Doctoral theses at NTNU, 2017:290

Håkon Jacobsen

A Modular Security Analysis of EAP and IEEE 802.11

Doctoral thesis

Håkon Jacobsen

ISBN 978-82-326-2648-9 (printed ver.)
ISBN 978-82-326-2649-6 (electronic ver.)
ISSN 1503-8181

NTNU
Norwegian University of
Science and Technology

NTNU

Norwegian University of Science and Technology
Thesis for the Degree of
Philosophiae Doctor
Faculty of Information Technology and Electrical
Engineering
Department of Information Security and
Communication Technology

Doctoral theses at NTNU, 2017:290

NTNU
Norwegian University of
Science and Technology

Håkon Jacobsen

A Modular Security Analysis of EAP and IEEE 802.11

Thesis for the Degree of Philosophiae Doctor

Trondheim, October 2017

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Information Security and Communication Technology

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the Degree of Philosophiae Doctor

Faculty of Information Technology and Electrical Engineering
Department of Information Security and Communication Technology

ISBN 978-82-326-2648-9 (printed ver.)
ISBN 978-82-326-2649-6 (electronic ver.)
ISSN 1503-8181

Doctoral theses at NTNU, 2017:290

Printed by NTNU Grafisk senter

Abstract

This thesis presents a computational reduction-based security analysis of the Extensible Authentication Protocol (EAP) and the IEEE 802.11 protocol. EAP is a widely used authentication framework while IEEE 802.11 is the most commonly used standard for creating wireless local area networks (WLANs), better known as Wi-Fi. The typical use case of EAP is to allow a client on a WLAN to connect to an access point through the use of mutually trusted server. EAP is a general framework that specifies how different sub-protocols can be combined to securely achieve this goal. IEEE 802.11 is usually one of the sub-protocols used within the EAP framework.

There are three main contributions of this thesis. The first is a modular security analysis of the general EAP framework. This includes two generic composition theorems that reflect the modular nature of EAP, and which establish sufficient criteria on its sub-protocols in order for the whole framework to be instantiated securely. Having proven the soundness of the general EAP framework, it remains to find suitable sub-protocols that satisfy the security criteria of the composition results.

Our second main contribution is a security analysis of one such concrete sub-protocol, namely the EAP-TLS protocol which is used to establish a shared key between the wireless client and the trusted server. We prove that EAP-TLS is a secure two-party authenticated key exchange protocol by presenting a generic compiler that transforms secure channel protocols into secure key exchange protocols.

Our third main contribution is a thorough security analysis of the IEEE 802.11 protocol. We study both the handshake protocol as well as the encryption algorithm used to protect the application data. On their own, our results on IEEE 802.11 apply to the usage found in wireless home networks where a key is shared between the client and access point *a priori*, e.g. using a password. However, by combining this with our composition theorems for the EAP framework, we also obtain a result for IEEE 802.11 in its "enterprise" variant, where the common key is instead established using a mutually trusted server.

Acknowledgments

I would like to thank my two supervisors Danilo Gligoroski and Colin Boyd for all their guidance and support throughout my studies.

Much of the work in this thesis is the result of collaboration with others. First of all, I want to thank my co-authors on the two papers on which the main parts of this thesis are based: Chris Brzuska and Douglas Stebila. I especially want to acknowledge Chris Brzuska for showing me how fun (and exhausting!) research can be, but also for being a friend, a mentor, a role-model, and in effect a third supervisor for me. Without him this thesis would simply not have been possible.

Additionally, I want to thank Bogdan Warinschi and Cas Cremers for many helpful discussions. I am also indebted to those who volunteered their time and effort into proofreading my thesis: Colin Boyd, Cristina Onete, Chris Brzuska, and Gareth Davies—I express my deepest gratitude to all of you.

A big thanks also goes to my office mates Simona Samardjiska, Britta Hale, and Chris Carr for the great company during my PhD at NTNU.

Finally, I would like to thank my family for their unwavering support and encouragement throughout the years, and last but not least, Vilde for always believing in me. Thank you.

Contents

Chapter 1

Introduction

Contents

Designing secure cryptographic protocols is difficult. Over the years a large number of security protocols have been proposed that later turned out to be flawed. This is mostly due to the inherent complexity of the protocols themselves, but it can also be partly ascribed to the paradigm in which they were traditionally designed. Typically, a protocol designer would start out by proposing some concrete protocol construction P. Next, the protocol would get analyzed, often revealing some flaw. The designer would then revise the original design of P to (hopefully) include a fix for the discovered flaw. The whole cycle would then repeat itself, with a new round of analysis discovering new flaws, yielding more fixes, and so on.

Over time, a body of prudent practices emerged [AN96], identifying common pitfalls when designing cryptographic protocols. However, these practices represented no more than useful heuristics and guidelines, rather than necessary and sufficient criteria for creating secure protocols. Within the academic cryptography community this realization led to an interest in finding more rigorous and formal approaches towards assessing the security of a protocol.

Traditionally, two distinct approaches have been taken in order to formally model cryptographic protocols. The first, and the one we will be following in this thesis, is the *computational* approach. As its name suggests, it has its

roots in computational complexity theory and views cryptographic operations as algorithms working on bitstrings. Adversaries are modeled as probabilistic Turing machines and security is expressed in terms of the probability and computational complexity of carrying out a successful attack. We will have more to say about the computational model below, as well as in Chapter 3.

The second approach is the *symbolic* approach, also called *formal methods*. It has its roots in logic and formal language theory and views cryptographic operations as functions on formal symbolic expressions. A symbolic security model consists of a set of axioms and inference rules that can be applied to the symbolic expressions. For example, a formal expression of the form $\{\!|M|\!\}_K$ could represent the encryption of a message M under some key K. Note, however, that both M and K are also formal expressions, and thus carry no inherent meaning. An inference rule could say that, given $\{\!|M|\!\}_K$ and K, one can conclude M. That is, the inference rule allows you to decrypt M from $\{\!|M|\!\}_K$ given K. On the other hand, without K it is impossible to deduce M. In particular, since only operations derivable from the inference rules are possible, cryptographic primitives in the symbolic model are *perfect*. Security in the symbolic model is expressed as saying that one cannot reach a certain configuration by applying the inference rules, starting from the given axioms. Unlike in the computational approach, there is no probabilistic reasoning in the symbolic world.

A major benefit of the symbolic model is that it readily allows for machine-checkable proofs, or even automatic derivation of proofs. Many tools exist for this purpose, including ProVerif [Bla16], Scyther [Cre08], and Tamarin [Mei+13]. On the other hand, a common criticism of the symbolic approach is that its assumption of perfect black-box primitives is unrealistic. A protocol proven secure in the symbolic model may nevertheless have an attack in the computational model. Still, there have been attempts to bridge the gap between the computational model and the symbolic model, beginning with [AR00].

Finally, there have also been much recent development in tools that can automatically verify proofs in the computational model, such as CryptoVerif [Bla08], EasyCrypt [Bar+13], and miTLS [Bha+13]. Although this thesis will be based on the computational model, it will, however, not be making use of any of these tools. Instead, it will follow the more traditional style of "pen-and-paper" proofs. Moreover, since our security models will be in the computational setting, we will not be saying more about the symbolic model in this thesis. As a result, most of our literature references will be to results in the computational model. At the same time, we acknowledge that there is a vast body of cryptographic research that consequently will not be covered here.

1.1 Computational modeling of cryptographic protocols

The idea of formalizing cryptography within a computational complexity theoretical setting was introduced by Goldwasser and Micali in 1984 [GM84]. Central to their work was the formal *definition* of what it means for a cryptographic scheme to be *secure*. Specifically, they focused on the goal of public-key encryption and formalized the now fundamental definition of *semantic security*. To go along with their new definition, they also created a concrete scheme which they could now *prove* satisfied the definition of semantic security by giving a reduction to a number-theoretic assumption. Soon after, many other common cryptographic primitives, like digital signatures, symmetric encryption, pseudorandom functions, and message-authentication schemes were formalized (and proven secure) in a similar manner.

However, it would go almost 10 years from Goldwasser and Micali's initial paper until the first formal model for cryptographic protocols was presented by Bellare and Rogaway [BR94] in 1993. On the other hand, their model became highly influential for the formal research on protocols, in particular *key exchange* protocols, and it is still the basis for many of the models used today.

The BR-model. The starting point of the BR-model is a set of principals that want to communicate over an insecure network. Every pair of principals shares a common long-term key, and their goal is to negotiate a temporary *session key* which they will use to secure their further communication. In the formal model the details of the communication network is mostly abstracted away, leaving only the principals themselves and a specification of how they behave on receiving input from the network. How the messages are delivered to each principal is left to the adversary's discretion, i.e., in the BR-model the adversary *is* the network. In particular, while the adversary can chose to forward messages as intended by the protocol, it also has full freedom to arbitrarily change, delay, reorder, reroute or drop messages as it sees fit. It is important that we allow the attacker this kind of flexibility since we want our protocols to be secure from *any* choice of adversarial strategy. That is, in general it is impossible to enumerate every possible way that a protocol might get attacked, so the only thing we can reasonably make assumptions about is the attacker's *computational powers*.

Depending on the type of protocol, its security goals may vary. Classically, the goals considered by Bellare and Rogaway [BR94], were those of *authenticated key exchange* and *entity authentication*. The first property focuses on the security of the established session keys themselves. The formal definition of this borrows from the idea of semantic security for public-key encryption

schemes, and demands that an adversary should learn nothing about the distributed keys. The second property focuses on the authenticity of the protocol conversation, meaning that two protocol participants can be assured that they have in fact been speaking to each other at the end of the protocol run. There are also protocols goals beyond those of authenticated key exchange and entity authentication, for example focusing on the secure *usage* of the distributed session keys. This will all be covered in detail in Chapter 3.

Simulation-based vs. game-based security. Within the computational setting, there are two main approaches to defining the security of protocols. One is the *simulation-based* approach and the other is the *game-based* approach. In the simulation-based approach, security is defined by considering two "worlds": an *ideal* world where the protocol is replaced with some *idealized functionality* that is secure by design; and a *real* world where the actual protocol is being used. Security is expressed by saying that for any attacker $\mathcal{A}$ against the protocol in the real world, there should exist a corresponding *simulator* $\mathcal{S}$ in the ideal world, such that the transcript that $\mathcal{A}$ generates through its interactions with the real protocol, is *computationally indistinguishable* from the transcript that $\mathcal{S}$ generates through its interaction with the ideal functionality. Since the ideal functionality is secure by design, the existence of $\mathcal{S}$ means that $\mathcal{A}$'s ability to break the real protocol must be limited.

A number of simulation-based models have been developed in order to analyze protocols. Examples include the model of Shoup [Sho99], the UC model of Canetti [Can01], the IITM model of Küsters and Tuengerthal [KT13], and the GNUC model of Hofheinz and Shoup [HS15]. Of these, the latter three are so-called *universal composability* models, where the emphasis is on very general composition results that allow secure sub-protocols to be arbitrarily composed in order to form larger and still secure protocols. Due to their generality, universal composability models tend to be quite complex.

The alternative to simulation-based models is *game-based* models. Here, security properties are formulated directly as *winning conditions* in a formal experiment, called a *game*, played between an honest entity $\mathcal{C}$ called the *challenger*, and an adversary $\mathcal{A}$. A protocol is said to be *secure* with respect to the property modeled by the game, if no computationally efficient adversary can manage to win in the game except with a small probability. What "efficient" and "small" means in this setting can be formalized in different ways; see Chapter 3.

The original BR-model [BR94] was in the game-based setting, and naturally so were also the large number of extensions and follow-up works that built on it, for example [BR95, BM97, BPR00, CK01, LLM07, Jag+12]. In this thesis we are going to take the game-based approach to security.

1.2 Content and contribution of thesis

This thesis provides a formal security analysis of the Extensible Authentication Protocol (EAP) [RFC3748] and the IEEE 802.11 [IEEE 802.11] protocol in a computational game-based setting. Compared to the Transport Layer Security (TLS) [RFC5246] protocol, which has been subject to a large amount of formal analysis, both EAP and IEEE 802.11 have received considerably less scrutiny. That is not to say that EAP and IEEE 802.11 are little used; quite the contrary. For instance, according to the Wireless Geographic Logging Engine (WiGLE)[1] project, there are more than 350 million Wi-Fi networks available worldwide today—Wi-Fi being the name more commonly associated with IEEE 802.11. Similarly, the *eduroam*[2] network alone, which is a roaming service provided to students and employees of educational institutions around the world, accounted for more than 3 billion user authentications in 2016[3]—all of these use EAP. The importance of these protocols should thus be clear from the sheer scale of their deployment.

The main contribution of this thesis is a formal analysis of the EAP and IEEE 802.11 protocols in a computational setting based on the BR-model. Our analysis will cover these protocols both separately and when combined (since EAP and IEEE 802.11 are often used together). Chapter 2 will describe EAP and IEEE 802.11 in detail, but here we nevertheless give a very brief description of these protocols so as to illustrate the main results of the thesis. Hopefully, Wi-Fi, and thus IEEE 802.11, should be well-known to everyone: a wireless client and an access point use a shared secret, typically a password, to protect the wireless link between them. This involves an initial key exchange phase, where the client and access point derive a cryptographic key from the common secret, and a channel encryption phase, where the application data is being sent. At the same time, IEEE 802.11 can also be used in situations where the client and access point do not share a common secret beforehand. This is exactly the setting of the eduroam network mentioned above. Here, they will first use a trusted third-party to help them establish a common secret. The protocol used to facilitate this is EAP.

EAP specifies a way for two parties to establish a common secret through the help of a trusted third-party. However, rather than viewing EAP as a single protocol, it can be better thought of as a protocol *framework* used to compose other concrete protocols. For the EAP framework to be secure the concrete protocols need to be safely instantiated, but EAP itself does not specify them. IEEE 802.11 is commonly used as one of the concrete sub-protocols in the EAP

[1] `https://wigle.net/`.
[2] `https://www.eduroam.org/`
[3] `https://www.eduroam.org/2017/03/07/2016-a-record-breaking-year-for-eduroam/`

framework, but it does not have to be; EAP is mostly protocol agnostic.

Given these high-level descriptions of EAP and IEEE 802.11, our results can be summarized as follows. Below we refer to security notions such as authenticated key exchange and secure channel protocols only informally. Their formal definitions will be made precise in Chapter 3.

- Our first result is a game-based security analysis of the general EAP framework. This involves two generic composition theorems that abstract away the concrete protocols used within EAP. Instead, the theorems establish sufficient criteria on the protocol building blocks in order for the EAP framework to be instantiated securely. The overall security goal of EAP that we aim for is that of a three-party authenticated key exchange. Having proven the soundness of the general EAP framework, it remains to find suitable concrete protocols that satisfy the security criteria laid down by the composition results.

- One such concrete protocol is EAP-TLS [RFC5216], which within the EAP framework is used between the client and the trusted third-party. We prove that EAP-TLS is a secure two-party authenticated key exchange, which is sufficient in order to be used in our compositions results. However, this result also has independent interest outside of the EAP framework, because of the way it is established. Essentially, we give a generic transformation that shows how secure channel protocols can be turned into secure key exchange protocols by exporting additional session keys from their handshake protocols. This has applications to the practice of exporting extra keys from the TLS handshake, since TLS is a secure channel protocol, but *not* a secure key exchange protocol (we return to this point in Chapter 3).

- Finally, we analyze the IEEE 802.11 protocol. Again, this analysis has independent interest outside of the EAP framework, since IEEE 802.11 is often used without EAP. Recall from our brief description above that IEEE 802.11 proper consists of a key exchange phase followed by a channel encryption phase. We prove that the former constitutes a secure two-party authenticated key exchange protocol, and that the latter satisfies the notion of a secure stateful authenticated encryption scheme. Although these results are of independent interest, they also combine with our EAP composition theorems to culminate in our biggest main result: namely the security of EAP and IEEE 802.11 used together.

The results outlined above roughly correspond to the contents of Chapter 4, Chapter 5, and Chapter 6, respectively.

Modularity. A common theme among all the results established in this thesis is an emphasis on reusing existing security results as far as possible. For example, the TLS protocol is an important component in both EAP and EAP-TLS, but we do not want to redo any analysis of TLS for the purposes of establishing our results. Instead, we want to be able to leverage the large amount of already existing analysis of TLS in a black-box manner. This requires generic and modular results, but it also requires security models that are comparable. This is one of the reasons why we have chosen to use a game-based formulation of security over a simulation-based formulation. Many of the existing results on the real-world protocols we care about, such as TLS, IPsec, and SSH, are for the most part proven in a game-based setting.

1.2.1 Publications

The material in this thesis is primarily based on the following two papers:

[BJ17] Chris Brzuska and Håkon Jacobsen. "A Modular Security Analysis of EAP and IEEE 802.11". In: *PKC 2017: 20th International Conference on Theory and Practice of Public Key Cryptography, Part II*. ed. by Serge Fehr. Vol. 10175. Lecture Notes in Computer Science. Amsterdam, The Netherlands: Springer, Heidelberg, Germany, Mar. 2017, pp. 335–365.

[BJS16] Christina Brzuska, Håkon Jacobsen, and Douglas Stebila. "Safely Exporting Keys from Secure Channels: On the Security of EAP-TLS and TLS Key Exporters". In: *Advances in Cryptology – EUROCRYPT 2016, Part I*. ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9665. Lecture Notes in Computer Science. Vienna, Austria: Springer, Heidelberg, Germany, May 2016, pp. 670–698. DOI: 10.1007/978-3-662-49890-3_26.

Specifically, the material found in Chapter 4 and Chapter 6 of this thesis is taken from [BJ17], while the material in Chapter 5 comes from [BJS16]. However, the content as it appears in this thesis has undergone major revisions compared to the original publications. Moreover, this thesis also introduces some new material not found in either of the published papers. In particular, Section 6.3 and Section 6.4 present some additional results and discussions on IEEE 802.11.

1.2.2 Outline of thesis

In Chapter 2 we give detailed protocol descriptions of EAP and IEEE 802.11. In Chapter 3 we provide the formal framework that will be used to analyze EAP,

EAP-TLS and IEEE 802.11 in the later chapters. Since our analyses cover a wide range of different protocols, a great number of definitions and notions are needed. We have tried to discuss and justify all of our definitional choices to the greatest extent possible.

In Chapter 4 we conduct our first security analysis, beginning with the general EAP framework. The main results of the chapter are two modular and generic protocol composition theorems. Then, in Chapter 5, we analyze one specific component in the EAP framework, namely the EAP-TLS protocol. However, although the starting point is the concrete EAP-TLS protocol, the main result of the chapter is again a generic result with applications beyond the immediate scope of EAP-TLS. Following this, in Chapter 6 we analyze the IEEE 802.11 protocol. The main technical result is an analysis of the IEEE 802.11 key exchange protocol when considered as a standalone protocol—as it is typically used in home networks. However, the result additionally combines with the composition theorems of Chapter 4 to yield a result for IEEE 802.11 combined with EAP. Furthermore, Chapter 6 also presents some new material on IEEE 802.11 which have not appeared elsewhere, including and analysis of the IEEE 802.11 data encryption algorithm called CCMP, as well as a discussion of the multi-ciphersuite and negotiation security of IEEE 802.11.

Finally, in Chapter 7 we conclude the thesis by putting our work in a larger context and discussing some of its limitations. We also point out some possible directions for future work.

Note. We use the symbol "▲" to denote the end of a remark or example, and use the symbol "■" to denote the end of a proof.

Chapter 2

Description of EAP and IEEE 802.11

Contents

This chapter describe EAP and IEEE 802.11 in detail from a functional perspective. In later chapters we will analyze their security.

2.1 EAP

The Extensible Authentication Protocol (EAP) is an authentication framework used to provide network access control. It is defined by the IETF in the base standard RFC3748 [RFC3748], but a large number of supporting RFCs also update or extend the base standard further in various ways.

The purpose of EAP is to provide central management of authentication in a network with many clients and network connection points. Specifically, EAP considers a setting consisting of three principal entities: *clients*, *authenticators*, and *authentication servers*. The clients are regular users that want to get access to the network using a device such as a laptop or a smartphone. The

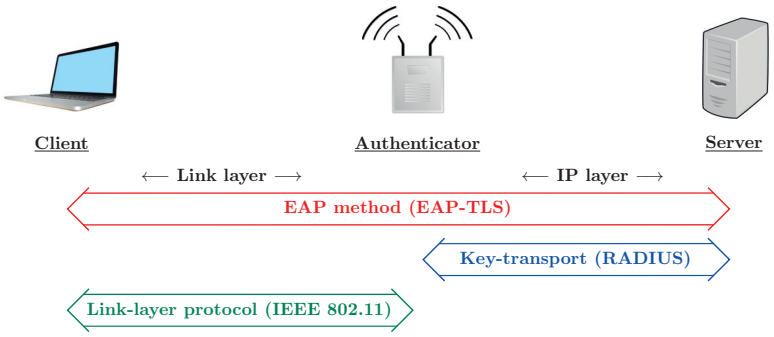


Figure 2.1: The three-party EAP architecture. Concrete example protocols shown in parenthesis.

authenticators control access to the network and are typically implemented in network devices such as switches and wireless access points. Authorized clients will be granted access to the network by the authenticators, unauthorized clients will be blocked.

The main difficulty of this scenario is that the clients and the authenticators do not have any common credentials *a priori*. In order to make their access control decisions, the authenticators will consult with an authentication server, which stores the credentials of every user that is authorized to access the network. On a network there will typically be many clients and authenticators, but only a few authentication servers.

Remark 2.1. Within the EAP standard [RFC3748] the client is usually referred to as the *peer* or the *supplicant*, but in this thesis we will be using the word *client* exclusively. Moreover, due the visual resemblance between the words "authenticator" and "authentication", from now on we will refer to authentication servers simply as *servers* in order to avoid confusion. ▲

EAP architecture. The general EAP architecture is shown in Figure 2.1. The exchange begins when a client wants to connect to a network controlled by an authenticator. Since the client and the authenticator do not share any common credentials, the idea is to first have the client authenticate itself towards the server and then let the server vouch for the client towards the authenticator. The client and the server can use any authentication method they like in order to authenticate each other. However, in order to do this in a uniform manner across different authentication methods, EAP defines four generic message types that are used to encapsulate the concrete authentication protocol. The four message types are `Request`, `Response`, `Success` and `Failure`, respectively.

The combination of a concrete authentication method, say TLS or IPsec, together with its encapsulation inside the generic EAP message types, is called an *EAP method*.

EAP methods. Individual EAP methods are defined in separate RFC documents that specifies how the concrete authentication method is to be used within the EAP framework. For example, RFC5216 [RFC5216] defines the EAP-TLS method, which provides certificate-based mutual authentication between the client and the server based on the TLS protocol. Numerous other EAP methods have also been defined; see Table 2.1 for a few examples.

Although EAP defines message formats in the form of `Request`, `Response`, `Success`, and `Failure` messages, it does not specify how these message should be transmitted through the network. For this, EAP depends on some lower-layer protocol to take care of the actual delivery of the messages. Thus, EAP messages will themselves be encapsulated inside other transport protocols. For example, when passing EAP messages over a LAN, a protocol known as *EAPOL* (*EAP over LAN*) is typically used. EAPOL is defined in the IEEE 802.1X [IEEE 802.1X] standard. Notably, EAP does *not* require IP connectivity in order to be used.

Besides authentication, EAP methods usually also derive some shared keying material between the client and the server. The keying material (if derived) needs to be at least 512 bits long and is referred to as the *master session key (MSK)*. The server will transport the MSK to the authenticator, so that it can be used in a subsequent authentication step directly between the authenticator and the client (more on this below). We are going to assume that *all* EAP methods derive keying material in this thesis, even though some of the originally defined EAP methods, such as EAP-MD5 and EAP-OTP, do not support this feature.

For as long as the EAP method is being run between the client and the server, the authenticator operates in so-called *pass-through* mode. This means that it merely relays the messages between the client and the server. In fact, the authenticator can be completely oblivious as to which concrete authentication method is being used since the whole exchange is wrapped inside the generic EAP message types.

Key transport. Once the client has successfully authenticated itself towards the server using an EAP method, the server will communicate this fact to the authenticator using an EAP `Success` message. This EAP `Success` message will also contain the keying material that the server and the client agreed upon in the prior EAP method exchange. If the client failed to authenticate towards the server, the server will instead send an EAP `Failure` message to the

Table 2.1: Examples of standardized EAP methods.

