A Modular Security Analysis of EAP and IEEE 802.11

Chris Brzuska[1] and Håkon Jacobsen[2,*]

[1] Hamburg University of Technology, Hamburg, Germany
brzuska@tuhh.de
[2] Norwegian University of Science and Technology, Trondheim, Norway
hakoja@item.ntnu.no

Abstract. We conduct a reduction-based security analysis of the Extensible Authentication Protocol (EAP), a widely used three-party authentication framework. We show that the main EAP construction, considered as a 3P-AKE protocol, achieves a security notion which we call AKE^w under the assumption that the EAP method employs channel binding. The AKE^w notion resembles two-pass variant of the eCK model. Our analysis is modular and reflects the compositional nature of EAP. Furthermore, we show that the security of EAP can easily be upgraded by adding an additional key-confirmation step. This key-confirmation step is often carried out in practice in the form of a link-layer specific AKE protocol that uses EAP for bootstrapping its authentication. A concrete example of this is the extremely common IEEE 802.11 4-Way-Handshake protocol used in WLANs. Building on our modular results for EAP, we get as our second major result the first provable security result for IEEE 802.11 with upper-layer authentication.

1 Introduction

The Extensible Authentication Protocol (EAP), specified in RFC 3748 [4], is a widely used authentication framework for network access control. It is particularly common in wireless networks, being used by protocols like IEEE 802.11 (Wi-Fi), IEEE 802.16 (WiMAX) and various 3G/4G mobile networks. The typical use case of EAP is in settings where a *client* seeks to gain access to a network controlled by an *authenticator*, but where the client and authenticator do not share any common credentials. EAP allows the client and authenticator to authenticate each other based on a mutually trusted *server*. Technically, EAP is not a specific authentication mechanism on its own, but rather specifies a certain generic three-party construction that composes other concrete authentication protocols into a unified framework. This provides applications of EAP the freedom to choose whatever concrete instantiation is suitable for their own specific setting. The success of this approach is apparent by the huge and diverse set of real-life deployments using the EAP framework.

*Håkon Jacobsen was supported by a STSM Grant from COST Action IC1306.

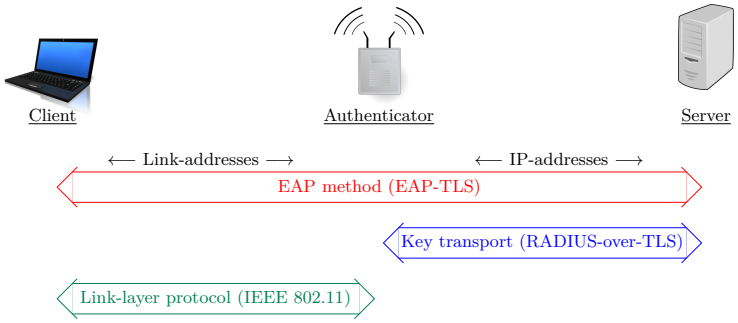

Fig. 1. The three-party EAP architecture. Example protocols shown in parenthesis.

Surprisingly then, given its prevalence and importance, there has been no formal reduction-based provable security analysis of EAP. One reason for this might be due to the general nature of EAP itself. As mentioned above, EAP is not a single protocol on its own, but relies on other sub-protocols to instantiate it. As such, many things in the EAP specification are left unspecified or considered out of scope. However, in order to conduct a formal security analysis of EAP, these details matter and require a careful treatment. Generally, the need to make assumptions on protocols outside of EAP makes analysis harder (see also [15]), because now it is not sufficient to consider a single protocol in isolation, but rather it has to be considered it in tandem with other protocols.

Another reason for the lack of provable security analyses of EAP might be the fact that it is a three-party protocol. As pointed out by Schwenk in his recent work on Kerberos [28], apart from a few papers like [8,3,24,5,28] relatively little work has been done on three-party protocols[3] in the computational setting compared to the huge literature on two-party protocols.

In this paper we aim to remedy this state-of-affairs by providing a formal reductionist analysis of EAP. We then build on our result to obtain a result for the extremely common IEEE 802.11 wireless standard *with* upper-layer authentication. Current results on IEEE 802.11 have so far only focused on the much simpler pre-shared key setting, while we can now provide an analysis of the full protocol. Below we will further expand upon our results, but first we provide a brief description of the EAP architecture and how IEEE 802.11 relates to it.

Review of EAP and IEEE 802.11. The general EAP architecture is shown in Fig. 1. The exchange begins with the client and trusted server authenticating each other using some concrete authentication protocol, like TLS. However, the whole TLS exchange is wrapped within a generic set of EAP messages, known as Request/Response messages. The combination of a concrete authentication protocol together with the EAP encapsulation is called an *EAP method*. Numerous EAP methods have been defined, with EAP-TLS being one of the most widely

[3] Considered distinct from *group-key exchange* protocols.

supported. Besides authenticating each other, the EAP-method usually also results in the client and server agreeing upon a shared key. The server will forward this key to the authenticator over some separately established channel. The EAP standard does not specify which protocol to use here, but in practice the de-facto standard is RADIUS [26].[4] Once the key is transported from the server to the authenticator—which so far has only operated in pass-through mode between the client and the server—the EAP exchange is technically complete. However, the client and authenticator now typically use the key distributed by EAP to authenticate each other using some link-layer specific protocol. If the link-layer media is a wireless link provided by the IEEE 802.11 protocol [2], then this entire exchange is usually referred to as "802.11 with upper-layer authentication".

On the difficulty of modeling EAP. In this paper we consider the provable security of both EAP and 802.11 with upper-layer authentication in the game-based setting. We do this in a modular way: first considering the security properties provided by EAP and 802.11 in isolation, then using a composition theorem to link them together. However, since EAP inherently depends on other protocols, assessing the exact security guarantees it provides is in a sense harder than for "standalone" protocols like TLS, IKE and SSH. While the EAP specification defines the security requirements of each EAP method ([4, §7]), this only covers the communication between the client and the trusted server. Still, as pointed out in the beginning, it is more accurate to think of EAP as a three-party protocol. But RFC 3748 leaves unspecified how, for example, the derived key should be transferred from the server to the authenticator. Hence, solely using the security claims from RFC 3748 is not sufficient to decide the security of EAP considered as a three-party protocol. In fact, without making further assumptions on the various protocols that make up EAP, it is impossible to talk about "the" EAP and its security. Consequently, in order to be able to analyze EAP, we will have to make some assumptions on these protocols.

Firstly, in this paper we are going to assume that the communication between the authenticator and the trusted server takes place over a secure channel. Specifically, we model the link as a two-party authenticated channel establishment protocol (2P-ACCE) based on symmetric long-term pre-shared keys[5] (see Section 2.3 for a formal definition). Since most key-transport protocols used between the server and the authenticator support to be run over a secure channel (see e.g. RADIUS-over-TLS [30]), this assumption seems reasonable.

Second, a well-known issue with the EAP architecture is the so-called "lying authenticator problem". Namely, a malicious authenticator may present false identity information to the client and the trusted server. Unless the EAP

[4] Within the EAP standard lingo, the protocol run between the server and authenticator is generally referred to as an *Authentication, Authorization and Accounting (AAA)* protocol.

[5] There is nothing fundamental about our assumption on symmetric PSKs here. We made the choice simply because the trust-relationship between the server and authenticator is commonly based on symmetric PSKs in practice. Our results work just as well for certificate-based authentication.

method provides a feature known as *channel binding* [14], there is no way for the client and server to verify that they are in fact talking to the same authenticator (see [14, §3] for examples of attacks that this may enable). Hence, in this paper we are generally going to assume that EAP provides channel binding, although we will also briefly explore the security guarantees provided by EAP without channel binding in Section 4.3. While there are a couple of ways to achieve channel binding in EAP (see [14, §4.1]), here we are only going to focus on the cryptographically simplest one, described in RFC draft `draft-ohba-eap-channel-binding-02` [25]. In this approach, the client and authenticator identities are being input to the key-derivation step of EAP, cryptographically binding the session key to the right pair of identities (see Section 4.2 for details).

Our contributions. The main contributions of this paper are the following.

- We provide the first reductionist-based provable security result for three-party EAP with channel binding.
- We show how the security guarantees of EAP can be upgraded by adding an additional key-confirmation step (modeled as a 2P-AKE). This corresponds to a common usage pattern where EAP is first used to bootstrap the establishment of a common key among the client and authenticator, then some lower-layer specific 2P-AKE is run between the client and authenticator to mutually prove possession of that key (in addition to establishing session keys for the lower-layer link).
- We provide the first game-based provable security result for the IEEE 802.11 4-way-handshake protocol in the pre-shared key setting. This corresponds to the setting typically found in home WLANs.
- More importantly however, the results above combine to provide the first reductionist-based provable-security result for the full IEEE 802.11 protocol with upper-layer authentication. This corresponds to the setting usually found in enterprise and university WLANs. For instance, the *eduroam* network, which is used to provide wireless roaming services to university and research institutions, uses IEEE 802.11 with upper-layer authentication.
- Our technical means for obtaining the above results are two modular composition theorems which may be of separate interest. Namely, the two theorems consider a fairly generic way of constructing a 3P-AKE protocol, using generic 2P-AKEs and secure channels as building blocks. For instance, both Kerberos and the AKA protocol used within the UMTS and LTE mobile networks, fit the description of our 3P-AKE construction. In particular, for the latter protocol, our theorems might enable a more general and modular analysis than the one recently provided by Alt et al. [5].

Technical overview of our results. The main technical contributions of this paper are two fairly generic composition theorems which correspond to the "cryptographic core" of EAP and IEEE 802.11 with upper-layer authentication, respectively. To obtain these theorems, however, we have to provide an appropriate

security model. Our starting point is the original 3P-AKE model of Bellare and Rogaway [8], but which we update to accommodate our needs. Most importantly, EAP and IEEE 802.11—both when considered separately and when combined—can achieve different levels of security. In order to capture these differences we have to define *three* different corruption models of differing strengths. These definitions are based on the eCK model[6] [21], and are primarily concerned with the level of adaptivity afforded to the adversary with respect to corruption queries. Preempting our own results a bit, we show that standalone EAP can achieve a restricted variant of forward secrecy, while IEEE 802.11 *without* upper-layer authentication achieves no forward secrecy (this is natural since it relies on symmetric primitives exclusively). However, when EAP and 802.11 are *combined*, the security is upgraded to achieve forward secrecy in our strongest corruption model. Briefly, the difference between the strongest security model and the intermediate one depends on what happens if the test-session does not have partner. When the test-session does not have a partner in the strongest model, the adversary is still allowed to learn all the long-term keys of the parties involved, as long as this happens after the test-session accepted. On the other hand, if the test-session does not have a partner in the intermediate model, then the adversary is forbidden from learning any of these long-term keys. If the test-session *does* have a partner, then there is no difference between the two models: the adversary is allowed to learn any long-term key at any time. The formal definitions of these models are provided in Section 2.2.

Intuitively, the reason why EAP on its own cannot achieve security in our strongest model is because it does not provide explicit entity authentication. Specifically, the client has no guarantee that the key-transport protocol between the server and authenticator actually took place without running some lower-link protocol to confirm. Suppose an adversary could learn the long-term key shared between the server and the authenticator *after* the client accepted, but *before* the key transport took place. Then it could simply impersonate the authenticator towards the server and have it send over the session key it previously established with the client. According to our strongest security model this adversary would be valid (since the exposure of the PSK happened after the client accepted this is allowed), whereas in the intermediate one it would not (since the client session does have a partner, the PSK cannot be exposed at all). Essentially, the purpose of the lower-layer protocol is to provide key-confirmation to the standalone EAP protocol, which ensures that the client will always have a partner before it accepts.

Besides the introduction of the three different corruption models, we only provide a few other changes to the original 3P-AKE model of Bellare and Rogaway [8]. For example, we support both asymmetric and symmetric long-term keys, and dispense with the explicit SendS query to the trusted server (now modeled simply as a regular Send query).

One thing we *do* keep from [8] however, is the concept of *partner functions*. Interestingly, the use of partner functions has seen rather limited adoption when

[6]However, we do not consider ephemeral key leakage in this paper.

compared to partnering based on matching conversations [7] or abstract session identifiers (SIDs) [6]. However, when modeling EAP, we are in the peculiar situation that the parties that we need to partner (the client and the authenticator) do not have any messages in common! Naturally, this makes partnering based on matching conversations more difficult, but it also severely limits our choice of SIDs: we are essentially forced to pick their session keys as the SID. While using the session key as the SID is reasonable in many settings (cf. [17]), it does not guarantee *public* partnering (see [11]). This is important for modular composition proofs like our own. While partnering functions have been criticized for being non-intuitive and hard to work with (even by Rogaway himself [27, §6]), they generalize more naturally to the three-party setting than SIDs. Essentially, this is because partner functions can take global transcript information into consideration rather than only the local views of the two partners. In a companion manuscript [10] we explore partner functions in more detail, showing their soundness as a partnering tool for analyzing key exchange protocols.

