An abridged version of this paper appears in the proceedings of PKC 2017, DOI: 10.1007/978-3-662-54388-7_12. This is the full version.

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

March 19, 2017

Abstract

We conduct a reduction-based security analysis of the Extensible Authentication Protocol (EAP), a widely used three-party authentication framework. EAP is often found in enterprise networks where it allows a client and an authenticator to establish a shared key with the help of a mutually trusted server. Considered as a three-party authenticated key exchange protocol, we show that the general EAP construction achieves a security notion we call 3P-AKE^w. The 3P-AKE^w security notion captures the idea of *weak forward secrecy* and is a simplified three-party version of the well-known eCK model in the two-pass variant. Our analysis is modular and reflects the compositional nature of EAP.

Additionally, we show that the security of EAP can easily be upgraded to provide *full* forward secrecy simply by adding a subsequent key-confirmation step between the client and the authenticator. In practice this key-confirmation step is often carried out in the form of a 2P-AKE protocol which uses EAP to bootstrap its authentication. A concrete example is the extremely common IEEE 802.11 protocol used in WLANs. In enterprise settings EAP is often used in conjunction with IEEE 802.11 in order to allow the wireless client to authenticate itself to a wireless access point (the authenticator) through some centrally administrated server. Building on our modular results for EAP, we get as our second major result the first reduction-based security result for IEEE 802.11 combined with EAP.

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

Chris: I removed some parts from this print-out. If you miss those, you can find them here: https://eprint.iacr.org/2017/253.pdf

Contents

Chris: We won't read the proof for explicit entity authentication but it might be nice to read the definition for future reference.

The actual proof is only 2.5 pages long. If you want, you can skip forward to page 42 right aways :-).

Chris: In this paper, the introduction is less relevant than in the previous paper we read. Also, the introduction of this paper contains more "stuff", i.e., discussions which might be hard to follow. However, it is probably still a good idea to read through the introduction once and mark how well you followed each part just to get some context and to see that we can still get something out of an introduction even if there are parts we don't understand.

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, rather it specifies a generic three-party framework for composing other concrete authentication protocols. 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.

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. On the other hand, in order to conduct a formal security analysis of EAP, these details matter and require a careful treatment. More generally, the need to make assumptions on protocols outside of the EAP standard makes it harder to analyze as described by Hoeper and Chen [23].

Another reason for the lack of reductionist-based security results on EAP might be due to the fact that it is a three-party protocol. As pointed out by Schwenk in his recent work on Kerberos [41], apart from a few papers like [10, 3, 37, 5, 41] relatively little work has been done on three-party protocols[1] 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 in the computational setting. Our result is modular and reflects the compositional nature of EAP. Building upon this result we extend our analysis to cover a very common application of the EAP framework: network authentication and access control in enterprise and university networks. In particular, we focus on wireless networks based on the IEEE 802.11 standard [2] when combined with EAP for centralized authentication. This setting is often referred to as WPA2-Enterprise. Current results on IEEE 802.11 have so far only focused on the much simpler WPA2-PSK setting where the client and access point (authenticator) already share a pre-established long-term key. WPA2-PSK is typically used in wireless home-networks and small offices where sharing a single long-term key among many users is feasible, while WPA2-Enterprise is used in larger organizations and businesses where central authentication is necessary. Based on our result on EAP we can now provide a

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

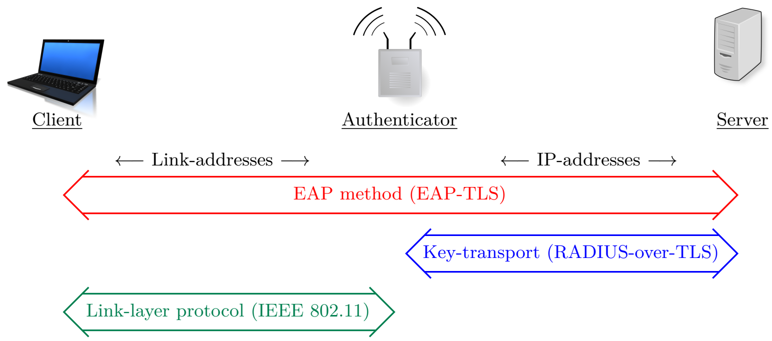

Figure 1: The three-party EAP architecture. Concrete example protocols shown in parenthesis.

reduction-based analysis of WPA2-Enterprise as well.

Review of EAP and IEEE 802.11. The general EAP architecture is shown in Fig. 1. The exchange begins when a client wants to connect to some access-controlled service protected by an authenticator. For most practical purposes the service is simply getting network access (e.g., to the Internet), and the authenticator is a wireless access point or link-layer switch. The client and authenticator do not share any common secrets a-priori but instead share credentials with a mutually trusted server. The purpose of EAP is to allow the client to authenticate itself to the server using whatever authentication protocol they like, for instance TLS or IPsec, and then having the server "vouch" for the client to the authenticator. In order to do this in a generic and uniform manner across different authentication protocols, EAP defines a frame format as well as a set of generic messages known as Request/Response messages. The Request and Response messages are used to encapsulate the concrete authentication protocol being used between the client and the server. This frees the authenticator—which often operates in so-called *pass-through* mode between the client and the server, meaning that all their messages pass through it—from having to support all the concrete authentication mechanisms itself. Instead, the authenticator only needs to inspect the generic EAP messages. This is valuable in settings where it can be difficult to update the authenticator(s).

The combination of a concrete authentication protocol, like TLS, together with the EAP encapsulation is called an *EAP method.* Numerous EAP methods have been defined, with EAP-TLS [44] being one of the most widely supported. In EAP-TLS the client and server mutually authenticate each other based on certificates. Besides authentication, the EAP method usually also supports the derivation of a shared key between the client and the server. In this paper we will assume that all EAP methods derive keys. The server will forward this key to the authenticator over some separate channel, where the choice of channel protocol is orthogonal to the choice of EAP method used between the client and the server. While the EAP standard does not specify the protocol to use between the server and the authenticator, the de-facto standard in practice is RADIUS [39].[2]

Once the key is transported from the server to the authenticator the EAP exchange is technically complete. Still, at this point the client does not actually have any guarantee that the authenticator is in possession of the same key as itself, since the communication between the server and the authenticator happens over a completely separate channel. Thus, in practice the client and the authenticator now typically run some link-layer specific protocol in order to prove mutual possession of the key distributed by EAP. Additionally, this also serves to implicitly authenticate the client and the authenticator to each other, since in order to get the same key they must have been able to authenticate themselves to the mutually trusted server.

Again, the subsequent link-layer protocol run between the client and the authenticator is outside the scope of EAP and could in principal be any one of a number of different protocols. In this paper we are going to focus particularly on the a very common setting of wireless LANs provided by the IEEE 802.11 protocol[2]. In this case the "key-confirmation" protocol run between the client and the authenticator (i.e., access point) is known to as the "4-Way-Handshake" (4WHS). The 4WHS is both an authentication protocol as well as a key-exchange protocol, meaning that the client and the access point also derive fresh session keys. This session key is used to protect the bulk data transfer between the client and the access point on the WLAN. Although technically incorrect, the security part of the IEEE 802.11 wireless standard is also commonly referred to by the name "WPA2". When EAP and IEEE 802.11 are combined, then the entire exchange is referred to as "IEEE 802.11 with upper-layer authentication" or WPA2-Enterprise.

On the difficulty of modeling EAP. In this paper we analyze the security of EAP both when considered on its own as well as when combined with IEEE 802.11 in a formal reduction-based setting. We do this in a modular way: first considering the security properties provided by EAP and IEEE 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 one 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 server. It 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, it is impossible to talk about "the" EAP and its security without making further assumptions

[2]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. Besides RADIUS is Diameter [18] another common AAA protocol.

on the various protocols that make up EAP. 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 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[3] (see Section 2.4 for a formal definition). Since most key-transport protocols used between the server and the authenticator support to be run inside a secure channel (see e.g. RADIUS-over-TLS [45] and Diameter [18]), this assumption seems reasonable.

Second, since the authenticator often works in pass-through mode, a well-known issue with the EAP architecture is the so-called "lying authenticator problem". Namely, a malicious authenticator may present false or inconsistent identity information to the client and the server. Unless the EAP method provides a feature known as *channel binding* [21], there is no way for the client and server to verify that they are in fact talking to the same authenticator (see [21, §3] for examples of attacks that this may enable). Hence, in this paper we are generally going to assume that EAP provides channel binding. However, we will also briefly explore the (in)security of EAP without channel binding in Section 4.3. While there are a couple of suggested ways to achieve channel binding in EAP (see [21, §4.1]), here we are only going to focus on the cryptographically simplest one, described in RFC draft `draft-ohba-eap-channel-binding-02` [38]. 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 reduction-based security result for EAP assuming it employs 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 the common scenario where EAP is first used to bootstrap the establishment of a common key among the client and the authenticator, then some link-layer specific protocol is run between the client and the authenticator in order to prove mutual possession of that key (in addition to establishing session keys for the lower-layer link).
- 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

[3]There is nothing fundamental about our assumption on symmetric PSKs here. The choice is made 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.