EAP method	Description	Reference
EAP-TLS	TLS-based mutual authentication using certificates.	[RFC5216]
EAP-TTLS	Tunneled TLS. Certificate-based authentication from server to client, followed by an arbitrary authentication method from client to server *inside* the established TLS tunnel.	[RFC5281]
PEAP	Similar to EAP-TTLS, but where the inner authentication method is the password-based protocol MS-CHAPv2 [RFC2759]	[PEAPv2]
EAP-IKEv2	IKEv2-based [RFC7296] authentication. Credentials can be based on certificates, pre-shared keys or passwords.	[RFC5106]
EAP-AKA	Authentication based on the Authentication and Key Agreement (AKA) protocol used in 3G and 4G mobile networks. Trust relationships are based on symmetric keys stored in a SIM card on the client side.	[RFC4187]
EAP-GTC	Authentication based on generic token cards and one-time passwords.	[RFC3748]
EAP-PSK	A lightweight authentication method based on PSKs.	[RFC4764]

authenticator and abort the exchange.

Since the server transports sensitive data such as keying material to the authenticator, the security of the connection between the server and the authenticator is also of great importance. However, just like the EAP standard does not mandate a single concrete authentication method to be used between the client and the server, it similarly does not mandate any particular protocol to be used between the server and the authenticator. Thus, implementors are free to choose whatever protocol they want as long as it supports the features required by the EAP framework. Still, in practice, the *de facto* standard is the RADIUS protocol [RFC2865] (and to some lesser extent its successor Diameter [RFC6733]). In fact, it is not uncommon to call the authentication server in the EAP framework a "RADIUS server". Note that RADIUS and Diameter are also used for purposes other than authentication and authorization, for example accounting, metering, and billing of network services. Because of this, RADIUS and Diameter are more generally referred to as *Authentication, Authorization, and Accounting (AAA)* protocols.

Link-layer protocol. Once the master session key MSK has been delivered from the server to the authenticator, the EAP exchange is technically com-

plete. At this point the client and the authenticator are both in possession of the same MSK. Since they could have obtained the same MSK only if they were both trusted by the server, they have also implicitly authenticated each other. However, rather than using the MSK directly to encrypt their subsequent communication, the client and the authenticator will instead use the MSK as input to a lower-layer authentication and key exchange protocol. This protocol directly authenticates the client and the authenticator to each other using the MSK as a shared key, in addition to deriving temporal encryption keys to protect their communication.

Again, the choice of authentication protocol to run between the client and the authenticator is independent of EAP and usually depends on the physical medium being used between them. Recall that the authenticators are normally implemented in devices such as switches and wireless access points. These devices operate at the link-layer in the network stack, so the authentication protocol between the client and the authenticator will also take place at this layer, as shown in Figure 2.1. While many different link-layer protocols exist, in this thesis we will primarily focus on the IEEE 802.11 [IEEE 802.11] protocol used in wireless LANs (Wi-Fi). IEEE 802.11 will be described in detail in the next section.

Example 2.2. Given the large number of acronyms and different protocols used within the EAP architecture, it might be helpful to look at a concrete example to see how all the different pieces fit together. As a use case, we consider the *eduroam* network. Recall from Chapter 1 that eduroam is a roaming service provided to students and employees of educational institutions and research organizations around the world. eduroam allows users from any participating institution to automatically connect to the eduroam network using a single set of credentials, even if visiting a different institution then their own, i.e., when *roaming*. To achieve this, eduroam uses EAP with a hierarchical network of RADIUS servers. However, in order to keep the example simple, we will only look at the case of a non-roaming user; that is, a user that wants to connect to the eduroam network at their home institution.

Suppose Alice is a student at the NTNU university, who wants to connect to the eduroam network. At the NTNU campus there are many wireless access points broadcasting the eduroam network identifier, and Alice can connect to any one of them. However, none of the access points have any *a priori* knowledge of Alice. Instead, NTNU maintains a central RADIUS server containing the credentials of all its users, including Alice. In this example we are going to assume that all users at NTNU are issued client certificates which uniquely identify them. Conversely, the RADIUS server also has a certificate of its own which is trusted by all its users. Furthermore, while the access points have no shared credentials with any of the users, they each share a (unique) long-term

symmetric key with the RADIUS server. Referring back to Figure 2.1 we thus have the following situation: Alice corresponds to the client, the access point is the authenticator, and the NTNU RADIUS server is the server.

When Alice wants to connect to the eduroam network, she first associates to a wireless access point broadcasting the eduroam network identifier. The access point will now ask Alice to identify herself using an EAP `Request` message. Since the communication between Alice and the access point takes place over the link-layer, the EAP `Request` message is encapsulated inside an EAPOL protocol frame. On receiving the EAP `Request` message, Alice responds with an EAP `Response` message (again encapsulated inside EAPOL) containing her username: `alice@ntnu.no`. The access point will now forward this EAP `Response` message to the NTNU RADIUS server by encapsulating it inside a RADIUS packet.[1] The RADIUS message will itself be transferred over IP.

Following the receipt of the EAP `Response` message, Alice and the RADIUS server will initiate an EAP-TLS exchange, using their respective certificates to authenticate each other. Alice will use EAP `Response` messages over EAPOL, while the server will use EAP `Request` messages over RADIUS. All of the messages pass through the access point, which continuously de-encapsulates the EAPOL frames coming from Alice, and re-encapsulates the containing EAP messages as RADIUS messages towards the server (and *vice versa*).

Once the EAP-TLS exchange is complete, corresponding to the red part in Figure 2.1, Alice and the RADIUS server are in possession of a shared key MSK exported by the EAP-TLS method. In order to securely transfer the MSK from the RADIUS server to the access point, the RADIUS protocol specifies a custom encryption scheme based on the Microsoft Point-to-Point Encryption (MPPE) protocol [RFC2548]. Basically, this encryption scheme uses the long-term secret shared between the access point and the RADIUS server to derive a key-stream which is XOR'ed together with the MSK. Using this method, the RADIUS server transfers the MSK to the access point in addition to an EAP `Success` message to indicate that its EAP-TLS exchange with Alice completed successfully. This is shown as the blue part in Figure 2.1.

Finally, Alice and the access point use the MSK as input to the IEEE 802.11 handshake protocol, which they now run directly between themselves. This is shown as the green part in Figure 2.1. The IEEE 802.11 protocol will be described in detail in the next section, but the result is that Alice and the access point prove mutual possession of the MSK, and derive an encryption key to protect their subsequent communication. Since Alice and the access point could only have obtained the same MSK if they have a mutual trust relationship with the RADIUS server, this implicitly proves that they are both authorized

[1] RADIUS is a challenge-response protocol just like EAP, having its own set of generic messages (called `Access/Request` in RADIUS).

members of the eduroam network. At this point Alice is allowed to access the eduroam network by the access point. ▲

2.2 IEEE 802.11

IEEE 802.11 [IEEE 802.11] is the most widely used standard for creating wireless local area networks (WLANs). IEEE 802.11 defines a set of specifications for the physical and medium access control (MAC) layer, describing how wireless devices within a WLAN can achieve connectivity. IEEE 802.11 supports three modes of operation depending on the network topology: infrastructure mode, ad-hoc mode, and mesh network mode.

Infrastructure mode is the most common topology currently in use, and involves one or more access points that coordinate the communication within the WLAN. In particular, in infrastructure mode all client traffic must pass through the access points. The access points usually also provide the clients with connectivity to a larger network, such as the Internet. Conversely, in ad-hoc and mesh-networking mode there is no central infrastructure. Wireless clients talk directly to each other and there might be no connectivity to a larger network. This thesis will only focus on the infrastructure mode of operation.

2.2.1 IEEE 802.11 basics

Most of the IEEE 802.11 standard is not directly concerned with security, but instead deals with communication and transmission aspects such as the choice of radio modulation, transfer rates, and frequency spectrums. In this section we give a very brief description of the IEEE 802.11 protocol in infrastructure mode from a non-security perspective, providing details only to the extent it will be needed for the rest of the thesis.

An IEEE 802.11 network in infrastructure mode is identified by its *Service Set Identifier (SSID)*. This is the network name that an access point broadcasts. Multiple interconnected access points may advertise the same SSID to form what is know as an *Extended Service Set (ESS)*. The union of all the access points advertising the same SSID forms an ESS. Note that a single access point might broadcast several SSIDs at the same time, hence serving multiple ESSs simultaneously. An access point broadcasts all the SSIDs it serves at regular intervals in short messages called *beacons*. The beacon messages allow wireless devices to discover the presence of a network by scanning the frequency bands on which they are sent.

Before a wireless client can send or receive data from an SSID served by an access point, it first needs to *associate* with the access point. This process includes presenting the access point with its *media access control (MAC)* address

so that the access point can address future messages directly to it. A MAC address is normally unique per physical network card, but it can be changed in software. We will usually refer to MAC addresses as *link-layer* addresses or *physical* addresses in order to avoid confusion with the cryptographic concept of a *message authentication code*.

Messages sent over a WLAN are called *frames*. All frames have a fixed format consisting of an IEEE 802.11 header, a frame body containing the application data, and an error-correcting code. Apart from the fact that the header includes the link-layer addresses of the sender and the receiver, we will not the describe the IEEE 802.11 header in any detail since it has no importance for security. An IEEE 802.11 frame can have a maximum size of roughly 8 kB, but is usually smaller; around 200–2000 bytes.

Like in the EAP standard [RFC3748], the clients in IEEE 802.11 [IEEE 802.11] are generally referred to as *supplicants*. However, we will only be using the word *clients*.

2.2.2 A brief history of security in IEEE 802.11

There have been several different security protocols defined within the IEEE 802.11 standard. Originally, the only security protocol defined for IEEE 802.11 networks was the *Wired Equivalent Privacy (WEP)* protocol, which revolved around the stream cipher RC4. After its introduction in 1997 there have been discovered flaws in virtually every part of WEP's design. Today the protocol can be broken within a matter of seconds. See [Wal00, FMS01, BGW01, SIR02, Cam+03, SIR04, Mis+04, BHL06, Tew07, TWP08, TB09, MT11, Sep+14] for some of the existing analysis of WEP.

As an interim solution until a long-term replacement for WEP could be defined by the IEEE, an industry consortium called the Wi-Fi Alliance[2] designed the *Temporal Key Integrity Protocol (TKIP)*. While officially called TKIP, it is probably better known under its marketing name *Wi-Fi Protected Access (WPA)*. An important design requirement for TKIP/WPA was that it should be able to run on the same legacy hardware as WEP in order to facilitate easy upgrades of existing IEEE 802.11 deployments. In particular, this led TKIP/WPA to reuse RC4 as its algorithm of choice for bulk data encryption. TKIP/WPA has received quite a bit of analysis [MRH04, Woo04, TB09, SVV11, Hal+09, MT11, Tod+12, VP13, Gup+15, PPS15, IM15, VP15]. Both WEP and TKIP are today deprecated by the IEEE.

Ultimately, the long-term replacement for WEP was specified by the IEEE in a 2004 amendment to the original IEEE 802.11 standard, denoted IEEE 802.11i

[2] `https://www.wi-fi.org/`

[IEEE 802.11i]. This amendment defines the concept of a *Robust Security Network (RSN)* which specifies the security capabilities that a wireless device needs to support. There are two main components to an RSN: a key establishment protocol called the *4-Way Handshake (4WHS)*; and a bulk data encryption algorithm which must either be TKIP or a new algorithm defined in IEEE 802.11i based on AES, called the *Counter Mode Cipher Block Chaining Message Authentication Code Protocol (CCMP)*. A client and an access point will first use the 4WHS protocol to establish a temporal session key, and then use this key with the CCMP encryption algorithm to protect the subsequent communication. We will explain the 4WHS and CCMP in more detail in Section 2.2.3.

Since IEEE 802.11 also supports multicast and broadcast communication, IEEE 802.11i additionally specifies a *Group Key Handshake*. The Group Key Handshake is used to establish a common (temporal) group key among all the devices currently connected to the WLAN. The common group key is used with CCMP to protect all multicast and broadcast messages within the WLAN.

Like TKIP, IEEE 802.11i and RSN are probably better known under the marketing name *Wi-Fi Protected Access 2 (WPA2)*. Compared to WEP and TKIP, there has been much less cryptanalysis of RSN/WPA2. Most existing analyses have focused on the 4WHS protocol and its susceptibility to dictionary attacks when using password-based authentication [Joh+15, Kam+16]. In particular, if the long-term key is derived from a low-entropy password, then a passive observer of the 4WHS can conduct an off-line dictionary attack in order to recover the password.[3] Besides dictionary attacks, there have also been a number of DoS attacks against the 4WHS [HM04, HM05, RLM06, Eia09, Eia10, EM12], as well as attacks focusing on various implementation aspects of RSN/WPA2 [Cas+13, VP16].

Finally, we note that in addition to IEEE 802.11i, there have been several other security-relevant amendments to the IEEE 802.11 standard as well. For example, amendment IEEE 802.11w [IEEE 802.11w] defines procedures for protection of management frames; amendment IEEE 802.11s [IEEE 802.11s] defines security in mesh networking (including a new password-based key establishment protocol called *Simultaneous Authentication of Equals (SAE)* [Har08], as well as using the AES-SIV mode of operation [RFC5297] for the protection of mesh management frames)); and amendment IEEE 802.11r [IEEE 802.11r] defines security procedures for fast transitioning between access points. All the above amendments have been incorporated into the current full IEEE 802.11 standard [IEEE 802.11].

In this thesis we will only focus on the RSN security procedures introduced in amendment IEEE 802.11i. Specifically, when in the following we talk about

[3] For instance the open-source tool `aircrack-ng` (`https://www.aircrack-ng.org/`) incorporates such an attack in an easy-to-use command-line program.

the security of IEEE 802.11, we mean the collection of the 4WHS, CCMP, and the Group Key Handshake defined in the current IEEE 802.11 standard [IEEE 802.11], excluding TKIP. In fact, most of our security analysis will be focused on the 4WHS protocol. For the remainder of the thesis we will use the terms IEEE 802.11, RSN and WPA2 interchangeably to refer to the security protocols that were introduced in amendment IEEE 802.11i.

2.2.3 Detailed description of the IEEE 802.11 security protocol

IEEE 802.11 in infrastructure mode is either a two-party protocol involving a wireless *client* and an *access point*, or a three-party protocol which additionally includes a trusted *server*. The goal is for the client and access point to establish a Robust Security Network (RSN) association, which involves running the 4WHS key exchange protocol to establish a session key, and using the CCMP encryption scheme to protect their data. The 4WHS protocol needs a shared symmetric key, which can either be configured as a pre-shared key (PSK) on both the client and the access point, or be derived from some process involving the trusted server. Which protocol to use for this purpose is technically outside the scope of the IEEE 802.11 standard [IEEE 802.11], but in practice it is usually based on EAP. In any case, when a third-party server is involved in establishing the shared key for the 4WHS protocol, we call it *IEEE 802.11 with upper-layer authentication*. The complete IEEE 802.11 establishment procedures consist of six stages and are shown in Figure 2.2.

Stage 1. Network and Security Capability Discovery. In this stage the client discovers available networks and their security capabilities. As mentioned in Section 2.2.1, an access point will advertise its presence by regularly broadcasting so-called beacon frames (Message (1) in Figure 2.2). A beacon frame contains the network SSID as well as all the capabilities supported by the access point. In particular, this includes the security protocols it is willing to use (WEP, TKIP, RSN), together with a list of *ciphersuites* $\overrightarrow{\mathtt{CS}}$ that it supports. An IEEE 802.11 ciphersuite specifies a collection of algorithms which is used either to protect the handshake itself, or the application data. We will specify the various algorithms supported by IEEE 802.11 when we describe the 4WHS protocol in Stage 4. Any client can learn the capabilities supported by an access point by passively listening for the information contained in the beacon frames. Alternatively, a client can actively ask for it by sending a probe request message (Message (2) in Figure 2.2). An access point that receives a probe request message will reply with a probe response message (Message (3)) containing the same information as in its beacon frame.

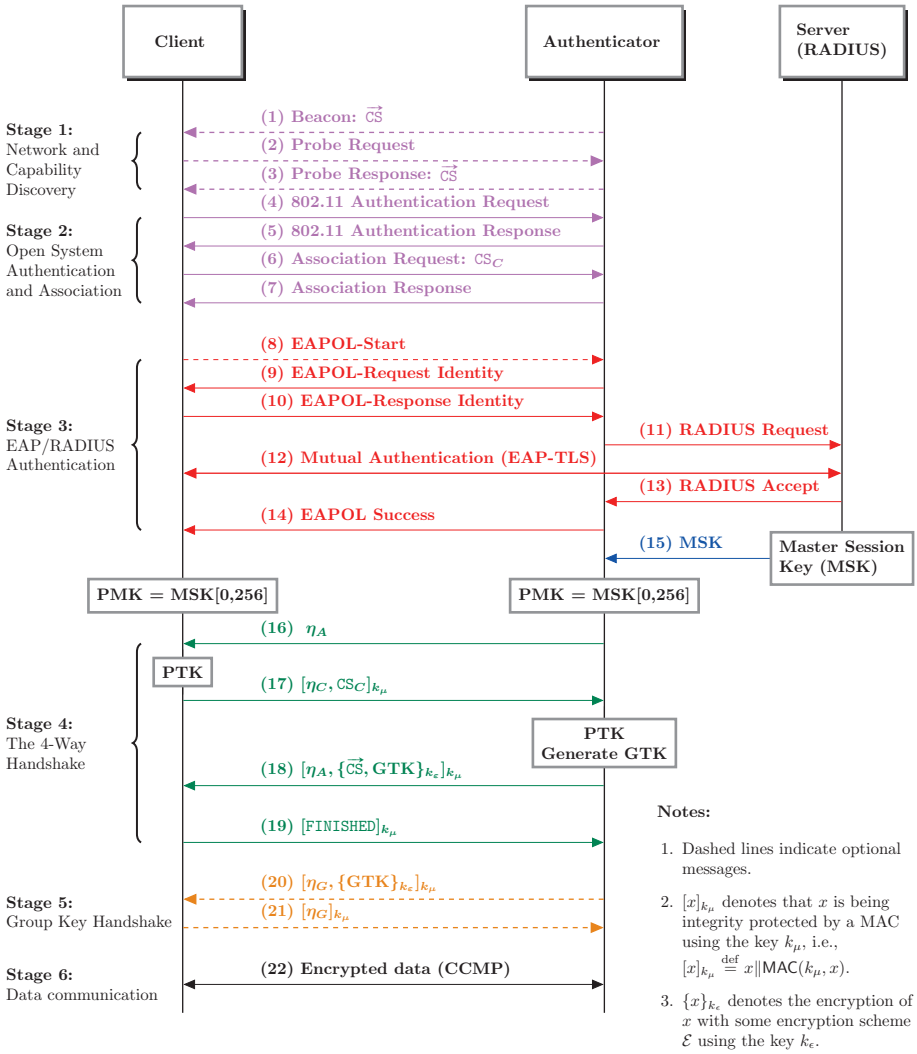


Figure 2.2: The IEEE 802.11 protocol in infrastructure mode. Diagram adapted from [HM05].

Stage 2. Open System Authentication and Association. In this stage the client selects the access point it wants to connect to. The first step involves a procedure called Open System Authentication (Message (4) and Message (5) in Figure 2.2). In terms of security this is a null operation; it is included simply to maintain backward compatibility with previous IEEE 802.11 specifications. The second step is client association as described in Section 2.2.1. The client sends an association request message (Message (6) in Figure 2.2) that specifies which of the capabilities of the access point it wants to use. In particular, this involves selecting a ciphersuite from the list $\overrightarrow{\mathtt{CS}}$ that the access point broadcast earlier. The ciphersuite chosen by the client is denoted $\mathtt{CS}_C$. Additionally, the client also indicates whether a PSK or upper-layer EAP authentication will be used in the following authentication stages. Provided the access point finds the client's choices acceptable, it replies with an association response message (Message (7)) and continues to the next stage of the protocol. If a pre-shared key is used for authentication, then Stage 3 as described below is omitted, and the protocol continues directly to Stage 4.

Stage 3. Upper-layer Authentication. When upper-layer authentication is being used, the client authenticates itself towards a trusted server, usually using EAP as described in Section 2.1. In Figure 2.2 we have assumed that EAP-TLS is the EAP method being used between the client and the server, and that RADIUS is being used as the key transport protocol between the server and the access point. The whole exchange is shown as Messages (8)–(15) in Figure 2.2, although note that Message (12) really constitutes several messages. The end result of a successful run of EAP is that a shared master session key MSK is distributed to both the client and the access point. The MSK will be used as the shared key input for the 4WHS protocol in Stage 4.

Stage 4. The 4-Way Handshake (4WHS). In this stage the client and the access point run the 4WHS protocol in order to authenticate each other, as well as to derive temporary session keys for protecting their subsequent communication. The 4WHS, shown in Messages (16)–(19) in Figure 2.2, is based on a shared symmetric key called the *pairwise master key (PMK)*.

If EAP was used in Stage 3 to distribute an MSK to the client and the access point, then the PMK is set to be the first 256 bits of the MSK (recall from Section 2.1 that the keying material exported by an EAP method needs to be at least 512 bits long). Otherwise, if no upper-level EAP authentication is being used, the PMK is a pre-shared key installed manually at the client and the access point. Usually, this pre-shared key is derived from a password using the password-based key derivation function PBKDF2 [RFC8018], but it can also be created in other ways.

Regardless of how the PMK was obtained, the 4WHS protocol proceeds as follows. In the first handshake message (Message (16) in Figure 2.2) the access point sends a nonce η_A to the client. On receiving this message, the client creates its own nonce η_C and derives a *Pairwise Transient Key (PTK)*, computed in the following way. Let $\hat{U}$ denote the 48 bit physical MAC address of a user U, and let min and max denote functions that compute, respectively, the smallest and largest of two MAC address based on their numerical values when treated as 48 bit unsigned integers. Then

$$\text{PTK} = k_\mu \| k_\varepsilon \| k_\alpha \leftarrow \mathsf{PRF}(\text{PMK}, \text{“}\texttt{Pairwise key expansion}\text{”}, \hat{P} \| \eta), \quad (2.1)$$

where $\hat{P} \leftarrow \min\{\hat{A}, \hat{C}\} \| \max\{\hat{A}, \hat{C}\}$ is the combination of the client's ($\hat{C}$) and access point's ($\hat{A}$) physical addresses, and $\eta \leftarrow \min\{\eta_A, \eta_C\} \| \max\{\eta_A, \eta_C\}$ is the combination of their nonces. The pseudorandom function PRF is based on HMAC [RFC2104]. The PTK is parsed into three sub-keys k_μ, k_ε, and k_α, having the following purposes:

- k_μ – this is a key for a message authentication code (MAC) used to provide integrity of the handshake messages.
- k_ε – this is an encryption key used to protect the distribution of a *Group Transient Key (GTK)* inside the 4WHS (see below), or in a dedicated Group Key Handshake step (see Stage 5).
- k_α – this is the session key used to encrypt the bulk data traffic in Stage 6.

After computing the PTK, the client creates the second protocol message of the 4WHS (Message (17) in Figure 2.2). This message contains the client's nonce η_C, as well as the ciphersuite $\texttt{CS}_C$ that it selected during the association step in Stage 2. The integrity of the entire message is protected by a MAC keyed with k_μ. The precise MAC algorithm to use is determined by the ciphersuite $\texttt{CS}_C$ that was chosen in Stage 2. The IEEE 802.11 standard specifies three legal MAC algorithms: HMAC-MD5 [RFC2104] (deprecated), HMAC-SHA1-128 [RFC2104], and AES-128-CMAC [FIPS:SP-800-38B].

On receiving the second handshake message, the access point first extracts the client's nonce η_C and derives the PTK according to Equation (2.1). Using the derived PTK, the access point first checks the validity of the MAC tag on the message, and compares the included ciphersuite $\texttt{CS}_C$ with the one it received during the association request in Stage 2 (Message (4)).

