Press, November 2020.

TV11. Cihangir Tezcan and Serge Vaudenay. On hiding a plaintext length by preencryption. In Javier Lopez and Gene Tsudik, editors, *ACNS 11*, volume 6715 of *LNCS*, pages 345–358. Springer, Heidelberg, June 2011.

WTSB16. David J. Wu, Ankur Taly, Asim Shankar, and Dan Boneh. Privacy, discovery, and authentication for the internet of things. In Ioannis G. Askoxylakis, Sotiris Ioannidis, Sokratis K. Katsikas, and Catherine A. Meadows, editors, *ESORICS 2016,*

Part II, volume 9879 of *LNCS*, pages 301–319. Springer, Heidelberg, September 2016.

Zha16. Yunlei Zhao. Identity-concealed authenticated encryption and key exchange. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 1464–1479. ACM Press, October 2016.

A Additional Definitions

Subsequently, we recall the strong Diffie-Hellman (SDH) assumption and the oracle Diffie-Hellman assumption (ODH) where the hash function is modeled as a random oracle which is implied by SDH in the ROM [BFGJ17]. Specifically, we consider the mmPRF-ODH assumption where the PRF is instantiated with a random oracle.

Definition 8 (SDH). *The strong Diffie-Hellman assumptions holds relative to* $\mathcal{G} = (\mathbb{G}, q, g)$ *and an oracle* $\mathsf{stDH}_{\mathsf{x}}(g^y, g^z)$ *that returns* 1 *if and only if* $xy = z$*, if for all PPT adversaries* $\mathcal{A}$*, there is a negligible function* ε *such that*

$$\Pr\left[\begin{matrix} x, y \xleftarrow{\$} \mathbb{Z}_q \\ h^* \leftarrow \mathcal{A}^{\mathsf{stDH}_{\mathsf{x}}(\cdot,\cdot)}(g^x, g^y) \end{matrix} : h^* = g^{xy}\right] \leq \varepsilon(\kappa).$$

Definition 9 (ODH). *The ODH assumptions holds relative to* $\mathcal{G} = (\mathbb{G}, q, g)$ *and an oracle* $H\colon \mathbb{G} \to \{0,1\}^\lambda$*, if for all PPT adversaries* $\mathcal{A}$*, there is a negligible function* ε *such that*

$$\left|\Pr\left[\begin{matrix} x, y \xleftarrow{\$} \mathbb{Z}_q, b \xleftarrow{\$} \{0,1\} \\ t^* \leftarrow \mathcal{A}^{H_x}(g^x) \\ \begin{cases} w \xleftarrow{\$} \{0,1\}^\lambda \textit{ if } b = 0 \\ w \leftarrow H(g^{xy}, t^*) \textit{ otherwise} \end{cases} \\ b^* \leftarrow \mathcal{A}^{H_x, H_y}(g^x, g^y, w) \end{matrix} : b = b^*\right] - \frac{1}{2}\right| \leq \varepsilon(\kappa)$$

where $H_x(h, t) = H(h^x, t)$ *and* $H_y(h, t) = H(h^y, t)$ *and the adversary may not query* H_x *on* (g^y, t^*) *and* H_y *on* (g^x, t^*)*, respectively.*

Public-key encryption. We briefly recall the definition and security notions of public-key encryption including the notion of key-privacy introduced by Bellare et al [BBDP01]. A public-key encryption (PKE) scheme PKE with message space $\mathcal{M}$ consists of the three PPT algorithms $(\mathsf{PSetup}, \mathsf{PGen}, \mathsf{PEnc}, \mathsf{PDec})$ defined as follows:

$\mathsf{PSetup}(\lambda)$: On input security parameter λ, outputs public parameters pp.
$\mathsf{PGen}(\mathsf{pp})$: On input public parameters pp, outputs public and secret keys $(\mathsf{pk}, \mathsf{sk})$.
$\mathsf{PEnc}_{\mathsf{pk}}(M)$: On input pk and message $M \in \mathcal{M}$, outputs a ciphertext ctxt.
$\mathsf{PDec}_{\mathsf{sk}}(\mathsf{ctxt})$: On input sk and ctxt, outputs $M \in \mathcal{M} \cup \{\perp\}$.

We note that we make the generation of shared public parameters, e.g., the choice of groups, explicit as separate algorithm PSetup. This is necessary for key privacy that we will discuss below.

A PKE scheme is correct if for all $\mathsf{pp} \leftarrow \mathsf{PSetup}(\lambda)$ and $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PGen}(\mathsf{pp})$, then

$$\Pr[c \leftarrow \mathsf{PEnc}_{\mathsf{pk}}(M) : \mathsf{PDec}_{\mathsf{sk}}(\mathsf{ctxt}) = M] = 1.$$

We say a PKE is PKE-IND-CCA-secure if and only if any PPT adversary $\mathcal{A}$ has only negligible advantage in the following security experiment. First, $\mathcal{A}$ gets an honestly generated public key pk. $\mathcal{A}$ outputs equal-length messages (M_0, M_1) and, in return, gets $\mathsf{ctxt}_b^* \leftarrow \mathsf{PEnc}_{\mathsf{pk}}(M_b)$, for $b \leftarrow\$ \{0,1\}$. Eventually, $\mathcal{A}$ outputs a guess b'. If $b = b'$, then the experiment outputs 1. During the experiment $\mathcal{A}$ has access to a decryption oracle $\mathsf{PDec}_{\mathsf{sk}}$ where the adversary can query decryptions of ciphertexts distinct from ctxt^*. If the adversary is not given access to the decryption oracle, then the scheme is PKE-IND-CPA-secure.

Definition 10. *For any PPT adversary $\mathcal{A}$ the advantage function*

$$\mathsf{Adv}^{\mathsf{pke\text{-}ind\text{-}cca}}_{\Pi,\mathcal{A}}(\lambda) := \left| \Pr\left[\mathsf{Exp}^{\mathsf{pke\text{-}ind\text{-}cca}}_{\mathsf{PKE},\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2} \right|,$$

is negligible in λ, where the experiment $\mathsf{Exp}^{\mathsf{pke\text{-}ind\text{-}cca}}_{\mathsf{PKE},\mathcal{A}}(\lambda)$ is given in Figure 2 and PKE *is a PKE as above.*

Exp. $\mathsf{Exp}^{\mathsf{pke\text{-}ind\text{-}cca}}_{\mathsf{PKE},\mathcal{A}}(\lambda)$

$\mathsf{pp} \leftarrow \mathsf{PSetup}(\lambda)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PGen}(\mathsf{pp})$
$(M_0, M_1) \leftarrow \mathcal{A}^{\mathsf{PDec}_{\mathsf{sk}}}(\mathsf{pk})$
$b \leftarrow\$ \{0,1\}$
$\mathsf{ctxt}^* \leftarrow \mathsf{PEnc}_{\mathsf{pk}}(M_b)$
$b' \leftarrow \mathcal{A}^{\mathsf{PDec}_{\mathsf{sk}}}(\mathsf{ctxt}^*)$
if $b = b'$ then **return** 1 **else return** 0

Fig. 2. PKE-IND-CCA security for PKE PKE.

We say a PKE PKE is PKE-IK-CCA-secure if and only if any PPT adversary $\mathcal{A}$ has only negligible advantage in the following security experiment. First, $\mathcal{A}$ gets two honestly generated public keys $\mathsf{pk}_0, \mathsf{pk}_1$. $\mathcal{A}$ outputs a message M and, in return, gets $\mathsf{ctxt}_b^* \leftarrow \mathsf{PEnc}_{\mathsf{pk}_b}(M)$, for $b \leftarrow\$ \{0,1\}$. Eventually, $\mathcal{A}$ outputs a guess b'. If $b = b'$, then the experiment outputs 1. During the experiment $\mathcal{A}$ has access to a decryption oracles $\mathsf{PDec}_{\mathsf{sk}_0}$ and $\mathsf{PDec}_{\mathsf{sk}_1}$ where the adversary can query decryptions of ciphertexts distinct from ctxt^*.