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].

- As a stepping stone towards our final result, we provide a reduction-based security result for the IEEE 802.11 4-Way-Handshake protocol in the preshared key setting without the use of EAP (i.e., WPA2-PSK). This corresponds to the setting typically found in home WLANs. To the best of our knowledge, we are the first do to such an analysis using standard game-based definitions of AKE. Previous analyses of WPA2-PSK have either been based on formal methods [22] or on universal composition frameworks [32, 33].

- Finally, the results above combine to provide the first reduction-based security result for EAP combined with IEEE 802.11 (WPA2-Enterprise). This corresponds to the setting usually found in enterprise and university WLANs. For instance, the *eduroam* network[4], which is used to provide wireless roaming services to university and research institutions, uses EAP and IEEE 802.11.

The structure of our paper is as follows. In Section 2 we provide our formal security definitions, including 2P/3P-AKE, ACCE (secure channels) and explicit entity authentication. In Section 3 we prove our two composition results for two generic protocol constructions. In Section 4 we show how the security of standalone EAP follows directly from our first composition result by making appropriate assumptions on the concrete protocols used to instantiate the EAP framework. Finally, in Section 5, we prove the security of the IEEE 802.11 4WHS protocol. Combined with our result on EAP in Section 4 and our second composition result, this immediately yields a result for IEEE 802.11 combined with EAP.

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 with or without a subsequent key-confirmation step, respectively. To obtain these theorems we first have to provide an appropriate security model. Our starting point is the original 3P-AKE model of Bellare and Rogaway [10]. However, due the different security guarantees provided by standalone EAP, EAP combined with IEEE 802.11 and standalone IEEE 802.11, we in fact define *three* different models of varying strength. The main difference between these models lays in the level of adaptivity afforded to the adversary in terms of long-term key leakage, capturing full, weak and no forward secrecy, respectively. The distinction between full and weak forward secrecy follows the definition given in the eCK model[5] [34].

[4] `https://www.eduroam.org`

[5] We do not consider ephemeral key-leakage in this paper however.

Briefly, the only difference between the strongest security model (full forward secrecy) and the intermediate one (weak forward secrecy) depends on what happens if the test-session does not have a partner. When this happens in the strongest model the adversary is still allowed to learn the long-term keys of the parties involved, provided this happens after the test-session accepted. On the other hand, in the intermediate model, if the test-session does not have a partner then the adversary is forbidden from learning these long-term keys. Finally, 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 2.3.

Preempting our own results a bit, we show that standalone EAP can achieve weak forward secrecy, while IEEE 802.11 *without* upper-layer authentication achieves no forward secrecy at all (this is natural since the 4WHS relies exclusively on symmetric primitives). However, when EAP and IEEE 802.11 are *combined*, the security is upgraded to achieve full forward secrecy in our strongest corruption model.

Intuitively, the reason why standalone EAP does not achieve full forward secrecy is because it does not provide *key confirmation*. Namely, after completing the EAP method with the server, the client has no guarantee that the key-transport protocol between the server and the authenticator actually took place. Specifically, the following attack illustrates why EAP does not provide full forward secrecy. Suppose that *after* the client accepted, but *before* the key-transport protocol between the server and authenticator starts running, an adversary learns the long-term PSK of the server and the authenticator. Now the adversary simply impersonates the authenticator towards the server and have it send over the session key it previously established with the client. According to the full forward secrecy model this attack would be valid since the exposure of the PSK happened after the client accepted. On the other hand, in the weak forward secrecy model the attack is not considered valid because client session doesn't have a partner, hence the PSK cannot be exposed.

Essentially, the purpose of the link-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. This is similar to how the security of the two-flow variant of HMQV can be upgraded from only providing weak forward secrecy to providing full forward secrecy simply by adding a third flow to it (see [28, §3]). While the property of key confirmation was recently formalized in [19], in this paper we model the key-confirmation step by assuming that the link-layer protocol provides *explicit* entity authentication (formally defined in Section 2.5).

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 [10]. For example, we support both asymmetric and symmetric long-term keys, and dispense with the explicit SendS query to the server (now modeled simply as a regular Send query).

One thing we *do* keep from [10] however, is the concept of *partner functions*. Interestingly, the use of partner functions has seen rather limited adoption when compared to partnering based on matching conversations [9] or session identifiers (SIDs) [8]. 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. [25]), it does not necessarily guarantee *public* partnering (see [13]). 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 [40, §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.

2 Formal models

2.1 Notation

For $m, n \in \mathbb{N}$ and $m \leq n$, let $[m, n] \stackrel{\text{def}}{=} \{m, m+1, \dots n\}$. We use $v \leftarrow x$ to denote the assignment of x to the variable v, and $x \twoheadleftarrow X$ to denote that x is assigned a value randomly according to the distribution X. If S is a finite set, then $x \twoheadleftarrow S$ means to sample x uniformly at random from S. Algorithms are in general randomized, and we let $y \twoheadleftarrow A(x_1, \dots, x_n)$ denote running A on inputs $x_1, \dots, x_n$ with random coins r, and assigning its output to the variable y. A function $g \colon \mathbb{N} \to \mathbb{R}$ is *negligible* if for every $c \in \mathbb{N}$ there is an integer n_c such that $g(n) \leq n^{-c}$ for all $n \geq n_c$.

2.2 A unified execution model

Protocol participants. A protocol is carried out by a set of *parties* $U \in \mathcal{P}$. Each party U can either take 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 the two-party case there are no servers, in which case $\mathcal{S} = \emptyset$.

Our model includes both long-term asymmetric keys as well as a symmetric pre-shared keys (PSKs). While there are in general many ways in which asymmetric and symmetric long-term keys could be combined in a three-party protocol, in this paper we are going to limit ourselves to the configuration typically found in EAP. That is, we assume that only initiators and servers are in possession of a long-term private/public key-pair, while all responders and servers share a symmetric PSK. On the other hand, for two-party protocols we assume that the long-term keys are either strictly based on asymmetric keys or strictly based on PSKs. For every U party holding a public key pk_U, we assume that all 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 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 run of the protocol.

- sk_U, pk_U – the (possibly empty) long-term private/public key of party U,
- peers – a list of the identities of the intended communication peers of π_U^i,
- $\mathsf{PK}[\cdot]$ – a map taking party identities to (possibly empty) public keys for each $V \in \pi_U^i.\mathsf{peers}$, i.e, $\mathsf{PK}[V] \mapsto pk_V/\bot$,
- $\mathsf{PSK}[\cdot]$ – a map taking party identities to (possibly empty) PSKs for each $V \in \pi_U^i.\mathsf{peers}$, i.e, $\mathsf{PSK}[V] \mapsto K_{UV}/\bot$,
- $\vec{\alpha} = (\alpha_1, \ldots, \alpha_n)$ – a vector of *accept states* $\alpha_i \in \{\mathsf{running}, \mathsf{accepted}, \mathsf{rejected}\}$,
- $k \in \{0,1\}^\lambda \cup \{\bot\}$ – the symmetric session-key derived by π_U^i.

Only initiators and responders accept sessions keys, i.e., if $S \in \mathcal{S}$, then we always have $\pi_S^i.k = \bot$. Note that this is pure formalism: we certainly expect many protocols in which the server might be in possession of the session key—in fact, the 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 $\vec{\alpha}$ rather than a *single* acceptance state more commonly found in other formal protocol models. We do this 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 acceptance state of the full protocol, and use $\alpha_F \stackrel{\text{def}}{=} \alpha_n$ to denote this state. A session is said to be *running*, *accepted* or *rejected*, based on the value of α_F. Thus, α_F has the same role as the single acceptance state variable α used in other protocol models.

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

$$\alpha_i = \mathsf{accepted} \implies \alpha_{i-1} = \mathsf{accepted}, \tag{1}$$
$$\alpha_i = \mathsf{rejected} \implies \alpha_{i+1} = \mathsf{rejected}, \tag{2}$$
$$\pi.\alpha_n = \mathsf{accepted} \implies \pi.k \neq \bot\ . \tag{3}$$

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.