If the verification is *not* successful, then the access point silently discards the message. Otherwise, the access point creates the third handshake message of the 4WHS (Message (18) in Figure 2.2). This message includes: (i) the nonce η_A that the access point sent in its previous handshake message (Message (16));

(ii) the list of ciphersuites $\overrightarrow{\mathtt{CS}}$ the access point advertised in Stage 1 of the IEEE 802.11 establishment procedures; and (iii) a group key GTK. The two latter values are encrypted with an encryption scheme $\mathcal{E}$ using the key k_ε, where the choice of encryption scheme is again determined by the selected ciphersuite $\mathtt{CS}_C$. The IEEE 802.11 standard specifies two legal encryption algorithms: RC4 (deprecated) and NIST AES Key Wrap [RFC3394]. The integrity of the entire message is protected by a MAC keyed with k_μ.

On receiving the third handshake message, the client first decrypts (with k_ε) the list of ciphersuites $\overrightarrow{\mathtt{CS}}$ and the group key GTK. If the ciphersuite list does not match what the access point broadcast in Stage 1, then the client aborts the protocol. Otherwise, the client proceeds by verifying the MAC tag. If the verification was successful, then the client creates the fourth and final message of the handshake (Message (19) in Figure 2.2). If the verification was not successful, then the client silently discards the message.

Remark 2.3. Some additional points about the 4WHS are worth emphasis.

- (No forward secrecy) The 4WHS does not provide forward secrecy. Anyone who knows the PMK and observes the nonces η_A and η_C can compute the PTK. Additionally, if the PMK is derived from a low-entropy password, then the PMK is subject to off-line dictionary attacks. As mentioned in Section 2.2.2, most existing security analyses of WPA2 have focused on this aspect of the 4WHS.

- (Replay protection mechanism) The 4WHS employs a somewhat unusual approach for protecting against replay attacks. Instead of explicitly acknowledging a nonce by repeating it in a following response message, the 4WHS instead mixes η_A and η_C into the derivation of the PTK. Replays are then detected implicitly by MAC verification failures.

- (Downgrade protection) To protect against ciphersuite downgrade attacks, the second and third messages of the 4WHS repeat the ciphersuites that were advertised earlier in the IEEE 802.11 establishment procedures (i.e., Message (1), (3) and (6) in Figure 2.2). However, note that if WEP is enabled alongside RSN/WPA/WPA2, then this downgrade protection can easily be bypassed by an attacker. Namely, since WEP does not involve running the 4WHS protocol at all, an attacker can remove the option of WPA/WPA2 from the access point's beacon and probe request messages, leading the client to believe that only WEP is supported. Since no subsequent ciphersuite verification is being done in this case, the downgrade will not be detected.

- (GTK selection) The group key GTK is chosen solely by the access point without any input from the clients. Although the IEEE 802.11 standard

suggests deriving the GTK from a *Group Master Key (GMK)*, the only formal requirement on the GTK is that it should be a random number.

▲

Stage 5. Group Key Handshake. This is an optional stage, providing a (new) group key (GTK) to all clients that are currently associated to the access point and have completed the 4WHS. The GTK is used to protect broadcast and multicast messages within the WLAN. The access point distributes the GTK to each client one by one, using their individually shared PTKs to protect the Group Key Handshake message carrying the GTK. The encryption and MAC algorithms used to protect the group handshake messages are the same as those used for the 4WHS. Note that the access point also includes a nonce η_G in its group handshake message (Message (20) in Figure 2.2) which the client is required to acknowledge (Message (21)).

Stage 6. Application Data. The final stage of the IEEE 802.11 protocol is the actual transmission of application data. Messages are protected by one of the two encryption algorithms TKIP and CCMP using the k_α sub-key of the PTK. Since TKIP is deprecated by the IEEE 802.11 standard, we only explain CCMP to some extent here. CCMP is a stateful authenticated encryption scheme based on the block cipher AES [FIPS:197-2001]. It ensures data confidentiality, integrity, and replay protection using the CCM mode of operation [RFC3610] to encrypt each frame. CCM itself is a combination of counter mode encryption with CBC MAC. CCMP will be explained in greater detail when we analyze it in Section 6.3.

Chapter 3

Formal models

Contents

In this chapter we define the formal security models that will be used to prove our results on EAP, EAP-TLS and IEEE 802.11 in the later chapters. We seek to establish two main definitions: the security of an *authenticated key exchange (AKE)* protocol and the security of an *authenticated and confidential channel establishment (ACCE)* protocol. EAP, EAP-TLS and the IEEE 802.11 4WHS protocol are all naturally modeled as AKE protocols. In fact, since EAP, EAP-TLS and the 4WHS all achieve different levels of security, we will actually

define *three* AKE models of varying strengths. ACCE protocols will be used as important building blocks in our analyses of EAP and EAP-TLS. Definitions of standard primitives, like pseudorandom functions and MACs, are provided in Appendix A.

3.1 Notation and preliminaries

For $m, n \in \mathbf{N}$ and $m \leq n$, let $[m, n] \stackrel{\text{def}}{=} \{m, m+1, \ldots, n\}$ and $[n] \stackrel{\text{def}}{=} [1, n]$. 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 random value according to the distribution X. If S is a finite set, then $x \twoheadleftarrow S$ means to sample x uniformly at random from S. We write $X \leftarrow X \cup x$ for adding an element x to a set X. The set of all bitstrings of length n is denoted by $\{0,1\}^n$ and the set of all finite length bitstrings is denoted by $\{0,1\}^*$. The string of zero length is denoted ε. The concatenation of two bitstrings x and y is written $x \| y$. Algorithms are in general randomized and we let $y \twoheadleftarrow A(x_1, \ldots, x_n)$ denote running the (possibly randomized) algorithm A on inputs $x_1, \ldots, x_n$, assigning A's output to the variable y. We write A^O for an algorithm being given *oracle access* to a function or algorithm $O(\cdot)$. If $\mathcal{O} = \{O_1, \ldots, O_t\}$ is a collection of functions or algorithms, then $A^{\mathcal{O}}$ means to give oracle access to all the O_i. We use a distinguished error symbol $\perp$ to denote cases where a computation might have failed, some value is missing, or if some precondition is not met.

3.1.1 Security games

All our security definitions are formulated in terms of formal experiments, called *games*. A game consists of an interaction between an *adversary* and an honest entity called the *challenger*. During a game, the adversary interacts with the challenger using a set of *queries*. The type of queries present, and how the challenger answers them, depends on the particular game. Associated to each game is one or more events that constitute the *winning conditions* of the game. A winning condition precisely defines what it means for an adversary to break a protocol and is meant to capture one or more of the intuitive security properties we might want a protocol to satisfy. Since both the adversary and the challenger will be probabilistic algorithms, a security game can also be thought of as a random variable over a probability space where the random coins of the challenger and adversary are drawn uniformly at random. In particular, the outcome of the game, i.e., whether the adversary has won or not, is a random variable on this probability space. Our formalization of games mostly follows the style of Shoup [Sho04], as opposed to the more syntactic version of Bellare and Rogaway [BR04].

Given that one has defined a formal security game, what does it mean for a protocol to be *secure*? Intuitively, a protocol is secure if any "efficient" adversary only has a "small" probability of satisfying the winning condition of the security game. In other words, a secure protocol provides the security property formalized by the winning condition. At the same time, it is important to remember that a security game is an abstraction of the real world. It represents an estimate of what we think the adversary might be able to do, as well as a hope that the associated winning condition truly models the security goal we set out to capture. Any statement about security always takes place in some choice of model, and this model is only an approximation of the real world.

3.1.2 Concrete vs. asymptotic security

In our informal definition of security we emphasized that adversaries should be "efficient" and their winning probabilities "small" but not necessarily zero. The reason for this is that most protocols cannot hope to achieve *unconditional* security in the face of *arbitrary* adversaries. But how do we define "efficient" and "small". There are two common approaches.

The first is the *asymptotic* approach, where "efficient" is equated with probabilistic polynomial-time (PPT) algorithms and "small" with *negligible* functions, where a function $g\colon \mathbf{N} \to \mathbf{R}$ is negligible if for all integers c there exists an integer N such that for all $n \geq N$, $g(n) < n^{-c}$. The asymptotic approach says nothing about a protocol's security for any particular choice of parameters. Instead, the adversary's winning probability, as well as its running time, is measured relative to some *security parameter* λ. A protocol is said to be (asymptotically) *secure* if for all PPT adversaries $\mathcal{A}$, the probability that $\mathcal{A}$ wins the security game is negligible in λ. The asymptotic approach has its roots in complexity theory and has been the traditional approach taken in cryptography, originating with the seminal work of Goldwasser and Micali [GM84].

The second approach, and the one we will be taking in this thesis, is called *concrete security*. It was originally introduced by Bellare, Killian, and Rogaway [BKR94]. In the concrete security approach one actually forgoes the whole question of defining "efficient" and "small" altogether. Instead, what is emphasized is the demonstration of an explicit *reduction* $\mathcal{R}$, which takes as input an adversary $\mathcal{A}$ that supposedly breaks the protocol, and transforms it into an algorithm that solves some other problem P. The reduction's success probability in solving problem P, as well as its resource usage, is explicitly expressed in terms of $\mathcal{A}$'s winning probability and resource usage (i.e., the number of queries it made in the security game). The conceptual idea of the reduction methodology is that if we believe that no "reasonable" algorithm can be found for solving problem P, then no algorithm for breaking the protocol can be found

either. However, the interpretation of "reasonable" is left to the user of the protocol to decide. Note that there are subtleties in what type of conclusions one can draw from a result expressed in the concrete security setting, especially when we know that efficient algorithms for solving P must *exist*, but we do not know how to actually to find them (see [Rog06, BL13]).

Whether to favor an asymptotic or a concrete security treatment depends on the application context. Asymptotic security is typically very useful when stating high-level results and feasibility results where the *qualitative* relationship between security notions is being emphasized. For example, the fact that one-way functions imply pseudorandom generators can be elegantly stated in the asymptotic language. Concrete security statements on the other hand are usually more precise, focusing on the *quantitative* relationship between notions. It promotes more application oriented results. Ultimately, the choice between asymptotic and concrete security is not fundamental. A concrete reduction can trivially be transformed into a statement about asymptotic security, and a proof showing that a protocol is asymptotically secure typically carries within it an explicit reduction.

A word on language. Technically speaking, since we are working in the concrete security setting, we cannot ever say that a scheme or protocol is actually *secure*. Unfortunately, this makes talking about our security results quite cumbersome. For instance, instead of being able to say things like "if scheme X is IND-CPA secure and scheme Y is EUF-CMA secure, then protocol Z is AKE secure", we need to say "given that algorithm $\mathcal{A}$ breaks protocol Z according to security game AKE, we can create explicit algorithms $\mathcal{B}_1$ and $\mathcal{B}_2$ that breaks scheme X according to security game IND-CPA, and scheme Y according to security game EUF-CMA, respectively". This quickly gets tedious. Thus, in our informal expositions we allow ourselves to use the first kind of statement rather than the second, safe in the knowledge that the reader can make the necessary translation in their head. However, we emphasize that all our formal definitions and theorem statements will be given in the second, precise form.

3.2 A unified protocol execution model

Our modeling of AKE and ACCE protocols follows the so-called *BR-model* which originates in the seminal work by Bellare and Rogaway [BR94], and includes a number of extensions and follow-up work [BR95, BM97, BPR00, CK01]. A protocol in the BR-model is formally just an algorithm. However, it is more useful to think of what this algorithm *represents*: a collection of principals interacting across an insecure network. Each principal sends and receives messages over the network by constructing and processing messages

according to some rule specific to the protocol being modeled. On the other hand, details of how the network routes and delivers these messages are abstracted away. Instead, the adversary is assumed to be in full control of the network, being able to decide exactly where and when messages are delivered to the principals. In particular, this means that the adversary can also choose to alter the messages, reorder them, drop them, or even inject messages of its own. Generally, whatever gets delivered to the principals happens at the behest of the adversary.

Principals hold both *long-term* and *short-term* keys (the latter usually called *session keys*), and the adversary will also be given the ability to obtain both of these types of keys at will. This models the fact that, in the real-world, keys which are supposed to be kept secret can nevertheless get lost for a large number of reasons. We return to this in Section 3.2.3.

The following subsections describe our variant of the BR-model in detail. Since AKE and ACCE protocols are mostly the same in terms of modeling, we use a unified protocol model that covers everything that is common to both. Material specific to AKE and ACCE protocols is covered in Section 3.3 and Section 3.4, respectively. Essentially, their main difference lies in the winning conditions of their respective security games.

3.2.1 Protocol participants

A protocol is carried out by a set of principals or *parties* $U \in \mathcal{P}$, each taking on a distinct *role* within the protocol run. In two-party protocols there is an *initiator* role and a *responder* role, and in three-party protocols there is also an additional role called the *server*. We consider only protocols where each party implements only one of the predefined roles in the protocol. That is, the set of party identities $\mathcal{P}$ is partitioned into three disjoint sets $\mathcal{I}$, $\mathcal{R}$, and $\mathcal{S}$ consisting of the initiators, responders, and servers, respectively. Of course, in two-party protocols there are no servers, so we have $\mathcal{S} = \emptyset$. As a convention, we will use A to denote a party having the initiator role, B to denote a party having the responder role, S to denote a party having the server role, and U, V, W to denote parties that could have any role.

Protocol roles serve as symmetry-breaking devices, requiring that each participant in the protocol be discernible from the others. Of course, in real-world protocols there might be no explicit variable that records a user's role. Instead, a participant's role may be implicitly present in the structure and message flow of the protocol itself, such as the naming of the protocol messages or the order in which different messages are delivered and processed. Indeed, the names "initiator" and "responder" reflect the intuitive idea that one party is expected to initiate the protocol, while the other is supposed to wait for some initial

incoming message before responding.

Conversely, there are protocols in which there are no fundamental differences in the actions being performed by the different protocol participants. For instance, in *role-symmetric* protocols (see [Cre09, Cre11a]) the messages are identically distributed, so up to their order, there is no discernible difference between the messages in the protocol. Examples of such protocols are MQV [Law+03] and HMQV [Kra05b]. Here, two initiators can establish a common key with each other. In fact, the two initiators might even belong to the same party. For role-symmetric protocols, care needs to be taken so that they are not vulnerable to so-called *reflection attacks* [Cre11b] where an attacker replays a sender's messages back to itself. On the other hand, we point out that whether a reflection attack should actually be considered problematic or not, might depend on the protocol's authentication goals; see Section 3.5 for further discussion on authentication

Finally, we remark that our model could easily be generalized to protocols that consist of N distinct roles instead of just two or three. However, for the purposes of this thesis, three distinct roles are enough. Similarly, many formal models also allow parties to take on different roles in different runs of the protocol (see e.g., [Jag+12]). That is, party U could in one run of the protocol take the role of an initiator, while in another take on the role of a responder (or a server). For simplicity and clarity of presentation we assume that a party only implements one role.

3.2.2 Long-term keys

Every party holds at least one long-term key. Our model includes both asymmetric private/public key-pairs as well as a symmetric pre-shared keys (PSKs). In principle, we could allow arbitrary configurations of long-term keys, where each party could hold multiple asymmetric keys and share multiple PSKs with arbitrary subsets of $\mathcal{P}$. However, we are going to restrict our attention to the following three specific classes of protocols in terms of their configurations of long-term keys.

1. Two-party protocols exclusively based on public-keys. In this case, every party U gets a single asymmetric private/public key-pair (sk_U, pk_U).

2. Two-party protocols exclusively based on PSKs. In this case, every pair of initiator/responder (A, B) shares a single symmetric long-term key K_{AB}.

3. Three-party protocols, where the long-term keys are configured as follows:

   - each initiator $A \in \mathcal{I}$ has one private/public key-pair (sk_A, pk_A);

- each responder $B \in \mathcal{R}$ has one PSK for every server $S \in \mathcal{S}$, denoted K_{BS};
- each server $S \in \mathcal{S}$ has one private/public key-pair (sk_S, pk_S) and one PSK K_{BS} for every responder $B \in \mathcal{R}$.

The choice of focusing only on the above three classes of protocols is not arbitrary. Rather, they model in a somewhat simplified manner the way long-term keys are used in, respectively, EAP-TLS, IEEE 802.11, and EAP. Specifically, Item 3 captures the setting of EAP where the EAP method that is run between the client and the server is based on public-keys, and the key-transport protocol between the server and the authenticator (normally RADIUS) is based on PSKs. It would be possible to also handle EAP methods that use PSKs or even a mix of both, but for ease of presentation we limit ourselves to the asymmetric case only.

Finally, when asymmetric long-term keys are used, then all users are given an authentic copy of every public key in the system. This is a standard assumption used in most key exchange models, but it is nevertheless a big assumption. It glosses over the big challenges faced with constructing and maintaining a public key infrastructure (PKI) needed for users to obtain authentic public keys. The alternative is to explicitly include PKIs into the formal models, which has been done in a handful of related papers [Bol+07, FW09, Boy+13]. This generally leads to more realistic modeling at the cost of making an already complex model even more complex. For the sake of keeping our model manageable we have chosen to omit any considerations of PKIs in this thesis and do not model any aspects relating to the actions and functioning of certificate authorities (CAs). In particular, this means that we do not consider attacks where the adversary can register its own public key(s) as authentic, or pass off somebody else's public key as its own; nor do we model attacks where the adversary registers invalid keys, which can have devastating effects on some protocols (see [MU06]). In short, in our model all long-term keys are honestly generated and authentically distributed at the beginning of the security game.

3.2.3 Protocol syntax

A *protocol* is a tuple $\Pi = (\mathsf{KG}, \mathsf{NextMsg}, \kappa)$, where KG is a key generation algorithm, $\mathsf{NextMsg}$ is an algorithm that specifies how honest parties process and construct protocol messages, and $\kappa \in \mathbf{N}$ is a session key length. Algorithm KG either takes no input, in which case it produces a long-term asymmetric key-pair (sk, pk) consisting of a private key sk and a public key pk; or it takes as input the string "$\mathtt{PSK}$", in which case it produces a long-term symmetric key K. Each party $U \in \mathcal{P}$ can take part in multiple executions of the protocol—

called *sessions*[1]—both concurrently and sequentially. We use an administrative label π_U^i to refer to the ith session at user U. Sometimes we simplify π_U^i 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. The type of variables that make up a session's state are in general highly protocol dependent, but in our model the following variables are always assumed to be present.

- peers – an unordered list of party identities $V \in \mathcal{P}$ representing the principals that π_U^i believes take part in this protocol run (including U itself),
- $\vec{\alpha} = (\alpha_1, \ldots, \alpha_n)$ – a vector of *acceptance states* $\alpha_i \in \{\mathsf{running}, \mathsf{accepted}, \mathsf{rejected}\}$,
- $k \in \{0,1\}^\kappa \cup \{\perp\}$ – the symmetric *session key* derived by π_U^i,
- $\tau \in \{0,1\}^*$ – the *local transcript* of π_U^i, consisting of all the messages it has sent and received,
- $st \in \{0,1\}^*$ – any additional auxiliary state that might be needed by the protocol.

We use the notation "$\pi_U^i.x$" to refer to some variable x of session π_U^i. For instance, $\pi_U^i.k$ denotes the session key of π_U^i, while $\pi_U^i.\mathsf{peers}$ denotes its list of peers.

In addition to the variables above, a session also has access to the long-term keys of the party to which it belongs, as well as the public keys of the other parties in the system. Specifically, to π_U^i we also associate the following variables:

- sk_U, pk_U – the long-term private/public key of party U,
- $\mathsf{PubKey}[\cdot]$ – a list of public keys indexed by their owner's identity $V \in \mathcal{P}$,
- $\mathsf{PSK}[\cdot]$ – a list of the PSKs that U shares with other parties, indexed by their identities $V \in \mathcal{P}$.

Of course, if we consider two-party protocols based on PSKs, then the variables sk_U, pk_U and PubKey are not needed. Likewise, for two-party protocols based on public keys, or for initiators in three-party protocols, then the variable PSK will be superfluous. In general, depending on the type of protocol under consideration, some of the long-term key variables may not be needed. By convention, unnecessary variables are set to $\perp$.

[1] What we call a *session* is also often called an *instance* in the literature.

Remark 3.1. Bellare, Pointcheval, and Rogaway [BPR00, Remark 2] points out the difference between accepting and terminating. When a session terminates, then it does not send any further messages in the protocol. On the other hand, a session might be able to accept—meaning that from this point on we expect some security property to hold—even if more messages will be exchanged subsequently. ▲

We use a *vector* $\vec{\alpha}$ of acceptance states to model protocols Π that are built out of sub-protocols Π_i run sequentially after each other. This is somewhat similar to the multi-stage model of Fischlin and Günther [FG14] where there is a separate accept state for each individual stage. However, we use sub-protocols purely to make our expositions in constructions and proofs easier. We do not define any security goals in terms of a protocol's sub-protocols, and a protocol is not required to have a logical sub-protocol structure.

Each entry of $\vec{\alpha}$ signifies whether the session believes that sub-protocol Π_i has operated correctly (accepted), something has gone wrong (rejected), or that the session hasn't come to a decision yet (running). Since we only consider sub-protocols run sequentially, we demand the following semantics from the variables $\vec{\alpha} = (\alpha_1, \ldots, \alpha_n)$ and k:

$$\alpha_i = \mathsf{accepted} \implies \alpha_{i-1} = \mathsf{accepted}, \tag{3.1}$$

$$\alpha_i = \mathsf{rejected} \implies \alpha_{i+1} = \mathsf{rejected}, \tag{3.2}$$

$$\alpha_n = \mathsf{accepted} \implies k \neq \bot. \tag{3.3}$$

In other words: a session can only accept in sub-protocol Π_i if it has already accepted in all prior sub-protocols; if it rejects in a sub-protocol Π_i, then this cascades to all subsequent sub-protocols; and finally, if a session accepts in the final sub-protocol Π_n, then it must have set its session key k. Moreover, we demand that the session key only be set once.

In our formal security experiments, the accepted state will be used as a reference point from which security properties are expected to hold for a session. Let $\alpha_F \stackrel{\text{def}}{=} \alpha_n$ denote the final acceptance state of $\vec{\alpha}$. As a convention we always use α_F to refer to the acceptance state of the full protocol Π (considered as a composition of $n-1$ sub-protocols). A session is said to have *accepted*, *rejected* or still be *running* in the full protocol based on the value of α_F. Thus, ignoring all the other states in $\vec{\alpha}$, the *single* acceptance state used in most other security models corresponds to α_F in ours.

Finally, note that the acceptance states are not intended to be secret, but will be explicitly given to the adversary.

3.2.4 Protocol correctness

Protocols are required to satisfy the following correctness requirement in the absence of any adversary. If an initiator session π_A^i, a responder session π_B^j—and possibly a server session π_S^k (if in the three-party setting)—run the protocol between them according to its specification, then we require that: (1) all sessions end up accepting, (2) all sessions have their peers variable set to the unordered list $\{A, B, [S]\}$, and (3) π_A^i and π_B^j derive the same session key $\pi_A^i.k = \pi_B^j.k$. Although correctness can be further formalized in various ways,[2] we hope that its intuitive meaning should be sufficiently clear as to obviate this need.

Note that we have only demanded that the initiator and the responder derive the same key. What about the server's key in the case of three-party protocols? The purpose of the server is to help the initiator and the responder establish a common key, but this does not imply that the server will necessarily derive a session key itself. Of course, there are protocols in which the server will be in possession of the session key—in fact, the server might be the one that chooses and distributes it!—but this is not always the case. Thus, in general it does not make sense to ask for correctness with respect to the server's key.

In order to maintain consistency with the requirement of Equation (3.3), we establish by convention that the session key variable k of all server sessions π_S^i is always set to the all-zero string 0^κ. Note that this is pure formalism: for protocols where the server does, in fact, derive the same session key as the initiator and responder, this value will simply be stored in the auxiliary state variable $\pi_S^i.st$ rather than the variable $\pi_S^i.k$.

3.2.5 Security experiment

As mentioned at the beginning of this section, security will be defined in terms of a formal experiment run between an adversary and a challenger. Technically, for each of the security properties we want to define—2P-AKE, 3P-AKE, and (2P-)ACCE—there will be a corresponding security experiment. However, since all these experiments are very similar in nature, we use a common experiment template, shown in Figure 3.1, that captures all of them. Experiment $\mathbf{Exp}_{\Pi,\mathcal{Q}}(\mathcal{A})$ is parameterized by a protocol Π, a *query set* $\mathcal{Q}$, and an adversary $\mathcal{A}$. The query set $\mathcal{Q}$ is a collection of the permissible queries that $\mathcal{A}$ can make during the experiment. Each query represents a function or algorithm implemented by the experiment.