After proving the two composition results in Section 3 for generic protocols, we show how to apply them to EAP with and without upper-layer authentication in Sections 4 and 5, respectively.

2 Formal models

2.1 A unified execution model

Protocol participants. An AKE protocol is carried out by a set of *parties* $U \in \mathcal{P}$, where U either takes on the role of *initiator*, *responder* or *server*, i.e., $\mathcal{P}$ is partitioned into three disjoint sets $\mathcal{I}$, $\mathcal{R}$ and $\mathcal{S}$, consisting of the initiators, responders and servers, respectively. In this paper we assume that all initiators and servers are in possession of a long-term asymmetric key-pair (sk_U, pk_U), while all responders and servers share a symmetric pre-shared key K. For every party holding a public key, we assume that the other parties have an authenticated copy of it.

Syntax. A *protocol* is a tuple $\Pi = (\mathsf{KG}, \Sigma)$ of probabilistic polynomial-time algorithms, where KG specifies how long-term keys are generated for each party, and Σ specifies how (honest) parties behave. Each party $U \in \mathcal{P}$ can take part in multiple executions of the protocol, both concurrently and subsequently, called a *session*. We use an administrative label π_U^i to refer to the ith session at user U. This will sometimes also be simplified to π. Associated to each session π_U^i, there is a collection of variables that embodies the (local) state of π_U^i during the protocol.

- sk_U, pk_U – the long-term private/public key of party U,
- peers – a list of the identities of the intended communication peers of π_U^i,
- peerPK – a list of the public keys of the parties in $\pi_U^i.\mathsf{peers}$,
- $\mathsf{peerPSK}$ – a list of the long-term PSKs shared between U and $\pi_U^i.\mathsf{peers}$,
- $\vec{\alpha} = (\alpha_1, \ldots, \alpha_n)$ – a vector of *accept states* $\alpha_i \in \{\mathsf{running}, \mathsf{accepted}, \mathsf{rejected}\}$,

$\underline{\mathsf{Exp}_{\Pi,\mathcal{Q},\mathcal{A}}(\lambda)}$:
1: **for all** $U \in \mathcal{I} \cup \mathcal{S}$ **do**
2: $\quad (sk_U, pk_U) \leftarrow\!\!\$\ \mathsf{KG}(1^\lambda)$
3: **for all** $(U, V) \in \mathcal{R} \times \mathcal{S}$ **do**
4: $\quad K_{UV} = K_{VU} \leftarrow\!\!\$ \{0,1\}^\lambda$
5: $\mathsf{pks} \leftarrow \{(U, pk_U) \mid U \in \mathcal{I} \cup \mathcal{S}\}$
6: $out \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{Q}}(1^\lambda, \mathsf{pks})$

Fig. 2. Generic security experiment for a three-party protocol where all initiators and servers are in possession of a public key, and all responders and servers share a symmetric PSK.

– $k \in \{0,1\}^\lambda \cup \{\perp\}$ – the symmetric session-key derived by π^i_U.

Only initiators and responders accepts sessions keys, i.e., if $S \in \mathcal{S}$, then we always have $\pi^i_S.k = \perp$. Note that this is pure formalism; we certainly except many protocols in which the trusted server might be in possession of the session key—in fact, the trusted server might be the one that choses and distributes it—we simply do no not associate it with the variable k.

Remark 1. We use a *list* of acceptance states $\vec{\alpha}$ in order to model protocols that are logically built out of sub-protocols. The individual acceptance states α_i provides a convenient way to signal to the adversary that a session has accepted in some intermediate sub-protocol Π_i of the full protocol Π. By convention, we will let α_n represent the running-state of the full protocol, and use $\alpha_F \stackrel{\text{def}}{=} \alpha_n$ to denote this state. Specifically, α_F has the same role as the *single* running-state variable α which is typically used by most other formal protocol models. Saying that π is *running*, or that it has *accepted* or *rejected*, refers to the value of α_F.

We require the following semantics of the variables $\vec{\alpha} = (\alpha_1, \ldots, \alpha_n)$ and k:

$$\alpha_i = \mathsf{accepted} \implies \alpha_{i-1} = \mathsf{accepted}, \tag{1}$$

$$\alpha_i = \mathsf{rejected} \implies \alpha_{i+1} = \mathsf{rejected}, \tag{2}$$

$$\pi.\alpha_n = \mathsf{accepted} \implies \pi.k \neq \perp\ . \tag{3}$$

By convention, whenever we set $\alpha_i = \mathsf{rejected}$, we also automatically set $\alpha_j = \mathsf{rejected}$ for all $i < j$, in accordance with (2). Moreover, we assume that the session key $\pi.k$ is set only once.

A unified security experiment. To define the security goals of both AKE and ACCE protocols we use the unified experiment shown in Fig. 2. Experiment $\mathsf{Exp}_{\Pi,\mathcal{Q},\mathcal{A}}(\lambda)$ is parameterized on the protocol Π, a *query set* $\mathcal{Q}$, and the adversary $\mathcal{A}$. While the query sets used to define AKE and ACCE security will be different, they will both contain the following "base" query set $\mathcal{Q}_{base}$:

– $\mathsf{NewSession}(U, [V, W])$: This query creates a new session π_U^i at party U, optionally specifying its intended communication peers V and W. The state variables are initiated as follows: $\pi_U^i.k = \perp$, $\pi_U^i.\vec{\alpha} = \{\mathsf{running}, \ldots, \mathsf{running}\}$, if V and/or W are specified as U's peers, then $\pi_U^i.\mathsf{peers} = \{V, W\}$, $\pi_U^i.sk = sk_U$, $\pi_U^i.pk = pk_U$, $\pi.\mathsf{peerPK} = \{pk_V, pk_W\}$[7] and $\pi.\mathsf{peerPSK} = \{K_{UV}, K_{UW}\}$.[7] It is required that U, V and W all have different roles. Finally, if $U \in \mathcal{I}$, then π_U^i also produces its first message m according to specification of protocol Π. Both the administrative label π_U^i and m are returned to $\mathcal{A}$.
– $\mathsf{Send}(\pi, m)$: If $\pi.\alpha_F \neq \mathsf{running}$, return $\perp$. Otherwise, π creates a response message m^* according to the specification Σ. This depends on π's role and current internal state. Both m^* and $\pi.\vec{\alpha}$ are returned to $\mathcal{A}$.
– $\mathsf{Reveal}(\pi_U^i)$: If $\pi.\alpha_F \neq \mathsf{accepted}$ or $U \in \mathcal{S}$, return $\perp$. Else, return $\pi_U^i.k$. From this point on π_U^i is said to be *revealed*. Note that π_U^i is *not* considered revealed if the Reveal query was made before π accepted.
– $\mathsf{LongTermKeyReveal}(U, [V])$: Depending on the second input parameter, this query returns a certain long-term key of party U.
  - $\mathsf{LongTermKeyReveal}(U)$: If U has an associated private-public key-pair (sk_U, pk_U), return the private key sk_U.
  - $\mathsf{LongTermKeyReveal}(U, V)$: If U and V share a symmetric long-term key K_{UV}, return K_{UV}.

  After a long-term key is leaked we say that it is *exposed* and the corresponding party *corrupted*.

Note that we are working in the post-specified peer model [13], meaning that the identities of a session's peers might not necessarily be known by the session at the onset of the protocol, but are instead learned as the protocol progresses.

Protocol correctness. It is required that a protocol satisfies the following correctness requirement. In an honest execution of the protocol between an initiator π_A^i, a responder π_B^j and a trusted server π_S^k—meaning that all messages are faithfully transmitted between them according the protocol description—then all sessions end up accepting, and π_A^i and π_B^j both hold the same session key $k \neq \perp$.

Remark 2. Note that experiment $\mathsf{Exp}_{\Pi,\mathcal{Q},\mathcal{A}}(\lambda)$ does not provide any output and does not define any "winning condition" for $\mathcal{A}$. Instead, it provides a single execution experiment on which we can define many different winning conditions. This is convenient when we later want to define AKE-security and ACCE-security.

Transcripts and partner functions. Consider a run of experiment $\mathsf{Exp}_{\Pi,\mathcal{Q},\mathcal{A}}(\lambda)$, where $\mathcal{Q}_{base} \subseteq \mathcal{Q}$. Let T be the ordered transcript consisting of all the Send and $\mathsf{NewSession}$ queries made by $\mathcal{A}$, together with their responses. A transcript T is a *prefix* of T', written $T \subseteq T'$, if the first $|T|$ entries of T' are identical to T. We let

[7] In case V or W does not hold a public key, or if U does not a share a PSK with V or W, then these values are set to $\perp$.

$\mathcal{T}$ denote the set of all possible transcripts generated from running experiment $\mathsf{Exp}_{\Pi,\mathcal{Q},\mathcal{A}}(\lambda)$. To define partnering in our security analysis we use the concept of partner functions as introduced by Bellare and Rogaway [8].

Definition 1 (Partner functions). *A* partner function *is a polynomial-time function* $f\colon \mathcal{T} \times (\mathcal{P} \setminus \mathcal{S}) \times \mathbb{N} \to ((\mathcal{P} \setminus \mathcal{S}) \times \mathbb{N}) \cup \{\perp\}$*, subject to the following requirement*

$$f(T,U,i) = (V,j) \implies f(T',U,i) = (V,j) \text{ for all } T \subseteq T'. \tag{4}$$

Instead of $f(T,U,i) = (V,j)$, we also write $f_T(\pi_U^i) = \pi_V^j$, or even just $f_T(\pi) = \pi'$ if the exact identities of the sessions are irrelevant.

Definition 2 (Partnering). *Let f be a partner function. A session π' is a* partner *to π if $f_T(\pi) = \pi'$.*

Remark 3. Partnering is only defined between initiators and responders. Servers are not considered partners to any session.

Partnering soundness. For a security analysis based on partner functions to be meaningful, the partner function needs to satisfy certain soundness properties. Briefly, soundness demands that partners should: (1) end up with the same session key, (2) agree upon who they are talking to, (3) have compatible roles, and (4) be unique. These requirements are essentially the same as those demanded for SIDs through the "Match-security" notion introduced by Brzuska et al. [11].

Definition 3 (Partnering soundness predicate). *Consider a run of experiment $\mathsf{Exp}_{\Pi,\mathcal{Q},\mathcal{A}}(\lambda)$, and let T be the corresponding transcript. Predicate* Sound *is true if and only if the following holds for all $T' \subseteq T$. If sessions π_U^i and π_V^j have both accepted and $f_{T'}(\pi_U^i) = \pi_V^j$, then*

1. $\pi_U^i.k = \pi_V^j.k \neq \perp$,
2. $\pi_U^i.\mathsf{peers} = \{V,W\}$, $\pi_V^j.\mathsf{peers} = \{U,W\}$, *and* $W \in \mathcal{S}$,
3. $U \in \mathcal{I} \wedge V \in \mathcal{R}$ *or* $U \in \mathcal{R} \wedge V \in \mathcal{I}$,
4. *there is no* $\pi' \neq \pi_U^i$ *such that* $f_{T'}(\pi') = f_{T'}(\pi_U^i)$.

We let $\mathsf{Exp}^{\mathsf{Sound}}_{\Pi,\mathcal{Q},\mathcal{A}}(\lambda) \Rightarrow 1$ *denote the event that predicate* Sound *evaluated to true.*

Remark 4. Note that predicate Sound depends on the partner function f.

Remark 5. The use of partner functions to analyze key exchange protocols is rare in the literature. To the best of our knowledge, besides the original paper by Bellare and Rogaway [8], it has only been used in one other paper by Shoup and Rubin [29].

2.2 2P-AKE and 3P-AKE

Syntax. A *2P/3P-AKE protocol* has the same syntax as the general protocol defined in Section 2.1. Moreover, in our framework, there is no syntactical difference between a 2P-AKE protocol and a 3P-AKE protocol. However, in a 2P-AKE protocol there is no trusted server session $S \in \mathcal{S}$, and the session variables $\pi.\mathsf{peers}$, $\pi.\mathsf{peerPK}$ and $\pi.\mathsf{peerPSK}$ contain at most a single entry.