$\underline{\mathbf{Exp}_{\Pi,\mathcal{Q},\mathcal{A}}(\lambda)}$:

Long-term key set-up:

  3P: For every $U \in \mathcal{I} \cup \mathcal{S}$ create $(sk_U, pk_U) \twoheadleftarrow \Pi.\mathsf{KG}(1^\lambda)$

    For every $(U, V) \in \mathcal{R} \times \mathcal{S}$ define $K_{UV} = K_{VU} \twoheadleftarrow \{0,1\}^\lambda$

    Define $\mathsf{pks} \leftarrow \{(U, pk_U) \mid U \in \mathcal{I} \cup \mathcal{S}\}$

  2P-Public-Key: for every $U \in \mathcal{I} \cup \mathcal{R}$ create $(sk_U, pk_U) \twoheadleftarrow \Pi.\mathsf{KG}(1^\lambda)$

    Define $\mathsf{pks} \leftarrow \{(U, pk_U) \mid U \in \mathcal{I} \cup \mathcal{R}\}$

  2P-PSK: For every $(U, V) \in \mathcal{I} \times \mathcal{R}$ define $K_{UV} = K_{VU} \twoheadleftarrow \{0,1\}^\lambda$

    Define $\mathsf{pks} \leftarrow \emptyset$

$out \twoheadleftarrow \mathcal{A}^{\mathcal{Q}}(1^\lambda, \mathsf{pks})$

Figure 2: Unified experiment used to simultaneously define AKE and ACCE security, including three-party and two-party settings as well as protocols using asymmetric and symmetric long-term keys.

Protocol correctness. It is required that an AKE 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 server π_S^k (if in the three-party setting)—meaning that all messages are faithfully transmitted between them according the protocol description—then all sessions end up accepting with the correct intended peers, and π_A^i and π_B^j both hold the same session key $k \neq \perp$.

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 $\mathbf{Exp}_{\Pi,\mathcal{Q},\mathcal{A}}(\lambda)$ 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. It is required that U, V and W all have different roles.

  The variables k and $\vec{\alpha}$ are initialized to $\pi_U^i.k = \perp$ and $\pi_U^i.\vec{\alpha} = (\mathsf{running}, \ldots, \mathsf{running})$, respectively. Additionally, depending on the type of protocol (two-party/three-party, symmetric/asymmetric long-term keys), as well as the roles of U, V and/or W, the variables sk, pk, peers, PK and PSK are initialized accordingly.

  Finally, if $U \in \mathcal{I}$, then π_U^i also produces its first message m according to the specification of protocol Π. Both the administrative label π_U^i and m are returned to $\mathcal{A}$.

- $\mathsf{Send}(\pi_U^i, m)$: If $\pi_U^i.\alpha_F \neq \mathsf{running}$, return $\perp$. Otherwise, π_U^i creates a response message m^* according to the specification of protocol Π. This depends on π_U^i's role and current internal state. Both m^* and $\pi_U^i.\vec{\alpha}$ 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_U^i 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 has been leaked we say that it is *exposed* and the corresponding party *corrupted*.

Remark 2. We are working in the post-specified peer model [15], meaning that the identities of a session's peers might not necessarily be known by the session at the beginning of the protocol run, but are rather learned as the protocol progresses.

Note that experiment $\mathbf{Exp}_{\Pi,\mathcal{Q},\mathcal{A}}(\lambda)$ 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 since it allows an easy way of specifying the many different security models needed in this paper in a uniform manner. Common for all of the security models will be the notion of *freshness* which decides whether $\mathcal{A}$ has managed to satisfy a winning condition in a non-trivial way. Our definition(s) of freshness further depends on the notion of *partnering*, defined next. Partnering is used to formalize the intuition that for any session π that ends up holding a session key, there will (possibly) be some *other* session π' whose loss of session key will also compromise that of π.

Transcripts and partner functions. To define partnering in our security models we will use the concept of *partner functions* as introduced by Bellare and Rogaway [10]. However, to simplify our later analysis, we will limit ourselves only to *symmetric* and *monotonic* partner functions. Basically, a partner function is symmetric if it is its own inverse (up to $\perp$), and monotonic if it never "changes its mind", i.e., once two sessions become partners they remain so forever. Bellare and Rogaway did not demand these properties directly in their original definition, but instead claimed that they could be inferred from the definition (see [10, §6]). We find it easier to require these properties at the definitional level.

To formally define partner functions, we first the need the notion of a protocol *transcript*, which is essentially records the public communication of a protocol run. More precisely, consider a run of experiment $\mathbf{Exp}_{\Pi,\mathcal{Q},\mathcal{A}}(\lambda)$. 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 another transcript T', written $T \subseteq T'$, if the first $|T|$ entries of T' are identical to T. Let $\mathcal{T}$ denote the set of all possible transcripts generated from running experiment $\mathbf{Exp}_{\Pi,\mathcal{Q},\mathcal{A}}(\lambda)$. We can now define partner functions.

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

1. $f(T, U, i) = (V, j) \implies f(T, V, j) = (U, i)$, (symmetry)
2. $f(T, U, i) = (V, j) \implies f(T', U, i) = (V, j)$ for all $T \subseteq T'$. (monotonicity)

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.

Since all partner functions in this paper are required to be both symmetric and monotonic, we drop these qualifiers from now on and simply talk about "partner functions". Note that both requirements are straightforwardly met by partner functions based on SIDs.

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

Of course, by the symmetry requirement above, if π' is the partner to π, then π will also be a partner to π'. Hence, we can simply talk about π and π' being *partners*.

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

Remark 4. 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 [10], it has only been used in one other paper by Shoup and Rubin [43]. In a currently unpublished manuscript [12], we provide a more detailed treatment of partner functions in general.

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. However, since we are limiting our attention to symmetric partner functions in this paper, the last requirement follows directly so we omit it.

Definition 3 (Partner function soundness)**.** A partner function is *sound* if the following holds for all transcripts T. If sessions $f_{T'}(\pi_U^i) = \pi_V^j$ then:

1. $\pi_U^i.\alpha_F = \pi_V^j.\alpha_F = \mathsf{accepted} \implies \pi_U^i.k = \pi_V^j.k \neq \bot$,

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 $U \in \mathcal{R} \wedge V \in \mathcal{I}$,

Soundness is essentially the partner function equivalent of the Match-security notion introduced by Brzuska et al. [13], used for partnering based on SIDs. However, unlike Match-security, we demand that properties (1)–(3) hold unconditionally instead of only with overwhelming probability. We note that this requirement is not fundamental, and only used to simplify our later analysis.

2.3 2P-AKE and 3P-AKE

Syntax. A *2P/3P-AKE protocol* has the same syntax as the general protocol defined in Section 2.2. Note that is also no syntactical difference between a 2P-AKE protocol and a 3P-AKE protocol, apart from the fact that the former has no server session $S \in \mathcal{S}$. Consequently, in the two-party case the session variables peers, PK and PSK contain at most a single entry.

AKE security. 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 $\bot$. 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 *test-key*. The Test query can only be made once.

Let $\mathcal{Q}_{\mathsf{AKE}} = \mathcal{Q}_{base} \cup \{\mathsf{Test}\}$. Experiment $\mathbf{Exp}_{\Pi,\mathcal{Q},\mathcal{A}}(\lambda)$ 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 key 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 gives rise to a corresponding security model, denoted AKE, AKE^w and AKE$^{\mathsf{static}}$ respectively. We describe the three models in more detail below and summarize their main differences in Table 1.