[2] For example, with an adequate definition of a *benign* adversary [BR95], one can easily formalize correctness using the game framework used in this thesis. Alternatively, correctness could be defined directly in terms of a protocol's message sequence diagram (see also the discussion on matching conversations in Section 3.2.6).

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

Long-term key set-up:
  3P: For every $U \in \mathcal{I} \cup \mathcal{S}$ create $(sk_U, pk_U) \twoheadleftarrow \Pi.\mathsf{KG}$
    For every $(U, V) \in \mathcal{R} \times \mathcal{S}$ define $K_{UV} = K_{VU} \twoheadleftarrow \Pi.\mathsf{KG}(\mathtt{PSK})$
    Define $\mathsf{pubkeys} \leftarrow \{(U, pk_U) \mid U \in \mathcal{I} \cup \mathcal{S}\}$

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

  2P-PSK: For every $(U, V) \in \mathcal{I} \times \mathcal{R}$ define $K_{UV} = K_{VU} \twoheadleftarrow \Pi.\mathsf{KG}(\mathtt{PSK})$
    Define $\mathsf{pubkeys} \leftarrow \emptyset$

$out \twoheadleftarrow \mathcal{A}^{\mathcal{Q}}(\mathsf{pubkeys})$

Figure 3.1: 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.

Experiment $\mathbf{Exp}_{\Pi,\mathcal{Q}}(\mathcal{A})$ begins with a set-up phase, where all the long-term keys in the system are being generated. Recall from Section 3.2.2 that we are considering three types of protocols in this thesis: two-party protocols based on public keys, two-party protocols based on PSKs, and three-party protocols that combine both public keys and PSKs. The set-up phase in Figure 3.1 reflects these three scenarios.

Following the creation of the long-term keys the adversary $\mathcal{A}$ is run, being given as input a list pubkeys containing all the public keys in the system (if any). In this phase the adversary gets to interact with the experiment using the queries contained in the query set $\mathcal{Q}$. The query sets used to define AKE and ACCE security will be different, but they will contain a common *base* query set $\mathcal{Q}_{base}$ consisting of the queries NewSession, Send, Reveal and Corrupt.

A NewSession query allows the adversary to create a new session at a given party. A Send query allows the adversary to interact with the sessions by sending (arbitrary) messages to them. This will yield a response based on the NextMsg algorithm of protocol Π. A Reveal query allows the adversary to learn the session key of a given session. This models the fact that in the real world, the adversary might know some of the session keys in the system for a number of reasons, e.g. because of break-ins, cryptanalysis, leakage due to application usage, or misconfigurations. Although the loss of a session key will certainly compromise the security for that particular session, one hopes that it will not impact the security of other sessions, using different session keys. Finally, a

Corrupt query allows the adversary to obtain a long-term secret key of a given party. This query models the fact that in the real world some of the secret long-term keys might become known to the adversary, for example by break-ins, subversions, or mishandling of credentials. The loss of a long-term secret key can potentially be even more devastating than losing a single session key, since now it might have ramifications for *all* the sessions of a given party, as well as all the sessions wanting to communicate with that party. Nevertheless, many protocols can mitigate the damage caused by leaking long-term keys. For example if a protocol has *forward secrecy* [MvV96, p. 496] then the loss of a long-term key should not affect the security of already established session keys. Similarly, if a protocol has resistance to *key compromise impersonation (KCI)* attacks [JV96, BM97], then the loss of an asymmetric long-term private key sk_U will *not* enable an attacker to impersonate *other* parties towards the holder of sk_U. We now give precise definitions of the queries contained in $\mathcal{Q}_{base}$.

- NewSession$(U, [V, W])$: This query creates a new session π_U^i at party U. It takes one mandatory input U, namely the party where the session is created, and two optional inputs V and W, representing the intended peers of U. It is required that U, V and W all have different roles.

  The variables associated to π_U^i are initialized as follows: $\pi_U^i.\mathsf{peers} = \{U, [V, W]\}$, $\pi_U^i.\vec{\alpha} = (\mathsf{running}, \ldots, \mathsf{running})$, $\pi_U^i.k = \perp$, $\pi_U^i.\tau = \perp$, and $\pi_U^i.st$ is set to whatever is needed by protocol Π.

  Additionally, depending on the type of protocol, the long-term key variables sk, pk, peers, PubKey and PSK are initialized accordingly, based on the long-term keys in the system.

  Finally, if $U \in \mathcal{I}$, then π_U^i also produces its first message m^* according to the message creation algorithm NextMsg of protocol Π. In this case m^* gets added to $\pi_U^i.\tau$. The administrative label π_U^i, the message m^*, and π_U^i's accept state $\pi_U^i.\vec{\alpha}$ are all returned to $\mathcal{A}$.

- Send(π_U^i, m): This query sends a message m to session π_U^i. The session computes a response message m^* according to the specification of protocol Π. This also updates π_U^i's current internal state. Both m^* and $\pi_U^i.\vec{\alpha}$ are returned to $\mathcal{A}$.

- Reveal(π_U^i): This query returns the session key $\pi_U^i.k$ of π_U^i. From this point on, π_U^i is said to be *revealed*.

- Corrupt$(U, [V])$: Depending on the second input parameter, this query returns a certain long-term key of party U.

  - Corrupt(U): If U has an associated private-public key-pair (sk_U, pk_U), return the private key sk_U.

– Corrupt(U, V): If U and V share a symmetric long-term key K_{UV}, return K_{UV}.

The long-term key returned from this query is said to be *exposed* and the owner(s) of the key, *corrupted*.

Since the inputs V, W to the NewSession query are optional, we are working in the *post-specified peer model* [CK02]. This means that a session might not know its peers at the beginning of the protocol, but will instead learn this as the protocol progresses.

Note that a Corrupt query returns a party's long-term key and nothing else. Particularly, the adversary does not take control of all the sessions at this party, nor learn their internal state (except for the leaked long-term key). This is in contrast to some protocol models [CK01, CK02], where corruption means to take full control of a party and all its sessions. We emphasize that in our model, sessions created by the NewSession query *always* behave honestly (i.e., according to the protocol specification), using whatever internal state they have. The adversary can learn some of this state using Reveal and Corrupt queries, but it never gets to directly control the sessions' actions. On the other hand, with the knowledge of a party's long-term key, the adversary can of course simulate a run of a session at this party. However, this "dishonest session" does not have a material representation in experiment $\mathbf{Exp}_{\Pi,\mathcal{Q}}(\mathcal{A})$ in the form of an administrative session label π.

Ultimately, the adversary will halt in experiment $\mathbf{Exp}_{\Pi,\mathcal{Q}}(\mathcal{A})$ with some (possibly empty) output *out*, which also ends the experiment. Note that experiment $\mathbf{Exp}_{\Pi,\mathcal{Q}}(\mathcal{A})$ does not in itself produce any output, nor define any winning condition for $\mathcal{A}$. Rather, it provides a single experiment on which we can define many different winning conditions. Sections 3.3 through 3.5 define the concrete winning conditions used to formalize the security goals of 2P/3P-AKE, ACCE, and explicit entity authentication, respectively. For some security properties, the output *out* produced by $\mathcal{A}$ will be used to define the winning condition of the security game.

3.2.6 Freshness predicates and partnering

Experiment $\mathbf{Exp}_{\Pi,\mathcal{Q}}(\mathcal{A})$ puts no restrictions on the adversary's usage of the queries in $\mathcal{Q}$. Specifically, the adversary can ask for any session key it wants using the Reveal query, and any long-term key using the Corrupt query. It follows that any meaningful winning condition needs to take into account the possibility of trivial attacks enabled by the adversary's unfettered access to

Reveal and Corrupt queries. An attacker should not be given credit for trivial attacks.

To precisely define what constitutes a trivial attack, we use the concept of a *freshness predicate*. A session will be considered a legitimate target in the security game if and only if it satisfies the prescribed freshness predicate. In fact, we will define several freshness predicates that encode different security properties. More specifically, the combination of a query set $\mathcal{Q}$, a freshness predicate F, and a winning condition W, is called a *security model* (following [CF12]). Although we don't do so here (see [CF12, FC14]), the different freshness predicates make it possible to formalize an ordering on the security models in terms of their "strength". Generally, a security model M is "stronger" than model M' if any protocol secure in model M is also secure in protocol M'. Provided their query sets and winning conditions are the same, the relative strength of two security models comes down to the permissiveness of their freshness predicates.

A key tool for defining freshness is the concept of *partnering* (also called *matching*). Suppose two sessions π and π' share the same session key k. If an adversary $\mathcal{A}$ reveals π, meaning that $\mathcal{A}$ obtains π's session key k, then $\mathcal{A}$ can also trivially attack π'. But if π and π' were *supposed* to obtain the same key k, then it doesn't seem fair that $\mathcal{A}$ should get any credit for this attack. Partnering aims to capture exactly this: two sessions that ought to have the same session key are called *partners*, and revealing one of them will automatically make its partner unfresh as well. We hasten to add that partnering can serve purposes other than this, something which will be discussed further in Section 3.3, but the main idea is to match sessions having the same key.

Although the concept of partnering is pervasive in cryptographic security models, nailing down exactly what partnering *is* can be surprisingly difficult. Below we discuss some of the common approaches that have been taken in the literature.

Matching conversations. In their original key exchange model, Bellare and Rogaway [BR94] defined partners using *matching conversations*. For two-party protocols (which was the topic of [BR94]), two sessions are said to have matching conversations if all the messages sent and received by one session match the messages received and sent by the other (save possibly for the last message, which might not have been delivered). For three-party protocols, or more generally, N-party protocols, defining matching conversations is less straightforward but can still be done (basically by appealing to the protocol's message sequence diagram). Notice that matching conversations are consistent with a protocol run in which all messages are being faithfully transmitted, so by protocol correctness, partners based on matching conversations do indeed have the same key.

Partner functions. Matching conversations do have a downside in that they focus on an inherently syntactical part of a protocol which ultimately may be irrelevant to its security. This can be illustrated by the following "folklore" example. Suppose a protocol Π has been proven secure using matching conversations as the mechanism for partnering. From Π create a new protocol Π^0 by adding a zero bit to the end of every message of Π. On receiving a message in Π^0, a session will ignore the last bit and otherwise proceed as in protocol Π. Intuitively, protocol Π^0 should be no less secure than Π. However, when matching conversations are used to define partnering, an adversary can simply flip the zero bit to cause two sessions to no longer be partners. Since protocol Π and Π^0 otherwise proceed identically, the unpartnered sessions will still end up with the same session key and can now be legitimately attacked by the adversary.

Partly due to this undesirable property of matching conversations, in their next key exchange model, Bellare and Rogaway [BR95] instead defined partnering using the notion of a *partner function*. The idea behind a partner function is to look at the global transcript of all the messages sent and received in the security experiment, and use this to determine a session's partner. This solves the problem of matching conversations since a partner function can ignore the parts of a protocol's transcript that are irrelevant for security. However, this begs the question of what parts actually *are* relevant for security. It is not immediately obvious how one should recognize this. Indeed, the partner function Bellare and Rogaway [BR95] themselves constructed in order to analyze their 3PKD protocol, turned out to be flawed for the purpose of proving security as shown by Choo et al. [Cho+05, CH05]. More generally, the connection between partner functions and our intuitive understanding of partnering seems less clear than for matching conversations. Similar remarks have also been made by Rogaway [Rog04, §6].

SIDs. Bellare, Pointcheval, and Rogaway (BPR) [BPR00] presented yet another way of doing partnering by introducing explicit *session identifiers (SIDs)*. Here, each session is equipped with an additional string called its SID, and for two sessions to be partners it is necessary that their SIDs are the same. Although simple at first sight, the exact usage and interpretation of SIDs as a partnering mechanism is not fully consistent in the literature. First there is the question of how the SID should be constructed. In BPR's original fomulation, the SID is constructed locally by the sessions themselves during the run of the protocol, whereas in [CK01, CK02] the SID is assumed to be handed to the sessions from some unspecified outside process (which could even be the adversary).

Second, what should the SID contain? At the definitional level this is usually

left unspecified, but when doing a concrete analysis of a protocol, the SID is often taken to be the concatenation of a session's sent and received messages. This was suggested by BPR [BPR00] and mirrors partnering based on matching conversations. However, the SID can also be computed as an arbitrary function of the sent and received messages [AFP05], thus more closely resembling partner functions.

Finally, the exact relationship between the SID and the session key is not always formulated identically in different models. For instance, in BPR's [BPR00] definition no explicit relationship between the SID and the session key is required apart from the fact that partners must have both the same SID *and* the same session key. This is in contrast to most of the models following it, where having equal keys is not taken as a requirement for two sessions to be partners. Instead, the implication "partners $\implies$ equal keys" is included as a security goal on its own (see, e.g., [CK01, CK02, LLM07, CF12]). This idea has been further distilled in the notion of "Match security" introduced by Brzuska et al. [Brz+11]. Here, several implications of the form "equal SID $\implies$ ..." are collected into a single Match predicate, and this predicate is then required to hold throughout the security experiment. Note that Match security mostly functions as a sanity check on the chosen SID, rather than being an interesting security goal on its own. When basing partnering on SIDs, it has now become common practice to split the security definition into two separate goals: one being Match security and another being the actual security property of interest; see e.g., [Brz+11, Brz+13a, FG14, Dow+15].

Key-partnering. Of the partnering mechanisms we have discussed so far it is matching conversations and SIDs which have seen the widest adoption in the literature; a small sample being [BR94, BM97, Kra05b, LM06, MU08, CF12, Jag+12, Ber+14, CCG16] (matching conversations) and [BPR00, CK01, CK02, JKL04, AFP05, RS09, Brz+11, Brz+13a, Brz+13b, KPW13b, FG14, Dow+15] (SIDs). Partner functions on the other hand, have to the best of our knowledge only been used in two independent analyses [BR95, SR96]. However, coming back to the central idea of partnering—two sessions holding the same key—why are these *mechanisms* even necessary? Stated differently: why not simply define partnering directly in terms of which sessions hold the same key? This approach, which we dub *key-partnering* here, has in fact been suggested by Kobara et al. [KSS09] and by George and Rackoff [GR13].

Despite this fact, partnering today is almost exclusively based on either matching conversations or SIDs. We suggest several possible reasons for this. One might be historical. When Bellare and Rogaway [BR94] presented their original model it was primarily in the context of entity authentication. Since matching conversations is a natural way of formulating the goal of entity au-

thentication (at least in hindsight!), and since the models for entity authentication and key exchange are almost the same, it might have made sense to re-use matching conversations as a mechanism for partnering. But as noted by Bellare and Rogaway [BR95], the goals of entity authentication and key distribution are very different and it is quite possible to consider one without the other. Hence, there is no reason *a priori* why a mechanism for defining entity authentication (matching conversations) needs to be tied up with a definition of partnering in key exchange. On the other hand, if both entity authentication and key exchange are wanted properties, then a single mechanism might be more convenient (see Section 3.5).

Public partnering. A more technical reason for the lack of key-partnering might be the issue of *public* partnering. Basically, a partnering mechanism is said to be public if the adversary can always tell, based on the messages exchanged in the protocol, what the partner of a session is. In other words, public partnering implies that the partnering mechanism must be some function of the public messages sent and received in the security experiment. For matching conversations and partner functions this is true by definition (a point emphasized by [BR95]), whereas for SID-based partnering this does not necessarily have to be the case. Specifically, in [BPR00] the definition of partnering depends both on the session SID *and* the keys. Although the SID is explicitly handed to the adversary and in that sense can be thought of as being public, as we remarked above, there was no implication that equal SIDs imply equal session keys. Thus, partnering in [BPR00] is not technically speaking public. However, as we also noted, most SID-based models following [BPR00] removed the requirement of equal session keys from the partnering definition itself, allowing the partnering decision to be based purely on public data.

So why is public partnering a desirable feature? The problem with partnering based on private data has to do with simulatability in security reductions. When proving the security of some protocol Π, one reduces the task of breaking Π to the problem of breaking one of its building blocks, or to solving some hard mathematical problem. Specifically, from some hypothetical adversary $\mathcal{A}$ that breaks protocol Π, one constructs an algorithm $\mathcal{B}$ that breaks one of the underlying building blocks or hardness assumptions. However, in order for $\mathcal{B}$ to be able to capitalize on $\mathcal{A}$'s ability to break protocol Π, it needs to properly simulate experiment $\mathbf{Exp}_{\Pi,\mathcal{Q}}(\mathcal{A})$. In particular, $\mathcal{B}$ needs to give consistent answers to $\mathcal{A}$'s Reveal queries. This might require that $\mathcal{B}$ is able to determine which sessions are partners. If protocol Π is only made up out of cryptographic primitives like encryption schemes and signature schemes, then this step is mostly straightforward. However, if one of the building blocks of Π is actually a protocol in itself, then this can become much more difficult. In fact, Brzuska et

al. [Brz+11] showed that a weak form of public partnering is actually necessary in order to establish a certain compositional result. Particularly, they proved the "folklore" result that a secure key exchange protocol can safely be composed with a protocol that uses the established session keys—assuming that the key exchange protocol provides public partnering. Conversely, they also showed that if two such protocols could be securely composed, then this must also imply a weak form of public partnering.

In contrast, key-partnering is inherently based on private data (the session keys!). While Kobara et al. [KSS09] make no mention of this point, George and Rackoff [GR13] include an oracle that allows the adversary to check whether two sessions have the same key. In this way they explicitly incorporate public partnering into their model.

Our choice of partner mechanism. Given all of the above, we have elected to use partner functions as the partner mechanism in this thesis. On the whole, we find partner functions to be the most conducive for the kind of modular security results we seek to establish. Also, partner functions seem an elegant way of doing partnering for three-party protocols. While matching conversations can be generalized beyond two-party protocols, the issue of making a secure protocol insecure by adding an independent bit to it remains. As for SIDs, when modeling EAP we are in the peculiar situation that the sessions that we want to partner, namely the client and authenticator sessions, don't actually have any messages in common! Since equal SIDs should imply equal session keys, we are essentially forced to pick the session keys as the SID—basically leaving us with key-partnering. However, as pointed out when discussing key-partnering, there is no guarantee that key-partnering necessarily provides public partnering. This is an important property to have, especially when analyzing the composition of several protocols as we do in this thesis. Of course, one could follow the approach of George and Rackoff [GR13] and assume that a partnering oracle is present. But since none of the protocols that we want to compose in thesis (TLS, IKEv2, SSH, etc.,) have been proven secure in this manner, this would essentially require us to redo the analysis of these protocols in this new setting. Since a major goal of this thesis is to be able to re-use existing analyses of these protocols in a modular way, we have chosen not to take this approach.

The remainder of this section is devoted to formally defining partner functions. However, before we can do so we need some language to talk about the protocol messages being exchanged in the security experiment.

Transcripts. Consider a run of experiment $\mathbf{Exp}_{\Pi,\mathcal{Q}}(\mathcal{A})$. The *transcript* of this execution is the ordered sequence T consisting of all the Send and NewSession queries made by the adversary $\mathcal{A}$, together with their corresponding responses.

We tacitly assume that $\mathcal{A}$ only makes Send queries to sessions that it previously created with a NewSession query, since sending messages to a non-existing session is meaningless. Thus, a transcript records all the public messages exchanged between the existing sessions in the experiment run.

Example 3.2. A typical transcript T might look something like the following. Below we have used different colors to indicate different messages, and we have simplified the sessions' acceptance state variable $\vec{\alpha}$ to only consist of a single value α.

$$\begin{gathered}
\langle \mathsf{NewSession}(A, B), \pi_A^1, \mathbf{m}, \mathsf{running} \rangle, \\
\langle \mathsf{NewSession}(B, A), \pi_B^1, \bot, \mathsf{running} \rangle, \\
\langle \mathsf{Send}(\pi_B^1, \mathbf{m}), \mathbf{m}, \mathsf{running} \rangle, \\
\langle \mathsf{Send}(\pi_A^1, \mathbf{m}), \mathbf{m}, \mathsf{accepted} \rangle, \\
\langle \mathsf{Send}(\pi_B^1, \mathbf{m}^*), \bot, \mathsf{rejected} \rangle, \\
\langle \mathsf{NewSession}(A, C), \pi_A^2, \mathbf{m}, \mathsf{running} \rangle, \\
\vdots \\
\langle \mathsf{Send}(\pi_D^{23}, \mathbf{m}), \bot, \mathsf{accepted} \rangle
\end{gathered}$$

In this example, $\mathcal{A}$ first creates an initiator session π_A^1 and a responder session π_B^1. It then forwards π_A^1's initial message $\mathbf{m}$ to π_B^1, which responds with its own message $\mathbf{m}$. This is shown in the first Send query. Next, $\mathcal{A}$ forwards π_B^1's message $\mathbf{m}$ to π_A^1 which responds with its own message $\mathbf{m}$ and accepts (second Send query). However, $\mathcal{A}$ now sends a forged message $\mathbf{m}^*$ to π_B^1 which leads it to reject (third Send query). The rest of the transcript can be explained in a similar manner. ▲

Note that a transcript does not include any of $\mathcal{A}$'s Test, Reveal, or Corrupt queries. So for the example given above, $\mathcal{A}$ could have made a number of Reveal and Corrupt queries (as well as a Test query) in between the NewSession and Send queries recorded on T.

We now define some useful notation for working with transcripts. 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. A transcript T is said to *contain* a session π if there is a NewSession query on T that created π. Let S be a set of sessions and let T be an arbitrary transcript. The *restriction of T to S*, written $T\big|_S$, is the transcript one gets from T by removing all queries that do not pertain to the sessions in S. That is, $T\big|_S$ consists only of the NewSession queries in T that created the sessions in S, as well as the Send queries directed to these sessions. Note that $T\big|_S$ is not necessarily a prefix of T because the Send and NewSession

queries of $T\big|_S$ could have been arbitrarily interspersed with all the other Send and NewSession queries of T.

Partner functions – formal definition. Given the language of transcripts we can now precisely define partner functions and partnering. We give the formal definitions first, then make several comments.

Definition 3.3 (Partner functions). A *partner function* is a function

$$f\colon (T,\pi) \mapsto \pi'/\perp \tag{3.4}$$

that takes as input a transcript T and a session π contained in T, and then outputs a session π' in T or $\perp$. A partner function is *symmetric* if $f(T,\pi) = \pi'$ implies that $f(T,\pi') = \pi$ for all transcripts T. A partner function is *monotone* if $f(T,\pi) = \pi'$ implies that $f(T',\pi) = \pi'$ for all $T \subseteq T'$. Instead of $f(T,\pi) = \pi'$ we will more commonly write $f_T(\pi) = \pi'$.

Definition 3.4 (Partnering). Let T be a transcript and f be a partner function. If $f_T(\pi) = \pi'$ then π' is said to be the *partner* of π. If $f_T(\pi) = \pi'$ and $f_T(\pi') = \pi$ then π and π' are *partners*.

In other words, a partner function assigns every session created in experiment $\mathbf{Exp}_{\Pi,\mathcal{Q}}(\mathcal{A})$ to its partner (if it has one) or to $\perp$ (if it doesn't). Technically speaking, a partner function also depends on the parties in the system and the roles they have, so a partner function should additionally have taken the sets $\mathcal{I}$, $\mathcal{R}$, and $\mathcal{S}$ as input. However, since these sets could easily have been provided to the partner function in some other way, say by writing them as the first entries of the transcript T, we assume that the configuration of $\mathcal{I}$, $\mathcal{R}$, and $\mathcal{S}$ is encoded into the partner function itself.