Definition 11. *For any PPT adversary $\mathcal{A}$ the advantage function*

$$\mathsf{Adv}^{\mathsf{pke\text{-}ik\text{-}cca}}_{\mathsf{PKE},\mathcal{A}}(\lambda) := \left| \Pr\left[\mathsf{Exp}^{\mathsf{pke\text{-}ik\text{-}cca}}_{\mathsf{PKE},\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2} \right|,$$

is negligible in λ, where the experiment $\mathsf{Exp}^{\mathsf{pke\text{-}ik\text{-}cca}}_{\mathsf{PKE},\mathcal{A}}(\lambda)$ is given in Figure 3 and PKE *is a PKE as above.*

Exp. $\mathsf{Exp}^{\mathsf{pke\text{-}ik\text{-}cca}}_{\mathsf{PKE},\mathcal{A}}(\lambda)$

$\mathsf{pp} \leftarrow \mathsf{PSetup}(\lambda)$
$(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow \mathsf{PGen}(\mathsf{pp}), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{PGen}(\mathsf{pp})$
$M \leftarrow \mathcal{A}^{\mathsf{PDec}_{\mathsf{sk}_0}, \mathsf{PDec}_{\mathsf{sk}_1}}(\mathsf{pk})$
$b \leftarrow\$ \{0,1\}$
$\mathsf{ctxt}^* \leftarrow \mathsf{PEnc}_{\mathsf{pk}_b}(M)$
$b' \leftarrow \mathcal{A}^{\mathsf{PDec}_{\mathsf{sk}_0}, \mathsf{PDec}_{\mathsf{sk}_1}}(\mathsf{ctxt}^*)$
if $b = b'$ then **return** 1 **else return** 0

Fig. 3. PKE-IK-CCA security for PKE PKE.

Exp. $\mathsf{Exp}^{\mathsf{pke\text{-}ik\text{-}cca}}_{\mathsf{PKE},\mathcal{A}}(\lambda)$

$\mathsf{pp} \leftarrow \mathsf{PSetup}(\lambda)$
$(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow \mathsf{PGen}(\mathsf{pp}), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{PGen}(\mathsf{pp})$
$b \leftarrow\$ \{0,1\}$
$b' \leftarrow \mathcal{A}^{\mathsf{PEnc}_{\mathsf{pk}_b}, \mathsf{PDec}_{\mathsf{sk}_0}, \mathsf{PDec}_{\mathsf{sk}_1}}(\mathsf{pk}_0, \mathsf{pk}_1)$
if $b = b'$ then **return** 1 **else return** 0

Fig. 4. PKE-IK-CCA security for PKE PKE with multiple queries.

We note that for both notions we presented the single-challenge notions. Using a hybrid argument, both can be extended to multi-challenge notions, e.g., see [BBM00].

Digital signatures. A signature scheme Σ consists of the PPT algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$, which are defined as follows:

$\mathsf{Gen}(1^\lambda)$: On input security parameter λ outputs a signing key sk and a verification key pk with associated message space $\mathcal{M}$.
$\mathsf{Sign}_{\mathsf{sk}}(M)$: On input, a secret key sk and a message $M \in \mathcal{M}$, outputs a signature σ.
$\mathsf{Verify}_{\mathsf{pk}}(M, \sigma)$: On input a public key pk, a message $M \in \mathcal{M}$ and a signature σ, outputs a bit b.

We assume that a signature scheme satisfies the usual (perfect) correctness notion, i.e. for all security parameters $\lambda \in \mathbb{N}$, for all $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$, for all $m \in \mathcal{M}$, we have that

$$\Pr\left[\mathsf{Verify}_{\mathsf{pk}}(M, \mathsf{Sign}_{\mathsf{sk}}(M)) = 1\right] = 1.$$

We say a signature Σ is EUF-CMA-secure if and only if any PPT adversary $\mathcal{A}$ has only negligible advantage in the following security experiment. First, $\mathcal{A}$ gets a honestly generated public key and outputs a message M^* and signature σ^*. During the experiment $\mathcal{A}$ has access to an singing oracle $\mathsf{Sign}_{\mathsf{sk}}$ where the adversary can query signatures for arbitrary messages. The experiment outputs 1 if and only if $\mathsf{Verify}_{\mathsf{pk}}(M^*, \sigma^*) = 1$ and M^* was not queried to the signing oracle.

Definition 12. *For any PPT adversary $\mathcal{A}$, we define the advantage in the EUF-CMA experiment $\mathsf{Exp}^{\mathsf{euf-cma}}_{\Sigma,\mathcal{A}}$ (cf. Figure 5) as*

$$\mathsf{Adv}^{\mathsf{euf-cma}}_{\Sigma,\mathcal{A}}(\lambda) := \Pr\left[\mathsf{Exp}^{\mathsf{euf-cma}}_{\Sigma,A}(\lambda) = 1\right].$$

Exp. $\mathsf{Exp}^{\mathsf{euf-cma}}_{\Sigma,\mathcal{A}}(\lambda)$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$
$(M^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}_{\mathsf{sk}}}(\mathsf{pk})$
if $\mathsf{Verify}(\mathsf{pk}, M^*, \sigma^*) = 1$ then **return** 1 **else return** 0

Fig. 5. The EUF-CMA experiment for a signature scheme Σ.

A signature scheme Σ is EUF-CMA-secure, if $\mathsf{Adv}^{\mathsf{euf-cma}}_{\Sigma,\mathcal{A}}(\lambda)$ is a negligible function in λ for all PPT adversaries $\mathcal{A}$.

Two-move key-exchange. We denote by Γ a two-move key exchange protocol between two PPT algorithms A and B running in three steps: $(\mathsf{st}_A, \mathsf{out}_A) \leftarrow \Gamma_A^{(1)}(1^\lambda)$ produces A's state and output; $(k_B, \mathsf{out}_B) \leftarrow \Gamma_B^{(1)}(1^\lambda, \mathsf{out}_A)$ on input A's output produces a key $k_B \in \mathcal{K}$ and finally on input B's output $k_A \leftarrow \Gamma_A^{(2)}(1^\lambda, \mathsf{out}_B, \mathsf{st}_A)$ produces a key $k_A \in \mathcal{K}$. Note that $\Gamma_A^{(1)}(\cdot)$ and $\Gamma_B^{(1)}(\cdot,\cdot)$ are stateless functions that only take the specified inputs. Specifically, they do not have access to any long-term keys. Correctness requires that for all $\lambda \in \mathbb{N}$ and all random tapes of A and B we have that $k_A = k_B$, except for a negligible error probability. We use the shorthand $(k, \mathsf{trans}) \leftarrow \Gamma_{A,B}(1^\lambda)$ to denote a run of the protocol where $\mathsf{trans} = (\mathsf{out}_A, \mathsf{out}_B)$. We say that a two-move key exchange protocol is secure against eavesdroppers if and only if any PPT adversary $\mathcal{A}$ has only negligible advantage in the following security experiment.

Exp. $\mathsf{Exp}^{\mathsf{eav}}_{\Gamma,\mathcal{A}}(\lambda)$

$k_0 \leftarrow\!\$\ \mathcal{K}$
$b \leftarrow\!\$\ \{0,1\}$
$(k_1, \mathsf{trans}) \leftarrow \Gamma(1^\lambda)$
$b' \leftarrow \mathcal{A}(k_b, \mathsf{trans})$
if $b = b'$ then **return** 1 **else return** 0

Fig. 6. The EAV experiment for a two-move key exchange protocol Γ.

Definition 13. *For any PPT adversary $\mathcal{A}$ the advantage function*

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

is negligible in λ, where the experiment $\mathsf{Exp}^{\mathsf{eav}}_{\Gamma,\mathcal{A}}(\lambda)$ is given in Figure 6 and Γ is a two-move key exchange protocol as above.

For brevity, we will call Γ secure if it is EAV-secure and we will write $\mathsf{out}_A \leftarrow \Gamma(0)$ for A's first message and $\mathsf{out}_B \leftarrow \Gamma(1, \mathsf{out}_A)$ for B's message and denote with $\Gamma.key$ the resulting key if everything is clear from the context.