AKE with forward secrecy: the AKE and AKE^w models. The AKE model is our "partner function analogue" of the standard eCK model (as defined in the updated version [34] of the original paper [35]), 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

$\underline{\mathsf{Fresh}_M(\pi_U^i)}$:

$\mathsf{fresh} \leftarrow \texttt{true}$
$\{V, W\} \leftarrow \pi_U^i.\mathsf{peers}$
$\mathsf{LTKeys} \leftarrow \{\pi_U^i.\mathsf{PK}[V], \pi_U^i.\mathsf{PK}[W], \pi_U^i.\mathsf{PSK}[V], \pi_U^i.\mathsf{PSK}[W], K_{VW}\}$

$\mathsf{fresh} \leftarrow \mathsf{fresh} \wedge (\pi_U^i.\alpha_F = \mathsf{accepted})$
$\mathsf{fresh} \leftarrow \mathsf{fresh} \wedge (\pi_U^i \text{ not revealed})$
$\mathsf{fresh} \leftarrow \mathsf{fresh} \wedge (f_T(\pi_U^i) \text{ not revealed})$
$\mathsf{fresh} \leftarrow \mathsf{fresh} \wedge (\mathsf{corrupt}_M = \texttt{false})$

return fresh

- $\mathsf{corrupt}_{\mathsf{AKE/ACCE}} = \texttt{true} \iff (f_T(\pi_U^i) = \perp) \wedge$ (a key in LTKeys was exposed before π_U^i accepted)
- $\mathsf{corrupt}_{\mathsf{AKE}^w} = \texttt{true} \iff (f_T(\pi_U^i) = \perp) \wedge$ (a key in LTKeys is exposed)
- $\mathsf{corrupt}_{\mathsf{AKE}^{\mathsf{static}}} = \texttt{true} \iff$ a key in LTKeys is exposed

Figure 3: Freshness predicate for security model $M \in \{\text{AKE}, \text{AKE}^w, \text{AKE}^{\mathsf{static}}, \text{ACCE}\}$. Some of the keys in LTKeys might be undefined, e.g., if $W \in \mathcal{S}$, then $\pi_U^i.\mathsf{PK}[W]$, $\pi_U^i.\mathsf{PK}[W]$ and K_{VW} are undefined in the two-party case, and $\pi_U^i.\mathsf{PK}[V]$ is undefined if V is a responder party in the three-party case. Undefined keys are ignored.

key can always be exposed by the adversary. Forward secrecy is captured since the adversary can additionally learn the long-term keys of the peers of the test-session after it accepted.

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 [34, 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.

AKE without forward secrecy: the AKE$^{\mathsf{static}}$ *model.* To accommodate protocols that does not provide forward secrecy we introduce the AKE$^{\mathsf{static}}$ model. Unlike the AKE and AKE^w models, the AKE$^{\mathsf{static}}$ model disallows the adversary

Table 1: Summary of the three AKE security models in terms of the amount of corruption allowable by the adversary (i.e., long-term key reveals). The table assumes π_A^i is the test-session having peers B and S (in the three-party case).

		Corrupt B or S	
Model	Corrupt A	if π_A^i has a partner	if π_A^i has no partner
AKE	allowed	allowed	allowed[1]
AKE^w	allowed	allowed	$\times$
AKE$^{\mathsf{static}}$	$\times$	$\times$	$\times$

[1] Only after π_A^i accepted.

from exposing the long-term keys altogether, no matter whether a session has a partner or not (of course, the adversary is allowed to expose long-term keys unrelated to the test-session and its peers).

On the other hand, for technical reasons (see the explanation following Lemma 9 in Section 3.2), we slightly strengthen the AKE$^{\mathsf{static}}$ model compared to the AKE and AKE^w-models along a different axis. Namely, we give the adversary the capability of *key registration*. That is, when creating a new session, the adversary is allowed to (optionally) specify the long-term key(s) that the session will use in its protocol run. Of course, any session for which the adversary supplies the long-term keys will be considered unfresh, so the key registration capability does not substantially strengthen the model.

Technically, key registration in the AKE$^{\mathsf{static}}$ model is handled by modifying the $\mathsf{NewSession}$ query. Furthermore, since we are only going to use the AKE$^{\mathsf{static}}$ model to analyze PSK-based two-party protocols in this paper, we specialize the definition to this specific case:

- $\mathsf{NewSession}(U, [V], [\widehat{K}])$: This query works exactly like the $\mathsf{NewSession}$ query defined in Section 2.2, except that if the adversary supplies an optional long-term key $\widehat{K}$, then the newly created session π_U^i stores $\widehat{K}$ in $\pi_U^i.\mathsf{PSK}[V]$ rather than K_{UV}. In this case $\pi_U^i.\mathsf{PSK}[V]$ is considered *exposed*.

If the adversary makes a $\mathsf{NewSesssion}$ query where it provides a long-term key, then the key is nevertheless omitted from the $\mathsf{NewSession}$ query that gets added to the protocol transcript T. Thus, the protocol transcripts generated from the AKE$^{\mathsf{static}}$ model are syntactically the same as those generated from the AKE and AKE^w models, even if the latter does not include key registration.

Security definitions. Let $\mathcal{Q}_{\mathsf{AKE}}$ denote the query set either used in the AKE or AKE^w models (having $\mathsf{NewSession}$ queries without key registration), or in the AKE$^{\mathsf{static}}$ model (having $\mathsf{NewSession}$ queries with key registration).

Definition 4 (AKE winning event)**.** Suppose π was the test-session chosen by $\mathcal{A}$ in a run of experiment $\mathbf{Exp}_{\Pi,\mathcal{Q}_{\mathsf{AKE}},\mathcal{A}}(\lambda)$, b was the random bit used in answering the Test query, and suppose b' was the final output of $\mathcal{A}$. Fix a partner function f and let $\mathsf{AKE}^* \in \{\mathsf{AKE}, \mathsf{AKE}^w, \mathsf{AKE}^{\mathsf{static}}\}$ be the following random variable defined on experiment $\mathbf{Exp}_{\Pi,\mathcal{Q}_{\mathsf{AKE}},\mathcal{A}}(\lambda)$:

$$\mathsf{AKE}^* \stackrel{\text{def}}{=} \begin{cases} (b = b'), & \text{if } \mathsf{Fresh}^*_{\mathsf{AKE}}(\pi) = \texttt{true} \\ \texttt{true} \text{ with probability } 1/2, & \text{if } \mathsf{Fresh}^*_{\mathsf{AKE}}(\pi) = \texttt{false} \end{cases} \tag{4}$$

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

Definition 5 (AKE security)**.** For AKE$^* \in \{$AKE, AKE^w, AKE$^{\mathsf{static}}\}$ and some partner function f, define the *AKE*-advantage* of adversary $\mathcal{A}$ to be

$$\mathbf{Adv}^{\mathsf{AKE}^*}_{\Pi,\mathcal{A},f}(\lambda) \stackrel{\text{def}}{=} 2 \cdot \Pr[\mathbf{Exp}^{\mathsf{AKE}^*}_{\Pi,\mathcal{Q}_{\mathsf{AKE}},\mathcal{A}}(\lambda) \Rightarrow 1] - 1 \tag{5}$$

A protocol Π is *AKE*-secure*, if there exists a sound partner function f, such that for all PPT adversaries $\mathcal{A}$, its advantage $\mathbf{Adv}^{\mathsf{AKE}^*}_{\Pi,\mathcal{A},f}(\lambda)$ is negligible in security parameter λ.

If we want to emphasize that a protocol is two-party or three-party, we write $\mathbf{Adv}^{\mathsf{2P\text{-}AKE}^*}_{\Pi,\mathcal{A},f}(\lambda)$ or $\mathbf{Adv}^{\mathsf{3P\text{-}AKE}^*}_{\Pi,\mathcal{A},f}(\lambda)$, respectively.

Remark 5. Note that in our formulation of security we are quantifying over *all* PPT adversaries, not only those that satisfy the freshness predicate. Instead, if the adversary violates the freshness predicate, we "penalize" it in the winning condition (Def. 4) by having the challenger output a random bit on its behalf. This *penalty-style* of formulating security has previously been used in other papers like e.g., [7] and [20].

Chris: I removed Section 2.4 from this print-out. It's an interesting topic, but a bit of a tangent for what we want to look at.

Chris: We won't look at the proof for explicit entity authentication in week 2, but it might be nice to read the definition.

2.5 Explicit entity authentication

Explicit entity authentication, as opposed to *implicit* entity authentication, adds "aliveness" guarantees to a protocol in the sense that if a session at party A accepts with peer B, then A can be certain that there exists a corresponding session at B that contributed to this protocol run. While the need for AKE protocols to provide explicit entity authentication has been somewhat disputed in the literature (see e.g. [10, §1.6], [40, §6] or [27, §2.1]), our use of it in this paper mostly serve as an approximation of the (intuitively) simpler notion of *key confirmation* (see [19] for a detailed treatment of this property). On the other hand, explicit entity authentication has always been part of the security requirements of an ACCE protocol [24, 30, 26], and we are going to assume that in this paper too.

Since the definition of explicit 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 (in any of the three security models), and let $\mathcal{Q}_{\mathsf{ACCE}}$ denote the query set of the ACCE experiment.

Definition 8 (Entity authentication predicate)**.** For $\mathsf{X} \in \{\mathsf{AKE}, \mathsf{ACCE}\}$, let T be the transcript of experiment $\mathbf{Exp}_{\Pi,\mathcal{A},\mathcal{Q}_{\mathsf{X}}}(\lambda)$. Predicate Auth is true if and only if the following holds for all fresh sessions π:

$$\pi.\alpha_F = \mathsf{accepted} \implies \exists \pi' \text{ such that } f_{T'}(\pi) = \pi'. \tag{8}$$

Let $\mathbf{Exp}^{\mathsf{X\text{-}Auth}}_{\Pi,\mathcal{Q}_{\mathsf{X}},\mathcal{A}}(\lambda) \Rightarrow 1$ denote the event that Auth is true. A fresh session that accepts without a partner is said to have *accepted maliciously*.