Except for notational differences, our definition of partner functions mostly mirrors that of Bellare and Rogaway [BR95]. However, unlike Bellare and Rogaway, we will always demand that our partner functions are symmetric and monotone. Both properties are fairly natural to expect from a partnering mechanism. Symmetry says that if π' is a partner of π, then π is also the partner of π'; while a partner function is monotone if once π and π' becomes partners, then they remain so forever. Partnering based on matching conversations, SIDs, or key-partnering are usually both symmetric and monotone[3]. Moreover, Bellare

[3]Partnering can fail to be monotone if a requirement of uniqueness is baked into the definition. For instance, if partnering was defined as "π and π' are partners if and only if they are the *only* two sessions having the same SID", then they would cease to be partners if a third session were to compute the same SID. The original SID-based partner definition of Bellare, Pointcheval, and Rogaway [BPR00], as well as the key-partnering definition of Kobara, Shin, and Strefler [KSS09], were of this form. However, most other models today are monotone. The issue of three sessions computing the same SID is instead formulated as a security goal of its own.

and Rogaway [BR95, Thm 5] even state (although without proof) that a protocol proven secure with a general partner function can also be proven secure with a symmetric and monotone partner function. At any rate, we find it easier to simply demand these properties at the definitional level. Since we are always going to demand that our partner functions are symmetric and monotone, we drop these adjectives from now on and talk only about "partner functions".

Because of its generality, the partner function definition technically admits some rather non-intuitive functions. For instance, the trivial partner function which partners no sessions at all is a valid partner function. Clearly, no protocol can be secure with this partner function. So what does security based on partner functions actually mean? Essentially, security is a statement about the *existence* of some partner function for which no adversary can have a good advantage in breaking the protocol. Particularly, security means that there exists a partner function so that any attack on the protocol can be translated into an attack on its building blocks. However, when protocols are built out of sub-protocols, the meaning of security is more subtle since the security of the sub-protocols is itself expressed in terms of partner functions. The problem is that for any protocol there *exists* a partner function for which the protocol can trivially be broken (like the trivial partner function mentioned above). Thus, a statement of the form "an attack on protocol Π under partner function f implies an attack on its sub-protocol Π_1 for *some* partner function g" is meaningless. Instead, a meaningful reduction from a protocol to its sub-protocol needs to hold for *every* choice of g. Alternatively, it should hold for a *particular* choice of g, and then one shows that an attack on the sub-protocol under *this* g implies an attack on its building blocks. Note that security based on SIDs is also fundamentally a statement about the existence of some SID, although this point is seldom emphasized in papers that use SIDs for partnering.

Finally, we define a special class of partner functions, called *local* partner functions, which will be useful in one of our later analyses. Local partner functions capture the idea that deciding whether two sessions π and π' are partners or not should only depend on π's and π''s local transcripts, i.e., the messages they sent and received. However, there is one issue with this approach: since partner functions are indeed *functions*, this notion could be ambiguous if two sessions at the same party have exactly the same local transcript τ. Thus, we only define local partnering for *unique* transcripts, where a transcript is said to be unique if no two sessions at the same party have the same local transcripts.

Definition 3.5 (Local partnering). A partner function is *local* if for all unique transcripts T, and for all sets S of sessions contained in T, we have

$$f(T, \pi) = \pi' \iff f(T\big|_S, \pi) = \pi' \tag{3.5}$$

for all $\pi, \pi' \in S$.

Although we have presented local partner functions as being a special class of partner functions, they are in fact the norm. Both matching conversations and SIDs are local as partner mechanisms.

Soundness of partner functions. As already noted, the generality of the partner function definition allows for some nonsensical constructions. In fact, the definition does not even mandate that partners end up with the same key. Thus, following the Match security approach of Brzuska et al. [Brz+11], we define a *soundness* predicate Sound which aims to capture those properties that we intuitively expect to hold for two partnered sessions.

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. Note that beyond having the same key, these requirements also express authentication goals in terms of the partner function. Since partner functions are indeed *functions*, the uniqueness requirement of (4) follows automatically. Hence, we can skip it in our formal definition.

Definition 3.6 (Soundness security). Let f be a partner function. Consider a run of experiment $\mathbf{Exp}_{\Pi,\mathcal{Q}}(\mathcal{A})$. Predicate Sound_f is true if and only if, for any two partnered sessions π_U^i and π_V^j, all of the following requirements hold:

1. $\pi_U^i.\alpha_F = \pi_V^j.\alpha_F = \mathsf{accepted} \implies \pi_U^i.k = \pi_V^j.k$,
2. $\pi_U^i.\alpha_F = \pi_V^j.\alpha_F = \mathsf{accepted} \implies \pi_U^i.\mathsf{peers} = \pi_V^j.\mathsf{peers} = \{U, V, [W]\}$,
3. $(U \in \mathcal{I} \wedge V \in \mathcal{R} \wedge W \in \mathcal{S})$ or $(U \in \mathcal{R} \wedge V \in \mathcal{I} \wedge W \in \mathcal{S})$.

We let $\mathbf{Exp}_{\Pi,\mathcal{Q}}^{\mathsf{Sound}_f}(\mathcal{A}) \Rightarrow 1$ denote the event that $\mathsf{Sound}_f = \texttt{false}$. The *soundness* advantage of an adversary $\mathcal{A}$ against a protocol Π under partner function f is

$$\mathbf{Adv}_{\Pi,f}^{\mathsf{Sound}}(\mathcal{A}) \stackrel{\text{def}}{=} \Pr[\mathbf{Exp}_{\Pi,\mathcal{Q}}^{\mathsf{Sound}_f}(\mathcal{A}) \Rightarrow 1]. \tag{3.6}$$

If clear from context, we write Sound instead of Sound_f. Note that Item 3 excludes role-symmetric protocols (recall Section 3.2.1), but it also encodes the fact that server sessions will not be considered partners to anyone.

Formulated as a security game, soundness says that if two sessions become partners, then they will agree upon the same session key (say) except with some "small" error probability $\mathbf{Adv}_{\Pi,f}^{\mathsf{Sound}}(\mathcal{A})$. However, in order not to always having to condition on two partners having the same session key (or any of the other properties), we will in most of our proofs make the simplifying assumption that soundness is perfect, i.e., $\mathbf{Adv}_{\Pi,f}^{\mathsf{Sound}}(\mathcal{A}) = 0$. This is a mild assumption;

partnering mechanisms such as matching conversations and SIDs usually always have this property provided the protocol employs a deterministic key derivation function.

3.3 2P-AKE protocols and 3P-AKE protocols

In this section we define our security model for AKE. In fact, we define *three* AKE models. One provides what we call full forward secrecy, one provides weak forward secrecy, and one provides no forward secrecy. Since most of the groundwork has already been done in the previous section, this section merely defines the AKE winning condition as well as providing a detailed discussion of the freshness predicates that make up the different AKE security models.

AKE syntax and security. The syntax of an AKE protocol is exactly the same as presented in Section 3.2.3. For AKE security, we want that an adversary should learn nothing about the distributed session keys except for those keys it can obtain by trivial means using Reveal and $\mathsf{Corrupt}$ queries. The standard way of expressing this is by saying that the session keys held by fresh sessions should be indistinguishable from random strings. Formally, this is captured by adding to the base query set $\mathcal{Q}_{base}$ an additional query Test, defined as follows.

- $\mathsf{Test}(\pi_U^i)$: 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 $\widetilde{k} \twoheadleftarrow \{0,1\}^\kappa$ 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.

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. Whether a session is fresh or not depends on the security model M. Specifically, it is decided by a *freshness predicate* Fresh. In this thesis we consider three AKE security models:

- AKE^{fs} which captures full forward secrecy (fs);
- $\text{AKE}^{\mathsf{wfs}}$ which captures weak forward secrecy (wfs); and
- $\text{AKE}^{\mathsf{nfs}}$ which has no forward secrecy (nfs).

Each model is determined by its corresponding freshness predicate $\mathsf{Fresh}_{\text{AKE}^{\mathsf{fs}}}$, $\mathsf{Fresh}_{\text{AKE}^{\mathsf{wfs}}}$, or $\mathsf{Fresh}_{\text{AKE}^{\mathsf{nfs}}}$. The definitions of these freshness predicates (together with the freshness predicate of the ACCE model) are presented jointly in Figure 3.2 using the parametrized predicate Fresh_M. Before we describe these predicates in greater detail, we first give the formal definition of AKE security.

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

1: *// Record the long-term keys of π_U^i's peers in the list* LTKeys.
2: *// LTKeys depends on the type of protocol under consideration*
3: *// (2P-PSK vs. 2P-Public Keys vs. 3P).*
4: $\{U, V, [S]\} \leftarrow \pi_U^i.\mathsf{peers}$
5: 2P-Public-Key:
6: LTKeys $\leftarrow \{sk_V\}$
7: 2P-PSK:
8: LTKeys $\leftarrow \{K_{UV}\}$
9: 3P:
10: **if** U is an initiator**:**
11: LTKeys $\leftarrow \{K_{VS}, sk_S\}$
12: **if** U is a responder**:**
13: LTKeys $\leftarrow \{K_{VS}, sk_S, sk_V\}$
14:
15: *// π_U^i is fresh...*
16: fresh $\leftarrow$ `true`
17: *// ... if it has accepted*
18: fresh $\leftarrow$ fresh $\wedge$ ($\pi_U^i.\alpha_F$ = accepted)
19: *// ... and it has not been revealed*
20: fresh $\leftarrow$ fresh $\wedge$ (Reveal(π_U^i) has not been called)
21: *// ... and its partner has not been revealed*
22: fresh $\leftarrow$ fresh $\wedge$ (Reveal($f_T(\pi_U^i)$) has not been called)
23: *// ... and no keys in* LTKeys *have been exposed in violation of security model M*
24: fresh $\leftarrow$ fresh $\wedge$ ($\mathsf{Corrupt}_M$ = `false`)
25:
26: **return** fresh

$\mathsf{Corrupt}_M$:

27: corrupt $\leftarrow$ `false`
28: **if** $M \in \{\mathsf{AKE}^{\mathsf{fs}}, \mathsf{ACCE}\}$**:**
29: corrupt $\leftarrow$ ($f_T(\pi_U^i) = \perp$) $\wedge$ (a key in LTKeys was exposed *before* π_U^i accepted)
30: **if** $M = \mathsf{AKE}^{\mathsf{wfs}}$**:**
31: corrupt $\leftarrow \big(($$f_T(\pi_U^i) = \perp$) $\wedge$ (a key in LTKeys is exposed)$\big)$
32: **if** $M = \mathsf{AKE}^{\mathsf{nfs}}$**:**
33: corrupt $\leftarrow$ (a key in LTKeys is exposed)
34: **return** corrupt

Figure 3.2: Freshness predicate Fresh_M parameterized on security model $M \in \{\mathsf{AKE}^{\mathsf{fs}}, \mathsf{AKE}^{\mathsf{wfs}}, \mathsf{AKE}^{\mathsf{nfs}}, \mathsf{ACCE}\}$.

Definition 3.7 (AKE security). Consider a run of experiment $\mathbf{Exp}_{\Pi,\mathcal{Q}_{\mathsf{AKE}}}(\mathcal{A})$. Suppose π was the test-session chosen by $\mathcal{A}$, b was the random bit used in answering the Test query, and b' was the final output of $\mathcal{A}$. Fix a partner function f and define $\mathsf{AKE}^* \in \{\mathsf{AKE}^{\mathsf{fs}}, \mathsf{AKE}^{\mathsf{wfs}}, \mathsf{AKE}^{\mathsf{nfs}}\}$ to be the following random variable on experiment $\mathbf{Exp}_{\Pi,\mathcal{Q}_{\mathsf{AKE}}}(\mathcal{A})$:

$$\mathsf{AKE}^* \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } (b' = b) \wedge \mathsf{Fresh}_{\mathsf{AKE}^*}(\pi) = \mathtt{true} \\ 0 & \text{if } (b' \neq b) \wedge \mathsf{Fresh}_{\mathsf{AKE}^*}(\pi) = \mathtt{true} \\ \widetilde{b} \twoheadleftarrow \{0,1\} & \text{if } \mathsf{Fresh}_{\mathsf{AKE}^*}(\pi) = \mathtt{false} \end{cases} \tag{3.7}$$

Let $\mathbf{Exp}^{\mathsf{AKE}^*}_{\Pi,\mathcal{Q}_{\mathsf{AKE}}}(\mathcal{A}) \Rightarrow d$ denote the event that $\mathsf{AKE}^* = d$. The *AKE** *advantage* of an adversary $\mathcal{A}$ is

$$\mathbf{Adv}^{\mathsf{AKE}^*}_{\Pi,f}(\mathcal{A}) \stackrel{\text{def}}{=} 2 \cdot \Pr[\mathbf{Exp}^{\mathsf{AKE}^*}_{\Pi,\mathcal{Q}_{\mathsf{AKE}}}(\mathcal{A}) \Rightarrow 1] - 1. \tag{3.8}$$

Note that in Definition 3.7 we are quantifying over *all* adversaries, not only those that satisfy the freshness predicate. Instead, if the adversary violates the freshness predicate then it gets penalized in the winning condition by having the challenger output a random bit on its behalf. This *penalty-style* formulation of security has previously been used in other works like [BHK15] and [GR13].

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

3.3.1 Comparing the three AKE security models

We now explain our AKE security models in detail, beginning with the $\mathsf{AKE}^{\mathsf{fs}}$ model, which is the strongest of the three.

$\mathsf{AKE}^{\mathsf{fs}}$. Given that the adversary has the ability to obtain any keys it wants using the Reveal and $\mathsf{Corrupt}$ queries, the purpose of the $\mathsf{Fresh}_{\mathsf{AKE}^{\mathsf{fs}}}$ predicate is to limit the scope of what is considered a valid attack within the model. The goal of the $\mathsf{AKE}^{\mathsf{fs}}$ model is to restrain the adversary as little as possible (hence leading to a strongest possible model), while at the same time being satisfiable. Although the freshness predicate can be applied to any session π^i_U, we ultimately only care about the freshness of the test-session, so in the following we assume that the session π^i_U in Figure 3.2 is the test-session.

First, the adversary should not be allowed to reveal the test-session, since otherwise it could trivially tell whether the test-key is random or not by checking whether the test-key is equal to the revealed key. This restriction is shown at Line 20 in Figure 3.2. Second, as we elaborated in Section 3.2.6, the adversary should also not be allowed to reveal the session key of the test-session's partner.

Table 3.1: Summary of the long-term key corruption models for the three AKE security models considered in this thesis. The table assumes that π_A^i is the test-session having B and S (in the three-party case) as its peers. The adversary is allowed to corrupt a party $U \notin \{A, B, S\}$ in all three models (not shown).

		Corrupt B or S	
Model	Corrupt A	if π_A^i has a partner	if π_A^i has no partner
AKE^{fs}	allowed[1]	allowed	allowed[2]
$\text{AKE}^{\mathsf{wfs}}$	allowed[1]	allowed	$\times$
$\text{AKE}^{\mathsf{nfs}}$	allowed[1]	$\times$	$\times$

[1] Only when using asymmetric long-term keys.
[2] But only after π_A^i accepted.

This is shown at Line 22 in Figure 3.2. Note that this makes the freshness predicate dependent on the partner function f, although this is not visible from the notation $\mathsf{Fresh}_{\mathsf{AKE}^{\mathsf{fs}}}$. Finally, we come to the issue of leakage of long-term keys. We refer to the extent to which an adversary can obtain long-term keys as the *(long-term key) corruption model* of the security model. In fact, the only difference between our three AKE models lies in their respective corruption models as shown at Line 24 of Figure 3.2. In Table 3.1 we summarize the corruption models of our three AKE models.

The corruption model of AKE^{fs} is quite liberal. First of all, any long-term key not contained in the variable LTKeys (Lines 4 through 13 in Figure 3.2) can be obtained at any point without affecting the freshness of the test-session. Note in particular that this includes the test-session's own private key sk_U (if it has one). Thus, the AKE^{fs} model captures KCI attacks (refer back to Section 3.2.5 or [JV96, BM97]).

On the other hand, for the long-term keys contained in LTKeys some restrictions apply. The keys in LTKeys are the long-term keys of the parties that the test-session believes it is talking to (as recorded in its peers variable at Line 4 in Figure 3.2). If these keys could be arbitrarily corrupted the adversary could trivially impersonate the corresponding parties towards the test-session. There are two cases to consider: (1) either the test-session has a partner ($f_T(\pi_U^i) \neq \perp$), or (2) it does not ($f_T(\pi_U^i) = \perp$).

In the first case the AKE^{fs} model is maximally lenient: *any* long-term key can be corrupted—even before the test-session has accepted! This is an example of where the partnering concept is used to model something beyond the notion of two sessions having the same key. In this setting, partnering is instead used to model *passiveness* by the adversary. Basically, the presence of

a partner is used as a sign that the adversary did not actively interfere with the communication of the test-session. Of course, when partnering is based on matching conversations, this connection is explicit. However, by letting the existence of a partner represent passiveness, we have lifted this intuition from matching conversations to partner functions.

Remark 3.8. Note that there are other, more fine grained, mechanisms for capturing passiveness besides partnering. For instance, the notions of *origin-sessions* [CF12] and *contributive session identifiers* [Dow+15] are two ways to express the idea that the adversary did not actively influence those parts that determine the session key, say like a Diffie-Hellman share or a nonce. A similar concept is the notion of a *protocol core* introduced by Krawczyk [Kra16]. ▲

Example 3.9. Modeling an attacker which has access to the parties' private long-term keys but does not actively interfere with the communication can be relevant in a real-world scenario. Namely, consider a Big Brother type of adversary, like an ISP or a governmental three letter agency, which has massive data collection capabilities, and might even be able to obtain many of the users' long-term keys. Still, this Big Brother adversary will probably not be able to actively interfere with, say, *every* TLS connection on the Internet, even though it might be able to passively collect all communications. Thus, in this case we can still have security for those connections where the adversary does not *actively* use its knowledge of the private keys. The AKE^{fs} corruption model captures this possibility. ▲

So far we have discussed the AKE^{fs} corruption model in the scenario where the test-session has a partner. We now turn to the situation where the test-session does *not* have a partner. Going with the idea that existence of partners implies passiveness, the absence of a partner must mean that the adversary was active. In this scenario, if the adversary can obtain the long-term keys of the test-session's peers *before* it accepted, then we cannot in general give any security guarantees. This is because the adversary could be impersonating the peers of the test-session. However, the AKE^{fs} model still assures security as long as the Corrupt queries happen *after* the test-session accepts. This is shown at Line 29 in Figure 3.2.

To summarize, in the AKE^{fs} model the adversary can:

- reveal any session keys that does not belong to the test-session or its partner,
- (when passive) obtain any long-term keys it wants, at any time,
- (when active) obtain the long-term keys of unrelated parties at any point it wants, but the long-term keys of the test-session's peers can only be obtained *after* the test-session accepted.

AKE$^{\mathsf{wfs}}$. Many protocols that provide forward secrecy nevertheless fail to be secure according to the AKE$^{\mathsf{fs}}$ model. For example, the Diffie-Hellman based protocol HMQV [Kra05b] and NAXOS [LLM07] cannot be secure in this model (see [Kra05a, §3.2] for a description of the attack). But also EAP used without key-confirmation is insecure in the AKE$^{\mathsf{fs}}$ model. To see this, recall from Chapter 2 that EAP without key-confirmation consists of running an EAP method between the client and the server in order to establish a common session key MSK. This key is then transferred from the server to the authenticator using some secure key transport protocol. Recall also that the server and the authenticator use a long-term PSK to authenticate each other. Now consider the following attack. First, the adversary runs the EAP method to completion as normal. At this point the client has accepted so the adversary can select it as its test-session. Next, the adversary exposes the PSK shared between the server and authenticator and uses this to impersonate the authenticator towards the server. As a result, the server will send the MSK directly to the adversary using the key transport protocol. Note that this attack on basic EAP is valid in the AKE$^{\mathsf{fs}}$ model since the corruption of the PSK happened *after* the client test-session accepted. Hence, basic EAP cannot be secure in the AKE$^{\mathsf{fs}}$ model.

The problem is that the client in basic EAP does not have any guarantees that the second part of the protocol actually took place. As long as the second part has not completed, we cannot allow the adversary to obtain the long-term keys of the client's peers since this enables it to impersonate the authenticator. In other words, we require that the client must have a partner before we can safely leak the long-term keys. This is the idea of *weak forward secrecy*. The AKE$^{\mathsf{wfs}}$ model is obtained from the AKE$^{\mathsf{fs}}$ model by moving to a corruption model using weak forward secrecy instead of full forward secrecy. This is shown at Line 31 in Figure 3.2.

Bellare, Pointcheval, and Rogaway [BPR00] and Krawczyk [Kra05b] originally introduced the concept of weak forward secrecy in the context of two-flow (Diffie-Hellman based) protocols. There it was used to capture the problem that the adversary could modify the final message of the responder and then corrupt the initiator in order to learn the session key of the responder. Weak forward secrecy then demanded that the responder must have a partner before the initiator could be corrupted. Since the (SID-based) partnering mechanism used in [BPR00, Kra05b] included the sessions' local message transcripts, weak forward secrecy essentially amounted to saying that the adversary must be passive with respective to the test-session. This is another example of how partnering is used to encode passiveness.

A standard trick to upgrade protocols from weak forward secrecy to full forward secrecy is to add an additional key confirmation message to the protocol. For instance, the three-message variant of HMQV, called HMQV-C [Kra05a],

is constructed in exactly this way and can be shown to satisfy full forward secrecy. Foreshadowing our own results a bit, the combination of EAP with IEEE 802.11 can also be seen as an instance of this trick, where the IEEE 802.11 4WHS protocol provides key-confirmation to EAP. Basically, this idea lies at the center of one of our main composition results (Theorem 4.12 in Chapter 4), which shows generically that any 3P-AKE protocol with weak forward secrecy can be upgraded to achieve full forward secrecy by composing it with a 2P-AKE protocol with no forward secrecy.

AKE$^{\mathsf{nfs}}$. In order to accommodate protocols that do not provide forward secrecy—like the IEEE 802.11 4WHS protocol—we introduce the AKE$^{\mathsf{nfs}}$ model. The AKE$^{\mathsf{nfs}}$ model follows in the same vein as the AKE$^{\mathsf{fs}}$ and AKE$^{\mathsf{wfs}}$ models, but now the adversary is banned from obtaining any of the relevant long-term keys at *all* times. Long-term keys that are not relevant for the test-session can still be obtained as before.

3.3.2 Comparison with other models

Our three AKE models correspond almost one-to-one to three comparable models in [BPR00]. More precisely, our AKE$^{\mathsf{nfs}}$ model with no forward secrecy corresponds to their "basic" model, our AKE$^{\mathsf{fs}}$ model with full forward secrecy corresponds to their "forward secrecy" model, and our AKE$^{\mathsf{wfs}}$ model with weak forward secrecy corresponds to their "weak forward secrecy" model. To see this, compare our freshness predicates with the freshness notions given in Figure 2 of [BPR00] (where the freshness notion for the "weak forward secrecy" model is described in [BPR00, Remark 7]). However, there are two major differences between our models and those of [BPR00]. The first is that we are using partner functions and they are using SIDs. We already elaborated on the difference between partner functions and SIDs in Section 3.2.6. The second difference is that the corruption model in [BPR00] additionally allows the adversary to obtain a session's full state, including its internal randomness used to generate ephemeral values, and not only their secret long-term keys. Bellare, Pointcheval, and Rogaway call this the *strong* corruption model, as opposed to the *weak* corruption model [BPR00, Remark 3] where the adversary can only obtain the long-term keys (not to be confused with the notion of weak forward secrecy described above). Thus, using this language, our AKE models can be seen to directly correspond to those of [BPR00] in the *weak* corruption model (save for the use of different partnering mechanisms).

Let us expound upon the strong corruption model in order to explain why we are not covering it in this thesis. The possibility of giving the adversary access to the sessions' internal state has been a central motif in the so-called

Canetti–Krawczyk [CK01] and extended Canetti–Krawczyk [LLM07] models. Formally, this is captured by giving the adversary access to an additional query, usually called SessionStateReveal or EphemeralKeyReveal. The idea is that if a protocol mixes both ephemeral and long-term keys into the derivation of the session keys, then it is not sufficient for the adversary to only obtain one of them. Thus, security can still be achieved in the face of ephemeral key leakage. More generally, the aim of modern AKE models has been to capture more and more of the real-world threats that exists, such as bad randomness generators [FC14], side-channel attacks [ADW09, MO11, ASB14, ASB15], PKI subversion [Boy+13], and total long-term key compromise [CCG16]. Typically, this is achieved by defining increasingly stronger models that grant the adversary access to progressively more of a protocol's secret data and internal computations.