Lemma 4. *Let Γ be an EAV-secure two-move key exchange protocol Γ, then it holds that:*

$$\Pr[\mathsf{out}_A = \mathsf{out}'_A] \leq \mathsf{negl}(\lambda) \ \textit{and} \ \Pr[\mathsf{out}_B = \mathsf{out}'_B] \leq \mathsf{negl}(\lambda)$$

where $\mathsf{out}_A, \mathsf{out}'_A$ *and* $\mathsf{out}_B, \mathsf{out}'_B$ *are results of independent calls to* $\Gamma(0)$ *and* $\Gamma(1, \mathsf{out}_A)$*, respectively.*

Proof. Note that correctness demands that a specific transcript fully determines a single key. If one pair $(\mathsf{out}_A, \mathsf{out}_B)$ could be produced in multiple runs with different resulting keys, then A and B would have no way to tell which key to agree on in a specific run, since they have no common information besides the transcript.

Assume the lemma does not hold, and view the case that $\Pr[\mathsf{out}_A = \mathsf{out}'_A]$ is non-negligible (the other case can be treated analogously). Construct an adversary $\mathcal{A}$ against the security of Γ as follows:

1. Upon receiving $k, (\mathsf{out}_A, \mathsf{out}_B)$, simply call $(\mathsf{st}'_A, \mathsf{out}'_A) \leftarrow \Gamma_A^{(1)}(1^\lambda)$.
2. **Case 1.** $\mathsf{out}'_A \neq \mathsf{out}_A$.
   Output a random bit b'.
3. **Case 2.** $\mathsf{out}'_A = \mathsf{out}_A$.
   Call $k_A \leftarrow \Gamma_A^{(2)}(1^\lambda, \mathsf{out}_B, \mathsf{st}'_A)$. As discussed before, k_A must be identical to the actual key that was derived in the challenger's protocol run. Hence output b' according to whether k_A equals k.

Since Case 2 has non-negligible probability of happening, this gives $\mathcal{A}$ a non-negligible advantage.

Symmetric encryption. A symmetric encryption with padding (SE) scheme Ω with key space $\mathcal{K}$ and message space $\mathcal{M}$ consists of the PPT algorithms (E, D) defined as follows:

$E_k(M, l)$: On input secret key k, message $M \in \mathcal{M}$ and length l ($l \geq |M|$), outputs a ciphertext ctxt.

$D_k(\mathsf{ctxt})$: On input secret key k and ctxt, outputs $M \in \mathcal{M} \cup \{\perp\}$.

A SE Ω is correct if for all $k \leftarrow\!\!\$\ \mathcal{K}$, $M \leftarrow\!\!\$\ \mathcal{M}$, $l \geq |M|$ it holds that

$$\Pr[\mathsf{ctxt} \leftarrow E_k(M, l) : D_k(\mathsf{ctxt}) = M] = 1.$$

We say a SE Ω is SE-LH-IND-CCA-secure (length-hiding indistinguishable under chosen ciphertext attacks) if and only if any PPT adversary $\mathcal{A}$ has only negligible advantage in the following security experiment. $\mathcal{A}$ outputs messages (M_0, M_1) and length ℓ with $\ell \geq \max\{M_0, M_1\}$ and, in return, gets $\mathsf{ctxt}^* \leftarrow E_k(M_b, \ell)$, for $b \leftarrow\!\!\$\ \{0, 1\}$. Eventually, $\mathcal{A}$ outputs a guess b'. If $b = b'$, then the experiment outputs 1. During the experiment $\mathcal{A}$ has access to an encryption oracle E_k and decryption oracle D_k where the adversary can query decryptions of ciphertexts distinct from ctxt^*.

Definition 14. *For any PPT adversary* $\mathcal{A}$ *the advantage function*

$$\mathsf{Adv}^{\mathsf{se\text{-}ind\text{-}cca}}_{\Omega,\mathcal{A}}(\lambda) := \left| \Pr\left[\mathsf{Exp}^{\mathsf{se\text{-}ind\text{-}cca}}_{\Omega,\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2} \right|,$$

is negligible in λ*, where the experiment* $\mathsf{Exp}^{\mathsf{se\text{-}ind\text{-}cca}}_{\Omega,\mathcal{A}}(\lambda)$ *is given in Figure 7 and* Ω *is a SE as above.*

In our protocols we write $E_k(M)$ instead of $E_k(M, \ell)$ if we assume there to be a suitable publicly known maximum length ℓ.

Exp. $\mathsf{Exp}^{\mathsf{se\text{-}ind\text{-}cca}}_{\Omega,\mathcal{A}}(\lambda)$

$k \leftarrow\$ \mathcal{K}$
$(M_0, M_1, \ell) \leftarrow \mathcal{A}^{E_k, D_k}(\mathsf{pk})$
$b \leftarrow\$ \{0,1\}$
$\mathsf{ctxt}^* \leftarrow E_k(M_b, \ell)$
$b' \leftarrow \mathcal{A}^{E_k, D_k}(\mathsf{ctxt}^*)$
if $b = b'$ then **return** 1 **else return** 0

Fig. 7. SE-LH-IND-CCA security for SE Ω.

B Proofs

B.1 Proofs for Protocol Π_{ss}

In this section we prove the security of Π_{ss}:

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.*

We prove this theorem in two parts as Lemmas 5 and 6.
For the proof of Lemma 5, we require the following definition:

Definition 15 ($\mathsf{LookupOrRandom}(\exists X_1, \ldots, X_k \colon H(V_1, \ldots, V_m), \phi)$)**.** *Let $X_1, \ldots, X_k$ denote variables. Let $V_1, \ldots, V_m$ denote expressions, containing the variables $X_1, \ldots, X_k$. Let ϕ be a logical formula, containing the variables $X_1, \ldots, X_k$. The proof shortcut $\mathsf{LookupOrRandom}(\exists X_1, \ldots, X_k \colon H(V_1, \ldots, V_m)\phi)$ is evaluated as follows.*

1. *If there are some $X_1, \ldots, X_k$ s.t. ϕ is true and the RO was queried before for $H(V_1, \ldots, V_m)$, then return this result $H(V_1, \ldots, V_m)$.*
2. *Else, draw Z randomly. Program the RO s.t. when it receives a query of the form $H(V_1, \ldots, V_m)$ for any $X_1, \ldots, X_k$ s.t. ϕ is true, answer Z. Return Z.*

Lemma 5. *If the ODH assumptions holds, then Π_{ss} in Protocol 1 provides explicit authentication.*

Proof. Assume $\mathcal{A}$ breaks explicit authentication.

- **Case 1.** π_i^s is initiator, let m_1 be its first message. Let $j = \mathsf{Pid}^{\mathsf{s}}_{\mathsf{i}}$ and $b = \mathsf{sk}_j$, $B = \mathsf{pk}_j$. The proof below is based on the following fact. Since π_i^s accepted, we know it received $m_2 = (Y, c_2, U)$ s.t. $D_{k'}(c_2) = B$ and $H(B^x, g^{xy}, g^x, g^y, c_2, s) = U$. We show that g^{xb} is hard to construct for $\mathcal{A}$. Let (X^*, B^*) be an arbitrary ODH-challenge.
  - Game 0: The original game.
  - Game 1: Before the start of the game, replace pk_j with B^* (thereby also modifying Cert_j). After some π_j^t receives X as the first message, let the value $U^* = \mathsf{LookupOrRandom}(\exists T \colon H(T, X^y, X, g^y, c_2, s), \mathsf{stDH}_{\mathsf{y}}(X, T) = 1))$. $m_2 = (g^y, E_{k'}(\mathsf{Cert}_j), U^*)$, where g^y and k' is computed normally. Record (U^*, m_1, m_2, k'). π_j^t returns m_2.

At the end of a protocol run, π_j^t determines its session key k as follows: Let $a = \mathsf{Pid}_\mathsf{j}^\mathsf{t}$, $A = \mathsf{pk}_a$. $k = \mathsf{LookupOrRandom}(\exists T\colon H(X^y, T, A^y, s), \mathsf{stDH}_\mathsf{y}(X, T) = 1)$.