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

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

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

Remark 6. Note that the explicit entity authentication of an AKE (resp. ACCE) protocol needs to hold with the *same* partner function as used to prove its AKE (resp. ACCE) security.

Chris: I removed Section 3 and 4 from this print-out.

Chris: Game-Hopping

The proof in this section uses game-hopping. There are two styles of game-hopping when trying to prove the indistinguishability of G^0 from G^1.

Style 1:

PPT adversaries can only distinguish with negligible advantage.   PPT adversaries can only distinguish with negligible advantage.   PPT adversaries can only distinguish with negligible advantage.

G^0=H_0   H_1   H_2   ...   H_k-1   H_k=G^1

=> The advantage of a PPT adversary A in distinguishing G^0 from G^1 is upper bounded by the sum of the advantages of A distinguishing between H_i and H_i+1. Since summing a constant number of negligible functions yields a negligible function. A has negligible advantage in distinguishing G^0 from G^1.

The proof in this section uses game-hopping. There are two styles of game-hopping when trying to prove the indistinguishability of G^0 from G^1.

Style 2: In the following, G^b denotes the game which flips a random bit b in the beginning and then, depending on this bit either emulates G^0 or G^1.

PPT adversaries can only distinguish with negligible advantage.   PPT adversaries can only distinguish with negligible advantage.   PPT adversaries can only distinguish with negligible advantage.

G^b=H_0   H_1   H_2   ...   H_k-1   H_k   Any PPT adversary has advantage 0 in H_k.

=> The advantage of a PPT adversary A in distinguishing G^0 from G^1 is upper bounded by the sum of the advantages of A distinguishing between H_i and H_i+1 plus the advantage of A in H_k. Since summing a constant number of negligible functions yields a negligible function. A has negligible advantage in distinguishing G^0 from G^1.

This paper adopts the 2nd game-hopping style.

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, which usually also 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.

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

The IEEE 802.11 protocol is a link-layer protocol, aiming to secure the wireless link between the client and the access point. It defines two main security protocols: the *4-Way-Handshake (4WHS)*, used to authenticate and establish session keys between the client and the access point; 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 access point. The PMK can either be pre-configured at the client and access point or distributed through some other means, like for instance EAP. The first alternative is most typically found in wireless home networks where a static PMK is manually configured at the access point and at every connecting device.[12] This variant is also commonly referred to as WPA2-PSK. The second alternative, often referred to as WPA2-Enterprise, is normally used in large organization like universities and big companies where there are many users and access points. In this setting it is infeasible for every user and access point to share the same PMK. Instead, a central authentication server is used to manage authentication as well as distributing new PMKs for every established session. Usually the protocol used to access the authentication server is EAP.

In Section 5.2 we analyze the pre-shared key variant of the 4WHS, while in Section 5.3 we analyze it when combined with EAP.

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 $\Sigma = (\mathsf{MAC}, \mathsf{Vrfy})$; see Appendix A for their formal definitions. We use the notation $[x]_k \stackrel{\text{def}}{=} x \| \sigma$ to denote a message x together with its MAC tag $\sigma \leftarrow \mathsf{MAC}(k, x)$. Identities in the 4WHS are based on the parties' 48-bit link-layer addresses which makes it possible to compare them 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 when interpreted as 48-bit integers.

In our modeling we will mostly ignore the exact encoding of the IEEE 802.11 packets as used by the 4WHS. For our purposes it sufficient to model them as consisting of a nonce plus a fixed constant p_i that uniquely determines each handshake message m_i. If a received message does not match the expected format, including the value of the constant p_i, it is silently discarded. The 4WHS proceeds as follows:

1. The 4WHS begins with the access point AP sending the message $m_1 = \eta_{AP} \| p_1$ to the client C, where η_{AP} is a nonce and p_1 a constant.

2. On receiving m_1, C generates its own nonce η_C and derives a so-called *pairwise transient key (PTK)* using the pseudorandom function PRF and

[12] Usually the PMK is not configured directly, but instead derived from a password using a password-based KDF. We ignore this distinction here.

the long-term key it shares with AP. Specifically, PTK $\stackrel{\text{def}}{=} k_\mu \| k_\alpha \leftarrow \mathsf{PRF}_K(P\|\eta)$, where $P\|\eta = \min\{AP, C\}\| \max\{AP, C\}\| \min\{\eta_{AP}, \eta_C\}\| \max\{\eta_{AP}, \eta_C\}$. The sub-key k_α will be the session key eventually output by the client, while the sub-key k_μ will be used in the MAC Σ to protect the handshake messages. After deriving PTK, C creates and sends the next protocol message $m_2 = [\eta_C \| p_2]_{k_\mu}$.

3. On receiving $m_2 = [\eta_C \| p_2]_{k_\mu}$, AP uses the containing nonce η_C to derive $k_\mu \| k_\alpha \leftarrow \mathsf{PRF}_K(P\|\eta)$. With the sub-key k_μ, it verifies the MAC tag of m_2. If the verification succeeds, then AP stores PTK $\leftarrow k_\mu \| k_\alpha$ as its PTK and sends out the third protocol message $m_3 = [\eta_{AP} \| p_3]_{k_\mu}$. If the verification fails, then AP silently discards m_2, as well as the derived keys k_μ, k_α, and continues running.

4. On receiving $m_3 = [\eta'_{AP} \| p_3]_{k_\mu}$, C first verifies its MAC tag with key k_μ, and checks that η'_{AP} equals the nonce η_{AP} that C previously received in message m_1. If the verification succeeds then C sends out the final handshake message $m_4 = [p_4]_{k_\mu}$. Additionally, it sets its own acceptance state to $\alpha = \mathsf{accepted}$. If the verification fails, C silently discards m_3 and continues running.

5. On receiving m_4, AP verifies its MAC tag using the key k_μ. If the verification succeeds, it sets its own acceptance state to $\alpha = \mathsf{accepted}$. If the verification fails, AP silently discards m_4 and continues running.

Remark 10. An adversary can freely modify message m_1 since it has no integrity protection. However, since every recipient of an m_1 message will check that it matches the excepted format of "$x\|p_1$", the adversary is in reality limited to only modifying the value of the nonce. Of course, this is a simplification compared to the real IEEE 802.11 header, where there are actually multiple different bit fields which the adversary could manipulate—in principle. Still, the fact is that except for the nonce η_{AP}, all bit fields in the IEEE 802.11 header of the first m_1 message have pre-determined values. Thus, the attacker does not have more opportunities to manipulate the real IEEE 802.11 m_1 message as opposed to in our simplified modeling of it.

On the other hand, in the IEEE 802.11 header of messages m_2, m_3 and m_4, there *are* bit fields that the adversary could potentially influence. But since these messages are protected by a MAC, the adversary will be unable to modify them. Whether we model p_2, p_3 and p_4 as constants or as arbitrary distinct values makes no difference for our analysis.

Remark 11. 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.

AKE$^{\mathsf{static}}$-security. We begin by proving that the 4WHS constitutes a secure 2P-AKE in the AKE$^{\mathsf{static}}$ model. Following that, we show that it also achieves

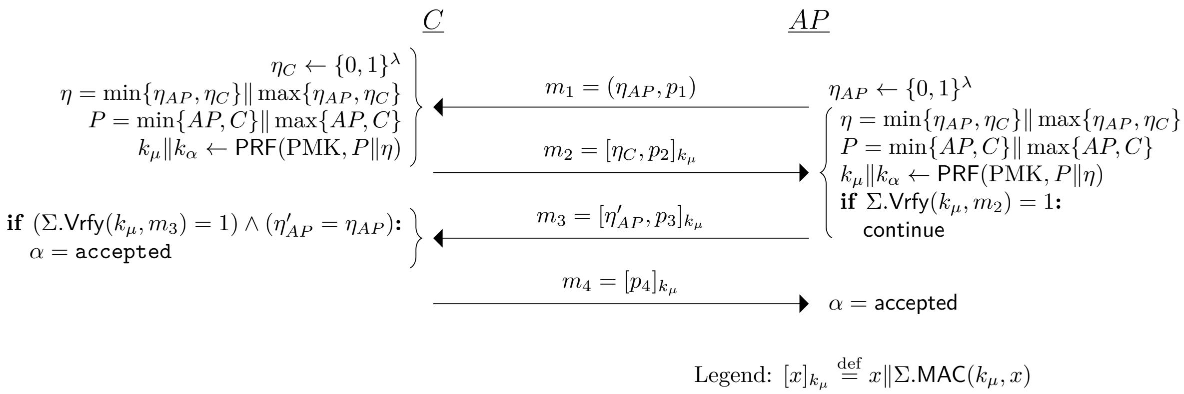

Figure 6: The IEEE 802.11 4-Way-Handshake protocol. The client C and the access point AP share a long-term symmetric key PMK.

explicit entity authentication. In the following, let $\mathcal{P}_{AP} = \mathcal{I}$ and $\mathcal{P}_C = \mathcal{R}$, i.e., in the 4WHS protocol the access point has the initiator role while the client has the responder role. According to the IEEE 802.11 standard, each client and access point is allowed to share multiple long-term PMKs with each other. In the following analysis we make the simplifying assumption that every client–access point pair only shares a single PMK.