The reason why we are not capturing these more advanced features in our security models is because the real-world protocols of interest to this thesis, e.g., EAP, IEEE 802.11, TLS, IKEv2, and SSH, do not provide them. Thus, looking at stronger models is out of scope for this thesis. Nevertheless, since our composition results are quite generic and modular, we believe that they should be fairly robust in the face of changing models. That is to say, by making comparatively stronger assumptions on our underlying (protocol) building blocks, we should also be able to achieve correspondingly stronger results for our composed protocols as well. From this perspective, the choice of model should be rather orthogonal to the results of this thesis.

3.4 ACCE protocols

The world's most important security protocol, TLS, fails to be a secure AKE protocol in all its currently standardized versions (up to TLS 1.2 [RFC5246]). The reason is banal: some of TLS's key exchange messages are encrypted using the session key itself. Since AKE security is defined in terms of session key indistinguishability, this trivially makes it impossible to prove TLS secure as an AKE protocol. Specifically, after receiving the test-key from the challenger in experiment $\mathbf{Exp}_{\mathsf{TLS},\mathcal{Q}_{\mathsf{AKE}}}(\mathcal{A})$, the adversary can try to decrypt one of the encrypted handshake messages using the test-key. If the decryption succeeds, then the adversary knows that it got the real key, otherwise, it must have gotten a random key.

On the other hand, it seems unlikely that this property should make TLS any less secure in practice. More specifically, for the purpose of establishing a secure channel between two parties, TLS might be perfectly fine. In order to analyze TLS from this point of view, Jager et al. [Jag+12] introduced the notion of an *authenticated and confidential channel establishment (ACCE)* pro-

tocol. Intuitively, an ACCE protocol combines an ordinary AKE protocol with a *stateful authenticated encryption (stAE) scheme* into a monolithic protocol, where the session key from the AKE protocol is used to key the stAE scheme. The security goal is then shifted from providing indistinguishable session keys to instead providing secure channels using the established session keys. As a consequence, ACCE protocols have less utility than AKE protocols, in the sense that they provide no assurance on the use of their sessions keys beyond the fact that they are safe to use with the corresponding authenticated encryption scheme in the manner described by the protocol. By contrast, a classic result of Canetti and Krawczyk [CK01] shows that AKE protocols can be used to construct secure channels in a modular fashion. The more recent result of Brzuska et al. [Brz+11] further generalizes this to show that AKE protocols can be securely composed with essentially any type of symmetric key functionality in a similarly modular way.

Despite the merits of modularity, most real-world designs are unfortunately not as clean. Like TLS, many protocols use the established session key within the key exchange phase. This early session key usage prevents a modular analysis that can treat the AKE part and the channel part of a protocol separately. As a result, the ACCE model has been used to analyze several real-world protocols after its introduction, including multiple variants of TLS [Jag+12, KPW13b, KSS13, Li+14], SSH [Ber+14], and QUIC [Lyc+15]. In this thesis we are only going to apply the ACCE notion to two-party protocols, so we only define it in that setting.

Syntax. An ACCE protocol is a two-party protocol as defined in Section 3.2.3, together with an associated stAE scheme $\Lambda = (\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$. The formal definition of an stAE scheme is given in Appendix A.4. The notion of a session is the same as before, but the session state is extended with two additional variables st_{E} and st_{D} in order to store the encryption/decryption state of the stAE scheme.

ACCE security. The security of a (2P-)ACCE protocol Π is based on experiment $\mathbf{Exp}_{\Pi,\mathcal{Q}}(\mathcal{A})$ defined in Section 3.2.5. However, the base query set $\mathcal{Q}_{base}$ is extended with two additional queries, LR and $\mathsf{Decrypt}$, shown in Figure 3.3. The two additional queries allow the adversary to interact with the channels established in the protocol. The LR query takes in a session π, two messages M_0, M_1, and some optional additional data A. It returns the stateful encryption of either M_0 or M_1 under π's session key $\pi.k$. The $\mathsf{Decrypt}$ takes in a session π, a ciphertext C, and additional data A. It either always returns $\perp$ or potentially the decryption of C, provided the query was *out-of-sync* with respect to the LR query.

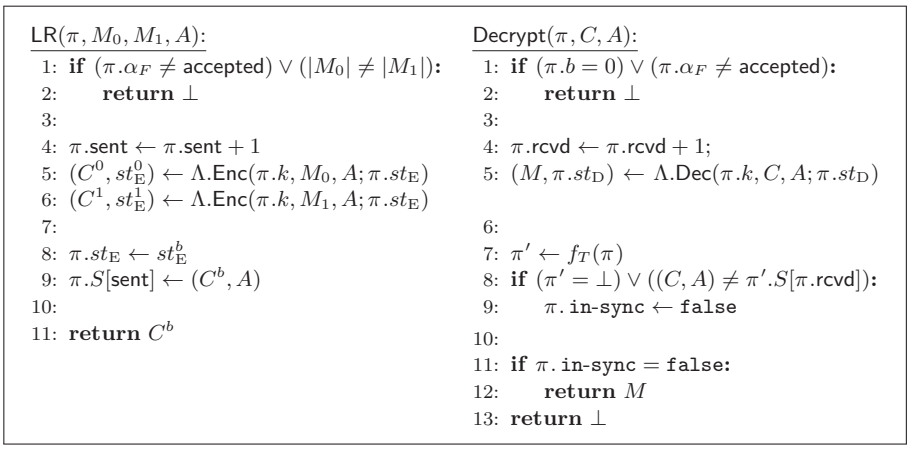


Figure 3.3: The LR and Decrypt queries for the ACCE security experiment.

The LR and Decrypt queries associate some additional variables to each session π_U^i, namely:

- b – a random bit drawn at the creation of session π_U^i;
- sent, rcvd – counters initialized to 0 and incremented for each call to $\mathsf{LR}(\pi_U^i, \cdot, \cdot, \cdot)$ and $\mathsf{Decrypt}(\pi_U^i, \cdot)$, respectively;
- $S[\cdot]$ – a list containing the sent ciphertexts and additional data returned from calls to $\mathsf{LR}(\pi_U^i, \cdot, \cdot, \cdot)$, we have $S[x] = \perp$ for all $x \notin [1, \mathsf{sent}]$;
- `in-sync` – a flag used to detect trivial wins by the adversary.

The variables b, sent, ..., are part of the security experiment $\mathbf{Exp}_{\Pi,\mathcal{Q}}(\mathcal{A})$, and not technically part of the syntax of an ACCE protocol. However, by abuse of notation, we use $\pi_U^i.b$, $\pi_U^i.\mathsf{sent}$, ..., also when referring to these variables.

The goal of an ACCE adversary is to guess the secret bit b of one of the sessions π. As before, the session needs to be fresh according to the freshness predicate $\mathsf{Fresh}_{\mathsf{ACCE}}$ (see Figure 3.2). Although the ACCE experiment is formulated as a distinguishing game, it captures both confidentiality and integrity goals. Particularly, for each session π, the adversary is challenged to distinguish between two worlds: one where the LR query returns the encryption of its left plaintext input and the Decrypt query always returns $\perp$ ($\pi.b = 0$); and one where the LR query returns the encryption of its right plaintext input and the Decrypt query returns the decryption of the supplied ciphertext provided it was out-of-sync ($\pi.b = 1$).

If the underlying stAE scheme does not provide confidentiality, then the LR query alone is enough to guess b. On the other hand, integrity is captured implicitly through the Decrypt query. If the adversary can successfully *forge* a ciphertext—meaning that it can produce an out-of-sync ciphertext which decrypts to something other than $\perp$—then it can use the output from the Decrypt query ($\perp$ vs $\neq \perp$) to determine the value of b. Notice that the out-of-sync requirement is needed in order to avoid trivial wins, since otherwise the adversary could just feed the output from the LR query directly to the Decrypt query and learn b. Finally, note that *stateless* authenticated encryption schemes cannot satisfy this definition. Specifically, for a stateless encryption scheme the adversary could use the LR query to first obtain a ciphertext C, and then query Decrypt on C *twice*. If $\pi.b = 0$, then the Decrypt query would return $\perp$ both times. If $\pi.b = 1$, then the first Decrypt query would return $\perp$ and the second query would return the decryption of C (since it is out-of-sync).

Let $\mathcal{Q} = \mathcal{Q}_{base} \cup \{\mathsf{LR}, \mathsf{Decrypt}\}$. Experiment $\mathbf{Exp}_{\Pi,\mathcal{Q}}(\mathcal{A})$ stops when $\mathcal{A}$ outputs a pair (π, b').

Definition 3.10 (ACCE security). Consider a run of experiment $\mathbf{Exp}_{\Pi,\mathcal{Q}}(\mathcal{A})$. Suppose (π, b') was the final output by $\mathcal{A}$. Fix a partner function f and define ACCE to be the following random variable on experiment $\mathbf{Exp}_{\Pi,\mathcal{Q}}(\mathcal{A})$:

$$\mathsf{ACCE} \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } (b' = \pi.b) \wedge \mathsf{Fresh}_{\mathsf{ACCE}}(\pi) = \texttt{true} \\ 0 & \text{if } (b' \neq \pi.b) \wedge \mathsf{Fresh}_{\mathsf{ACCE}}(\pi) = \texttt{true} \\ \widetilde{b} \twoheadleftarrow \{0,1\} & \text{if } \mathsf{Fresh}_{\mathsf{ACCE}}(\pi) = \texttt{false} \end{cases} \tag{3.9}$$

Let $\mathbf{Exp}^{\mathsf{ACCE}}_{\Pi,\mathcal{Q}}(\mathcal{A}) \Rightarrow d$ denote the event that $\mathsf{ACCE} = d$. The *ACCE advantage* of an adversary $\mathcal{A}$ is

$$\mathbf{Adv}^{\mathsf{ACCE}}_{\Pi,f}(\mathcal{A}) \stackrel{\text{def}}{=} 2 \cdot \Pr[\mathbf{Exp}^{\mathsf{ACCE}}_{\Pi,\mathcal{Q}}(\mathcal{A}) \Rightarrow 1] - 1. \tag{3.10}$$

3.5 Explicit entity authentication

One can observe the following consequence of our AKE[4] security definition. If a fresh session comes to accept a session key, then there can be at most one other session holding the same key, and this session must necessarily be its partner. Why? Suppose not, i.e., assume that π and π' accepts the same key but are not partners. If so, then the adversary can reveal one of them, test the other, and trivially break the AKE protocol—contradicting its supposed AKE security. Thus, π and π' must either have been partners or not accepted the

[4]The following applies equally to the ACCE security definition.

same key. By soundness (Definition 3.6), this implies that a fresh session π that accepts a key is assured that this key will be shared by at most one other session π', and that this session will reside at π's intended peer. However, notice that this authentication property is *implicit* in the sense that a session has no guarantee that its partner actually exists. For example, consider the client in EAP: after completing the EAP method with the server, it cannot know whether the subsequent key transfer from the server to the authenticator actually took place (at least without any further communication).

The opposite of implicit authentication is *explicit* authentication. Here the existence of a partner *is* guaranteed. Thus, explicit authentication adds the following aliveness property to a protocol. If a session at party A comes to accept, believing it has talked to party B, then some corresponding (unique) session at B must actually have contributed to this protocol run.

Note that the question of whether explicit (entity) authentication should be considered a requirement of secure AKE protocols is somewhat disputed in the literature (see [BR95, §1.6], [Rog04, §6], and [Kra03, §2.1]). Basically, the argument is that whether or not a partner is actually "out there" is ultimately irrelevant; it is the usage of the key that matters. For instance, even though the EAP client might not have a parter, once it starts using its accepted key, no one except its intended peer will actually be able to communicate with it. Thus, in the bigger picture, it is not so clear what benefits explicit authentication brings over implicit authentication. Consequently, most formal AKE models today do not require explicit entity authentication as a necessary security feature.

Remark 3.11. As opposed to computational AKE protocol models, which mostly treat authentication as an ancillary to the goal of key exchange, *symbolic* protocol models have historically focused extensively on the goal of authentication itself. As a result, they also have more refined definitions of authentication, including elaborate *authentication hierarchies* that consist of notions like "weak-aliveness", "injective-agreement", and "non-injective-sync"; see Chapter 4 of the book by Cremers and Mauw [CM12]. Our colloquial usage of the term "aliveness" in the preceding paragraph should not be interpreted in the literal sense of these technical notions (although the closest thing would probably be a combination of "weak-aliveness-role" and "injective-agreement"—see Figure 4.13 in [CM12]). ▲

Within our framework, the notion of explicit entity authentication is interchangeable with another property called *key-confirmation*. Key-confirmation is the property that if a session accepts a key, then it is assured that some other session must also have computed the same key (see [Fis+16] for a formal treatment of key-confirmation). Again, it might be debatable how useful this property is in practice, but it nevertheless is the key feature that allows us to

upgrade the security of EAP from weak forward secrecy to full forward secrecy. For this reason we find it useful to provide a formal definition of explicit entity authentication (and hence key-confirmation). However, we stress that the security properties we ultimately aim to satisfy in this thesis are key indistinguishability (AKE) and channel security (ACCE), as defined by Definition 3.7 and Definition 3.10, respectively. Explicit entity authentication is mostly used as a means to an end in order to achieve these goals.

Since explicit entity authentication is defined identically for AKE and ACCE protocols, we provide a single generic definition here.

Definition 3.12 (Explicit entity authentication). Let M be a security model. Consider a run of experiment $\mathbf{Exp}_{\Pi,\mathcal{Q}}(\mathcal{A})$ and fix a partner function f. A session π is said to have *accepted maliciously* if all of the following hold:

1. $\pi.\alpha_F = \mathsf{accepted}$,
2. $\mathsf{Fresh}_M(\pi) = \mathtt{true}$,
3. $f_T(\pi) = \perp$.

Let $\mathbf{Exp}^{\mathsf{M\text{-}EA}}_{\Pi,\mathcal{Q}}(\mathcal{A}) \Rightarrow 1$ denote the event that a session has accepted maliciously. The *EA advantage* of an adversary $\mathcal{A}$ is

$$\mathbf{Adv}^{\mathsf{M\text{-}EA}}_{\Pi,f}(\mathcal{A}) \stackrel{\text{def}}{=} \Pr[\mathbf{Exp}^{\mathsf{M\text{-}EA}}_{\Pi,\mathcal{Q}}(\mathcal{A}) \Rightarrow 1], \tag{3.11}$$

where $\mathsf{M} \in \{\mathsf{AKE}^*, \mathsf{ACCE}\}$.

Note that Definition 3.12 needs to be paired with soundness in order to be meaningful. That is, consider the partner function f that partners each session to itself the moment it is created. For this choice of partner function $\mathbf{Adv}^{\mathsf{M\text{-}EA}}_{\Pi,f}(\mathcal{A})$ equals 0 for all adversaries $\mathcal{A}$. Likewise, when we combine explicit entity authentication with, say, AKE security, then we only care about an adversary's advantage given the *same* partner function for both notions.

Chapter 4

Security of EAP

Contents

In this chapter we analyze the security of EAP using the formal models introduced in Chapter 3. However, first we need to precisely define what we mean by EAP and what type of security properties we expect it to provide. Recall from Section 2.1 that EAP is not a single protocol but rather a protocol framework which inherently depends on other concrete protocols. As summarized in Figure 2.1, EAP is essentially a composition of three separate protocols: an EAP method between the client and the server, a key-transport protocol between the server and the authenticator, and a link-layer specific protocol between the client and the authenticator. The EAP standard [RFC3748] is mostly agnostic as to which concrete protocol to actually use for each of these different parts.

4.1 Modeling EAP

With respect to security requirements, the base EAP standard [RFC3748] only specifies what the security properties of the EAP method should be. However, a supplementary RFC [RFC5247] describes the security goals of EAP as a whole. Section 1.5 of this document spells out the main requirements:

> The goal of the EAP conversation is to derive fresh session keys between the EAP [client] and authenticator that are known only to those parties, and for both the EAP [client] and authenticator to demonstrate that they are authorized to perform their roles either by each other or by a trusted third party (the backend authentication server).
>
> [...]
>
> The backend authentication server is trusted to transport keying material only to the authenticator that was established with the [client], and it is trusted to transport that keying material to no other parties. [...]

In other words, the goal of EAP is to be a secure 3P-AKE protocol. We now explain how we are going to model each of the components that make up the EAP framework. Since EAP is agnostic with respect to its components, we want to reflect this in our modeling as well. As far as possible, we try not to make any assumptions on the internal structure of the sub-protocols that are used to instantiate the EAP framework.

4.1.1 Client–server EAP method

The modeling of the EAP method between the client and the server is fairly straightforward. Since its goal is to distribute a shared key between the two parties, it can be naturally modeled as a 2P-AKE protocol with mutual (implicit) authentication. For simplicity, we are going to assume that the EAP method is based on public keys for its long-term credentials. This corresponds to EAP methods such as EAP-TLS [RFC5216] which we will study further in Chapter 5. Nevertheless, there is nothing fundamental about this assumption. Our results can easily be modified to also handle symmetric long-term keys, or even a combination of the two.

Channel binding. There is a well-known issue with the EAP architecture called the "lying authenticator problem" [RFC3748, Section 7.15], where a malicious authenticator may present false or inconsistent identity information to the different sides. Specifically, during the EAP method the client needs to

signal to the server which authenticator it is connecting to, so that the server can know where it is supposed to transfer the shared key. Unfortunately, many EAP methods do not authenticate this information. This can enable a rogue authenticator to impersonate another authenticator towards the client.

Concretely, suppose client A wants to connect to an authenticator B. Assume that they are both associated with the mutually trusted server S. Additionally, suppose there is also a malicious authenticator C associated with the same server S. When client A begins its EAP method exchange with server S, it also communicates the identity "B" to S. However, since this information is not authenticated, the authenticator C can change it to say "C". Consequently, once the EAP method completes, the server S will believe that A wanted to talk to C. As a result, S transfer the key it established with A to C instead of B.

Unless the EAP method authenticates the identify information there is no way for the client and server to verify that they are talking to the same authenticator. More generally, the process of ensuring that what the authenticator said to the client is consistent with what it said to the server is known in EAP as *channel binding*. There are two principal ways in which one can achieve channel binding in EAP [RFC6677, Section 4.1]. The first is to have the EAP method authenticate the necessary information directly during the exchange or in a separate integrity-protected channel after the shared key between the client and the server has been established. The second is to have the information that needs to be authenticated included into the derivation of the EAP session key.

There are advantages and disadvantages to both approaches. For example, the former allows for policy-based comparison of network properties where not all information necessarily have to match bit-for-bit on both ends, while in the latter this does not work. In contrast, authenticating the information by explicitly transferring it in an integrity-protected channel might require larger changes to the existing EAP methods than just including it into the key derivation.

Since we find it to be the cryptographically cleanest, we only consider channel binding based on key derivation in this thesis. Consequently, we are going to assume that there is a pseudorandom function PRF associated to each EAP method. On the other hand, we are not going to assume that the EAP method itself provides integrity protection of the identity information in any way. In fact, we are going to treat the communication of the authenticator's identity from the client to the server as being completely independent of the EAP method. This has the benefit of making it possible to analyze the EAP methods purely in terms of their concrete underlying authentication protocol.

4.1.2 Server–authenticator key transport protocol

After the EAP session key has been established between the client and the server, it needs to be transmitted to the authenticator. Without a doubt the most popular protocol for this purpose is RADIUS [RFC2865]. It is based on a long-term symmetric secret, i.e., a PSK, shared between the server and the authenticator. In order to transfer the EAP session key from the RADIUS server to the authenticator, the RADIUS protocol specifies a custom encryption scheme based on the Microsoft Point-to-Point Encryption (MPPE) algorithm [RFC2548]. Basically, MPPE is a stream cipher based on the cipher feedback (CFB) mode of operation using the MD5 hash function as its internal pseudorandom function. To encrypt an EAP session key $K = K_1 \| K_2 \| \cdots \| K_t$, consisting of 128 bit blocks K_i (with K_t possibly zero-padded), MPPE proceeds as follows:

$$C_1 \leftarrow \mathrm{MD5}(S \| R \| A) \oplus K_1, \tag{4.1}$$

$$C_i \leftarrow \mathrm{MD5}(S \| C_{i-1}) \oplus K_i. \tag{4.2}$$

Here S is the PSK shared between the server and the authenticator, R is a 128 bit random nonce, and A is a 16 bit salt. Peculiarly, the nonce R is *not* chosen by the server itself. Instead, it is generated by the authenticator and sent to the server in a previous RADIUS message. The ciphertext $C = C_1 \| C_2 \| \cdots \| C_t$ is integrity protected using HMAC-MD5[1] before being sent to the authenticator. The HMAC tag is computed with the same secret S that was used to encrypt K.

Although RADIUS is the most common server-to-authenticator protocol when using EAP, we choose not to model it explicitly in this thesis. There are a couple of reasons for this. First, while RADIUS is certainly the predominant choice of key transport protocol used together with EAP, it is not the *only* one. In particular, protocols like Diameter [RFC6733] and Cisco's TACACS+[Dah+17], are also frequently used. Thus, in keeping with our goal of capturing the generality of the EAP framework, we want our modeling to cover these protocols as well. Second, the RADIUS encryption mechanism described above has received very little scrutiny. That is to say, its CFB and HMAC building blocks have been heavily analyzed and are well understood (see e.g. [Woo08] and [Bel15]), but their specific usage within the RADIUS protocol is not. In particular, the non-standard way in which the random nonce R is chosen, as well as the reuse of the secret S in both MPPE and HMAC, are cause for concern. Ultimately, the security of RADIUS is largely unknown.

On the other hand, RADIUS is often used on top of other security protocols, like IPsec and TLS (see e.g. RADIUS-over-TLS [RFC6614]). Thus, it seems

[1] The original RADIUS standard [RFC2865] does not specify HMAC-MD5, but rather the MAC construction $\mathrm{MD5}(M \| S)$. However, a later RFC [RFC3579] dealing specifically with the combination of EAP + RADIUS, prescribes the use of HMAC-MD5.

reasonable to model the key transport protocol between the server and the authenticator as a generic ACCE protocol based on a symmetric PSK. Again, there is nothing fundamental about our choice of PSKs for long-term credentials, and our model could very well have included public keys as well. However, since RADIUS is so often configured with PSKs, it seems like a natural choice. It also has the added benefit of making our security analyses cleaner, since the long-term keys used by the EAP method and the key transport protocol are of distinct types (recall that we have assumed that EAP methods use public keys for long-term credentials).

4.1.3 Client–authenticator protocol

Let us call the combination of an EAP method and the subsequent key transport protocol *basic EAP*. Normally, basic EAP is followed by a link-layer specific protocol between the client and the authenticator, called a *Security Association Protocol* in [RFC5247]. Like the key transport protocol between the server and the authenticator, the Security Association Protocol is technically out of scope of the base EAP standard [RFC3748]. Nevertheless, Section 3.1 of [RFC5247] lists a number of recommended features that it ought to have. Chief among these are: mutual proof of possession of the EAP session key, generation of link-layer specific encryption keys, entity authentication, and secure negotiation of protocol capabilities. Save possibly for the last one, these are all features we expect from a PSK-based 2P-AKE protocol providing explicit entity authentication.

Let us call the combination of basic EAP with a subsequent Security Association Protocol *full EAP*. Given that full EAP is usually what is used in practice, our main aim in this chapter is to analyze this composition. However, rather than analyzing the full EAP all at once, we prefer a more modular approach. First, we establish the security of basic EAP under the assumptions we made on the EAP method and the key transport protocol in Section 4.1.1 and Section 4.1.2, respectively. Then, rather than viewing the full EAP as consisting of an EAP method, a key transport protocol, and a Security Association Protocol, we instead think of it as consisting of a black-box 3P-AKE protocol (i.e., basic EAP) combined with a PSK-based 2P-AKE protocol. Consequently, the two main results in this chapter are two generic composition theorems which correspond to the "cryptographic core" of basic and full EAP, respectively. Our results are modular and capture the compositional nature of EAP. Figure 4.1 gives a roadmap for the two composition results.