- Game 2: Replace m_1 from π_i^s with X^*. π_i^s, after receiving $m_2 = (Y, c_2, U)$, validate U by checking if (a) (U, m_1, m_2, k') for some k' is in the secret table or (b) the RO produced U for the call $H(T, Z, X^*, Y, c_2, s)$ for any T, Z such that $\mathsf{stDH}_\mathsf{x}(B^*, T) = \mathsf{stDH}_\mathsf{x}(Y, Z) = 1$ (since B^* is pk_j). In case (b) set $k' = H(Z, X^*, Y, s)$ and $V^* = \mathsf{LookupOrRandom}(\exists Z_2\colon H(Y^a, Z_2, X^*, Y, m_2, s), \mathsf{stDH}_\mathsf{x}(Y, Z_2) = 1)$. π_i^s outputs $m_3 = (E_{k'}(\mathsf{pk}_i), V^*)$.
  At the end of a protocol run, π_i^s determines its session key k as follows: Let there be a second secret table. If the second secret table contains an entry for (m_1, m_2, m_3, k) for any k, take that k. Otherwise: Let $a = \mathsf{sk}_i$. $k = \mathsf{LookupOrRandom}(\exists R, O\colon H(R, O, Y^a, s), \mathsf{stDH}_\mathsf{x}(Y, R) = \mathsf{stDH}_\mathsf{x}(B^*, O) = 1)$. Record (m_1, m_2, m_3).[9]

Indistinguishability of game hops.

- Game 0 $\rightarrow$ Game 1: The change of pk_j cannot be detected from the initial public key distribution since B^* is drawn under the same distribution. The change of U to U^* cannot be detected, as the RO behaves accordingly. Since $\mathsf{Pid}_\mathsf{i}^\mathsf{s}$ must not be corrupted, $\mathsf{RevLTK}(j)$ cannot be called.
- Game 1 $\rightarrow$ Game 2: Replacing the first message cannot be detected, since X^* is drawn at the same probability distribution. The validation is indeed indistinguishable from a normal protocol run. k' either corresponds to the value of the other modified oracle or can be computed correctly.

Consequences. Notice in Game 2, π_i^s only accepts if it receives U s.t. either (a) (U, m_1, m_2) was put into the secret table by some π_j^t or (b) the RO was called for $H(T, Z, X^*, Y, c_2, s)$ where $\mathsf{stDH}_\mathsf{x}(B^*, T) = 1$. Since (a) implies that π_j^t has a matching conversation to π_i^s, it contradicts the initial assumption. On the other hand if (b) holds, we can construct an adversary against the ODH-assumption, by running Game 2 and outputting T.

– **Case 2.** π_i^s is responder, let $m_1 = X, m_3 = (c_3, V)$ be the messages it receives and $m_2 = (g^y, c_2, U)$ be the sent message. Let $j = \mathsf{Pid}_\mathsf{i}^\mathsf{s}$ and $a = \mathsf{sk}_j$, $A = \mathsf{pk}_j$. Since π_i^s accepted, we know that $V = H(A^y, X^y, X, g^y, m_2, s)$. This time $A^y = g^{ay}$ is difficult to produce for the adversary.
  Consider analogous game hops to Case 1, again the long-term key of π_j^t (in this case A) and the random group element of π_i^s (in this case g^y) are replaced with the ODH-challenge. The proof that these game hops are indistinguishable to the adversary is analogous. Again, either π_j^t has matching conversations to π_i^s or we can construct an adversary against the ODH-assumption.

Lemma 6. *If (E_k, D_k) is a SE-LH-IND-CCA secure symmetric encryption scheme and the DDH assumptions holds, then Π_{ss} in Protocol 1 is secure, (weakly) 2-way MITM anonymous and forward private.*

[9] In case both our modified oracles talk, they need to decide on the same random value, hence the second secret table and the check for (m_1, m_2, m_3, k).

Proof. Assume some PPT adversary $\mathcal{A}$ wins $\mathsf{PPAKE}_{\mathcal{A}}^{\text{2-way-priv}}$ with non-negligible probability. Without loss of generality, assume that only one type of $\mathsf{Test}(m)$ query is issued. (If not, we can construct an adversary $\mathcal{A}_X$ for $X \in \{\mathsf{TestForwardPriv}, \mathsf{Test\text{-}w\text{-}MITMPriv}, \mathsf{TestKeyIndist}\}$ that abort if a $\mathsf{Test}(m)$ query without i is called. At least one of them has non-negligible probability to win $\mathsf{PPAKE}_{\mathcal{A}}^{\text{2-way-priv}}$.) We view the different types of $\mathsf{Test}(m)$ queries separately.

1. Assume $\mathcal{A}$ used $\mathsf{TestKeyIndist}$. Let π_i^s be the tested oracle. In the cases that $\mathcal{A}$ wins, π_i^s conforms to freshness (see Definition 3). If clause 3a is satisfied, we know that there is a partner oracle π_j^t and in particular that π_i^s and π_j^t have matching conversations. If clause 3b is satisfied, we know that $j := \mathsf{Pid}_{\mathsf{i}}^{\mathsf{s}}$ is not corrupted. Also from the $\mathsf{Test}(m)$ query succeeding we know that π_i^s has accepted. Since our protocol provides explicit authentication (see Lemma 5), it follows that there is an oracle π_j^t having matching conversations to π_i^s. Therefore in both cases we have such an oracle π_j^t that has matching conversations to π_i^s.
   - Game 0: The original game.
   - Game 1: Guess s, i for the oracle that $\mathsf{Test}(m)$ will be called with. If guessed wrong: abort.
   - Game 2: Guess j, t and abort if π_j^t does not have matching conversations to π_i^s at the end.
   - Game 3: Let X, Y, Z be the challenge of a DDH-Challenger. Instead of randomly drawing x, y and thereby calculating g^x, g^y, π_i^s and π_j^t transmit X, Y (notice that due to the matching conversations, these exact values also reach the respective other oracle). Whenever g^{xy} should be used, i.e. the key derivation, instead use Z.

   Indistinguishability of game hops.
   - Game 0 $\rightarrow$ Game 1: This guessing leads to a polynomial loss of winning probability.
   - Game 1 $\rightarrow$ Game 2: This guessing leads to a polynomial loss of winning probability.
   - Game 2 $\rightarrow$ Game 3: Since $\mathcal{A}$ is not allowed to call $\mathsf{RevSessKey}(i, s)$ for π_i^s and π_j^t, it is not able to detect the embedding of X and Y, which are also drawn uniformly at random. In order to distinguish the keys k and k' from random or to distinguish them from the correctly calculated ones in Game 2, it needs to call (1) $H(Z, \ldots)$ or (2) $H(g^{xy}, \ldots)$, where x and y are the secret exponents used for X and Y. If $Z \neq g^{xy}$, it was drawn at random and hence cannot be guessed. We conclude $\mathcal{A}$ needed to call $H(g^{xy}, \ldots)$. Under the assumption that $\mathcal{A}$ is able to distinguish k from random or distinguish games 2 and 3, we hence guess which of the $poly(n)$ many queries issued to H by $\mathcal{A}$ contained g^{xy}. Thus, if Z is equal to this value g^{xy}, output 0 to the DDH-Challenger, otherwise output 1.
2. Assume $\mathcal{A}$ used $\mathsf{Test\text{-}w\text{-}MITMPriv}$ and can now interact with $\pi_{i|j}^1$. In order to win, $\mathcal{A}$ must not corrupt any oracles. We show that $\mathcal{A}$ must be able to break SE-LH-IND-CCA-security of (E, D).
   Case 1: $\pi_{i|j}^1$ did not send its last message (i.e. m_2 or m_3, depending on the role of $\pi_{i|j}^1$). If follows $\mathcal{A}$ can only base its decision on (a) m_1 and (b) the fact that $\pi_{i|j}^1$

might have rejected the previous message (i.e. m_1 or m_2, depending on the role of $\pi^1_{i|j}$). Since (a) does not reveal information (m_1 is random and independent on the test bit b), only (b) is possible. Since m_1 is only rejected if it is malformed, we only need to view the case that m_2 was rejected, i.e. $\pi^1_{i|j}$ is $\mathsf{Initiator}$. Since m_1, m_2 and the corresponding validations are independent on the ID of the initiator, m_2's acceptance/rejection cannot yield any information to the attacker.