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

$$\mathbf{Adv}^{\mathsf{2P\text{-}AKE}^{\mathsf{static}}}_{\mathsf{4WHS},\mathcal{A},f}(\lambda) \leq 2 \cdot |\mathcal{P}_C| \cdot |\mathcal{P}_{AP}| \cdot \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{PRF}}(\mathcal{D}) + \frac{(n_P n_\pi)^2}{2^{\lambda+1}}, \tag{23}$$

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

Proof. Recall that in the AKE$^{\mathsf{static}}$ model the adversary is allowed to register the PMK a session will use when creating it via the $\mathsf{NewSession}$ query. Of course, in this case the session's PMK will be considered exposed and the session will thus not be fresh according to predicate $\mathsf{Fresh}_{\mathsf{AKE}^{\mathsf{static}}}$.

Defining the partner function f. For the analysis of the 4HWS it would be natural to use SIDs as the partnering mechanism. Namely, the SID of a session π would be the string $P \| \eta$ that π input to its PRF in order to create its session key (see Fig. 6).[13] However, because our paper is phrased in terms of partnering functions, we "synthetically" encode the SID as a partnering function by saying that π's partner session is the *first* session—different from π—that sets the same SID as π. Taking the first one is important because a partner function is a function and not a relation.

[13]For an access point the SID would only be set if the verification of the received m_2 message succeeded.

In more detail, suppose π^i_{AP} is an access point session having $C \in \mathcal{P}_C$ as its intended peer. If π^i_{AP} itself created the nonce η_{AP} for message m_1, and later successfully verified an incoming m_2 message containing the nonce η_C, then $f_T(\pi^i_{AP}) = \pi^j_C$ if and only if (1) π^j_C has AP as its indented peer and (2) π^j_C was the first session at C that used the nonces η_C and η_{AP} to derive its PTK.

Similarly, suppose π^i_C is a client session having $AP \in \mathcal{P}_{AP}$ as its intended peer. If π^i_C used the nonces η_C and η_{AP} to derive its PTK after receiving message $m_1 = (\eta_{AP} \| p_1)$, then $f_T(\pi^i_C) = \pi^j_{AP}$ if and only if (1) π^j_{AP} has C as its intended peer, (2) π^j_{AP} created the nonce η_{AP} and (3) π^j_{AP} was the first session at AP that successfully verified an m_2 message containing the nonce η_C.

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

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

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

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

$$\mathbf{Adv}^{\mathsf{G}_0}_{\mathsf{4WHS},\mathcal{A},f}(\lambda) \leq \mathbf{Adv}^{\mathsf{G}_1}_{\mathsf{4WHS},\mathcal{A},f}(\lambda) + \frac{(n_P n_\pi)^2}{2^{\lambda+1}} \ . \tag{24}$$

Game 2: This game implements a selective AKE security game where at the beginning of the game the adversary has to “commit” to the pre-shared PMK that will be used by the test-session.

Specifically, at the beginning the of the game, the adversary has to output two party identities $C \in \mathcal{P}_C$ and $AP \in \mathcal{P}_{AP}$. The game then proceeds as in Game 1, except that it aborts if the test-session selected by the adversary did not use the PMK shared between C and AP.

Lemma 11. $\mathbf{Adv}^{\mathsf{G}_1}_{\mathsf{4WHS},\mathcal{A},f}(\lambda) \leq |\mathcal{P}_{AP}| \cdot |\mathcal{P}_C| \cdot \mathbf{Adv}^{\mathsf{G}_2}_{\mathsf{4WHS},\mathcal{A}',f}(\lambda)$.

Proof. From an adversary $\mathcal{A}$ that wins against the adaptive game in Game 1, we create an adversary $\mathcal{A}'$ that wins against the selective game in Game 2 as follows. First, $\mathcal{A}'$ randomly selects two party identities $C \in \mathcal{P}_C$ and $AP \in \mathcal{P}_{AP}$. It outputs C and AP as its choice to the selective security game it is playing. $\mathcal{A}'$ then runs $\mathcal{A}$ and answers all of its queries by forwarding them to its own selective security game. When $\mathcal{A}$ stops with output b', then $\mathcal{A}'$ stops and outputs the same bit as well.

Algorithm $\mathcal{A}'$ perfectly simulates Game 1 for $\mathcal{A}$, so $\mathcal{A}'$'s choice of selective security targets matches those of $\mathcal{A}$ with probability at least $1/(|\mathcal{P}_{AP}| \cdot |\mathcal{P}_C|)$. When $\mathcal{A}'$'s guess is correct it wins with the same probability as $\mathcal{A}$, while when it is wrong, $\mathcal{A}'$ gets penalized in its selective security game, hence wins with probability 1/2. $\square$

In the remainder of the proof, let C and AP denote the parties that the adversary commits to in Game 2, and let PMK* denote the PMK shared between them. Note that by the requirements of the $\mathsf{Fresh}_{\mathsf{AKE}^{\mathsf{static}}}$ predicate (Fig. 3), PMK* cannot be exposed if the test-session is to be fresh. In particular, this means that the adversary cannot make a $\mathsf{LongTermKeyReveal}(C, AP)$ query, nor create the test-session via a $\mathsf{NewSession}$ query where it registers PMK* as its long-term key.

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

Lemma 12. $\mathbf{Adv}^{\mathsf{G}_2}_{\mathsf{4WHS},\mathcal{A},f}(\lambda) \leq \mathbf{Adv}^{\mathsf{G}_3}_{\mathsf{4WHS},\mathcal{A},f}(\lambda) + 2 \cdot \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{PRF},\mathcal{D}}(\lambda)$.

Proof. Algorithm $\mathcal{D}$ has access to an oracle $\mathcal{O}$, which either implements the function $\mathsf{PRF}(\widetilde{\mathrm{PMK}}, \cdot)$ 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_{sim} and creating all the PMKs for all client-access points pairs different from the selective security targets C and AP. It then runs $\mathcal{A}$.

For all of $\mathcal{A}$'s queries that does not involve computations with the PMK of C and AP, $\mathcal{D}$ answers itself using the keys it created. On the other hand, for queries that would normally involve computations with the PMK of C and AP, algorithm $\mathcal{D}$ uses its oracle $\mathcal{O}$ to do these computations, and the answers the queries accordingly. Finally, when $\mathcal{A}$ stops with some output b', then $\mathcal{D}$ stops and outputs 0 to its PRF experiment if $b' = b_{sim}$, and 1 otherwise.

When $\mathcal{O} = \mathsf{PRF}(\widetilde{\mathrm{PMK}}, \cdot)$, 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. The lemma follows. $\square$

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 which is unavailable to the adversary provided the test-session is to remain AKE$^{\mathsf{static}}$-fresh. 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, $\mathbf{Adv}^{\mathsf{G}_3}_{\mathsf{4WHS},\mathcal{A},f}(\lambda) = 0$, and Theorem 4 follows. $\square$

Chris: I removed the proof of explicit entity authentication from this print-out.

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.

A Additional definitions

A.1 Pseudorandom functions

A *pseudorandom function (PRF)* is a family of polynomial-time functions $F\colon \{0,1\}^\lambda \times \{0,1\}^\ell \to \{0,1\}^L$, having key-length key-length λ, input length ℓ and output length L. Let $\mathsf{Func}(\ell, L)$ denote the family of all functions from $\{0,1\}^\ell$ to $\{0,1\}^L$. The security of a PRF is defined by the experiments shown in Fig. 8.