Preempting our results a bit, we show that basic EAP can achieve security in our weak forward secrecy model $\mathrm{AKE}^{\mathsf{wfs}}$, while full EAP can achieve security in our full forward secrecy model $\mathrm{AKE}^{\mathsf{fs}}$. Intuitively, the reason why basic

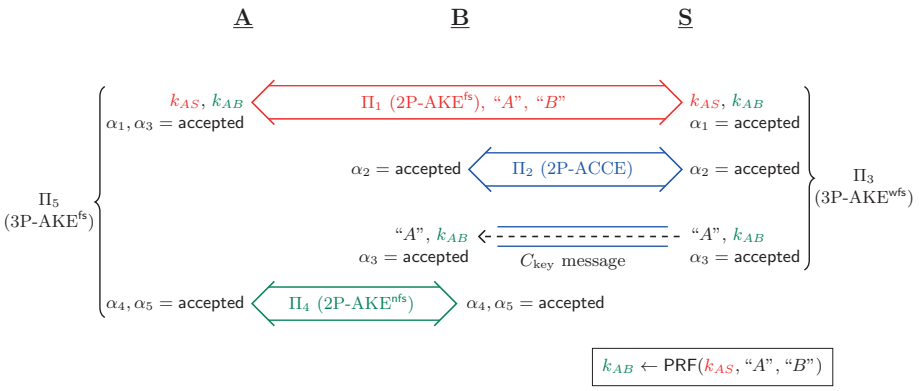


Figure 4.1: (Right) Construction of a 3P-AKE$^{\mathsf{wfs}}$ secure protocol Π_3 from a 2P-AKE$^{\mathsf{fs}}$ secure protocol Π_1, an ACCE secure protocol Π_2, and a pseudorandom function PRF. (Left) Construction of a 3P-AKE$^{\mathsf{fs}}$ secure protocol Π_5 from a 3P-AKE$^{\mathsf{wfs}}$ secure protocol Π_3 and a 2P-AKE$^{\mathsf{nfs}}$ secure protocol Π_4.

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, recall the following attack from Section 3.3 which illustrates why basic EAP does not provide full forward secrecy.

Suppose *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. The adversary can now impersonate 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 AKE$^{\mathsf{fs}}$ this attack is valid since the exposure of the PSK happened after the client accepted. On the other hand, in the weak forward secrecy model AKE$^{\mathsf{wfs}}$ the attack is not valid, because the client session does not have a partner and hence the PSK cannot be exposed.

To that end, the purpose of the link-layer protocol can be seen as providing key-confirmation to the basic 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 by adding a third flow to it [Kra05b, §3]. Interestingly, this also means that the forward secrecy of full EAP depends solely on the forward secrecy of the EAP method and not at all on the forward secrecy of the Security Association Protocol.

Finally, we point out that for technical reasons we cannot use the link-layer protocol in a completely black-box way, but need to assume a little bit of structure on it. Specifically, we need to assume that the probability that two sessions at the same party end up with the same local transcript is statistically bounded. Recall that a session's local transcript τ consists of all the messages it has sent and received. The reason for this assumption is that in the proof of our second composition result we need to rely on a local partner function, which are only defined for unique (global) transcripts (see Definition 3.5).

Note that this assumption is quite mild. It is met by any protocol where each protocol participant contributes some randomness, e.g., a nonce or Diffie-Hellman share. In particular, it is met by TLS, SSH, IKEv2 and the IEEE 802.11 4WHS protocol.

4.1.4 Related work on EAP

He et al. [He+05] have conducted a formal analysis of the combination of EAP-TLS and IEEE 802.11 in a symbolic model called the Protocol Compositional Logic. However, they do not threat the full generality of the EAP framework since they assume that the server and authenticator belong to the same entity (hence omitting the key transport stage). In the computational setting there are to the best of our knowledge no papers that have treated the general EAP framework. However, from a proof-technical point of view, our composition theorems are reminiscent of the composition result proven by Abdalla, Fouque and Pointcheval [AFP05].

Hoeper and Chen [HC07] have criticized the lack of a clear trust model and precise security definitions in the EAP standard [RFC3748], pointing out that this makes it difficult to formally assess its security. Regarding EAP methods specifically, Clancy and Hoeper [CH09] have stressed the importance of channel binding, demonstrating several real-world attacks that might be possible in its absence. Somewhat related to channel binding is the concept of *tunneled* EAP methods. These are EAP methods that first establish a secure channel (or tunnel) between the client and the authenticator having only unilateral or even no authentication. Then, a second authentication protocol is run inside the secure channel in order to provide mutual authentication. Examples of tunneled EAP methods are EAP-TTLS [RFC5281] and PEAP [PEAPv2] (see Table 2.1). A classic result by Asokan, Niemi and Nyberg [ANN03] shows that a man-in-the-middle attack is possible on tunneled authentication protocols unless the inner and outer layers are cryptographically bound together. Hoeper and Chen [HC10] have demonstrated that several existing tunneled EAP methods fail to properly bind the layers together. This has also been exploited in later attacks on EAP-TTLS and PEAP [Bha+14a]. However, we stress that we are

not covering tunneled EAP methods specifically in this thesis.

Finally, Horst et al. [Hor+16] have cryptanalysed the Point-to-Point Tunneling (PPTP) protocol in combination with RADIUS. PPTP is used to establish a virtual private network (VPN) between a client and a VPN endpoint with the help of a mutually trusted RADIUS server. The VPN endpoint functions very much like the authenticator in the EAP framework, and the RADIUS protocol is used to transfer a session key from the server to the VPN endpoint using the same MPPE encryption scheme described in Section 4.1.2. However, a crucial difference between the usage of MPPE in PPTP vs. EAP, is that in the former, the random nonce R provided by the VPN endpoint to the RADIUS server and used as input to the MPPE algorithm (see Equation (4.1)), is *not* protected by a MAC. This makes the attack of Horst et al. [Hor+16] possible against PPTP, but not against EAP.

4.2 First composition theorem: 2P-AKE$^{\mathsf{fs}}$ + 2P-ACCE $\Longrightarrow$ 3P-AKE$^{\mathsf{wfs}}$

In this section we state and prove our first composition theorem. This result corresponds to the cryptographic core of basic EAP.

Construction. From a 2P-AKE protocol Π_1 (based on public keys), a 2P-ACCE protocol Π_2 (based on PSKs), and a pseudorandom function PRF, we construct the 3P-AKE protocol Π_3 shown in Figure 4.1. Specifically, protocol Π_3 works as follows. First, sub-protocol Π_1 is run between client A and 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 the authenticator that A wants to talk to. These identities are sent independently of sub-protocol Π_1 and have no integrity protection.

Note that A knows the identities of 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 the key k_{AS} derived in sub-protocol Π_1, both A and S further derive the key $k_{AB} \leftarrow \mathsf{PRF}(k_{AS}, A, B)$. The key k_{AB} will ultimately be the session key shared between A and B in protocol Π_3. In order for S to transfer k_{AB} to B, they first establish a secure channel using sub-protocol Π_2. Once established, S sends the session key k_{AB} together with the identity of A over the channel to B. For simplicity, we assume that k_{AB} and "A" are transfered with a *single*

channel message, which we call the C_{key} *message*. Specifically,[2]

$$C_{\text{key}} \leftarrow \Lambda.\mathsf{Enc}(ck, k_{AB} \| \text{“}A\text{”}), \tag{4.3}$$

where Λ is the stAE scheme associated with the ACCE protocol Π_2; ck being the channel key that S and B established in Π_2. Unlike the identities “A”, “B” sent over the A–S link in Figure 4.1, the identity “A” sent over the S–B link enjoys privacy and integrity protection from the secure channel between S and B.

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, obtaining the session key k_{AB} as well as the identity “A” which it uses to update its peers variable.

Remark 4.1. Technically speaking, it would be possible to include the identity “A” only as associated data when creating the C_{key} message, since it does not need privacy protection. However, when RADIUS is being run on top of a secure channel protocol, like TLS or IPsec, everything is transmitted inside the encrypted channel anyway, so our approach in Equation (4.3) more closely matches real-world practice. ▲

Result. Our first composition result shows that protocol Π_3 is 3P-AKE$^{\mathsf{wfs}}$ secure if sub-protocol Π_1 is 2P-AKE$^{\mathsf{fs}}$ secure, sub-protocol Π_2 is 2P-ACCE secure, and PRF is a pseudorandom function. Note that protocol Π_3 does not provide explicit entity authentication. In fact, no session at the initiator A will have a partner at the time it accepts. As a consequence, protocol Π_3 cannot achieve security in the strongest AKE$^{\mathsf{fs}}$ model due to the attack on basic EAP described in Section 4.1.3.

Theorem 4.2. *Let Π_3 be the 3P-AKE protocol constructed from a 2P-AKE protocol Π_1, a ACCE protocol Π_2 and a pseudorandom function* PRF *as described in the construction above. Let f_1 and f_2 be partner functions. Then for any adversary $\mathcal{A}$ in security game AKE$^{\mathsf{wfs}}$ against Π_3, we can create a partner function f_3 and algorithms $\mathcal{B}_1$, $\mathcal{B}_2$, $\mathcal{B}_3$ and $\mathcal{D}$, such that*

$$\begin{aligned}\mathbf{Adv}^{\text{3P-AKE}^{\mathsf{wfs}}}_{\Pi_3,f_3}(\mathcal{A}) \leq\ & \mathbf{Adv}^{\text{ACCE-EA}}_{\Pi_2,f_2}(\mathcal{B}_1) + 2n^2 \cdot \mathbf{Adv}^{\text{2P-AKE}^{\mathsf{fs}}}_{\Pi_1,f_1}(\mathcal{B}_2,) \\ & + 2n^2 \cdot \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{PRF}}(\mathcal{D}) + 4n^2 \cdot \mathbf{Adv}^{\text{ACCE}}_{\Pi_2,f_2}(\mathcal{B}_3),\end{aligned} \tag{4.4}$$

where $n = (n_\pi + 1) \cdot |\mathcal{I} \cup \mathcal{R}|$, and n_π is the maximum number of sessions that $\mathcal{A}$ creates at each party.

[2] To simplify our exposition we omit both the associated data and the encryption state when writing the inputs to the stAE scheme Λ for the remainder of this chapter.

The proof of Theorem 4.2 roughly works as follows. The 2P-AKE$^{\mathsf{fs}}$ security of sub-protocol Π_1 allows us to swap out the intermediate key k_{AS} of the test-session with a random key. The PRF-security of the key-derivation function PRF then allows us to replace the derived session key k_{AB} with random key. Finally, the ACCE-security of sub-protocol Π_2 ensures that the adversary cannot modify any C_{key} messages nor can it learn anything about the session keys transferred inside them. Thus, at this point the adversary has zero advantage in winning in its 3P-AKE experiment.

Proof. We begin by defining the partner function f_3 using the given partner functions for sub-protocols Π_1 and Π_2. Remember that throughout this thesis we always assume that all partners functions are symmetric and monotone.

Defining the partner function for Π_3. Intuitively, f_3 is constructed by composing the two partner functions f_1 and f_2 as follows. If π_A^i is an initiator session and π_B^j is a responder session, then π_A^i and π_B^j are partners in protocol Π_3 according to f_3 if and only if there exists a server session π_S^k, such that π_A^i and π_S^k are partners in sub-protocol Π_1 according to f_1; and that π_S^k and π_B^j are partners in sub-protocol Π_2 according to f_2. That is, π_A^i and π_B^j are partners if there exists a server session π_S^k that acts as the bridge between them in the two sub-protocols Π_1 and Π_2.

To make this formally precise, one needs to extract from the 3P-AKE transcript T_3 of experiment $\mathbf{Exp}_{\Pi_3,\mathcal{Q}}(\mathcal{A})$ two transcripts T_1 and T_2 that contain the queries pertaining to the two-party sub-protocols Π_1 and Π_2. Then f_3 is defined on T_3 by running f_1 and f_2 on the corresponding transcripts T_1 and T_2. Admittedly, the details on how to do this are a bit tedious, so they are relegated to Appendix B. They can safely be skipped on first reading.

Suppose we have extracted transcripts T_1 and T_2 from T_3. We say that π_A^i and π_S^k are f_1-*partners* if $f_{1,T_1}(\pi_A^i) = \pi_S^k$. Since f_1 is symmetric this is equivalent to $f_{1,T_1}(\pi_S^k) = \pi_A^i$. Similarly, we say that π_S^k and π_B^j are f_2-*partners* if $f_{2,T_2}(\pi_S^k) = \pi_B^j$ (or equivalently $f_{2,T_2}(\pi_B^j) = \pi_S^k$). Finally, π_A^i and π_B^j are f_3-*partners*, or just *partners*, if $f_{3,T_3}(\pi_A^i) = \pi_B^j$, where f_3 is defined as follows.

- $f_{3,T_3}(\pi_A^i) = \pi_B^j$ if and only if:
  1. π_A^i and π_S^k are f_1-partners,
  2. π_S^k and π_B^j are f_2-partners,
  3. $\pi_A^i.\mathsf{peers} = \pi_B^j.\mathsf{peers} = \pi_S^k.\mathsf{peers} = \{A, B, S\}$.

Note that Item 3 implies that π_S^k received the same identities that π_A^i sent over the A–S link in Figure 4.1. By its construction, f_3 is monotone and symmetric provided f_1 and f_2 are. The soundness of f_3 follows from the soundness of f_1 and f_2 and the ACCE security of sub-protocol Π_2 (particularly its channel integrity). That is, it can be shown that

$$\mathbf{Adv}^{\mathsf{Sound}}_{\Pi_3,f_3}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{Sound}}_{\Pi_1,f_1}(\mathcal{A}') + \mathbf{Adv}^{\mathsf{Sound}}_{\Pi_2,f_2}(\mathcal{A}'') + \mathbf{Adv}^{\mathsf{ACCE}}_{\Pi_2,f_2}(\mathcal{A}'''). \tag{4.5}$$

However, for simplicity we are going to assume that f_1 and f_2 have perfect soundness in this proof, i.e., $\mathbf{Adv}^{\mathsf{Sound}}_{\Pi_1,f_1}(\mathcal{A}') = 0$ and $\mathbf{Adv}^{\mathsf{Sound}}_{\Pi_2,f_2}(\mathcal{A}'') = 0$. Unfortunately, this does not imply that f_3 has perfect soundness too, since an adversary could potentially forge a C_{key} message so that two partners end up with different keys. Thus, in order to enforce perfect soundness for f_3 as well, we extend its definition with the following requirement:

4. the C_{key} message received by π_B^j was identical to the one produced by π_S^k.

This removes the $\mathbf{Adv}^{\mathsf{ACCE}}_{\Pi_2,f_2}(\mathcal{A}''')$ term in Equation (4.5) and gives f_3 perfect soundness. In the remainder we can thus always assume that f_3-partners have the same session key.

AKE$^{\mathsf{wfs}}$ security. Our proof for protocol Π_3, as well as most of the other proofs in this thesis, make use of a proof technique called *game hopping* ([Sho04, BR04]). In a game hopping proof, one incrementally introduce small changes to the original security game—each change being called a *game hop*—which, after a finite number of hops, eventually leads to a situation where the adversary cannot win by definition, or where it is easy to bound its advantage in terms of something else. Each individual game hop is justified by showing that the change does not substantially affect the adversary's winning chances.

The most fundamental result on game hoping proofs is the so-called Difference Lemma [Sho04], sometimes also called the Fundamental Lemma of Game-Playing [BR04]. Basically, the Difference Lemma states that if two games proceed identically unless the some event F occurs, then the difference between the adversary's advantage in the two games is bounded by $\Pr[F]$. We will use this result many times in our thesis.

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$^{\mathsf{wfs}}$ security game, hence

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

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

Claim 4.3.

$$\mathbf{Adv}^{\mathsf{G}_0}_{\Pi_3,f_3}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{G}_1}_{\Pi_3,f_3}(\mathcal{A}) + \mathbf{Adv}^{\mathsf{ACCE\text{-}EA}}_{\Pi_2,f_2}(\mathcal{B}_1). \tag{4.6}$$

Proof. Let E be the event that a server or responder session accepts maliciously in sub-protocol Π_2. Game 0 and Game 1 proceed identically as long as event E does not occur, so by the Difference Lemma we have

$$\mathbf{Adv}^{\mathsf{G}_0}_{\Pi_3,f_3}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{G}_1}_{\Pi_3,f_3}(\mathcal{A}) + \Pr[E]. \tag{4.7}$$

To bound $\Pr[E]$ we create an algorithm $\mathcal{B}_1$ which breaks the explicit entity authentication of sub-protocol Π_2 whenever event E occurs in Game 0. Algorithm $\mathcal{B}_1$ begins by creating all the long-term keys for sub-protocol Π_1, selects a random bit b_{sim}, and then runs $\mathcal{A}$. Since $\mathcal{B}_1$ has created all the long-term keys for sub-protocol Π_1, it can derive all the session keys k_{AB} itself, and simulate all of $\mathcal{A}$'s queries pertaining to sub-protocol Π_1.

When $\mathcal{A}$ makes a Send query that pertains to sub-protocol Π_2, then $\mathcal{B}_1$ answers it by making a corresponding Send query to its 2P-ACCE-EA security game. When a server session π^k_S accepts in the 2P-ACCE-EA security game, then $\mathcal{B}_1$ creates its C_{key} message by making the query $\mathsf{LR}(\pi^k_S, k_{AB}, k_{AB})$[3], where k_{AB} is session key $\mathcal{B}_1$ derived for π^k_S in sub-protocol Π_1. When $\mathcal{A}$ issues a $\mathsf{Test}(\pi^i_U)$ query, then, depending on bit b_{sim}, $\mathcal{B}_1$ returns π^i_U's real session key or a random key. Finally, when $\mathcal{A}$ terminates, then $\mathcal{B}_1$ terminates too (in this case event E has not occurred).

To analyze $\mathcal{B}_1$'s winning probability, observe that $\mathcal{B}_1$ provides a perfect simulation of protocol Π_3 for $\mathcal{A}$. Moreover, since the freshness predicate $\mathsf{Fresh}_{\mathsf{AKE}^{\mathsf{wfs}}}$ is strictly more restrictive than predicate $\mathsf{Fresh}_{\mathsf{ACCE}}$, if event E occurs in Game 0, then a malicious accept also occurs in $\mathcal{B}_1$'s 2P-ACCE-EA security game. ■

Remark 4.4. 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, no initiator session will have a partner at the time when it accepts, because at that point sub-protocol Π_2 has not even started yet. ▲

Game 2: This game implements a *selective* AKE security game [KPW13a, §3.3], rather than the normal adaptive one. That is, at the beginning of the game, the adversary is required to commit to its choice of test-session and its

[3] For the remainder of this proof we omit the associated data input to the LR query, since the C_{key} message does not depend on it.

partner (if any). Technically, at the beginning of the game the adversary must output two pairs (U, i) and (V, j), with $i \in [1, n_\pi]$ and $j \in [0, n_\pi]$, where n_π is an upper bound on the number of sessions at each party, and a choice of $j = 0$ means that π_U^i is not intended to get a partner. Game 2 then proceeds as in Game 1, except that if either of the following events occur, then the challenger penalizes the adversary by outputting a random bit at the end.

(i) π_U^i was not selected as the test-session by $\mathcal{A}$.

(ii) π_U^i gets a different partner than π_V^j (including the case that it *gets* a partner if $j = 0$).

Claim 4.5.

$$\mathbf{Adv}^{\mathsf{G}_1}_{\Pi_3, f_3}(\mathcal{A}) \leq (n_\pi + 1)^2 \cdot |\mathcal{I} \cup \mathcal{R}|^2 \cdot \mathbf{Adv}^{\mathsf{G}_2}_{\Pi_3, f_3}(\mathcal{A}'). \tag{4.8}$$

Proof. From an adversary $\mathcal{A}$ that wins against the adaptive game in Game 1, we create the following adversary $\mathcal{A}'$ that wins against the selective game in Game 2. First, $\mathcal{A}'$ randomly selects a pair $(U, i) \twoheadleftarrow (\mathcal{I} \cup \mathcal{R}) \times [1, n_\pi]$ and a pair (V, j), which, depending on the role of U, is either selected as $(V, j) \twoheadleftarrow \mathcal{I} \times [0, n_\pi]$ or $(V, j) \twoheadleftarrow \mathcal{R} \times [0, n_\pi]$. It outputs (U, i) and (V, j) 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 $1/\left((n_\pi + 1) \cdot |\mathcal{I} \cup \mathcal{R}|\right)^2$. 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 $1/2$ in Game 2. ■

In the remaining games, let π_U^i and π_V^j denote the targets that the adversary commits to in Game 2; π_U^i being the test-session, and π_V^j being its (potentially empty) partner. Define the *co-partner* of π_U^i to be the server session being involved in the protocol run between π_U^i and π_V^j. Specifically, if π_U^i has the initiator role, then its co-partner is defined to be $f_{1,T_1}(\pi_U^i)$; while if π_U^i has the responder role, then its co-partner is defined to be $f_{2,T_2}(\pi_U^i)$.

Note that if π_U^i has the initiator role, then it does not necessarily have a co-partner when it accepts. On the other hand, if π_U^i has the responder role, then its co-partner is guaranteed to exist by Game 1.

Game 3: This game proceeds as the previous one, except that it replaces the intermediate key k_{AS} derived in sub-protocol Π_1 with a random key in the protocol run involving the test-session. That is, for the session out of π_U^i

and π_V^j that has the initiator role in protocol Π_3, the challenger replaces its intermediate key k_{AS} in sub-protocol Π_1 with a random key. Moreover, the intermediate key derived by the co-partner of π_U^i (if any) is also replaced with the same random key.

Claim 4.6.

$$\mathbf{Adv}^{\mathsf{G}_2}_{\Pi_3,f_3}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{G}_3}_{\Pi_3,f_3}(\mathcal{A}) + 2 \cdot \mathbf{Adv}^{\text{2P-AKE}^{\mathsf{fs}}}_{\Pi_1,f_1}(\mathcal{B}_2). \tag{4.9}$$

Proof. We show that if it is possible to distinguish Game 2 and Game 3, then we can create an algorithm $\mathcal{B}_2$ that breaks the 2P-AKE$^{\mathsf{fs}}$ security of sub-protocol Π_1. Algorithm $\mathcal{B}_2$ begins by drawing a random bit b_{sim} and creates all the long-term PSKs for sub-protocol Π_2. $\mathcal{B}_2$ then runs $\mathcal{A}$ and forwards all its queries that pertain to sub-protocol Π_1 to its own 2P-AKE$^{\mathsf{fs}}$ security game. All queries that pertain to sub-protocol Π_2, $\mathcal{B}_2$ answers itself using the PSKs it created. It also implements all the abort conditions of the previous games. To answer $\mathcal{A}$'s $\mathsf{Test}(\pi_U^i)$ query, $\mathcal{B}_2$ proceeds as follows.

If $b_{sim} = 1$, then $\mathcal{B}_2$ responds as normal by drawing a random key and returning this to $\mathcal{A}$. If $b_{sim} = 0$ and π_U^i is an initiator session, then $\mathcal{B}_2$ makes a corresponding $\mathsf{Test}(\pi_U^i)$ query to its own 2P-AKE$^{\mathsf{fs}}$ security game to obtain a key k_{AS}^* (which is either π_U^i's real session key in sub-protocol Π_1 or a random key). From k_{AS}^*, $\mathcal{B}_2$ derives $k_{AB} \leftarrow \mathsf{PRF}(k_{AS}^*, A, B)$ which it returns back to $\mathcal{A}$ (A, B being the party identities accepted by the test-session).

If $b_{sim} = 0$ and π_U^i is a responder session, then π_U^i must have a co-partner π_S^k by Game 1. To obtain the intermediate key k_{AS}^* needed to derive the session key k_{AB}, $\mathcal{B}_2$ does the same thing as above, but this time by issuing the Test query to π_S^k.

When $\mathcal{A}$ outputs its guess b', then $\mathcal{B}_2$ stops and outputs 0 to its 2P-AKE$^{\mathsf{fs}}$ security game if $b' = b_{sim}$, and 1 otherwise.