Case 2: $\pi^1_{i|j}$ did send its last message. In case $\pi^1_{i|j}$ is initiator, this means $\pi^1_{i|j}$ has accepted. Since Π_{ss} has explicit authentication, this means that there is some partnered oracle π^r_k. Refer to the proof of the case "$\mathcal{A}$ used $\mathsf{TestForwardPriv}$" below. Otherwise, $\pi^1_{i|j}$ is responder and m_2 its only output. Since a fitting U cannot be produced by the adversary due to not knowing s, the adversary cannot modify m_2 in a valid way. Oracles that receive m_2 behave independent of the test bit b (and hence the $\mathsf{pk}_{i|j}$) except that they might set their state to Reject, which is invisible to the adversary. Therefore only m_2 itself might leak information. We show that this is not the case by implicitly replacing k' with a random k^* (only detectable by solving CDH, formal proof below) and the ciphertext c_2, which depends on the bit b, with a specific ciphertext c_2^* that is independent of b. We show that this indistinguishable to the adversary, unless it is able to break CPA-security of (E, D).

- Game 0: The original game.
- Game 1: Choose k^* randomly. Replace k' used by $\pi^1_{i|j}$ with k^*. If any oracle has sent the m_1, which was received by $\pi^1_{i|j}$ and receives the modified m_2 by $\pi^1_{i|j}$, it also replaces its k' with k^*.
- Game 2: Choose $m \xleftarrow{\$} \{\mathsf{Cert}_i, \mathsf{Cert}_j\}$ and replace c_2' with $c_2^* = E_{k^*}(m)$.

In Game 3 all challenge bit b related ID information of $\pi^1_{i|j}$ was removed, hence the adversary only has probability 0.5 of winning the game.

Indistinguishability of game hops.

- Game 0 $\Rightarrow$ Game 1: Since k' is the output of a random oracle, it is indistinguishable to the attacker that does not know s, since corruptions are not allowed.
- Game 1 $\Rightarrow$ Game 2: Due to SE-LH-IND-CCA security of E, this change is not noticeable. To show this, since the used key k^* is random and only used once, we can embed the challenge produced by a SE-LH-IND-CCA-challenger for $(m_0 = \mathsf{Cert}_i, m_1 = \mathsf{Cert}_j)$ in c_2.

3. Assume $\mathcal{A}$ used $\mathsf{TestForwardPriv}$. Hence $\pi^1_{i|j}$ is partnered to some π^r_k, i.e. $\mathcal{A}$ was passive during the communication.
   - Game 0: Original Game.
   - Game 1: Guess i, j, k, r. If wrong, abort.
   - Game 2: Sample k^* randomly. Produce the same transcript, but instead of sending $E_{k'}(cert_X)$ where $X \in \{i, j, k\}$, send $E_{k^*}(cert_X)$. Also replace U and V with random values.
   - Game 3: Replace the ciphertext sent by $\pi^1_{i|j}$ with the result of querying $\mathsf{Test}(cert_i, cert_j)$ at the CPA challenger.

   Since g^x, g^y, U, V, k^* are random, and the ciphertext sent by $\pi^1_{i|j}$ is identical for $b = 0$ and $b = 1$, the adversary has probability 0.5 of winning Game 3.

   Indistinguishability of game hops.

- Game 0 → Game 1: This leads to a polynomial loss.
- Game 1 → Game 2: If detectable, then $\mathcal{A}$ can solve CDH (g^{xy}).
- Game 2 → Game 3: Clearly, $\mathcal{A}$ behaves identical in Game 1 and Game 2 if b chosen by our game is equal to b' chosen by the CPA-challenger, since k'. If $\mathcal{A}$ can detect the game hop with non-negligible probability, then $b \neq b'$ with non-negligible probability. Hence send b to the challenger if $\mathcal{A}$ did not detect the game hop and $1 - b$ otherwise.

B.2 Proofs for Protocol Π^4_{PKE}

Lemma 7. *Π^4_{PKE} in Protocol 2 provides explicit authentication if* PKE *is a PKE-IND-CCA secure public-key encryption scheme, Ω is a SE-LH-IND-CCA secure symmetric encryption scheme and Σ is an EUF-CMA secure signature scheme.*

Proof. Assume for contradiction that $\mathcal{A}$ breaks explicit authentication, i.e. for some π^s_i, that has accepted and its peer $j = \mathsf{Pid}^{\mathsf{s}}_{\mathsf{i}}$ is not corrupted, there is no π^t_j that has matching conversations. We view the two cases of π^s_i's role separately.

Case 1. $\mathsf{role}^s_i = \mathsf{Initiator}$. It follows that π^s_i has received a valid m_4. Except with $\mathsf{negl}(\lambda)$ probability, this means that $H(x, \mathsf{ctxt}_2)$ was queried. Note that m_3 is the only available source to reproduce x.

- Game 0: The original game.
- Game 1: Guess i, s, j. Abort if wrong.
- Game 2: Let x be the value that π^s_i computes for sending m_3. (This is determined before the game using π^s_i's randomness tape.) Pick x^* randomly so that $||x^*|| = ||x||$. Modify π^s_i to use $c^* = \mathsf{PEnc}_j(x^*)$ instead of $\mathsf{PEnc}_j(x)$ in its message. Modify all instances π^t_j of P_j to not actually decrypt c^* but instead treat x as the result of the decryption. (Hence in this game all oracles act as if x was still used everywhere, except that m_3 and hence the ctxts have changed. Note that m_3 is now independent of x.)

Notice that in Game 2, x is only ever used as input to the RO. Hence the adversary can only guess x. Also the adversary or an oracle must produce $m_4 = H(x, \mathsf{ctxt}_2)$. If an oracle used the correct ctxt_2 this means has matching conversations to π^s_i and agrees on the identities and roles, which contradicts the initial assumption. Therefore the adversary must have guessed x or m_4 and the probability of winning Game 2 is $\mathsf{negl}(\lambda)$.

Indistinguishability of game hops.

- Game 0 → Game 1: This guessing leads to a polynomial loss of winning probability.
- Game 1 → Game 2: If $\mathcal{A}$ notices this change, we can break PKE-IND-CCA-security of the PKE. For this, modify Game 2 as follows: Let pk, PDec be the public key and oracle provided by the PKE-IND-CCA-challenger. Set $\mathsf{pk}_j = \mathsf{pk}$. All messages m_3 sent to instances of P_j can be decrypted using PDec. Replace c^* sent by π^s_i with the ciphertext c obtained from the PKE-IND-CCA-challenger for the messages $a_0 = x, a_1 = x^*$. Notice that $\mathsf{PDec}(c)$ is never queried as due to the definition

of Game 2. Now if the challengers bit $b = 0$, this game is behaving identical to Game 1. If $b = 1$, then this game behaves like Game 2. Therefore, our constructed adversary against PKE-IND-CCA-security outputs 1 if $\mathcal{A}$ notices the Game Hop from 1 to 2. Otherwise output a random bit.

Case 2. $\mathsf{role}_i^s = \mathsf{Responder}$. Then π_i^s received a valid m_3, which contains $\sigma = \mathsf{Sign}_j(j||i||c_0||\mathsf{ctxt})$.

Case 2a. Assume some π_j^t computed σ at any point. It follows π_j^t has matching m_1, m_2, c_0. Since m_1, m_2 are matching, $\Gamma.key$ is also matching. Since c_0 is matching, x and hence k' is also matching. The only way that π_j^t does not have matching conversations is if π_j^t produced a different c_1.

We show that this is only possible if $\mathcal{A}$ can break SE-LH-IND-CCA security of E_k or the EUF-CMA security of Sign. Let E, D be the oracles provided by some SE-LH-IND-CCA-challenger.

- Game 0: The original game.
- Game 1: Guess i, s, j, t. Abort if wrong.
- Game 2: π_i^s and π_j^t use a random k'.
- Game 3: π_j^t, instead of outputting c_1, outputs c_1^* which is received from the SE-LH-IND-CCA-challenger for $(a_0 = (\mathsf{Cert}_j, \mathsf{Sign}_j(A||B||c_0||\mathsf{ctxt})), a_1)$ where a_1 is a random message. π_i^s uses D for decryption and treats a_1 as veryfing. (a_0 will verify just like any other pair $(\mathsf{Cert}_j, \mathsf{Sign})$ where Sign is a valid signature by P_j.)

Indistinguishability of game hops.