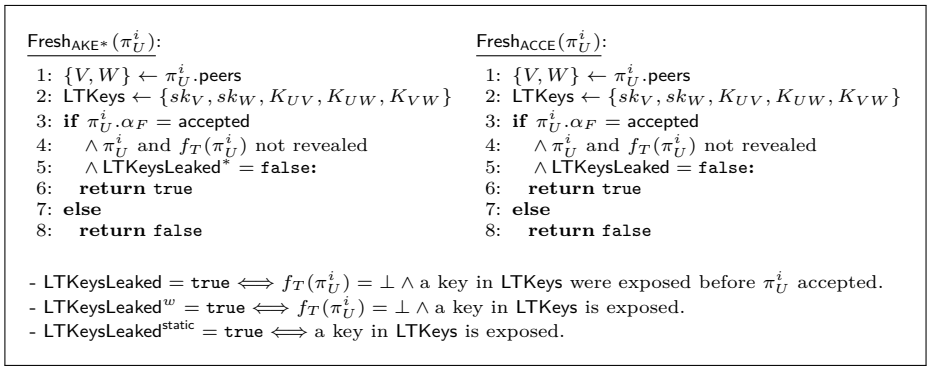

Fig. 3. Freshness predicates for security models $\mathrm{AKE}^* \in \{\mathrm{AKE}, \mathrm{AKE}^w, \mathrm{AKE}^{\mathsf{static}}\}$ and ACCE. The list LTKeys only contains actually existing long-term keys, e.g., if V is a responder party, then there is no corresponding private key sk_V.

AKE security. Besides soundness, a secure AKE protocol is supposed to provide secrecy of the distributed session keys. To capture this, the base query set $\mathcal{Q}_{base}$ is extended with the following query.

– $\mathsf{Test}(\pi_U^i)$: If $\pi_U^i.\alpha_F \neq \mathsf{accepted}$ or $U \in \mathcal{S}$, return $\perp$. Otherwise, draw a random bit b, and return π_U^i's session key if $b = 0$, or a random key if $b = 1$. We call π_U^i the *test-session* and the returned key the *challenge-key*. The Test query can only be made once.

Let $\mathcal{Q} = \mathcal{Q}_{base} \cup \{\mathsf{Test}\}$. Experiment $\mathsf{Exp}_{\Pi,\mathcal{Q},\mathcal{A}}(\lambda)$ stops when $\mathcal{A}$ outputs a bit b'. The goal of the adversary is to correctly guess the secret bit b used to answer the Test query. However, $\mathcal{A}$ is only given "credit" if the chosen test-session was *fresh*. A session is fresh if the adversary did not learn its session by trivial means, for example by revealing it or by impersonating its peers after having obtained their long-term keys etc. Formally, in Fig. 3, we specify three *freshness predicates* $\mathsf{Fresh}_{\mathsf{AKE}}$, $\mathsf{Fresh}_{\mathsf{AKE}^w}$, and $\mathsf{Fresh}_{\mathsf{AKE}^{\mathsf{static}}}$, of various permissiveness with respect to long-term key leakage. Each freshness predicate also give rise to a corresponding security notion AKE, AKE^w and $\mathrm{AKE}^{\mathsf{static}}$.

The AKE model is our "partner function analogue" of the standard eCK model (as defined in the updated version [21] of the original paper [22]), with the main difference being that we do not consider leakage of ephemeral values. In particular, the AKE model captures both key-compromise impersonation (KCI) attacks and forward secrecy. KCI attacks are captured since the test-session's own long-term private key can always be exposed by the adversary. Forward secrecy is captured since the adversary can, under certain conditions, learn the long-term keys of the peers of the test-session too. Specifically, the forward secrecy guarantees provided by the AKE model are rather strong: if a session has a partner, then the adversary is allowed to expose *any* long-term key it wants,

while if the session does not have a partner, then the adversary must wait until after the session accepted before it can expose the relevant keys. Note that partnering is used to model *passiveness* by the adversary in the test-session. Intuitively, even if the adversary knew all the long-term keys before the test-session started, if the test-session ends up with a partner, then the adversary cannot actually have exploited its knowledge of the keys.

Compared to the AKE model, the AKE^w model is more restrictive with respect to forward secrecy: if the test-session does not have partner, then the adversary is disallowed from exposing any of the relevant long-term keys. The AKE^w model is similar to the two-pass variant of the eCK model (see [21, Def. 3]). As mentioned in the introduction, standalone EAP does not achieve security in the AKE model, but we will show that it *is* secure in the AKE^w model.

Finally, the AKE$^{\mathsf{static}}$ model targets protocols that do not provide any forward secrecy, hence it disallows the adversary from exposing the long-term keys altogether (of course, the adversary is allowed to expose long-term keys unrelated to the test-session and its peers).

Definition 4 (Key-indistinguishability predicate). *Suppose π was the test-session chosen by $\mathcal{A}$ in a run of experiment $\mathsf{Exp}_{\Pi,\mathcal{Q},\mathcal{A}}(\lambda)$, b was the random bit used in answering the* Test *query, and suppose b' was the final output of $\mathcal{A}$. Define predicate $\mathsf{AKE}^* \in \{\mathsf{AKE}, \mathsf{AKE}^w, \mathsf{AKE}^{\mathsf{static}}\}$ as follows:*

$$\mathsf{AKE}^* \stackrel{\text{def}}{=} \begin{cases} b = b', & \text{if } \mathsf{Fresh}^*_{\mathsf{AKE}}(\pi) = \mathtt{true} \\ \mathtt{true} \text{ with probability } 1/2, & \text{if } \mathsf{Fresh}^*_{\mathsf{AKE}}(\pi) = \mathtt{false}. \end{cases} \tag{5}$$

Let $\mathsf{Exp}^{\mathsf{AKE}^*}_{\Pi,\mathcal{Q},\mathcal{A}}(\lambda) \Rightarrow 1$ denote the event that AKE^* evaluated to true.

Definition 5 (AKE security). *A protocol Π is* AKE*-secure*, if there exists a partnering function f, such that for all PPT adversaries $\mathcal{A}$,*

- $\mathsf{Adv}^{\mathsf{Sound}}_{\Pi,\mathcal{A},f}(\lambda) \stackrel{\text{def}}{=} 1 - \Pr[\mathsf{Exp}^{\mathsf{Sound}}_{\Pi,\mathcal{Q}\mathcal{A}}(\lambda) \Rightarrow 1]$ *is negligible in security parameter λ, and*
- $\mathsf{Adv}^{\mathsf{AKE}^*}_{\Pi,\mathcal{A},f}(\lambda) \stackrel{\text{def}}{=} |2 \cdot \Pr[\mathsf{Exp}^{\mathsf{AKE}^*}_{\Pi,\mathcal{Q},\mathcal{A}}(\lambda) \Rightarrow 1] - 1|$ *is negligible in security parameter λ,*

where AKE$^* \in \{\mathrm{AKE}, \mathrm{AKE}^w, \mathrm{AKE}^{\mathsf{static}}\}$.

2.3 (2P)-ACCE

Syntax. A (2P)-ACCE protocol is a two-party protocol as defined in Section 2.1, together with an associated *stateful authenticated encryption scheme (stAE)* $\mathsf{stE} = (\mathsf{st.Gen}, \mathsf{stE.Init}, \mathsf{stE.Enc}, \mathsf{stE.Dec})$ (following [20] [8]). Intuitively, an ACCE protocol is an amalgamation of an ordinary 2P-AKE protocol and a secure channel based on symmetric keys, where the session keys of the 2P-AKE protocol are used to key the secure channel.

Correctness of the stAE scheme demands that if the *deterministic* algorithm $\mathsf{st.Init}$ produced initial states st^0_E, st^0_D; and the ACCE session key k was used to

[8] For simplicity, we omit the properties of *length-hiding* and *associated data* in our treatment of ACCE. This omission is immaterial for the results established in this paper.

$\underline{\mathsf{Encrypt}(\pi, m_0, m_1)}$:

1: **if** $\pi.\alpha_F \neq \mathsf{accepted} \vee |m_0| \neq |m_1|$**:**
2: **return** $\perp$
3: $u \leftarrow u + 1$
4: $(C^0, st_{\mathrm{E}}^0) \leftarrow \mathsf{stE.Enc}(k, m_0, st_{\mathrm{E}})$
5: $(C^1, st_{\mathrm{E}}^1) \leftarrow \mathsf{stE.Enc}(k, m_1, st_{\mathrm{E}})$
6: $(\vec{C}[u], st_{\mathrm{E}}) \leftarrow (C^b, H, st_{\mathrm{E}}^b)$
7: **return** $\vec{C}[u]$

$\underline{\mathsf{Decrypt}(\pi, C)}$:

1: **if** $\pi.\alpha_F \neq \mathsf{accepted}$**:**
2: **return** $\perp$
3: $\pi' \leftarrow f_T(\pi)$
4: $v \leftarrow v + 1$;
5: $(m, st_{\mathrm{D}}) \leftarrow \mathsf{stE.Dec}(k, C, st_{\mathrm{D}})$
6: **if** $\pi' = \perp \vee v > \pi'.u \vee C \neq \pi'.\vec{C}[v]$**:**
7: `in-sync` $\leftarrow$ `false`
8: **if** `in-sync` $=$ `false`**:**
9: **return** m
10: **return** $\perp$

Fig. 4. The Encrypt and Decrypt queries for the ACCE security experiments. The variables $k, b, st_{\mathrm{E}}, st_{\mathrm{D}}, \vec{C}, u, v$ and `in-sync` all belong to the internal state of π. At the creation of every session π, a bit b is drawn uniformly at random from $\{0, 1\}$, st_{E} and st_{D} and are initialized by $\mathsf{stE.Gen}$, the list $\vec{C}$ is initialized to $\emptyset$, the counters u and v are set to 0, and `in-sync` is set to `true`.

produce a sequence of ciphertext/state pairs $(C_i, st_E^{i+1}) \leftarrow \mathsf{stE.Enc}(k, m_i, st_E^i)$ such that $C_i \neq \perp$ for all $i \geq 0$; then one must have, for all $i \geq 0$, that $m'_i = m_i$ in the sequence of decryptions $(m'_i, st_D^{i+1}) \leftarrow \mathsf{stE.Dec}(k, C_i, st_D^i)$.

ACCE security. To define security of an ACCE protocol, we extend the base query set $\mathcal{Q}_{base}$ with two additional queries, Encrypt and Decrypt, that allow the adversary to interact with the channels established in the protocol. The two queries are specified in Fig. 4.

Let $\mathcal{Q} = \mathcal{Q}_{base} \cup \{\mathsf{Encrypt}, \mathsf{Decrypt}\}$. Experiment $\mathsf{Exp}_{\Pi,\mathcal{Q},\mathcal{A}}(\lambda)$ stops when $\mathcal{A}$ outputs a pair (π, b'), consisting of a session π and a bit b'. The goal of the adversary, formally captured in the following predicate, is to break either the confidentiality or integrity of one of the channels established by a fresh session.

Definition 6 (ACCE predicates). *Consider a run of experiment* $\mathsf{Exp}_{\Pi,\mathcal{Q},\mathcal{A}}(\lambda)$*, and let* T *be the corresponding transcript. Suppose* (π, b') *was the final output by* $\mathcal{A}$*.*

- *Predicate* ACCE-int *is true if and only if, sometime during* $\mathsf{Exp}_{\Pi,\mathcal{Q},\mathcal{A}}(\lambda)$*,* $\mathcal{A}$ *made a* Decrypt *query that output something other than* $\perp$ *for a fresh session* π*.*
- *Predicate* ACCE-priv *is defined as follows:*

$$\mathsf{ACCE\text{-}priv} \stackrel{\text{def}}{=} \begin{cases} \pi.b = b', & \text{if } \mathsf{Fresh}(\pi) = \mathtt{true} \\ \mathtt{true} \text{ with probability } 1/2, & \text{if } \mathsf{Fresh}(\pi) = \mathtt{false}. \end{cases} \tag{6}$$

Let $\mathsf{Exp}^{\mathsf{ACCE\text{-}int}}_{\Pi,\mathcal{Q},\mathcal{A}}(\lambda) \Rightarrow 1$ *(resp.* $\mathsf{Exp}^{\mathsf{ACCE\text{-}priv}}_{\Pi,\mathcal{Q},\mathcal{A}}(\lambda) \Rightarrow 1$ *) denote the event that* ACCE-int *(resp.* ACCE-priv*) evaluated to true.*

Definition 7 (ACCE security). *A protocol* Π *is* ACCE-secure*, if there exists a partnering function* f*, such that for all PPT adversaries* $\mathcal{A}$*, the following are all negligible in the security parameter* λ*,*

- $\mathsf{Adv}^{\mathsf{Sound}}_{\Pi,\mathcal{A},f}(\lambda) \stackrel{\text{def}}{=} \Pr[\mathsf{Exp}^{\mathsf{Sound}}_{\Pi,\mathcal{Q},\mathcal{A}}(\lambda) \Rightarrow 1]$,
- $\mathsf{Adv}^{\mathsf{ACCE\text{-}int}}_{\Pi,\mathcal{A},f}(\lambda) \stackrel{\text{def}}{=} \Pr[\mathsf{Exp}^{\mathsf{ACCE\text{-}int}}_{\Pi,\mathcal{Q},\mathcal{A}}(\lambda) \Rightarrow 1]$,
- $\mathsf{Adv}^{\mathsf{ACCE\text{-}priv}}_{\Pi,\mathcal{A},f}(\lambda) \stackrel{\text{def}}{=} |2 \cdot \Pr[\mathsf{Exp}^{\mathsf{ACCE\text{-}priv}}_{\Pi,\mathcal{Q},\mathcal{A}}(\lambda) \Rightarrow 1] - 1|$.