$\mathbf{Exp}^{\mathsf{PRF\text{-}0}}_{\mathsf{PRF},\mathcal{A}}(\lambda)$:	$\mathbf{Exp}^{\mathsf{PRF\text{-}1}}_{\mathsf{PRF},\mathcal{A}}(\lambda)$:
1: $K \twoheadleftarrow \{0,1\}^\lambda$	1: $f \twoheadleftarrow \mathsf{Func}(\ell, L)$
2: $b \leftarrow \mathcal{A}^{\mathsf{PRF}(K,\cdot)}(1^\lambda)$	2: $b \leftarrow \mathcal{A}^{f(\cdot)}(1^\lambda)$
3: **return** b	3: **return** b

Figure 8: Experiments defining PRF security.

Definition 10 (PRF)**.** Let PRF be a PRF. The *PRF-advantage* of an adversary $\mathcal{A}$ is

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{PRF},\mathcal{A}}(\lambda) \stackrel{\text{def}}{=} \Pr[\mathbf{Exp}^{\mathsf{PRF\text{-}0}}_{\mathsf{PRF},\mathcal{A}}(\lambda) \Rightarrow 1] - \Pr[\mathbf{Exp}^{\mathsf{PRF\text{-}1}}_{\mathsf{PRF},\mathcal{A}}(\lambda) \Rightarrow 1] \tag{29}$$

A PRF is *(PRF-)secure* if $\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{PRF},\mathcal{A}}(\lambda)$ is negligible in security parameter λ for all PPT adversaries $\mathcal{A}$.

A.2 Message Authentication Codes

A *message authentication code (MAC)* is a pair of polynomial-time algorithms $\Sigma = (\mathsf{MAC}, \mathsf{Vrfy})$, where

- $\mathsf{MAC}\colon \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^*$ is a deterministic *tag-generation* algorithm that takes in a key $K \in \{0,1\}^\lambda$, a *message* $m \in \{0,1\}^*$ and returns a *tag* $\tau \in \{0,1\}^*$.
- $\mathsf{Vrfy}\colon \{0,1\}^\lambda \times \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ is a deterministic *verification-algorithm* that takes in a key $K \in \{0,1\}^\lambda$, a message $m \in \{0,1\}^*$ and a candidate tag $\tau \in \{0,1\}^*$; and produces a *decision* $d \in \{0,1\}$. Algorithm $\mathsf{Vrfy}(K,\cdot,\cdot)$ works as follows on inputs m and τ: if $\tau = \mathsf{MAC}(K, m)$ then return 1 (ACCEPT) else return 0 (REJECT).

$\underline{\mathbf{Exp}^{\mathsf{SUF\text{-}CMA}}_{\Sigma,\mathcal{A}}(\lambda)}$:

$K \twoheadleftarrow \{0,1\}^{\lambda}$
$\mathsf{forgery} \leftarrow 0$
$T[\cdot] \leftarrow \emptyset$

$\mathcal{A}^{\mathsf{MAC}(K,\cdot),\mathsf{Vrfy}(K,\cdot,\cdot)}(1^{\lambda})$
return $\mathsf{forgery}$

$\underline{\mathsf{MAC}(K,m)}$:

$\tau \leftarrow \Sigma.\mathsf{MAC}(K,m)$
$T[m] \leftarrow T[m] \cup \{\tau\}$
return τ

$\underline{\mathsf{Vrfy}(K,m,\tau)}$:

$d \leftarrow \Sigma.\mathsf{Vrfy}(K,m,\tau)$
if $(d = 1) \wedge (\tau \notin T[m])$**:**
    $\mathsf{forgery} \leftarrow 1$
return d

Figure 9: Experiment defining SUF-CMA security for a MAC $\Sigma = (\mathsf{MAC}, \mathsf{Vrfy})$.

The security of a MAC is defined by the experiment shown in Fig. 9.

Definition 11 (SUF-CMA security)**.** Let $\Sigma = (\mathsf{MAC}, \mathsf{Vrfy})$ be a MAC. The *SUF-CMA-advantage* of an adversary $\mathcal{A}$ is

$$\mathbf{Adv}^{\mathsf{SUF\text{-}CMA}}_{\Sigma,\mathcal{A}}(\lambda) \stackrel{\text{def}}{=} \Pr[\mathbf{Exp}^{\mathsf{SUF\text{-}CMA}}_{\Sigma,\mathcal{A}}(\lambda) \Rightarrow 1]. \tag{30}$$

We say that Σ is *strongly-unforgeable against chosen-message attacks (SUF-CMA)*, or simply *SUF-CMA-secure*, if $\mathbf{Adv}^{\mathsf{SUF\text{-}CMA}}_{\Sigma,\mathcal{A}}(\lambda)$ is negligible in security parameter λ for any PPT adversary $\mathcal{A}$.

Chris: I removed Appendix B and C from this print-out.

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)*, pages C1–205, Feb 2010. https://doi.org/10.1109/IEEESTD.2010.5409813. (Cited on page 38.)

[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*, pages 1–2793, March 2012. https://dx.doi.org/10.1109/IEEESTD.2012.6178212. (Cited on pages 3, 5, and 39.)

[3] Michel Abdalla, Pierre-Alain Fouque, and David Pointcheval. Password-based authenticated key exchange in the three-party setting. In Serge Vaudenay, editor, *PKC 2005: 8th International Workshop on Theory and Practice in Public Key Cryptography*, volume 3386 of *Lecture Notes in Computer Science*, pages 65–84, Les Diablerets, Switzerland, January 23–26, 2005. Springer, Heidelberg, Germany. (Cited on page 3.)

[4] Bernard Aboba, Larry J. Blunk, John R. Vollbrecht, James Carlson, and Henrik Levkowetz. Extensible Authentication Protocol. RFC 3748, RFC Editor, June 2004. https://tools.ietf.org/html/rfc3748. (Cited on pages 3, 5, and 37.)

[5] Stéphanie Alt, Pierre-Alain Fouque, Gilles Macario-Rat, Cristina Onete, and Benjamin Richard. A cryptographic analysis of UMTS/LTE AKA. In Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider, editors, *ACNS 16: 14th International Conference on Applied Cryptography and Network Security*, volume 9696 of *Lecture Notes in Computer Science*, pages 18–35, Guildford, UK, June 19–22, 2016. Springer, Heidelberg, Germany. (Cited on pages 3, 7, and 39.)

[6] Mihir Bellare, Oded Goldreich, and Anton Mityagin. The power of verification queries in message authentication and authenticated encryption. Cryptology ePrint Archive, Report 2004/309, 2004. http://eprint.iacr.org/2004/309. (Cited on page 45.)

[7] Mihir Bellare, Dennis Hofheinz, and Eike Kiltz. Subtleties in the definition of IND-CCA: When and how should challenge decryption be disallowed? *Journal of Cryptology*, 28(1):29–48, January 2015. (Cited on page 17.)

[8] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155, Bruges, Belgium, May 14–18, 2000. Springer, Heidelberg, Germany. (Cited on page 9.)

[9] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In *CRYPTO*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249. Springer, 1993. (Cited on page 9.)

[10] Mihir Bellare and Phillip Rogaway. Provably secure session key distribution: The three party case. In *27th Annual ACM Symposium on Theory of Computing*, pages 57–66, Las Vegas, NV, USA, May 29 – June 1, 1995. ACM Press. (Cited on pages 3, 7, 8, 8, 12, 12, 13, and 19.)

[11] Karthikeyan Bhargavan, Cédric Fournet, Markulf Kohlweiss, Alfredo Pironti, Pierre-Yves Strub, and Santiago Zanella Béguelin. Proving the TLS handshake secure (as it is). In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part II*, volume 8617 of *Lecture Notes in Computer Science*, pages 235–255, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany. (Cited on page 37.)

[12] Chris Brzuska, Cas Cremers, Håkon Jacobsen, and Bogdan Warinschi. Partner mechanisms in key exchange protocols. Unpublished manuscript, 2017. (Cited on page 13.)

[13] Christina Brzuska, Marc Fischlin, Bogdan Warinschi, and Stephen C. Williams. Composability of Bellare-Rogaway key exchange protocols. In Yan Chen, George Danezis, and Vitaly Shmatikov, editors, *ACM CCS 11: 18th Conference on Computer and Communications Security*, pages 51–62, Chicago, Illinois, USA, October 17–21, 2011. ACM Press. (Cited on pages 9 and 14.)

[14] Christina Brzuska, Håkon Jacobsen, and Douglas Stebila. Safely exporting keys from secure channels: On the security of EAP-TLS and TLS key exporters. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 670–698, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany. (Cited on page 37.)