- Game 0 $\rightarrow$ Game 1: This guessing leads to a polynomial loss of winning probability.
- Game 1 $\rightarrow$ Game 2: Notice both have matching m_1, m_2 i.e. honestly generated $\Gamma(0)$, $\Gamma(1)$. Hence we have an indistinguishable-from-random $\Gamma.key$ and consequently an indistinguishable-from-random k'.
- Game 2 $\rightarrow$ Game 3: In case m_3^* is the encrypted a_0, this change is unobservable. Hence if $\mathcal{A}$ can detect this change, we again break SE-LH-IND-CCA-security. Our constructed adversary against SE-LH-IND-CCA-security outputs $b' = 1$ if $\mathcal{A}$ detects the change and a random bit b' otherwise.

Since π_j^t does not have matching conversations as per assumption, D is never queried for m_3^*.

We now discuss how Game 3 can be translated into an adversary against SE-LH-IND-CCA security or EUF-CMA security.

- Note that the final game is identical to Game 2 from $\mathcal{A}$'s view, if the SE-LH-IND-CCA-challenger's bit b_C was 0.
- It follows that (a) the game hop 2 $\rightarrow$ 3 cannot be detected by $\mathcal{A}$ (otherwise this yields an adversary for the SE-LH-IND-CCA-challenge) and (b) $\mathcal{A}$ has non-negligible advantage if $b_C = 0$ (since it is the same as Game 2).
- Hence $\mathcal{A}$ must have non-negligible advantage as well if $b_C = 1$ (otherwise simply construct an adversary against SE-LH-IND-CCA-security that outputs 0 if $\mathcal{A}$ wins or a random bit otherwise).

– View the case that $b_C = 1$. In order to win, $\mathcal{A}$ needs to produce some c_1^*, $c_1^* \neq c_1$ (where c_1 was produced by π_j^t) and c_1^* is decrypted to a_1 or the pair $(\mathsf{Cert}_j, \mathsf{Sign})$ for some valid Sign of P_j.
– If in this case $\mathcal{A}$ produces some c_1 that is decrypted by π_i^s to a_1 with non-negligible probability, this allows us to construct an adversary against the SE-LH-IND-CCA security of the symmetric encryption scheme since a_1 was randomly chosen and can hence only be recovered by the adversary if $b_C = 1$.
– If on the other hand $\mathcal{A}$ produces some c_1 that is decrypted by π_i^s to $(\mathsf{Cert}_j, \mathsf{Sign})$ (where Sign is a valid signature) with non-negligible probability, we will show that this means that $\mathcal{A}$ was able to break EUF-CMA.
  First of all, start an EUF-CMA challenger to receive pk^* and gain access to the oracle Sign. Since we now attack EUF-CMA security, the SE-LH-IND-CCA challenger is considered part of our game and can be modified.
  - Game 3.0: Our current game, including the SE-LH-IND-CCA challenger.
  - Game 3.1: The SE-LH-IND-CCA challenger always uses $b_C = 1$.
  - Game 3.2: Instead of querying the SE-LH-IND-CCA challenger for (a_0, a_1) as defined before, query it for (a^*, a_1), where a^* is a random message.
  - Game 3.3: Replace the pk of P_j with pk^* and all protocol instances of party P_j use the provided oracle Sign instead of computing signatures themselves.

  Indistinguishable Game Hops:
  - Game 3.0 $\rightarrow$ 3.1: If this is noticeable to $\mathcal{A}$, this yields a trivial distinguisher against the SE-LH-IND-CCA security.
  - Game 3.1 $\rightarrow$ 3.2: This cannot be detected by $\mathcal{A}$, since a_0 is never used by the SE-LH-IND-CCA challenger anyways.
  - Game 3.2 $\rightarrow$ 3.3: Since corruptions of P_j are not allowed, $\mathcal{A}$ cannot notice this change.

  Deriving an adversary against EUF-CMA If $\mathcal{A}$ wins Game 3.3, it has to provide a valid signature (as discussed before). The message of this signature was never queried using the Sign oracle, since π_j^t will not have called this query as per game design and other oracles will never arrive at the same ctxt (recall Lemma 4). Hence the signature that $\mathcal{A}$ provided can be used to win the EUF-CMA game.

Case 2b. If no π_j^t produced $\sigma_j(\mathsf{ctxt})$: Break EUF-CMA as illustrated below. Let pk, Sign be given by a EUF-CMA challenger.

– Game 0: The original game.
– Game 1: Guess i, s, j, abort if wrong.
– Game 2: Set $\mathsf{pk}_j \leftarrow \mathsf{pk}$, implicitly setting sk_j to the sk by the EUF-CMA challenger. Any signing operations done by instances of P_j are done by calling Sign.

Our constructed adversary against EUF-CMA outputs the received $\sigma_j(\mathsf{ctxt})$.
Indistinguishability of game hops.

– Game 0 $\rightarrow$ Game 1: This guessing leads to a polynomial loss of winning probability.
– Game 1 $\rightarrow$ Game 2: This change is unobservable, since P_j must not be corrupted.

It follows that in all cases some π_j^t has matching conversations to π_i^s. □

We now restate Theorem 3 before proving it.

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.*

Proof. We divide the proof into separate parts for the different properties.

Π^4_{PKE} provides explicit authentication. Explicit authentication follows from Lemma 7.

Π^4_{PKE} is secure. Let π^s_i be the tested oracle. Let $j = \mathsf{Pid}^{\mathsf{s}}_{\mathsf{i}}$. π^s_i must conform to freshness (Definition 3). Case (a) Clause 3a was fulfilled, i.e. there is a partner oracle π^t_j. It follows that π^t_j has matching conversations to π^s_i. Case (b) Clause 3b was fulfilled, which means j must not be corrupted. Together with the fact that π^s_i accepted, from Π^4_{PKE} providing explicit authentication follows that there is some π^t_j that has matching conversations to π^s_i. It follows that in any case there is some π^t_j that has matching conversations to π^s_i.

To distinguish the session key k from random, $\mathcal{A}$ needs to query $H(\Gamma.key, x, \mathsf{ctxt}_3)$. We show that $\Gamma.key$ cannot be produced by the adversary.

- Game 0: The original game.
- Game 1: Guess i, s, j, t, abort if guessed wrong.
- Game 2: Before the game, query the challenger for the security of Γ and receive a transcript (m^*_1, m^*_2) and a key k^*. π^s_i and π^t_j send m^*_1, m^*_2 instead of newly computed m_1, m_2 and use k^* instead of $\Gamma.key$.
- Game 3: π^s_i and π^t_j use a random value u instead of k^*.

Indistinguishable Game Hops:

- Game $0 \to 1$: This results in a polynomial loss of winning probability.
- Game $1 \to 2$: If this game hop can be detected, it means that k^* is not the session key that corresponds to the transcript, and hence $\mathcal{A}$ can break Γ's security.
- Game $2 \to 3$: Analogous argument as for Game Hop $1 \to 2$.

In Game 3, $\mathcal{A}$ cannot deduce u since it is only used as input to the RO. Hence $\mathcal{A}$ has $\mathsf{negl}(\lambda)$ chance to win Game 3.

Π^4_{PKE} is strongly MITM-private.

- **Case 1.** $\pi^1_{i|j}$ is $\mathsf{Initiator}$. Therefore $p := \mathsf{Pid}^{\mathbf{1}}_{\mathsf{i|j}}$ is immediately set, and P_p must not be corrupted. Clearly, m_1 and m_2 are independent of the test bit b (i.e. independent of $\mathsf{pk}_{i|j}, \mathsf{sk}_{i|j}$).
  - Game 0: The original game.
  - Game 1: Guess i, j, p. Abort if $\mathsf{Test}(m)$ does not return $i|j$ or $p \neq \mathsf{Pid}^{\mathbf{1}}_{\mathsf{i|j}}$ at the end of the game.
  - Game 2: Replace part of the message m_3 by $\pi^1_{i|j}$ as follows: Instead of sending $c_0 = E_{\Gamma.key}(\mathsf{PEnc}_p(x))$ send $c^*_0 = E_{\Gamma.key}(\mathsf{PEnc}_p(z))$ where z is a random number of equal length to x. Program all other oracles to treat c^*_0 as c_0 (i.e. the decryption is x).