Remark 6. Our definition of ACCE security is slightly different from the standard one introduced by Jager et al. [16]. Specifically, in the standard formulation of ACCE, the Decrypt oracle is *conditional*, meaning that if $\pi.b = 0$, then Decrypt always returns $\perp$ irregardless of whether the supplied ciphertext was a valid forgery or not. This is done in order to encode both the channel privacy and the channel integrity goal as a single distinguishing game. However, this makes proofs relying on ACCE security more cumbersome since the Decrypt query does not actually provide a proper decryption oracle. By casting ACCE channel security as two separate security goals, the Decrypt query becomes a proper decryption oracle. In the full version we prove that our definition of ACCE is equivalent with the standard one.

2.4 Explicit entity authentication

Explicit entity authentication is frequently considered one of the required security properties of a protocol. However, in this paper we will only prove/assume it for *some* protocols, because some of the protocols we consider simply cannot achieve it. The need for AKE protocols to provide explicit entity authentication has actually been somewhat disputed in the literature (see e.g. [8, §1.6], [27, §6] or [19, §1.2]). On the other hand, explicit entity authentication has always been part of the requirements of ACCE security [16,20,18]. Since the definition of entity authentication is formulated identically for both AKE and ACCE protocols, we give a merged definition here. Let $\mathcal{Q}_{\mathsf{AKE}}$ denote the query set of the AKE experiment, and let $\mathcal{Q}_{\mathsf{ACCE}}$ denote the query set of the ACCE experiment.

Definition 8 (Entity authentication predicate). *Let T be the transcript of experiment $\mathsf{Exp}_{\Pi,\mathcal{A},\mathcal{Q}_{\mathsf{X}}}(\lambda)$. Predicate* Auth *is true if and only if the following holds for all $T' \subseteq T$. For all fresh sessions π in T':*

$$\pi.\alpha = \mathsf{accepted} \implies \exists \pi' \text{ such that } f_{T'}(\pi) = \pi'. \tag{7}$$

Let $\mathsf{Exp}^{\mathsf{X\text{-}Auth}}_{\Pi,\mathcal{Q}_{\mathsf{X}},\mathcal{A}}(\lambda) \Rightarrow 1$ denote the event that Auth *is true, where $\mathsf{X} \in \{\mathsf{AKE}, \mathsf{ACCE}\}$.*

Definition 9 (Explicit entity authentication). *A protocol Π provides* explicit entity authentication *if there exists partner function f, such that for all PPT adversaries $\mathcal{A}$, it holds that*

1. *Π is X-secure, and*
2. *$\mathsf{Adv}^{\mathsf{X\text{-}EA}}_{\Pi,\mathcal{A},\mathcal{Q}}(\lambda) \stackrel{\text{def}}{=} 1 - \Pr[\mathsf{Exp}^{\mathsf{X\text{-}Auth}}_{\Pi,\mathcal{A},f}(\lambda) \Rightarrow 1]$ is negligible in security parameter λ,*

where $\mathsf{X} \in \{\mathsf{AKE}, \mathsf{AKE}^{w}, \mathsf{AKE}^{\mathsf{static}}, \mathsf{ACCE}\}$.

Remark 7. Note that the explicit entity authentication of an AKE (resp. ACCE) scheme needs to hold with the *same* partner function as used to prove its AKE (resp. ACCE) security.

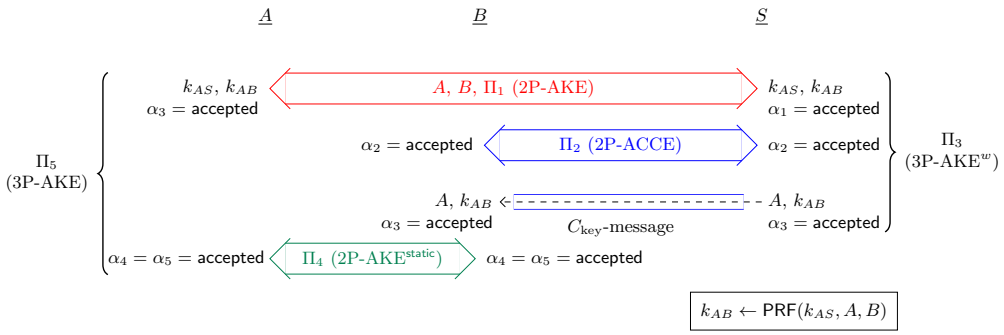

Fig. 5. (Right) Construction of a 3P-AKE^w-secure protocol Π_3, using as building blocks a 2P-AKE-secure protocol Π_1, an ACCE-secure protocol Π_2, and a pseudorandom function PRF. (Left) Construction of a 3P-AKE secure protocol Π_5, using as building blocks a 3P-AKE^w secure protocol Π_3 and a 2P-AKE$^{\text{static}}$-secure protocol Π_4.

3 Generic composition results

In this section we prove two composition theorems for two fairly generic constructions of 3P-AKE protocols. The first construction, shown as protocol Π_3 in Fig. 5, resembles the standalone EAP. It uses as building blocks any secure 2P-AKE protocol (in the strongest AKE model), any secure 2P-ACCE protocol, and a pseudorandom function for channel binding. The second construction, shown as protocol Π_5 in Fig. 5, resembles the EAP combined with a subsequent key-confirmation step, modeled here as a 2P-AKE protocol secure in the weakest AKE$^{\text{static}}$ model. We emphasize that the 3P-AKE protocol used as the underlying building block by protocol Π_5, does not necessarily have to be based on the Π_3 construction. Any 3P-AKE protocol secure according to the AKE^w model works. In Section 4 and Section 5, we will see how these generic constructions can be instantiated with EAP and IEEE 802.11 with upper-layer authentication, respectively.

3.1 2P-AKE + 2P-ACCE + channel binding $\Longrightarrow$ 3P-AKE^w

Construction. From a 2P-AKE protocol Π_1 (based on public keys), a 2P-ACCE protocol Π_2 (based on pre-shared symmetric keys), and a pseudorandom function PRF, we construct the 3P-AKE protocol Π_3 shown in Fig. 5. Specifically, protocol Π_3 works as follows. First, sub-protocol Π_1 is run between the initiator A and the trusted server S to derive an intermediate key k_{AS}. A also communicates the identities A and B to S, where B is the identity of responder that A wants to talk to. Note that A knows both S and B at the beginning of the protocol whereas S learns about B from the identities communicated by A. Technically, this means that a session at A needs to be initialized with the identities of S and B (setting the peers variable accordingly), while a session at S will update its peers variable to include B after receiving this identity from A.

From k_{AS}, both A and S derive the key $k_{AB} \leftarrow \mathsf{PRF}(k_{AS}, A, B)$. This key will be the ultimate session key shared between A and B in protocol Π_3. In order for S to transfer k_{AB} to B they establish a secure channel using sub-protocol Π_2. Once established, S sends the session key k_{AB} over the channel to B. Alongside k_{AB}, the server S also sends the identity of A to B (causing the receiving B to update its peers variable). For simplicity, we assume that the transfer of A and k_{AB} is done using a *single* channel message, which we call the C_{key} *message*. Note that the initiator A accepts in protocol Π_3 when it has derived k_{AB}, while the responder B accepts once it has received—and properly decrypted—the C_{key} message, finally obtaining k_{AB}.

Result. Our first composition result shows that protocol Π_3 is 3P-AKE^w-secure if sub-protocol Π_1 is 2P-AKE-secure, sub-protocol Π_2 is 2P-ACCE-secure, and PRF is a pseudorandom function. Note that since Π_3 does not provide explicit entity authentication—in fact, no initiator session A will have a partner at the time it accepts—it cannot achieve security in the strongest AKE model due to the attack mentioned for standalone EAP in the introduction.

Roughly, the proof of the first composition theorem works as follows. The 2P-AKE-security of sub-protocol Π_1 allows us to swap out the intermediate keys k_{AS} with random ones. The PRF-security of the function PRF then allows us to replace the derived session keys k_{AB} with random ones. Finally, the ACCE channel-privacy of sub-protocol Π_2 ensures that the adversary learns nothing about the session keys transfered in the C_{key} messages.

However, in order to make our proof work, we have to make one technical assumption on the partner function of sub-protocol Π_2. Namely, we have to assume that it is *symmetric*, meaning that $f_2(\pi) = \pi'$ implies $f_2(\pi') = \pi$. Note that this requirement is straightforwardly met by partner functions based on SIDs.

Theorem 1. *Let Π_3 be the protocol described in Section 3.1. If protocol Π_1 is 2P-AKE-secure, Π_2 is ACCE-secure using a symmetric partner function, and PRF is a secure PRF, then there exists a partner function f_3, such that protocol Π_3 is 3P-AKE^w-secure.*

Concretely, if Π_1 is AKE-secure with the partner function f_1, and Π_2 is ACCE-secure with the symmetric partner function f_2, then we can create a partner function f_3, and adversaries $\mathcal{B}_1, \ldots, \mathcal{B}_4$ and $\mathcal{D}$, such that

$$\begin{aligned}\mathsf{Adv}^{\mathsf{3P\text{-}AKE}^w}_{\Pi_3,\mathcal{A},f_3}(\lambda) \leq\ & \mathsf{Adv}^{\mathsf{ACCE\text{-}EA}}_{\Pi_2,\mathcal{B}_1,f_2}(\lambda) + \mathsf{Adv}^{\mathsf{ACCE\text{-}int}}_{\Pi_2,\mathcal{B}_2,f_2}(\lambda) \\ & + (n_\pi + 1)^2 \cdot |\mathcal{I} \cup \mathcal{R}|^2 \cdot \Big(\mathsf{Adv}^{\mathsf{ACCE\text{-}priv}}_{\Pi_2,\mathcal{B}_3,f_2}(\lambda) + \mathsf{Adv}^{\mathsf{2P\text{-}AKE}}_{\Pi_1,\mathcal{B}_4,f_1}(\lambda) + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{PRF},\mathcal{D}}(\lambda)\Big),\end{aligned} \tag{8}$$

where n_π is an upper bound on the number of sessions at each party.

Proof. We begin by defining the partner function f_3 using the partner functions for sub-protocols Π_1 and Π_2.

Defining the partner function for Π_3. Intuitively, f_3 is constructed by "composing" the two partner functions f_1 and f_2 assumed to exist for sub-protocols Π_1 and Π_2. For example, if π_A^i is an initiator session, then $f_3(\pi_A^i) = \pi_B^j$ if there exists a trusted server session π_S^k, such that $f_1(\pi_A^i) = \pi_S^k$ and $f_2(\pi_S^k) = \pi_B^j$, That is, π_B^j is π_A^i's f_3-partner if there exists a server session π_S^k that acts as the connection between them in the two sub-protocols Π_1 and Π_2.[9]

More detailed, when π_A^i is an initiator session having intended peers B (responder) and S (server), then:

- $f_{3,T_3}(\pi_A^i) = \pi_B^j$ if,
  1. $f_{1,T_1}(\pi_A^i) = \pi_S^k$ and $f_{2,T_2}(\pi_S^k) = \pi_B^j$,
  2. $\pi_B^j.\mathsf{peers} = \{A, S\}$,
  3. $\pi_S^k.\mathsf{peers} = \{A, B\}$ (in particular, this means that π_S^k received the same identities that π_A^i sent on the A-S link Fig. 5),
- $f_{3,T_3}(\pi_A^i) = \perp$, otherwise.

When π_B^j is a responder session having intended peers A and S, then f_3 is defined similarly by "reversing" the order of f_1 and f_2:

- $f_{3,T_3}(\pi_B^j) = \pi_A^i$ if,
  1. $f_{2,T_2}(\pi_B^j) = \pi_S^k$ and $f_{1,T_1}(\pi_S^k) = \pi_A^i$;
  2. $\pi_A^i.\mathsf{peers} = \{B, S\}$,
  3. $\pi_S^k.\mathsf{peers} = \{A, B\}$,
- $f_{3,T_3}(\pi_B^j) = \perp$, otherwise.

Soundness. The soundness of f_3 follows from the soundness of f_1 and f_2, the ACCE-security of protocol Π_2 (specifically, its channel integrity), together with the fact that PRF is deterministic. The proof is given in the full version.