[15] Ran Canetti and Hugo Krawczyk. Security analysis of IKE's signature-based key-exchange protocol. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 143–161, Santa Barbara, CA, USA, August 18–22, 2002. Springer,

Heidelberg, Germany. http://eprint.iacr.org/2002/120/. (Cited on page 12.)

[16] T. Charles Clancy and Katrin Hoeper. Making the case for EAP channel bindings. In *2009 IEEE Sarnoff Symposium, Princeton, NJ, March 30-31 & April 1*, pages 1–5. IEEEXplore, March 2009. https://doi.org/10.1109/SARNOF.2009.4850319. (Cited on page 38.)

[17] Morris J. Dworkin. SP 800-38B. Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication. Technical report, National Institute of Standards & Technology, Gaithersburg, MD, United States, October 2016. https://dx.doi.org/10.6028/NIST.SP.800-38B. (Cited on page 45.)

[18] Victor Fajardo, Jari Arkko, John Loughney, and Glen Zorn. Diameter Base Protocol. RFC 6733, IETF RFC Editor, October 2012. https://tools.ietf.org/html/rfc6733. (Cited on pages 5, 6, and 37.)

[19] Marc Fischlin, Felix Günther, Benedikt Schmidt, and Bogdan Warinschi. Key confirmation in key exchange: A formal treatment and implications for TLS 1.3. In *2016 IEEE Symposium on Security and Privacy*, pages 452–469, San Jose, CA, USA, May 22–26, 2016. IEEE Computer Society Press. (Cited on pages 8 and 19.)

[20] Wesley George and Charles Rackoff. Rethinking definitions of security for session key agreement. Cryptology ePrint Archive, Report 2013/139, 2013. http://eprint.iacr.org/2013/139. (Cited on page 17.)

[21] Sam Hartman, T. Charles Clancy, and Katrin Hoeper. Channel-Binding Support for Extensible Authentication Protocol (EAP) Methods. RFC 6677, RFC Editor, July 2012. https://tools.ietf.org/html/rfc6677. (Cited on pages 6, 6, 6, and 38.)

[22] Changhua He, Mukund Sundararajan, Anupam Datta, Ante Derek, and John C. Mitchell. A modular correctness proof of IEEE 802.11i and TLS. In Vijayalakshmi Atluri, Catherine Meadows, and Ari Juels, editors, *ACM CCS 05: 12th Conference on Computer and Communications Security*, pages 2–15, Alexandria, Virginia, USA, November 7–11, 2005. ACM Press. (Cited on page 7.)

[23] Katrin Hoeper and Lidong Chen. Where EAP security claims fail. In Victor Leung and Sastri Kota, editors, *4th International ICST Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness, QSHINE 2007, Vancouver, Canada, August 14-17, 2007*, page 46. ACM, 2007. (Cited on pages 3 and 38.)

[24] Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk. On the security of TLS-DHE in the standard model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume

7417 of *Lecture Notes in Computer Science*, pages 273–293, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany. (Cited on pages 17, 19, and 37.)

[25] Kazukuni Kobara, SeongHan Shin, and Mario Strefler. Partnership in key exchange protocols. In Wanqing Li, Willy Susilo, Udaya Kiran Tupakula, Reihaneh Safavi-Naini, and Vijay Varadharajan, editors, *ASIACCS 09: 4th ACM Symposium on Information, Computer and Communications Security*, pages 161–170, Sydney, Australia, March 10–12, 2009. ACM Press. (Cited on page 9.)

[26] Florian Kohlar, Sven Schäge, and Jörg Schwenk. 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`. (Cited on pages 19 and 37.)

[27] Hugo Krawczyk. SIGMA: The "SIGn-and-MAc" approach to authenticated Diffie-Hellman and its use in the IKE protocols. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 400–425, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany. (Cited on page 19.)

[28] Hugo Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 546–566, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Heidelberg, Germany. (Cited on page 8.)

[29] Hugo Krawczyk, Mihir Bellare, and Ran Canetti. HMAC: Keyed-Hashing for Message Authentication. RFC 2104 (Informational), February 1997. Available at `http://www.ietf.org/rfc/rfc2104.txt`. (Cited on page 45.)

[30] Hugo Krawczyk, Kenneth G. Paterson, and Hoeteck Wee. On the security of the TLS protocol: A systematic analysis. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 429–448, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany. (Cited on pages 19 and 37.)

[31] Hugo Krawczyk, Kenneth G. Paterson, and Hoeteck Wee. On the security of the TLS protocol: A systematic analysis. Cryptology ePrint Archive, Report 2013/339, 2013. `http://eprint.iacr.org/2013/339`. (Cited on page 23.)

[32] Ralf Küsters and Max Tuengerthal. Composition theorems without pre-established session identifiers. In Yan Chen, George Danezis, and Vitaly Shmatikov, editors, *ACM CCS 11: 18th Conference on Computer and Communications Security*, pages 41–50, Chicago, Illinois, USA, October 17–21, 2011. ACM Press. (Cited on page 7.)

[33] Ralf Küsters and Max Tuengerthal. Ideal key derivation and encryption in simulation-based security. In Aggelos Kiayias, editor, *Topics in Cryptology – CT-RSA 2011*, volume 6558 of *Lecture Notes in Computer Science*, pages 161–179, San Francisco, CA, USA, February 14–18, 2011. Springer, Heidelberg, Germany. (Cited on page 7.)

[34] Brian LaMacchia, Kristin Lauter, and Anton Mityagin. Stronger security of authenticated key exchange. Cryptology ePrint Archive, Report 2006/073, 2006. `http://eprint.iacr.org/2006/073`. (Cited on pages 7, 14, and 15.)

[35] Brian A. LaMacchia, Kristin Lauter, and Anton Mityagin. Stronger security of authenticated key exchange. In Willy Susilo, Joseph K. Liu, and Yi Mu, editors, *ProvSec 2007: 1st International Conference on Provable Security*, volume 4784 of *Lecture Notes in Computer Science*, pages 1–16, Wollongong, Australia, November 1–2, 2007. Springer, Heidelberg, Germany. (Cited on page 14.)

[36] Yong Li, Sven Schäge, Zheng Yang, Florian Kohlar, and Jörg Schwenk. On the security of the pre-shared key ciphersuites of TLS. In Hugo Krawczyk, editor, *PKC 2014: 17th International Conference on Theory and Practice of Public Key Cryptography*, volume 8383 of *Lecture Notes in Computer Science*, pages 669–684, Buenos Aires, Argentina, March 26–28, 2014. Springer, Heidelberg, Germany. (Cited on page 37.)

[37] Junghyun Nam, Kim-Kwang Raymond Choo, Juryon Paik, and Dongho Won. 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`. (Cited on page 3.)

[38] Yoshihiro Ohba, Mohan Parthasarathy, and Mayumi Yanagiya. Channel Binding Mechanism based on Parameter Binding in Key Derivation. RFC (Informational), IETF RFC Editor, December 2006. `https://tools.ietf.org/html/draft-ohba-eap-channel-binding-02`. (Cited on pages 6, 37, and 38.)

[39] Carl Rigney, Allan Rubens, William Allen Simpson, and Steve Willens. Remote Authentication Dial In User Service (RADIUS). RFC 2865, IETF RFC Editor, June 2000. `https://tools.ietf.org/html/rfc2865`. (Cited on page 5.)

[40] Phillip Rogaway. On the of role of definitions in and beyond cryptography. In *ASIAN*, volume 3321 of *Lecture Notes in Computer Science*, pages 13–32. Springer, 2004. (Cited on pages 9 and 19.)

[41] Jörg Schwenk. Nonce-based kerberos is a secure delegated AKE protocol. Cryptology ePrint Archive, Report 2016/219, 2016. `http://eprint.iacr.org/2016/219`. (Cited on pages 3 and 3.)

[42] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. `http://eprint.iacr.org/2004/332`. (Cited on pages 23, 27, and 32.)

[43] Victor Shoup and Aviel D. Rubin. Session key distribution using smart cards. In Ueli M. Maurer, editor, *Advances in Cryptology – EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 321–331, Saragossa, Spain, May 12–16, 1996. Springer, Heidelberg, Germany. (Cited on page 13.)

[44] Dan Simon, Bernhard Aboba, and Ryan Hurst. The EAP-TLS Authentication Protocol. RFC 5216 (Proposed Standard), March 2008. Available at `http://tools.ietf.org/html/rfc5216`. (Cited on page 4.)

[45] Stefan Winter, Mike McCauley, Stig Venaas, and Klaas Wierenga. Transport Layer Security (TLS) encryption for RADIUS. RFC 6614 (Experimental), IETF RFC Editor, May 2012. `https://tools.ietf.org/html/rfc6614`. (Cited on pages 6 and 37.)