- Game 3: Pick k^* randomly. Program all oracles to use k^* instead of computing their original k', if k' should be computed using $k' \leftarrow H(\Gamma.key, x, \mathsf{ctxt})$ where x is the value computed by $\pi^1_{i|j}$ (determined before the game using $\pi^1_{i|j}$'s randomness tape) and $\Gamma.key, \mathsf{ctxt}$ are arbitrary values.
- Game 4: Replace part of the message m_3 by $\pi^1_{i|j}$ as follows: Instead of sending $c_1 = E_{k^*}(u)$ where $u = (\mathsf{Cert}_A, \mathsf{Sign}_A(A||B||c_0||\mathsf{ctxt}))$ send $c_1^* = E_{k^*}(w)$ for some random w. (Note that E is length hiding.)

Indistinguishability of game hops.

- Game 0 $\Rightarrow$ Game 1: This guessing leads to a polynomial loss of winning probability.
- Game 1 $\Rightarrow$ Game 2: The indistinguishability of this game hop follows from the PKE-IND-CCA security of PKE, see the proof for Lemma 7.
- Game 2 $\Rightarrow$ Game 3: Since in Game 2, x is only used as input to the RO, the adversary cannot obtain x and hence not check whether the RO actually produced this output. It follows that this change is only detectable with $\mathsf{negl}(\lambda)$ probability as well. (Compare with proof for explicit authentication.)
- Game 3 $\Rightarrow$ Game 4: The indistinguishability of this game hop follows from the SE-LH-IND-CCA security of the symmetric encryption Ω (see the proof for Π^4_{PKE} having explicit authentication).

Notice that in Game 4, $\mathsf{pk}_{i|j}, \mathsf{sk}_{i|j}$ were not used at all. Hence in Game 4 all oracles behave independent of b. It follows the probability of $\mathcal{A}$ winning Game 4 is $\frac{1}{2}$.

– **Case 2.** $\pi^1_{i|j}$ is Responder. m_1 and m_2 do not depend on $\mathsf{pk}_{i|j}, \mathsf{sk}_{i|j}$. Below we argue why m_3 does not reveal the key that was used for encryption. m_4 only depends on the key in the sense that it is only valid if m_3 can be decrypted.
Since PKE is PKE-IK-CCA, we can replace $\mathsf{pk}_{i|j}$ with a random key. To show this, consider the game hops below. Note that a valid or invalid m_4, i.e. $\pi^1_{i|j}$ being able to decrypt m_1 or not, does not give any information about the secret bit b anymore if $\mathsf{pk}_{i|j}$ is replaced with a random key.

- Game 0: The original game.
- Game 1: Whenever any oracle sends m_3 with intended recipient $\pi^1_{i|j}$, it saves the message it produced in a secret table together with the data that was encrypted. $\pi^1_{i|j}$, instead of decrypting incoming messages, will look up the content in the secret table. If the message is not in the table, it will attempt to decrypt the message normally.
- Game 2: $\pi^1_{i|j}$ treats the incoming message m_3 as malformed if there is no matching entry in the secret table. (Note that in this game, under no circumstances $\pi^1_{i|j}$ actually decrypts any messages.)
- Game 3: Instead of using the public key of i or j (depending on the secret bit b), the party $P_{i|j}$'s public key is set to a randomly generated key pk'.

Indistinguishability of game hops.

- Game 0 $\Rightarrow$ Game 1: This is only a conceptual change.
- Game 1 $\Rightarrow$ Game 2: This change can only be detected, if $\pi^1_{i|j}$ receives a valid m_3, where m_3 is not the (exact) output of some other oracle. $\pi^1_{i|j}$ would then wrongfully respond with a randomly generated m_4 (since it treats m_3 as malformed) whereas in Game 1 it would respond with a valid m_4.However in these

cases, in which $\pi^1_{i|j}$ produces a valid response in Game 1, $\pi^1_{i|j}$ is set to the accept state. Since Π^4_{PKE} has explicit authentication, it follows that m_3 authenticates a corrupted user (since no oracle has matching conversations to $\pi^1_{i|j}$), except with $\mathsf{negl}(\lambda)$ probability. This means that only in a setting in which the adversary would not have won Game 1 (except with $\mathsf{negl}(\lambda)$ probability), it can detect the change to Game 2. Hence we only lose a $\mathsf{negl}(\lambda)$ amount of winning probability in this scenario.

- Game 2 $\Rightarrow$ Game 3: If this change is detectable with non-negligible probability, then the adversary can break the PKE-IK-CCA security. To show this, use a PKE-IK-CCA challenger $\mathcal{C}$ and modify our Game 3 so that if $\mathcal{C}$ has chosen secret bit $b_C = 0$ we have Game 2 and if $b_C = 1$ we have Game 3. To do so, we first have to guess i and j that will be used by the adversary in $\mathsf{Test}(m)$ (resulting in polynomial loss of winning probability).
  Before the game starts, obtain pk_0 and pk_1 from C. Set $\mathsf{pk}_i = \mathsf{pk}_0$ or $\mathsf{pk}_j = \mathsf{pk}_0$ (depending on the secret bit b), so that $\mathsf{pk}_{i|j} = \mathsf{pk}_0$. Furthermore, set pk' (defined in Game 3) to pk_1. Now any oracle that starts communications with $\pi^1_{i|j}$ does not encrypt its first message on its own, but rather queries the encryption oracle provided by $\mathcal{C}$. If $b_C = 0$ this encrypted message exactly resembles Game 2 and if $b_C = 1$ it exactly resembles Game 3. The behaviour of $\pi^1_{i|j}$ does not need to be changed as it never actually decrypts the incoming messages.

Π^4_{PKE} is forward private. Consider the following game hops, that end in a game in which the transcript does not depend on b.

- Game 0: The original game.
- Game 1: Guess i, j, k, r. Abort if $\pi^1_{i|j}$ is not partnered to π^r_k at the end.
- Game 2: $\pi^1_{i|j}$ and π^r_k use a random r instead of the computed $\Gamma.key$.
- Game 3: $\pi^1_{i|j}$ and π^r_k use a random k^* instead of k'.
- Game 4: Instead of m_3, the initiator sends $m_3^* = (E_r(z), E_{k^*}(v))$, where z, v are random values. (Note that the PKE ciphertext, the certificate and the signature are removed from the transcript.) The receiver treats m_3^* as if it was the normally computed m_3.

Indistinguishability of game hops.

- Game 0 $\Rightarrow$ 1: This leads to a polynomial loss of winning probability.
- Game 1 $\Rightarrow$ 2: This change cannot be detected with noticeable probability due to the security of Γ. To show this, simply embed the messages produced by the challenger for eavesdropper-security of Γ in the first message of $\pi^1_{i|j}$ and π^r_k. If the attacker is then able to distinguish the computed key of Γ from a random one, it is able to win the $\mathsf{Exp}^{\mathsf{eav}}_{\Gamma,\mathcal{A}}(\lambda)$ game.
- Game 2 $\Rightarrow$ 3: This change cannot be detected, since the input to the RO (specifically r) is hidden.
- Game 3 $\Rightarrow$ 4: This change cannot be detected due to the length-hiding CCA security of (E, D) (SE-LH-IND-CCA security). To show this, we show that c_0 being replaced is undetectable (c_1 can be treated analogously). Ask some SE-LH-IND-CCA-challenger $\mathcal{C}$ for the normal input for encrypting c_0 as M_0 and $M_1 = z$,

receiving ctxt^*. Set $c_0 = \mathsf{ctxt}^*$. If the challengers bit $b_{\mathcal{C}} = 0$ this looks like Game 3 to the adversary. Hence if $\mathcal{A}$ can detect the modification of this game, it can break SE-LH-IND-CCA-security.

B.3 Proofs for Protocol Π^2_{PKE}

In this section we prove the security of Π^2_{PKE}.

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.*

The proofs in this section will be given in Lemmas 8 to 10 and we follow the same strategy as in Appendix B.2, hence we may skip some details.

Lemma 8. *If* PKE *is a length-hiding and PKE-IND-CCA secure PKE, and Σ is a EUF-CMA secure signature scheme, then Π^2_{PKE} in Protocol 3 provides explicit authentication.*

Proof. Assume for contradiction that $\mathcal{A}$ breaks explicit authentication, i.e., for some π_i^s, that has accepted and its peer $j = \mathsf{Pid}_\mathsf{i}^\mathsf{s}$ is not corrupted, there is no π_j^t that has matching conversations. We view the two cases of π_i^s's role separately.