AKE^w-security. The proof of AKE^w-security of protocol Π_3 is structured as a sequence of games. In the following, when we say that a certain game *aborts*, we mean that the challenger stops the execution of the experiment and outputs a random bit on $\mathcal{A}$'s behalf.

Game 0: This is the real 3P-AKE^w security game, hence

$$\mathsf{Adv}^{\mathsf{G}_0}_{\Pi_3,\mathcal{A},f_3}(\lambda) = \mathsf{Adv}^{\mathsf{3P\text{-}AKE}^w}_{\Pi_3,\mathcal{A},f_3}(\lambda) \ .$$

Game 1: This game proceeds as the previous one, but aborts if a fresh responder or trusted server session *accepts maliciously* in sub-protocol Π_2, meaning that it accepted without a partner in Π_2 according to f_2.

Lemma 1. $\mathsf{Adv}^{\mathsf{G}_0}_{\Pi_3,\mathcal{A},f_3}(\lambda) \leq \mathsf{Adv}^{\mathsf{G}_1}_{\Pi_3,\mathcal{A},f_3}(\lambda) + \mathsf{Adv}^{\mathsf{ACCE\text{-}EA}}_{\Pi_2,\mathcal{B}_1,f_2}(\lambda)$.

[9] Technically, to make this formally precise, one needs to extract from the 3P-AKE transcript T two transcripts T_1 and T_2, containing the queries pertaining to the two-party sub-protocols Π_1 and Π_2, respectively, so that running f_1 and f_2 on them is well-defined. The details are provided in the full paper.

Proof (sketch). Reduction $\mathcal{B}_1$ begins by creating all the long-term keys for sub-protocol Π_1 and selecting a random bit b. Essentially, $\mathcal{B}_1$ will simulate the Π_1 part of Π_3 itself, while forwarding all messages pertaining to Π_2 to its 2P-ACCE challenger. In particular, $\mathcal{B}_1$ creates all the intermediate keys k_{AS} itself, and from them derive the session keys k_{AB}. In order to create the C_{key} message of some trusted server session π, $\mathcal{B}_1$ issues an $\mathsf{Encrypt}(\pi, k_{AB}, k_{AB})$ query to its own ACCE experiment. Moreover, when $\mathcal{A}$ issues a Test query, then depending on bit b, $\mathcal{B}_1$ returns the real session key or a random key. When $\mathcal{A}$ terminates, then $\mathcal{B}_1$ terminates too (in this case no malicious accept has occurred).

To analyze $\mathcal{B}_1$'s winning probability, we only have to observe that $\mathcal{B}_1$ provides a perfect simulation of Π_3 for $\mathcal{A}$. This means that if a malicious accept occurs in sub-protocol Π_2, then a malicious accept also occurs in $\mathcal{B}_1$'s ACCE experiment. □

Remark 8. Note that the abort condition in Game 1 does not mean that every session in protocol Π_3 will have a partner (according to f_3). In fact, all the initiator sessions in protocol Π_3 will accept without a partner.

Game 2: This game proceeds as the previous one, but it aborts if a fresh responder session accepts on receiving a C_{key} message that was not legitimately produced by its partner in Π_2.

Lemma 2. $\mathsf{Adv}^{\mathsf{G}_1}_{\Pi_3,\mathcal{A},f_3}(\lambda) \leq \mathsf{Adv}^{\mathsf{G}_2}_{\Pi_3,\mathcal{A},f_3}(\lambda) + \mathsf{Adv}^{\mathsf{ACCE\text{-}int}}_{\Pi_2,\mathcal{B}_2,f_2}(\lambda)$.

Proof (sketch). $\mathcal{B}_2$ works exactly like algorithm $\mathcal{B}_1$ in the previous proof, but it also simulates the abort on malicious accept. This simulation is possible because the partnering function f_2 is based on the *public* transcript T_2. It only remains to argue that the new abort event of Game 2 implies a forgery in $\mathcal{B}_2$'s ACCE experiment. This amounts to showing that if a session in Π_3 is fresh according to $\mathsf{Fresh}_{\mathsf{AKE^w}}$, then the corresponding session in Π_2 is fresh according to $\mathsf{Fresh}_{\mathsf{ACCE}}$. But this is true because the $\mathsf{Fresh}_{\mathsf{AKE^w}}$ predicate is more restrictive than the $\mathsf{Fresh}_{\mathsf{ACCE}}$ predicate. □

Game 3: In this game the challenger tries to guess the test-session chosen by $\mathcal{A}$, together with its eventual partner (if any). If the guess is wrong, or if $\mathcal{A}$ violates the freshness of the guessed test-session, the challenger aborts with a random output. Technically, the challenger proceeds as follows.

For $m \leq n$, let $[m,n] \stackrel{\text{def}}{=} \{m, m+1, \ldots n\}$. First, the challenger randomly guesses the test-session $(U,i) \leftarrow\!\$\, (\mathcal{I} \cup \mathcal{R}) \times [1, n_\pi]$, where n_π is an upper bound on the number of sessions at each party. Then, depending on the role of U, the challenger either guess $(V,j) \leftarrow\!\$\, \mathcal{I} \times [0, n_\pi]$ or $(V,j) \leftarrow\!\$\, \mathcal{R} \times [0, n_\pi]$ as the expected partner of (U,i), where a pick of $j = 0$ means that (U,i) is not expected to get any partner session at its peer V. Finally, the challenger aborts by outputting a random bit if either of the following bad event occurs:

(i) (U,i) was not selected as the test-session by $\mathcal{A}$.
(ii) (U,i) was guessed to be without a partner, but gets one.

(iii) (U, i) was guessed to have a partner, but either gets none or someone different from (V, j).
(iv) $\mathcal{A}$ makes a Reveal or $\mathsf{Corrupt}$ query that would make (U, i) unfresh.

Lemma 3.

$$\mathsf{Adv}^{\mathsf{G}_2}_{\Pi_3,\mathcal{A},f_3}(\lambda) \leq (n_\pi + 1)^2 \cdot |\mathcal{I} \cup \mathcal{R}|^2 \cdot \mathsf{Adv}^{\mathsf{G}_3}_{\Pi_3,\mathcal{A},f_3}(\lambda). \tag{9}$$

Proof. The occurrence of the bad events is independent from $\mathcal{A}$'s view up to the moment of where the bad event occurs. When none of the bad events occurs, then $\mathcal{A}$'s success probability is the same in G_2 and G_3, and the challenger guesses the right (pair of) session(s) with probability at least $1/\left((n_\pi + 1) \cdot |\mathcal{I} \cup \mathcal{R}|\right)^2$. And if a bad event occurs, then $\mathcal{A}$ wins G_3 with probability at least $1/2$. □

In the remaining games, let $\pi^* = \pi^i_U$ denote the guessed test-session, and let $\pi' = \pi^j_V$ denote its expected partner. Define the *co-partner* of π^* to be the trusted server session being involved in the protocol run between π^* and π'. Specifically, if π^* is an initiator, then its co-partner is defined to be $f_{1,T_1}(\pi^*)$; while if π^* is a responder, then its co-partner is defined to be $f_{2,T_2}(\pi^*)$.

Game 4: This game proceeds as the previous one, except that it swaps out the intermediate key k_{AS} derived in sub-protocol Π_1 with a random key for the guessed initiator session (either π^* or π') and its co-partner (if any).

Lemma 4. $\mathsf{Adv}^{\mathsf{G}_3}_{\Pi_3,\mathcal{A},f_3}(\lambda) \leq \mathsf{Adv}^{\mathsf{G}_4}_{\Pi_3,\mathcal{A},f_3}(\lambda) + \mathsf{Adv}^{\mathsf{2P\text{-}AKE}}_{\Pi_1,\mathcal{B}_3,f_1}(\lambda)$.

Proof (sketch). Reduction $\mathcal{B}_3$ begins by drawing a random bit b and creates all the long-term PSKs for sub-protocol Π_2. It also guesses the sessions π^* and π' as in Game 3. $\mathcal{B}_3$ then runs $\mathcal{A}$ and forwards all of its queries pertaining to sub-protocol Π_1 to its own AKE experiment, while all queries pertaining to sub-protocol Π_2 reduction $\mathcal{B}_3$ answers itself using the PSKs it created. It also implements all the abort conditions of the previous games. To answer $\mathcal{A}$'s $\mathsf{Test}(\pi^*)$ query, $\mathcal{B}_3$ does the following. If $b = 1$ then it responds with a random key as normal. If $b = 0$ and π^* is an initiator session, then $\mathcal{B}_3$ forwards $\mathcal{A}$'s $\mathsf{Test}(\pi^*)$ query to its own AKE game to obtain π^*'s intermediate key k_{AS} in sub-protocol Π_1. $\mathcal{B}_3$ then uses k_{AS} to derive the session key k_{AB} which it returns to $\mathcal{A}$. If $b = 0$ and π^* is a responder session, then by our abort conditions, π^* must have a co-partner π^k_S by Game 1. To obtain the intermediate key k_{AS} needed to derive k_{AB}, $\mathcal{B}_3$ queries $\mathsf{Test}(\pi^k_S)$ to its own AKE experiment and returns k_{AB} to $\mathcal{A}$. When $\mathcal{A}$ outputs its guess b', then $\mathcal{B}_3$ stops and outputs 0 if $b = b'$, and 1 otherwise.

Note that if the test-query in $\mathcal{B}_3$'s own AKE experiment returns real keys k_{AS}, then $\mathcal{B}_4$ perfectly simulates Game 3, while if it returns random keys then $\mathcal{B}_3$ simulates Game 4. However, we still need to argue that the test-session chosen in $\mathcal{B}_3$'s experiment is fresh. If π^* is an initiator session then $\mathcal{B}_3$ also uses π^* as the test-session in its own AKE experiment, hence it is fresh since the predicate $\mathsf{Fresh}_{\mathsf{AKE}^w}$ is more restrictive than $\mathsf{Fresh}_{\mathsf{AKE}}$. If π^* is a responder session, then

the test-session chosen by $\mathcal{B}_3$ is π^*'s co-partner π_S^k, so we need to argue that π_S^k is fresh in $\mathcal{B}_3$'s AKE experiment. There are two cases to consider: either π^* has an f_3-partner or it does not. If π^* does have a partner (which by Game 3 must be π'), then $\mathcal{A}$ cannot have made any $\mathsf{Reveal}(\pi')$ queries since this would violate the AKE^w-freshness of π^*. Moreover, since f_3 is constructed from f_1 and f_2, π' must be π_S^k's f_1-partner. Thus, $\mathcal{B}_3$ is also allowed to forward any $\mathsf{Corrupt}$ query to either A or S without violating the freshness of π_S^k according to $\mathsf{Fresh}_{\mathsf{AKE}}$. If π^* does not have an f_3-partner, then $\mathcal{A}$ cannot have made any $\mathsf{Corrupt}$ query to A or S (since this would violate AKE^w-freshness), and thus neither has $\mathcal{B}_3$. Moreover, if π^* does not have an f_3-partner then in particular its co-partner π_S^k cannot have an f_1-partner. Thus, $\mathcal{B}_3$ can safely forward all of $\mathcal{A}$'s Reveal queries without violating the AKE-freshness of π_S^k. □

Game 5: This game proceeds as the previous one, except that when deriving the session key k_{AB} for the guessed initiator session (either π^* or π') and its co-partner (if it exists), the challenger uses a random function $\$(\cdot,\cdot)$ rather than $\mathsf{PRF}(k_{AS},\cdot,\cdot)$.

Lemma 5. $\mathsf{Adv}^{\mathsf{G}_4}_{\Pi_3,\mathcal{A},f_3}(\lambda) \leq \mathsf{Adv}^{\mathsf{G}_5}_{\Pi_3,\mathcal{A},f_3}(\lambda) + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{PRF}}(\mathcal{D})$.

Proof. Algorithm $\mathcal{D}$ has access to an oracle $\mathcal{O}$ which either implements the function $\mathsf{PRF}(\widetilde{k},\cdot,\cdot)$ with an independent and uniformly distributed key $\widetilde{k}$, or a random function $\$(\cdot,\cdot)$. $\mathcal{D}$ begins by drawing a random bit b and creates all the long-term keys for sub-protocols Π_1 and Π_2. Next, it runs $\mathcal{A}$ and answers all its queries according to Game 4 by using the keys it created, except that it answers $\mathcal{A}$'s $\mathsf{Test}(\pi^*)$ query as follows. If $b = 1$, then $\mathcal{D}$ returns a random key. If $b = 1$, then $\mathcal{D}$ answers as follows. If π^* is an initiator session, then $\mathcal{D}$ answers with $\mathcal{O}(U,V)$ (recall that $\pi^* = \pi_U^i$ and $\pi' = \pi_V^j$). If π^* is a responder session, then $\mathcal{D}$ answers with $\mathcal{O}(V',U')$, where V' and U' were the identities that the co-partner of π^* received over the initiator-server link in Fig. 5 (recall that if π^* is a responder session it is guaranteed to have a co-partner by Game 1). When $\mathcal{A}$ outputs its guess b', then $\mathcal{D}$ stops and outputs 0 if $b = b'$, and 1 otherwise.