Case 1. $\mathsf{role}_i^s = \mathsf{Initiator}$. It follows that π_i^s has received a valid m_2, which contains σ_B. This means there are two cases.

The first case is that no oracle computed σ_B, which means $\mathcal{A}$ breaks the $EUF-CMA$ security of Σ (following a similar argument as the proof of Lemma 7).

The second case is that the adversary used a σ_B that was produced by some π_j^t. We now show that this yields a contradiction.

- Game 0: The original game.
- Game 1: Guess i, s, j. Abort if wrong.
- Game 2: Let x be the value that π_i^s computes for its first message. (This is determined before the game using π_i^s's randomness tape.) Pick a random x^* of equal length. Modify π_i^s to actually send $m_1^* = \mathsf{PEnc}_j(x^*, \mathsf{Cert}_A, \sigma_A)$. Modify all instances of P_j to treat the first argument x^* as x when receiving m_1^*. (Hence in this game all oracles act as if x was still used everywhere, except that m_1, which is now independent of x, has changed.)
  Note that if the adversary wins Game 2, they must have taken σ_B from a message m_2, which was produced by some π_j^t after receiving m_1 (since otherwise there is no way to make π_j^t create a signature that contains x).
- Game 3: When π_j^t would send a signature of x, y (where x is the value that was read from the randomness tape in Game 2, not the transmitted value in m_1 of π_i^s), it now instead sends a random value U of equal length to the actual signature. Program all oracles to now treat U as equivalent to the original signature.

Indistinguishability of game hops.

– Game 0 → Game 1: This guessing leads to a polynomial loss of winning probability.
– Game 1 → Game 2: Follows from PKE-IND-CCA security of PKE similar to the proof of Lemma 7.
– Game 2 → Game 3: Follows from PKE-IND-CCA security of PKE similar to the proof of Lemma 7.

Note that in the final game, there is no trace of σ_B in the transcript. Hence if the adversary produces this value, this can be used to attack the $EUF-CMA$ security of Σ like in the first case. On the other hand, if $\mathcal{A}$ sends U to π_i^s, this means that the adversary was able to break PKE-IND-CCA security of PKE (the argument is similar to the proof of Lemma 7).

Case 2. $\mathsf{role}_i^s = \mathsf{Responder}$. Then π_i^s received a valid m_1, which contains $\sigma_j(\mathsf{ctxt})$.

Case 2a. If some π_j^t produced $\sigma_j(\mathsf{ctxt})$, similar to Case 2a. of Lemma 7 we can build an adversary against PKE-IND-CCA security of PKE or EUF-CMA security of Σ.

– Game 0: The original game.
– Game 1: Guess i, s, j, t. Abort if wrong.
– Game 2: π_j^t, instead of outputting m_1, outputs m_1^* which is received from the PKE-IND-CCA-challenger for $a_0 = (x, \mathsf{Cert}_j, \sigma_j(\mathsf{ctxt}))$ and a_1 being a random string (note that PKE is length-hiding). π_i^s uses the decryption oracle for decryption and treats both a_0 and a_1 as verifying.

Since π_j^t does not have matching conversations as per assumption, the decryption oracle is never queried for m_1^*. If in the final game, $\mathcal{A}$ wins, our constructed adversary against PKE-IND-CCA-security outputs b' according to the result of the decryption, i.e. a_0 or a_1. Otherwise it outputs a random bit b'.

Indistinguishability of game hops.

– Game 0 → Game 1: Guess this values incurs a polynomial loss.
– Game 1 → Game 2: In case m_1^* is the encrypted a_0, this change is unobservable. Hence if $\mathcal{A}$ can detect this change, we again break PKE-IND-CCA-security.

Case 2b. If no π_j^t produced $\sigma_j(\mathsf{ctxt})$, we construct and adversary against EUF-CMA of Σ in the same way as in Case 2b of Lemma 7, which we do not repeat here.

Lemma 9. *If KE Γ is unauthenticated and secure, and PKE PKE is length-hiding and PKE-IK-CCA secure, then Π^2_{PKE} in Protocol 3 is secure and strongly MITM-private.*

Proof. Π^2_{PKE} **is secure.** Let π_i^s be the tested oracle. Let $j = \mathsf{Pid}_\mathsf{i}^\mathsf{s}$. π_i^s must conform to freshness (Definition 3). Case (a) Clause 3a was fulfilled, i.e. there is a partner oracle π_j^t. It follows that π_j^t has matching conversations to π_i^s. Case (b) Clause 3b was fulfilled, which means j must not be corrupted. Together with the fact that π_i^s accepted, from Lemma 8 follows that there is some π_j^t that has matching conversations to π_i^s. It follows that in any case there is some π_j^t that has matching conversations to π_i^s.

To distinguish the session key k from random, $\mathcal{A}$ needs to query $H(\Gamma.key, x, \mathsf{ctxt}_3)$. Following the proof of Lemma 8, we can again use game hops that replace c_1 with the

ciphertext of some random value to show that no information about x is leaked by c_1. Since x is otherwise only used as input for the RO, $\mathcal{A}$ only has $\mathsf{negl}(\lambda)$ chance to win the game (e.g. by guessing x or sk_j).

Π^2_{PKE} is strongly MITM-private.

- **Case 1.** $\pi^1_{i|j}$ is Initiator.
  Therefore $k := \mathsf{Pid}^1_{\mathsf{i|j}}$ is immediately set.
  - Game 0: The original game.
  - Game 1: Guess i, j, k. Abort if $\mathsf{Test}(m)$ does not return $i|j$ or $k \neq \mathsf{Pid}^1_{\mathsf{i|j}}$ at the end of the game.
  - Game 2: Replace m_1 by $\pi^1_{i|j}$ with $m_1^* = \mathsf{PEnc}_k(z)$ where z is random bit string. Program all other oracles to treat m_1^* as m_1 (i.e., the decryption is x).

  The transitions are the same as in Lemma 8 hence we skip them here. Notice that in Game 2 $\mathcal{A}$ can only guess, hence the probability of winning Game 2 is $\frac{1}{2}$.
- **Case 2.** $\pi^1_{i|j}$ is Responder. m_2 does not depend $\mathsf{pk}_{i|j}$ and is sent even after receiving messages that are invalid or cannot be decrypted. Below we argue why m_1 does not reveal the key that was used for encryption. Since PKE is PKE-IK-CCA, we can replace $\mathsf{pk}_{i|j}$ with a random key. To show this, consider the game hops below. Note that a valid or invalid m_2, i.e. $\pi^1_{i|j}$ being able to decrypt m_1 or not, does not give any information about the secret bit b anymore if $\mathsf{pk}_{i|j}$ is replaced with a random key.
  - Game 0: The original game.
  - Game 1: Whenever any oracle is instructed to initiate communications with $\pi^1_{i|j}$, it saves the message it produced, i.e. m_1, in a secret table together with the data that was encrypted. $\pi^1_{i|j}$, instead of decrypting incoming messages, will look up the content in the secret table. If the message is not in the table, it will attempt to decrypt the message normally.
  - Game 2: $\pi^1_{i|j}$ treats the incoming message m_1 as malformed if there is no matching entry in the secret table. (Note that in this game, under no circumstances $\pi^1_{i|j}$ actually decrypts any messages.)
  - Game 3: Instead of using the public key of i or j (depending on the secret bit b), the party $P_{i|j}$'s public key is set to a randomly generated key pk'.

  The game hopes are based on the same indistinguishable game hops as discussed in Theorem 3, hence we do not repeat them here.

Lemma 10. *If PKE* PKE *is PKE-IK-CCA secure, then* Π^2_{PKE} *is completed-session private.*

Proof. Due to PKE-IND-CCA-security of PKE, m_1 and m_2 can be replaced with encryptions of random content in this proof (neither party may be corrupted). Due to PKE-IK-CCA security of PKE the used keys can be replaced with random keys (similar to proof of Theorem 3). It follows that the full transcript is randomly generated, i.e. independent of secret bit b.