When $\mathcal{D}$'s oracle $\mathcal{O}$ implements PRF, then $\mathcal{D}$ perfectly simulates Game 4, while if $\mathcal{O}$ implements a random function $\$(\cdot,\cdot)$, then $\mathcal{D}$ perfectly simulates Game 5. Thus, the advantage difference of $\mathcal{A}$ winning in Game 4 and Game 5 corresponds exactly to the probability difference that $\mathcal{D}$ outputs 1 when interacting with PRF or a random function $\$(\cdot,\cdot)$ as its oracle $\mathcal{O}$. □

Note that by the change in Game 5, the session key of π^* and π' is derived using a random function rather then the pseudorandom function PRF. In the following, let $\widetilde{k_{AB}}$ denote the session key derived in this manner at the co-partner of π^* (if it exists).

Game 6: This game proceeds as the previous one, but when creating the C_{key} message of the co-partner of π^*, the challenger encrypts the "dummy" string 0^λ instead of the session key $\widetilde{k_{AB}}$. If this C_{key} message is eventually delivered to

the intended responder session (either π^* or π'), then its session key is still set to $\widetilde{k_{AB}}$ however.

Lemma 6. $\mathsf{Adv}^{\mathsf{G}_5}_{\Pi_3,\mathcal{A},f_3}(\lambda) \leq \mathsf{Adv}^{\mathsf{G}_6}_{\Pi_3,\mathcal{A},f_3}(\lambda) + \mathsf{Adv}^{\mathsf{ACCE\text{-}priv}}_{\Pi_2,\mathcal{B}_4,f_2}(\lambda)$.

Proof (sketch). Reduction $\mathcal{B}_4$ begins by drawing a random bit $b$and creates all the long-term keys for sub-protocol Π_1. It also guesses the sessions π^* and π' as in Game 3, and implements all of the abort conditions introduced so far. All of $\mathcal{A}$'s queries pertaining to sub-protocol Π_1 $\mathcal{B}_4$ answers itself using the long-term keys it created, while queries pertaining to sub-protocol Π_2 $\mathcal{B}_4$ forwards to its own ACCE experiment. In particular, $\mathcal{B}_4$ creates the C_{key} message of a server session π^k_S as follows.

If π^k_S is not the co-partner of the test-session π^*, then $\mathcal{B}_4$ makes the query $\mathsf{Encrypt}(\pi^k_S, A\|k_{AB}, A\|k_{AB})$ to its ACCE experiment, where "A" is the identity of the initiator that π received on the A-S link in Fig. 5, and k_{AB} is the session key $\mathcal{B}_4$ derived from π's intermediate key k_{AS} in sub-protocol Π_1. The returned ciphertext is used as the C_{key} message of π^k_S. If π *is* the co-partner of π^*, then $\mathcal{B}_4$ instead makes the query $\mathsf{Encrypt}(\pi^k_S, A\|k_{AB}, A\|0^\lambda)$ to create C_{key}.

Finally, when $\mathcal{A}$ outputs its guess b', then $\mathcal{B}_4$ outputs the following to its ACCE experiment. If the test-session π^* has a co-partner π^k_S, then $\mathcal{B}_4$ outputs $(\pi^k_S, 0)$ if $b = b'$ and $(\pi^k_S, 1)$ otherwise. If the test-session does not have a co-partner, then $\mathcal{B}_4$ simply outputs an arbitrary session together with a random bit.

Note that if the test-session does not have a co-partner then there is no difference between Game 5 and Game 6, and $\mathcal{B}_4$ perfectly simulates it. If the test-session has a co-partner π^k_S, and $\pi^k_S.b = 0$ in $\mathcal{B}_4$'s ACCE experiment, then $\mathcal{B}_4$ perfectly simulates Game 5 (since the C_{key} message of π^k_S is an encryption of the actual session key k_{AB}). On the other hand, if $\pi^k_S.b = 1$ then $\mathcal{B}_4$ perfectly simulates Game 6 (since the C_{key} message of π^k_S is an encryption of 0^λ). What remains to show that π^k_S is fresh in $\mathcal{B}_4$'s ACCE experiment, i.e., that π^k_S is fresh according to predicate $\mathsf{Fresh}_{\mathsf{ACCE}}$.

Suppose first that the test-session π^* is a responder. This is where we will use the assumption that the partner function f_2 for sub-protocol Π_2 is symmetric. By Game 1 π^* has a co-partner $f_2(\pi^*) = \pi^k_S$, and by the symmetry of f_2 we also have $f_2(\pi^k_S) = \pi^*$. It follows that π^k_S is fresh according to $\mathsf{Fresh}_{\mathsf{ACCE}}$ (note that since $\mathcal{B}_4$ makes no Reveal query to π^k_S in its ACCE experiment, we only have to consider the exposure of its PSK).

Now suppose the test-session is an initiator. There are two cases to consider: either π^* has an f_3-partner or it does not have an f_3-partner. If π^* has an f_3-partner π', then by the construction of f_3 from f_1 and f_2, we have in particular that $f_2(\pi^k_S) = \pi'$. Again, this implies that π^k_S is fresh according to $\mathsf{Fresh}_{\mathsf{ACCE}}$. Conversely, if π^* does not have an f_3-partner, then none of the long-term keys and PSKs of its peers can be exposed if π^* is to be fresh according to $\mathsf{Fresh}_{\mathsf{AKE^w}}$. In particular, this means that the long-term PSK of π^k_S must be unexposed. Thus, π^k_S is fresh according to $\mathsf{Fresh}_{\mathsf{ACCE}}$ (this is regardless of whether it has an f_2-partner or not). □

Concluding the proof of Theorem 1. We argue that $\mathsf{Adv}^{\mathsf{G}_6}_{\Pi_3,\mathcal{A},f_3}(\lambda) = 0$. By the change in Game 5, the session key of the test-session π^* is derived using a random function $\$(A, B)$, where "$A$" and "$B$" are the identities of the initiator and responder that π^* believes took part in this protocol run. We claim that the only other session that holds a session key derived from $\$(\cdot,\cdot)$ using the same identities "A" and "B", is π^*'s partner π' (if it exists).

First, note that the random function is evaluated for at most two sessions: one initiator session and one server session. Second, the session key derived by the server session is delivered to at most one responder session. Finally, the identities used to evaluate $\$(\cdot,\cdot)$ at the initiator and server might be different since the adversary can modify the communicated identities at the A-S link in Fig. 5.

However, if the adversary modifies these identities, then the initiator and server derive independent keys, which ultimately means that the initiator and responder will have independent keys too. Moreover, the initiator and responder sessions will not be partners since the communicated identities at the S-B link in Fig. 5 will be different too (recall that f_3-partnering includes the sessions' recorded peers, and by Game 2 the adversary is unable to change the C_{key} message). On the other hand, if the identities were the same, then the initiator and responder session would necessarily be f_3-partners. This follows because the initiator has the server session as its co-partner (in sub-protocol Π_1), and the server session's C_{key} message is only delivered to *its* co-partner (in sub-protocol Π_2). Combined with their agreement on their peers, this means that they would be partners by the definition of f_3.

Altogether, since the session key of the test-session is derived using an independent random function, and since the corresponding C_{key} message leaks nothing about the session key by Game 6, the adversary has zero advantage in Game 6 as claimed. Combining all the lemmas yields the theorem. □

Note that the conclusion above only holds because of the channel binding. In particular, if the identities of A and B did not go into to the evaluation of the pseudorandom function PRF, then Π_3 would be vulnerable to a simple UKS attack: just change the responder identity sent over the (unauthenticated) A-S link from B to B'. Without channel binding, A and B' obtain the same session key but disagree on their intended peers.

3.2 3P-AKE^w + 2P-AKE $\implies$ 3P-AKE

Construction. From a 3P-AKE protocol Π_3 and a 2P-AKE protocol Π_4, we construct the 3P-AKE protocol Π_5 shown in Fig. 5. Specifically, protocol Π_5 works as follows. First, sub-protocol Π_3 is run between A, B and S in order to establish an intermediate "session key" K_{Π_3}. Then, sub-protocol Π_4 is run between A and B using K_{Π_3} as the their shared "long-term key". The session key derived in Π_4 becomes A and B's final session key in Π_5.

Result. Our second composition result shows that protocol Π_5 is 3P-AKE-secure if sub-protocol Π_3 is 3P-AKE^w-secure and sub-protocol Π_4 is 2P-AKE$^{\mathsf{static}}$-secure with explicit entity authentication. We remark that the last requirement is necessary in order for our proof to go through. In fact, Π_5 inherits the property of explicit entity authentication from sub-protocol Π_4. On the other hand, while Π_4 does not achieve forward secrecy on its own, protocol Π_5 does. The reason is that within Π_5, sub-protocol Π_4 is merely used to upgrade the security of Π_3, which does provide forward secrecy (albeit limited).

Theorem 2. *Let Π_5 be the protocol described in Section 3.2. If protocol Π_3 is 3P-AKE^w-secure and protocol Π_4 is 2P-AKE$^{\mathsf{static}}$-secure with explicit entity authentication, then there exists a partner function f_5 such that protocol Π_5 is 3P-AKE-secure.*

Concretely, for partner functions f_3 and f_4, we can create a partner function f_5, and adversaries $\mathcal{B}_1$, $\mathcal{B}_2$ and $\mathcal{B}_3$, such that

$$\begin{aligned}\mathsf{Adv}^{\mathsf{3P\text{-}AKE}}_{\Pi_5,\mathcal{A},f_5}(\lambda) \leq (n_\pi+1)^2 \cdot |\mathcal{I}\cup\mathcal{R}|^2 \cdot \left(2 \cdot \mathsf{Adv}^{\mathsf{3P\text{-}AKE}^w}_{\Pi_3,\mathcal{B}_1,f_3}(\lambda) + \mathsf{Adv}^{\mathsf{2P\text{-}AKE}^{\mathsf{static}}}_{\Pi_4,\mathcal{B}_2,f_4}(\lambda)\right) \\ + (n_\pi+1)^2 \cdot |\mathcal{I}\cup\mathcal{R}|^2 \cdot \mathsf{Adv}^{\mathsf{2P\text{-}AKE}^{\mathsf{static}}\text{-}\mathsf{EA}}_{\Pi_4,\mathcal{B}_3,f_4}(\lambda)\end{aligned} \tag{10}$$

where n_π is an upper bound on the number of sessions at each party.

The proof of Theorem 2 is very similar to that of Theorem 1 and is provided in the full version.

4 Security of EAP

4.1 EAP with channel binding

In this section we explore the security guarantees provided by EAP. As mentioned in the introduction, there is no single definitive version of EAP which we can use for this purpose, because the specification itself (RFC 3748 [4]) leaves many of its components undefined. Thus, any analysis of EAP will have to make assumptions on these components.

In Theorem 1, let us identify sub-protocol Π_1 with the EAP method run between the client and the trusted server. Let sub-protocol Π_2 be the key-transport protocol run between the server and the authenticator. Finally, suppose that EAP employs the channel binding mechanism defined in [25]. Then we immediately get the following result for EAP.

Theorem 3 (3P-AKE^w security of EAP). *If the chosen EAP method used within EAP is 2P-AKE-secure, the key-transport protocol is 2P-ACCE-secure, and the employed key derivation function is a secure PRF that provides channel binding on the client's and authenticator's identities, then EAP is 3P-AKE^w-secure.*

To be even more concrete, we can also instantiate sub-protocols Π_1 and Π_2 with some actual protocols. For example, Brzuska et al. [12] recently showed that the EAP-TLS method constitutes a secure 2P-AKE protocol, thus satisfying the requirements on sub-protocol Π_1. For sub-protocol Π_2 we take RADIUS-over-TLS [30], which then reduces to the security of TLS. Multiple papers [16,20,18,23,9] have shown TLS to be a secure 2P-ACCE protocol. Hence, RADIUS-over-TLS fulfills the requirement on sub-protocol Π_2.

4.2 Channel-binding scope

In Theorem 1, and 3, we assumed that the channel binding mechanism included the identity of the client and the authenticator in order to bind the identities cryptographically to the session key. Implicitly, this also assumes that all identities are globally unique and belong to the same namespace. This is a standard assumption when doing cryptographic modeling. However, in reality, the various links in EAP take place over different types of communication media with different types of identities and addressing schemes. For instance, in IEEE 802.11 with upper-layer authentication, the communication between the client and the access point is based on link-layer addresses, the communication between the client and the server is typically based on usernames (client) and domain names (server), while the communication between the server and the access point might be based solely on IP addresses. Mapping between these identifiers is not always straightforward (see [15]). In fact, some of the identifiers might not even be available to all the protocol participants. Specifically, since the communication between the client and the access point happens at the link-layer, the IP addresses used by the access point towards the server might not be available to the client unless the access point broadcasts it. In practice, most link-layer protocols have facilities for providing this kind of information to the client[10], but there is no guarantee that the authenticator will actually provide it.

Moreover, in some settings this information may not even be relevant. For example, in a WLAN supported by many access points, the client might not care about *which* specific access point it connects to, as long as it connects to a legitimate access point of that WLAN. Thus, in this case the granularity of the channel-binding should not be at the individual access point level, but rather at the WLAN level, defined by all the access points broadcasting the same network identifier (SSID). However, in this case the security guarantees provided by the channel-binding will be weaker. Specifically, when channel-binding occurs at the individual level, then the corruption of a single access point will not influence clients connecting to access points having a *different* identity. On the other hand, when channel-binding occurs at the network level, then a single corrupted access point will affect *all* connections within that network. In this case, the channel binding only protects connections occurring in networks having a different SSID.

More generally, the information included in the channel-binding defines the scope of the protection it provides, and can include more than just identities. For instance, physical media types, data rates, cost-information, channel frequencies, etc., can all be used as input to the channel-binding. The specifications for channel-binding within EAP [25,14] leaves open exactly the kind of information that should go into the binding, because the amount of information that will be available to both the client and the server can vary.

[10] For instance, the `Identity` type field in EAP `Request` messages are often "piggy-backed" by layer 2 protocols (like EAPOL/802.1X [1]) to include this information.

4.3 EAP without channel binding

Without channel binding, it suffices to compromise a single access point in order to compromise an entire network. As access points are typically not highly protected devices, this is a substantial attack vector on enterprise networks. Even if the channel binding only included the network name, it would clearly be an upgrade over EAP without channel binding, and comes at essentially no cost. The situation in the AKA protocol used in the UMTS and LTE mobile networks is similar. The AKA protocol is similarly structured as the EAP protocol[11], where a mobile client that wants to connect to a base station first has to authenticate to its home operator. So-called *authentication vectors*, which in particular includes a session key, are then forwarded from the operator to the base station in much the same way as the server forwards the session key to the authenticator in EAP. Moreover, similar to many EAP methods, the AKA protocol too lacks channel-binding for its authentication vectors. In their recent analysis of the AKA protocol, Alt et al. [5] noted (Section 5) this lack of channel-binding, and suggested a fix identical to the key-derivation approach analyzed in this paper.

5 Security of IEEE 802.11

5.1 Description of the IEEE 802.11 protocol

IEEE 802.11 [2] is the most widely used standard for creating WLANs. It supports three modes of operation depending on the network topology: infrastructure mode, ad-hoc mode, and mesh network mode. In ad-hoc mode and mesh-networking mode there is no central infrastructure, and the wireless *clients* talk directly to each other. On the other hand, in infrastructure mode the clients only communicate through an *access point (AP)*, which provides connectivity to a larger WAN. In this paper we only cover IEEE 802.11 in infrastructure mode, which is by far the most common mode.

The IEEE 802.11 protocol is a layer 2 protocol, aiming to secure the wireless link between the client and the AP. It defines two main security protocols: the *4-Way-Handshake (4WHS)*, used to authenticate and establish session keys between the client and the AP; and the *Counter Mode CBC-MAC protocol (CCMP)*, used to secure the actual application data. We will only cover the 4WHS in this paper.

The 4WHS is based on a symmetric *Pairwise Master Key* (PMK), shared between the client and the AP. The analysis of IEEE 802.11 will therefore crucially depend on how this PMK is obtained. In Section 5.2 we will analyze the 4WHS when the PMK is simply taken for granted, i.e., the PMK is a pre-shared key. This is already quite significant on its own because it corresponds to the setting found in virtually every wireless home-network. Still, in most enterprise and university environments, the PMK is not a pre-shared key, but is rather distributed to the client and AP through some upper-level authentication mechanism involving a mutually trusted server. While technically outside the scope

[11] In fact, EAP is widely used within mobile networks.

of the IEEE 802.11 standard, the de-facto protocol for this is EAP. The analysis of IEEE 802.11 with upper-level authentication is the topic of Section 5.3.

5.2 Analyzing the 4-Way-Handshake

The 4WHS is shown in Figure 6. It depends on a pseudorandom function PRF and a MAC scheme $\Sigma = (\mathsf{kg}, \mathsf{MAC}, \mathsf{Vrfy})$. Identities are based on the parties' 48-bit link-layer addresses. This makes it possible to compare the parties' identities based on their corresponding numerical values. Particularly, the functions $\max\{A, B\}$ and $\min\{A, B\}$ returns, respectively, the largest and the smallest of two link-layer addresses A and B. We use the notation $[x]_k \stackrel{\text{def}}{=} x \| \sigma$ to denote a message x together with its MAC tag σ, computed with $\Sigma.\mathsf{MAC}$ and key k.

The 4WHS begins with the AP sending the message $m_1 = \eta_{AP} \| p_1$ to the client C, where η_{AP} is a nonce and p_1 is some auxiliary information included in the IEEE 802.11 packet.

On receiving m_1, C generates its own nonce η_C and derives a key $\text{PTK} = k_\mu \| k_\alpha \leftarrow \mathsf{PRF}_K(P \| \eta)$ using the pseudorandom function PRF and the long-term key it shares with AP. Here $P \| \eta = \min\{AP, C\} \| \max\{AP, C\} \| \min\{\eta_{AP}, \eta_C\} \| \max\{\eta_{AP}, \eta_C\}$. The sub-key k_α will be the session key output by the 4WHS, while k_μ will be used by the MAC scheme Σ to protect the handshake messages. After deriving PTK, C creates and sends the next protocol message $m_2 = [\eta_C \| p_2]_{k_\mu}$.

On receiving $m_2 = [\eta_C \| p_2]_{k_\mu}$, AP uses the containing nonce η_C to derive the keys $\text{PTK} = k_\mu \| k_\alpha \leftarrow \mathsf{PRF}_K(P \| \eta)$. Using k_μ as the key, it verifies the integrity of m_2 with the MAC scheme $\Sigma.\mathsf{Vrfy}$. If the verification goes through, AP creates and send the third protocol message $m_3 = [\eta_{AP} \| p_3]_{k_\mu}$.

On receiving m_3, C first verifies it using the MAC key k_μ. If the check goes through, it sends out the final handshake message $m_4 = [p_4]_{k_\mu}$. Additionally, it sets its own acceptance state to $\alpha = \mathsf{accepted}$. Once AP receives and verifies m_4, it sets its acceptance status to $\alpha = \mathsf{accepted}$ too.

Remark 9. The fourth handshake message m_4 serves no cryptographic purpose and could safely have been omitted. However, to stay true to the actual 4WHS, we leave it in.

In the following analysis, let $\mathcal{P}_{AP} = \mathcal{I}$ and $\mathcal{P}_C = \mathcal{R}$, i.e., in the 4WHS protocol APs are the initiators and the clients are the responders.

Theorem 4. *The 4WHS protocol is $\text{AKE}^{\mathsf{static}}$-secure. In particular, for any PPT adversary $\mathcal{A}$, there exists a partner function f and algorithm $\mathcal{D}$, such that*

$$\mathsf{Adv}^{\mathsf{2P\text{-}AKE}^{\mathsf{static}}}_{\mathsf{4WHS},\mathcal{A},f}(\lambda) \leq |\mathcal{P}_C| \cdot |\mathcal{P}_{AP}| \cdot \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF}}(\mathcal{D}) + \frac{(n_P n_\pi)^2}{2^{\lambda+1}}, \tag{11}$$

where n_π is the number of sessions at each party, and $n_P = |\mathcal{P}_C| + |\mathcal{P}_{AP}|$.

For this protocol it is natural to use SIDs as our partnering mechanism. However, because our paper is phrased in terms of partnering functions, we

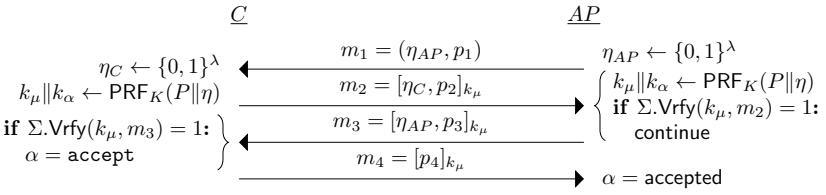

Fig. 6. The IEEE 802.11 4-Way-Handshake protocol. The client C and the access point AP share a symmetric key $\mathrm{PMK} = K$, $P\|\eta = \min\{AP, C\}\|\max\{AP, C\}\|\min\{\eta_{AP}, \eta_C\}\|\max\{\eta_{AP}, \eta_C\}$, and $\Sigma = (\mathsf{kg}, \mathsf{MAC}, \mathsf{Vrfy})$ is MAC scheme.

"synthetically" encode the SID as a partnering function by saying that the partner session is the *first* other session that gets the same SID $P\|\eta$. Taking the *first* one is important because a partner function is a function and not a relation.

Proof. Suppose $P\|\eta = \min\{U, V\}\|\max\{U, V\}\|\min\{\eta_U, \eta_V\}\|\max\{\eta_U, \eta_V\}$ was the string that π_U^i input to its pseudorandom function PRF. Then $f_T(\pi_U^i)$ is defined to be the *first* session at V that input the same string $P\|\eta$ to its PRF. Note that this can be computed based on publicly available transcript information.

Soundness. The soundness of f is immediate from its definition and PRF being deterministic.

AKE^{static}-security.

Game 0: This is the real 2P-AKE security game, hence

$$\mathsf{Adv}^{\mathsf{G}_0}_{\mathsf{4WHS},\mathcal{A},f}(\lambda) = \mathsf{Adv}^{\mathsf{2P\text{-}AKE}^{\mathsf{static}}}_{\mathsf{4WHS},\mathcal{A},f}(\lambda) \ .$$

Game 1: This game proceeds as the previous one, but aborts if not all the nonces in the game are distinct, hence

$$\mathsf{Adv}^{\mathsf{G}_0}_{\mathsf{4WHS},\mathcal{A},f}(\lambda) \leq \mathsf{Adv}^{\mathsf{G}_1}_{\mathsf{4WHS},\mathcal{A},f}(\lambda) + \frac{(n_P n_\pi)^2}{2^{\lambda+1}} \ . \tag{12}$$

Game 2: In this game the challenger guesses the pre-shared key that will be used by the test-session and aborts if that guess was wrong, hence

$$\mathsf{Adv}^{\mathsf{G}_1}_{\mathsf{4WHS},\mathcal{A},f}(\lambda) \leq |\mathcal{P}_{AP}| \cdot |\mathcal{P}_C| \cdot \mathsf{Adv}^{\mathsf{G}_2}_{\mathsf{4WHS},\mathcal{A},f}(\lambda) \ . \tag{13}$$

Let PMK^* denote the guessed pre-shared key. Note that by the $\mathsf{Fresh}_{\mathsf{AKE}^{\mathsf{static}}}$ requirement (Fig. 3), PMK^* cannot be exposed.

Game 3: In this game the challenger replaces the pseudorandom function PRF with a random function $\$(\cdot)$ in all evaluations using the guessed pre-shared key PMK^*. That is, calls of the form $\mathsf{PRF}(\mathrm{PMK}^*, \cdot)$ are instead answered by $\$(\cdot)$.

Lemma 7. $\mathsf{Adv}^{\mathsf{G}_2}_{\mathsf{4WHS},\mathcal{A},f}(\lambda) \leq \mathsf{Adv}^{\mathsf{G}_3}_{\mathsf{4WHS},\mathcal{A},f}(\lambda) + \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF},\mathcal{D}}(\lambda)$.

Proof. Algorithm $\mathcal{D}$ has access to an oracle $\mathcal{O}$, which either implements the function $\Pi.\mathsf{PRF}(\widetilde{\mathrm{PMK}}, \cdot)$ for some independently and uniformly distributed key $\widetilde{\mathrm{PMK}}$, or it implements a truly random function $\$(\cdot)$. $\mathcal{D}$ begins by choosing a random bit b and guessing a client-AP pair (C, AP). All computations that would normally involve the pre-shared key of C and AP, algorithm $\mathcal{D}$ will instead forward to its oracle $\mathcal{O}$. For all other client-AP pairs, $\mathcal{D}$ creates their the pre-shared keys itself, allowing it to simulate them perfectly. If $\mathcal{A}$ outputs b', then $\mathcal{D}$ outputs 1 if $b = b'$, and 0 otherwise.

When $\mathcal{O} = \Pi.\mathsf{PRF}(\widetilde{\mathrm{PMK}}, \cdot)$, then $\mathcal{D}$ perfectly simulates Game 2 since the PMKs are chosen independently and uniformly at random; while when $\mathcal{O} = \$(\cdot)$, then $\mathcal{D}$ perfectly simulates Game 3. □

Concluding the proof of Theorem 4. Suppose the test-session in Game 3 accepted with the "SID" $P\|\eta$. By Game 1 we know that the only sessions that evaluated the pseudorandom function on this SID was the test-session and possibly its partner. However, by Game 3 the PRF is now a truly random function unavailable to the adversary (since we are in the static corruption model). In particular, this means that the PTK derived by the test-session (and possibly its partner) is a truly random string $\widetilde{\mathrm{PTK}} = \widetilde{k_\mu}\|\widetilde{k_\alpha} \leftarrow \{0,1\}^{2\lambda}$, and where $\widetilde{k_\alpha}$ is independent of all other values. Thus, $\mathsf{Adv}^{\mathsf{G}_3}_{\mathsf{4WHS},\mathcal{A},f}(\lambda) = 0$, and Theorem 4 follows. □

We now turn to proving explicit entity authentication for the 4WHS.

Theorem 5. *The 4WHS provides explicit entity authentication. In particular, for any PPT adversary $\mathcal{A}$, there exists algorithms $\mathcal{D}$ and $\mathcal{F}$, such that*

$$\mathsf{Adv}^{\mathsf{2P\text{-}AKE}^{\mathsf{static}}\text{-}\mathsf{EA}}_{\mathsf{4WHS},\mathcal{A},f}(\lambda) \leq |\mathcal{P}_C| \cdot |\mathcal{P}_{AP}| \cdot \left(\mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF},\mathcal{D}}(\lambda) + \frac{(n_P n_\pi)^2}{2^{\lambda+1}} + 2n_\pi \cdot \mathsf{Adv}^{\mathsf{UF\text{-}CMA}}_{\Sigma,\mathcal{F}}(\lambda) \right), \tag{14}$$

where f, n_π, and n_P are the same as in Theorem 4.

Proof. This proof uses the exact same three game hops as in the proof of Theorem 4, differing only in its interpretation of the guessed pre-shared key PMK^*: instead of hoping that PMK^* belongs to the test-session, we now hope that it belongs to the first session that accepts maliciously. To recap, in Game 3 the challenger aborts if any nonces collide, or the first session that accepts maliciously uses a different pre-shared key then PMK^*. Moreover, all evaluations of $\mathsf{PRF}(\mathrm{PMK}^*, \cdot)$ are replaced with a truly random function $\$(\cdot)$. Since all the game hops are the same, we only have the analyze the probability that a session accepts maliciously in Game 3.

Lemma 8. $\mathsf{Adv}^{\mathsf{G_3\text{-}EA}}_{\mathsf{4WHS},f,\mathcal{A}}(\lambda) \leq 2n_\pi \cdot \mathsf{Adv}^{\mathsf{UF\text{-}CMA}}_{\Sigma,\mathcal{F}}(\lambda)$.

Proof. The forger $\mathcal{F}$ has access to two oracles $\mathcal{O}^{\mathsf{MAC}}$ and $\mathcal{O}^{\mathsf{Vrfy}}$, which implements the MAC and Vrfy algorithms of the MAC scheme Σ for some independent random key $\widetilde{k_\mu}$. Among all the sessions that use PMK*, $\mathcal{F}$ will guess a random session π^* and embed the oracles $\mathcal{O}^{\mathsf{MAC}}$ and $\mathcal{O}^{\mathsf{Vrfy}}$ into it. Let V^* denote the intended communication partner of π^*. We consider two cases based on whether π^* is a client or an AP.

Case $U^* \in \mathcal{P}_{AP}$. $\mathcal{F}$ will simulate Game 3 by creating all the pre-shared keys and implementing the random function $\$(\cdot)$ by lazy-sampling. However, when creating and verifying the handshake messages of π^*, it will use the oracles $\mathcal{O}^{\mathsf{MAC}}$ and $\mathcal{O}^{\mathsf{Vrfy}}$. Specifically, when receiving the handshake message m_2, π^* will accept only if $\mathcal{O}^{\mathsf{Vrfy}}(m_2) = 1$. Moreover, if any session accepts maliciously before π^*, then $\mathcal{F}$ aborts. Additionally, $\mathcal{F}$ also aborts if the nonce η_C contained in m_2 was created by a session at V^* that received the correct nonce η_{AP} from π^*. Note that this event simply means that $\mathcal{F}$'s guess of π^* was wrong, because if π^* were to accept on receiving this m_2 message, it could not have accepted maliciously by the definition of f, since the session creating η_C would be its partner (here we are also using that all the nonces are unique).

By the uniqueness of nonces, and the assumptions above, no session will evaluate $\$(\cdot)$ on the same input as π^*. Hence, embedding the oracles $\mathcal{O}^{\mathsf{MAC}}$, $\mathcal{O}^{\mathsf{Vrfy}}$ into π^* provides a perfect simulation of Game 3. But this means that π^* accepts maliciously iff $\mathcal{O}^{\mathsf{Vrfy}}(m_2) = 1$, with m_2 being a valid forgery.

Case $U^* \in \mathcal{P}_C$. Similar to the previous case, $\mathcal{F}$ embeds $\mathcal{O}^{\mathsf{MAC}}$, $\mathcal{O}^{\mathsf{Vrfy}}$ into π^*, and aborts if the guess was wrong. This again provides a perfect simulation of Game 3, and π^* accepts maliciously iff the call to $\mathcal{O}^{\mathsf{Vrfy}}$ is a valid forgery.

Since in both cases malicious acceptance by π^* implies a forgery for Σ, the lemma follows. □

5.3 Security of IEEE 802.11 with upper-layer authentication

In enterprise and university networks it is both inconvenient and less secure for every user to share a common PMK when accessing the WLAN. In these environments, user authentication is instead handled by a central authentication server, which is then accessed via some EAP variant. While the IEEE 802.11 standard technically allows for upper-level authentication mechanisms other than EAP, the de-facto standard is EAP. Since we have already proved that certain variants of EAP satisfies the 3P-AKE^w notion (Theorem 3), and that the 4WHS is a secure 2P-AKE protocol with static corruption (Theorem 4 and 5); the security of IEEE 802.11 with upper-level authentication now follows directly by applying our second composition theorem (Theorem 2) with $\Pi_3 = \text{EAP}$ and $\Pi_4 = \text{4WHS}$.

Theorem 6 (3P-AKE security of IEEE 802.11 w/upper-layer authentication). *If the* PMK *for the 4WHS is derived using a variant of EAP that is 3P-AKE^w-secure, then the IEEE 802.11 protocol with upper-layer authentication is 3P-AKE-secure.*

Acknowledgments

We would like to thank Colin Boyd, Britta Hale and Cas Cremers for helpful comments and discussions. Chris Brzuska is grateful to NXP for supporting his chair for IT Security Analysis.

References

1. IEEE Standard for Local and metropolitan area networks - Port-Based Network Access Control. IEEE Std 802.1X-2010 (Revision of IEEE Std 802.1X-2004) pp. C1–205 (Feb 2010)
2. IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Std 802.11-2012 pp. 1–2793 (March 2012)
3. Abdalla, M., Fouque, P.A., Pointcheval, D.: Password-based authenticated key exchange in the three-party setting. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 65–84. Springer, Heidelberg (Jan 2005)
4. Aboba, B., Blunk, L.J., Vollbrecht, J.R., Carlson, J., Levkowetz, H.: Extensible Authentication Protocol. RFC 3748, RFC Editor (June 2004), `https://tools.ietf.org/html/rfc3748`
5. Alt, S., Fouque, P.A., Macario-Rat, G., Onete, C., Richard, B.: A cryptographic analysis of UMTS/LTE AKA. In: Manulis, M., Sadeghi, A.R., Schneider, S. (eds.) ACNS 16. LNCS, vol. 9696, pp. 18–35. Springer, Heidelberg (Jun 2016)
6. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (May 2000)
7. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: CRYPTO. Lecture Notes in Computer Science, vol. 773, pp. 232–249. Springer (1993)
8. Bellare, M., Rogaway, P.: Provably secure session key distribution: The three party case. In: 27th ACM STOC. pp. 57–66. ACM Press (May / Jun 1995)
9. Bhargavan, K., Fournet, C., Kohlweiss, M., Pironti, A., Strub, P.Y., Zanella Béguelin, S.: Proving the TLS handshake secure (as it is). In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part II. LNCS, vol. 8617, pp. 235–255. Springer, Heidelberg (Aug 2014)
10. Brzuska, C., Cremers, C., Jacobsen, H., Kohbrok, K., Warinschi, B.: Partner mechanisms in key exchange protocols. Unpublished manuscript (2017)
11. Brzuska, C., Fischlin, M., Warinschi, B., Williams, S.C.: Composability of Bellare-Rogaway key exchange protocols. In: Chen, Y., Danezis, G., Shmatikov, V. (eds.) ACM CCS 11. pp. 51–62. ACM Press (Oct 2011)
12. Brzuska, C., Jacobsen, H., Stebila, D.: Safely exporting keys from secure channels: On the security of EAP-TLS and TLS key exporters. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part I. LNCS, vol. 9665, pp. 670–698. Springer, Heidelberg (May 2016)
13. Canetti, R., Krawczyk, H.: Security analysis of IKE's signature-based key-exchange protocol. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 143–161. Springer, Heidelberg (Aug 2002), `http://eprint.iacr.org/2002/120/`
14. Hartman, S., Clancy, T.C., Hoeper, K.: Channel-Binding Support for Extensible Authentication Protocol (EAP) Methods. RFC 6677, RFC Editor (July 2012), `https://tools.ietf.org/html/rfc6677`

15. Hoeper, K., Chen, L.: Where EAP security claims fail. In: QSHINE. p. 46. ACM (2007)
16. Jager, T., Kohlar, F., Schäge, S., Schwenk, J.: On the security of TLS-DHE in the standard model. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 273–293. Springer, Heidelberg (Aug 2012)
17. Kobara, K., Shin, S., Strefler, M.: Partnership in key exchange protocols. In: Li, W., Susilo, W., Tupakula, U.K., Safavi-Naini, R., Varadharajan, V. (eds.) ASIACCS 09. pp. 161–170. ACM Press (Mar 2009)
18. Kohlar, F., Schäge, S., Schwenk, J.: On the security of TLS-DH and TLS-RSA in the standard model. Cryptology ePrint Archive, Report 2013/367 (2013), http://eprint.iacr.org/2013/367
19. Krawczyk, H.: HMQV: A high-performance secure Diffie-Hellman protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (Aug 2005)
20. Krawczyk, H., Paterson, K.G., Wee, H.: On the security of the TLS protocol: A systematic analysis. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 429–448. Springer, Heidelberg (Aug 2013)
21. LaMacchia, B., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. Cryptology ePrint Archive, Report 2006/073 (2006), http://eprint.iacr.org/2006/073
22. LaMacchia, B.A., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 1–16. Springer, Heidelberg (Nov 2007)
23. Li, Y., Schäge, S., Yang, Z., Kohlar, F., Schwenk, J.: On the security of the pre-shared key ciphersuites of TLS. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 669–684. Springer, Heidelberg (Mar 2014)
24. Nam, J., Choo, K.K.R., Paik, J., Won, D.: Two-round password-only authenticated key exchange in the three-party setting. Cryptology ePrint Archive, Report 2014/017 (2014), http://eprint.iacr.org/2014/017
25. Ohba, Y., Parthasarathy, M., Yanagiya, M.: Channel Binding Mechanism based on Parameter Binding in Key Derivation. RFC (Informational), RFC Editor (December 2006), https://tools.ietf.org/html/draft-ohba-eap-channel-binding-02
26. Rigney, C., Willens, S., Rubens, A., Simpson, W.: Remote Authentication Dial In User Service (RADIUS). RFC 2865, RFC Editor (June 2000), https://tools.ietf.org/html/rfc2865
27. Rogaway, P.: On the of role of definitions in and beyond cryptography. In: ASIAN. Lecture Notes in Computer Science, vol. 3321, pp. 13–32. Springer (2004)
28. Schwenk, J.: Nonce-based kerberos is a secure delegated AKE protocol. Cryptology ePrint Archive, Report 2016/219 (2016), http://eprint.iacr.org/2016/219
29. Shoup, V., Rubin, A.D.: Session key distribution using smart cards. In: Maurer, U.M. (ed.) EUROCRYPT'96. LNCS, vol. 1070, pp. 321–331. Springer, Heidelberg (May 1996)
30. Winter, S., McCauley, M., Venaas, S., Wierenga, K.: Transport Layer Security (TLS) encryption for RADIUS. RFC 6614 (Experimental), RFC Editor (May 2012), https://tools.ietf.org/html/rfc6614