Note that if the key k_{AS}^* returned from the Test query in $\mathcal{B}_2$'s 2P-AKE$^{\mathsf{fs}}$ security game is real, then $\mathcal{B}_2$ perfectly simulates Game 2. On the other hand, if k_{AS}^* is a random key, then $\mathcal{B}_2$ perfectly simulates Game 3. Thus, the claim follows if we can show that the test-session chosen by $\mathcal{B}_2$ in its own 2P-AKE$^{\mathsf{fs}}$ security game is fresh according to predicate $\mathsf{Fresh}_{\mathsf{AKE}^{\mathsf{fs}}}$ whenever the test-session π_U^i chosen by $\mathcal{A}$ is fresh according to predicate $\mathsf{Fresh}_{\mathsf{AKE}^{\mathsf{wfs}}}$.

If π_U^i is an initiator session, then $\mathcal{B}_2$ uses the same session π_U^i as the test-session target in its own 2P-AKE$^{\mathsf{fs}}$ security game. Since the freshness predicate $\mathsf{Fresh}_{\mathsf{AKE}^{\mathsf{wfs}}}$ is strictly more restrictive than predicate $\mathsf{Fresh}_{\mathsf{AKE}^{\mathsf{fs}}}$, it follows that π_U^i is fresh in $\mathcal{B}_2$'s 2P-AKE$^{\mathsf{fs}}$ game whenever it is fresh in $\mathcal{B}_2$'s simulation for $\mathcal{A}$.

If π_U^i is a responder session, then the test-session chosen by $\mathcal{B}_2$ is π_U^i's co-partner π_S^k. We need to argue that π_S^k is fresh in $\mathcal{B}_2$'s 2P-AKE$^{\mathsf{fs}}$ security game whenever π_U^i is AKE$^{\mathsf{wfs}}$ fresh in $\mathcal{B}_2$'s simulation. There are two cases to

consider: either π_U^i has an f_3-partner or it does not. If π_U^i has a partner (which by Game 2 must be π_V^j), then $\mathcal{A}$ cannot have made a $\mathsf{Reveal}(\pi_V^j)$ query since this would violate the AKE$^{\mathsf{wfs}}$ freshness of π_U^i. Moreover, since f_3 is constructed from f_1 and f_2, π_V^j must be π_S^k's f_1-partner. Consequently, $\mathcal{B}_2$ is also allowed to forward any $\mathsf{Corrupt}$ query to either A or S without violating the freshness of π_S^k according to predicate $\mathsf{Fresh}_{\mathsf{AKE}^{\mathsf{fs}}}$.

If π_U^i 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$^{\mathsf{wfs}}$ freshness). Thus, neither has $\mathcal{B}_2$. Furthermore, if π_U^i does not have an f_3-partner, then this implies that its co-partner π_S^k cannot have an f_1-partner either. Thus, $\mathcal{B}_2$ can safely forward all of $\mathcal{A}$'s Reveal queries without violating the AKE$^{\mathsf{fs}}$ freshness of π_S^k. ■

Game 4: This game proceeds as the previous one, except that when deriving the session key k_{AB} in the protocol run involving the test-session π_U^i, the challenger uses a random function $\$(\cdot,\cdot)$ rather than the function $\mathsf{PRF}(k_{AS},\cdot,\cdot)$.

More specifically, if π_U^i has the initiator role then its session key k_{AB} is derived using the random function $\$(\cdot,\cdot)$ instead of the function $\mathsf{PRF}(k_{AS},\cdot,\cdot)$. Additionally, if π_U^i has a co-partner π_S^k, then π_S^k uses the same random function to derive the key k_{AB} that it will forward in its C_{key} message.

If π_U^i has the responder role, then it must have a co-partner π_S^k by Game 1. When deriving the key k_{AB} that π_S^k will use for its C_{key} message, the challenger uses the random function $\$(\cdot,\cdot)$ instead of the function $\mathsf{PRF}(k_{AS},\cdot,\cdot)$.

Claim 4.7.

$$\mathbf{Adv}^{\mathsf{G}_3}_{\Pi_3,f_3}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{G}_4}_{\Pi_3,f_3}(\mathcal{A}) + 2\cdot\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{PRF}}(\mathcal{D}). \tag{4.10}$$

Proof. We show that if it is possible to distinguish Game 3 and Game 4, then we can create a distinguisher algorithm $\mathcal{D}$ against the PRF security of the function PRF. Distinguisher $\mathcal{D}$ has access to an oracle $\mathcal{O}$ which either implements the function $\mathsf{PRF}(\widetilde{k},\cdot,\cdot)$ using an independent and uniformly distributed key $\widetilde{k}$, or it implements a random function $\$(\cdot,\cdot)$. $\mathcal{D}$ begins by drawing a random bit b_{sim} 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 3 by using the keys it created, except that it answers $\mathcal{A}$'s $\mathsf{Test}(\pi_U^i)$ query as follows.

If $b_{sim}=1$, then $\mathcal{D}$ returns a random key as normal. If $b_{sim}=0$ and π_U^i is an initiator session, then $\mathcal{D}$ answers with $\mathcal{O}(A,B)$, where A, B are the party identities accepted by π_U^i. If $b_{sim}=0$ and π_U^i is a responder session, then $\mathcal{D}$ does the same, but this time A, B are the identities that the co-partner of π_U^i received over the A–S link in Figure 4.1 (recall that if π_U^i is a responder session then 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 to its PRF-game if $b' = b_{sim}$, and 1 otherwise.

When $\mathcal{D}$'s oracle $\mathcal{O}$ implements $\mathsf{PRF}(\widetilde{k}, \cdot, \cdot)$, then $\mathcal{D}$ perfectly simulates Game 3; while if $\mathcal{O}$ implements a random function $\$(\cdot, \cdot)$, then $\mathcal{D}$ perfectly simulates Game 4. For $x \in \{3, 4\}$, let $G_x^{\mathcal{A}} \Rightarrow 1$ denote the event that $\mathcal{A}$ wins in Game x. Then

$$\mathbf{Adv}_F^{\mathsf{prf}}(\mathcal{A}) = \Pr[\mathcal{A}^{\mathsf{PRF}(\widetilde{k},\cdot,\cdot)} \Rightarrow 1] - \Pr[\mathcal{A}^{\$(\cdot,\cdot)} \Rightarrow 1] \tag{4.11}$$

$$= \Pr[G_3^{\mathcal{A}} \Rightarrow 1] - \Pr[G_4^{\mathcal{A}} \Rightarrow 1], \tag{4.12}$$

and the claim follows. ■

At this point one might expect that the adversary should be unable to distinguish the test-key from random since the session key of π_U^i is now derived using a random function rather than the pseudorandom function PRF. Unfortunately, we cannot (currently) rule out that $\mathcal{A}$ might be able to learn something about the session key through the C_{key} message delivered from the server to the responder. Furthermore, $\mathcal{A}$ could potentially also modify the C_{key} message in such a way that it still decrypts to the same session key. In this case π_U^i and π_V^j would end up with the same key while at the same time not being partners according to the definition of the partner function f_3. Hence, $\mathcal{A}$ could reveal π_V^j and trivially win in Game 4.

In the following two games we show that neither of these scenarios are possible due to the ACCE security of sub-protocol Π_2. In the first game we show that $\mathcal{A}$ is unable to successfully forge the C_{key} message in the protocol run involving π_U^i. In the second game we show that $\mathcal{A}$ is unable learn anything about the session key from observing the C_{key} message.

Game 5: Suppose π_U^i has a co-partner π_S^k and that the ciphertext C was the C_{key} message produced by π_S^k (if it created one at all). Let π^* be the f_2-partner of π_S^k in sub-protocol Π_2 required to exist by Game 1. Game 5 proceeds as Game 4, but if π^* receives a C_{key} message $C' \neq C$, then C' is automatically rejected, i.e., it is assumed to have decrypted to $\perp$. In this case π^*'s session key is not set.

Remark 4.8. Note that if π_U^i has the responder role, then π^* is π_U^i itself, while if π_U^i has the initiator role then π^* (if it exists) is some responder session. We write "if it exists" because if π_U^i has the initiator role then it might not actually have a co-partner at all since sub-protocol Π_1 does not give any guarantees of explicit entity authentication. However, in that case there is no difference between Game 4 and Game 5 since no relevant C_{key} message is being created. ▲

Claim 4.9.

$$\mathbf{Adv}^{\mathsf{G}_4}_{\Pi_3,f_3}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{G}_5}_{\Pi_3,f_3}(\mathcal{A}) + 2 \cdot \mathbf{Adv}^{\mathsf{ACCE}}_{\Pi_2,f_2}(\mathcal{B}'_3). \tag{4.13}$$

Proof. Assume π^i_U has a co-partner π^k_S, and that π^k_S produced the ciphertext C as its C_{key} message. Let session π^* be a session with the same definition as given in the game description above, and let F denote the event that $\mathcal{A}$ successfully forges the C_{key} message being delivered to π^*.

- *Event* F: $\mathcal{A}$ sends to π^* a C_{key} message $C' \neq C$, and C' decrypts to something other than $\perp$.

As long as event F does not occur then Game 4 and Game 5 are identical, so by the Difference Lemma we have

$$\mathbf{Adv}^{\mathsf{G}_4}_{\Pi_3,f_3}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{G}_5}_{\Pi_3,f_3}(\mathcal{A}) + \Pr[F]. \tag{4.14}$$

To bound $\Pr[F]$ we create an adversary $\mathcal{B}'_3$ that capitalizes on the event F in order to break the ACCE security of sub-protocol Π_2. Algorithm $\mathcal{B}'_3$ begins by drawing a random bit b_{sim} and creates all the long-term keys for sub-protocol Π_1. All of $\mathcal{A}$'s Send queries that pertain to sub-protocol Π_1, $\mathcal{B}'_3$ answers itself using the long-term keys it created. Particularly, $\mathcal{B}'_3$ can answer all $\mathsf{Corrupt}$ queries targeting the asymmetric long-term keys of initiators and servers itself. Moreover, it can also answer all of $\mathcal{A}$'s Reveal queries that target initiator sessions, and answer the Test query if π^i_U if $U \in \mathcal{I}$ (using the bit b_{sim}).

Send queries that pertain to sub-protocol Π_2, as well as $\mathsf{Corrupt}$ queries that target the PSKs shared between servers and responders, are forwarded to $\mathcal{B}'_3$'s 2P-ACCE security game. Whenever a server session π accepts in sub-protocol Π_2, then $\mathcal{B}'_3$ creates its C_{key} message by querying $\mathsf{LR}(\pi, A\|k_{AB}, A\|k_{AB})$[4] to its ACCE experiment, where "A" is the party identity that π received on the A–S link in Figure 4.1, and k_{AB} is the session key that $\mathcal{B}'_3$ derived from π's intermediate key k_{AS} in sub-protocol Π_1.

Whenever $\mathcal{A}$ forwards a C_{key} message to a responder session different from π^*, then $\mathcal{B}'_3$ first makes a Reveal query to that session in its ACCE security game in order to obtain its channel-key for sub-protocol Π_2. Using this channel-key, $\mathcal{B}'_3$ decrypts the received C_{key} message and simulates the responder session accordingly. Consequently, $\mathcal{B}'_3$ can also answer all of $\mathcal{A}$'s Reveal queries targeting responder sessions different from π^*.

If $\mathcal{A}$ at any point stops during $\mathcal{B}'_3$'s simulation, not having sent a C_{key} message to π^*, then $\mathcal{B}'_3$ stops too and outputs (π, b') to its ACCE security game where π is an arbitrary session and b' is a random bit, i.e., $b' \twoheadleftarrow \{0,1\}$.

[4]Here we are abusing notation and using "π" to denote both the server session that $\mathcal{B}'_3$ simulates for $\mathcal{A}$ in protocol Π_3, as well as the corresponding "proxy" server session that $\mathcal{B}'_3$ creates in its own ACCE security game in order to answer the queries to the former.

If $\mathcal{A}$ *does* forward a C_{key} message C' to π^*, then $\mathcal{B}'_3$ stops its simulation and outputs (π^*, b') to its ACCE game, where the bit b' is determined as follows. If $C' = C$, where C is the C_{key} message produced by π^i_U's co-partner π^k_S, then this cannot be a forgery, so $\mathcal{B}'_3$ lets b' be a random bit. On the other hand, if $C' \neq C$, meaning that C' is a potential C_{key} message forgery, then $\mathcal{B}'_3$ first makes the query $\mathsf{Decrypt}(\pi^*, C')$ to its ACCE game. If the $\mathsf{Decrypt}$ query returns something other than $\perp$, $\mathcal{B}'_3$ outputs 1, and in all other cases outputs a random bit.

We now analyze $\mathcal{B}'_3$. If $\mathcal{A}$ does not send a C_{key} message to π^* during $\mathcal{B}'_3$'s simulation then $\mathcal{B}'_3$ outputs (π, b') to its 2P-ACCE security, where π is an arbitrary session and b' a random bit. In this case $\mathcal{B}'_3$ wins with probability 1/2.

If $\mathcal{A}$ *does* send a C_{key} message to π^*, then $\mathcal{B}'_3$ picks π^* as its ACCE target. We begin by arguing that π^* is fresh according to predicate $\mathsf{Fresh}_{\mathsf{ACCE}}$, provided $\mathcal{A}$'s test-target π^i_U is fresh according to predicate $\mathsf{Fresh}_{\mathsf{AKE}^{\mathsf{wfs}}}$. Since $\mathcal{B}'_3$'s simulation stops immediately once π^* accepts, it never makes a Reveal query to π^*, and so we only have to consider the effects of $\mathsf{Corrupt}$ queries against the PSK shared between π^* and its server peer.

If π^i_U has the responder role then $\pi^i_U = \pi^*$. By Game 1, π^* must have an f_2-partner in sub-protocol Π_2 and its ACCE freshness follows immediately since then any long-term key can legally be exposed; in particular, this includes the PSK shared between π^* and its server peer.

Now suppose π^i_U has the initiator role. If π^i_U does not have a co-partner or this co-partner never reached the accept state (hence not producing a C_{key} message), then there is nothing to prove since then there is also no π^* session. On the other hand, if π^i_U has co-partner π^k_S which created a C_{key} message C, then by Game 1 there must be some session π^* being the f_2-partner of π^k_S. If $\mathcal{A}$ forwards C unmodified to π^*, then π^i_U and π^* would be f_3-partners and so the ACCE freshness of π^* would again follow immediately. Conversely, if $\mathcal{A}$ sends $C' \neq C$ to π^*, then π^i_U and π^* would not be f_3-partners. Hence, if π^i_U is to be fresh according to predicate $\mathsf{Fresh}_{\mathsf{AKE}^{\mathsf{wfs}}}$, then the long-term keys of its peers cannot have been exposed. In particular, this means that the PSK of π^* cannot have been exposed. It follows that π^* is fresh according to predicate $\mathsf{Fresh}_{\mathsf{ACCE}}$.

It remains to calculate $\mathcal{B}'_3$'s winning probability when $\mathcal{A}$ forwards a C_{key} message to π^*. That is, if $\mathcal{B}'_3$ picked π^* as its ACCE target. If the C_{key} message that π^* received was forwarded unmodified from its f_2-partner π^k_S, then $\mathcal{B}'_3$ outputs a random bit and thus wins with probability 1/2. On the other hand, if the C_{key} message that π^* received was different from the one that π^k_S sent out, then there is a potential for event F to occur. Note that $\mathcal{B}'_3$ perfectly simulates Game 4 until $\mathcal{A}$ sends a C_{key} message to π^*, so the

probability that F occurs in $\mathcal{B}_3'$'s simulation is the same as the probability that F occurs in Game 4.

Let C' be the C_{key} message that π^* received. Recall that $\mathcal{B}_3'$ outputs 1 only if the $\mathsf{Decrypt}(\pi^*, C')$ query returned something other than $\perp$, and a random bit otherwise. Since a $\mathsf{Decrypt}$ query in the ACCE experiment returns something other than $\perp$ only if $\pi^*.b = 1$, we have

$$\Pr[\mathbf{Exp}^{\mathsf{ACCE}}_{\Pi_2,\mathcal{Q}}(\mathcal{B}_3') \Rightarrow 1 \mid F \wedge \pi^*.b = 1] = 1, \tag{4.15}$$

and

$$\Pr[\mathbf{Exp}^{\mathsf{ACCE}}_{\Pi_2,\mathcal{Q}}(\mathcal{B}_3') \Rightarrow 1 \mid F \wedge \pi^*.b = 0] = \frac{1}{2}. \tag{4.16}$$

Finally, notice that the value of π^*'s secret bit b in $\mathcal{B}_3'$'s ACCE security game is independent of event F. This is because there is nothing in $\mathcal{B}_3'$'s simulation that depends on $\pi^*.b$ before π^* receives the C_{key} message, and $\mathcal{B}_3'$'s simulation stops immediately once this happens. Thus

$$\Pr[\pi^*.b = b \mid F] = \frac{1}{2}. \tag{4.17}$$

Conditioning on event F occurring, the winning probability of $\mathcal{B}_3'$ is

$$\begin{aligned} \Pr[\mathbf{Exp}^{\mathsf{ACCE}}_{\Pi_2,\mathcal{Q}}(\mathcal{B}_3') \Rightarrow 1 \mid F] &= \Pr[\mathbf{Exp}^{\mathsf{ACCE}}_{\Pi_2,\mathcal{Q}}(\mathcal{B}_3') \Rightarrow 1 \mid F \wedge \pi^*.b = 0] \cdot \frac{1}{2} \\ &\quad + \Pr[\mathbf{Exp}^{\mathsf{ACCE}}_{\Pi_2,\mathcal{Q}}(\mathcal{B}_3') \Rightarrow 1 \mid F \wedge \pi^*.b = 1] \cdot \frac{1}{2} \end{aligned} \tag{4.18}$$

$$= \frac{1}{2} \cdot \frac{1}{2} + 1 \cdot \frac{1}{2} = \frac{3}{4}. \tag{4.19}$$

Combing the above probability with the case when F does not occur yields Claim 4.9. ■

The previous game established that $\mathcal{A}$ cannot modify the C_{key} message in the protocol run involving the test-session π_U^i. The next and final game shows that $\mathcal{A}$ also cannot learn anything about π_U^i's session key by merely observing the C_{key} message.

Game 6: This game proceeds as the previous one, but when creating the C_{key} message of the co-partner of π_U^i, the challenger encrypts the all-zero string 0^κ instead of the session key k_{AB}. If this C_{key} message is eventually delivered to the intended responder session (being either π_U^i or π_V^j), then its session key is still set to k_{AB}.

Claim 4.10.

$$\mathbf{Adv}^{\mathsf{G}_5}_{\Pi_3,f_3}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{G}_6}_{\Pi_3,f_3}(\mathcal{A}) + 2 \cdot \mathbf{Adv}^{\mathsf{ACCE}}_{\Pi_2,f_2}(\mathcal{B}_3''). \tag{4.20}$$

Proof. We show that if it is possible to distinguish Game 5 and Game 6, then we can create an algorithm $\mathcal{B}_3''$ that breaks the ACCE security of sub-protocol Π_2. Algorithm $\mathcal{B}_3''$ is almost identical to algorithm $\mathcal{B}_3'$ in the previous proof of Claim 4.9, except for the following differences.

- When creating the C_{key} message of π_U^i's co-partner (if it exists), say π_S^k, algorithm $\mathcal{B}_3''$ makes the query $\mathsf{LR}(\pi_S^k, A\|k_{AB}, A\|0^\kappa)$ instead of the query $\mathsf{LR}(\pi_S^k, A\|k_{AB}, A\|k_{AB})$.

- If $\mathcal{A}$ sends a C_{key} message C' to π^* (with the same definition of π^* as in the description of Game 5), then $\mathcal{B}_3''$ does *not* stop its simulation. Instead $\mathcal{B}_3''$ continues the simulation as follows.

  If C' is equal to the C_{key} message that was previously output by the co-partner of π_U^i using the $\mathsf{LR}(\pi_S^k, A\|k_{AB}, A\|0^\kappa)$ query described above, then π^*'s peer and session key variables are set based on the left message input to the LR query.

  If C' is not equal to the C_{key} message, then C' is automatically rejected in accordance with Game 5.

- Finally, when $\mathcal{A}$ outputs its guess b', then $\mathcal{B}_3''$ outputs the following to its 2P-ACCE game. If the test-session π_U^i has a co-partner π_S^k, then $\mathcal{B}_3''$ outputs $(\pi_S^k, 0)$ if $b' = b_{sim}$ and $(\pi_S^k, 1)$ otherwise. If the test-session does not have a co-partner, then $\mathcal{B}_3''$ 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 $\mathcal{B}_3''$ perfectly simulates it.

If the test-session *has* a co-partner π_S^k, and $\pi_S^k.b = 0$ in $\mathcal{B}_3''$'s 2P-ACCE security game, then $\mathcal{B}_3''$ perfectly simulates Game 5 since the C_{key} message of π_S^k is an encryption of the actual session key k_{AB}. On the other hand, if $\pi_S^k.b = 1$ then $\mathcal{B}_3''$ perfectly simulates Game 6 since the C_{key} message of π_S^k is an encryption of 0^κ.

It remains to argue that whenever $\mathcal{B}_3''$ uses π_S^k as its ACCE target-session, then it is fresh according to predicate $\mathsf{Fresh}_{\mathsf{ACCE}}$ whenever π_U^i is fresh according to predicate $\mathsf{Fresh}_{\mathsf{AKE}^{\mathsf{wfs}}}$. But this follows by the same arguments that was used to show that π^* was ACCE fresh in the proof of Game 5, hence we omit it. ■

Concluding the proof of Theorem 4.2. We argue that $\mathbf{Adv}^{\mathsf{G}_6}_{\Pi_3, f_3}(\mathcal{A}) = 0$. By the change in Game 4, the session key of the test-session π_U^i is derived using a random function $\$(A, B)$, where "$A$" and "$B$" are the identities of the initiator and responder that π_U^i 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_U^i's partner π_V^j (if it exists).

First, note that the random function is evaluated at no more than 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 could potentially be different since $\mathcal{A}$ can modify the identities communicated at the A–S link in Figure 4.1.

However, if $\mathcal{A}$ modifies these identities, then the initiator and server derive independent keys, which means that the initiator and responder will ultimately have independent keys too. Moreover, since the communicated identities at the S–B link in Figure 4.1 will be different, the initiator and responder sessions will not be partners (recall that f_3-partnering includes the sessions' recorded peers, and by Game 5 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 f_1-partner in sub-protocol Π_1 and the server session's C_{key} message, if delivered at all, must be delivered honestly to its f_2-partner in sub-protocol Π_2. Combined with their agreement on their peers, this means the initiator and responder session would be partners by the definition of f_3.

Altogether, since the session key of the test-session is derived using a 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 the bounds from Claim 4.3 to Claim 4.10 we get

$$\begin{aligned}\mathbf{Adv}^{\mathsf{3P\text{-}AKE}^{\mathsf{wfs}}}_{\Pi_3,f_3}(\mathcal{A}) \leq\ & \mathbf{Adv}^{\mathsf{ACCE\text{-}EA}}_{\Pi_2,f_2}(\mathcal{B}_1) + 2n^2 \cdot \mathbf{Adv}^{\mathsf{2P\text{-}AKE}^{\mathsf{fs}}}_{\Pi_1,f_1}(\mathcal{B}_2) + 2n^2 \cdot \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{PRF}}(\mathcal{D}) \\ & + 2n^2 \cdot \mathbf{Adv}^{\mathsf{ACCE}}_{\Pi_2,f_2}(\mathcal{B}_3') + 2n^2 \cdot \mathbf{Adv}^{\mathsf{ACCE}}_{\Pi_2,f_2}(\mathcal{B}_3''),\end{aligned}$$

where $n = (n_\pi + 1) \cdot |\mathcal{I} \cup \mathcal{R}|$.

By letting $\mathcal{B}_3$ be the ACCE adversary that with probability 1/2 either implements algorithm $\mathcal{B}_3'$ or algorithm $\mathcal{B}_3''$, the concrete bound in the statement of Theorem 4.2 follows. ■

Remark 4.11. Note that the conclusion above only holds if protocol Π_3 employs channel binding. If the identities of A and B where not included in the evaluation of the pseudorandom function PRF, then Π_3 would be vulnerable to the simple UKS attack described in Section 4.1. That is, simply change the responder identity being sent over the (unauthenticated) A–S link from "B" to "C". Without channel binding, A and C would obtain the same session key but disagree on their intended peers, and hence not be partners. ▲

4.3 Second composition theorem: 3P-AKE$^{\mathsf{wfs}}$ + 2P-AKE$^{\mathsf{nfs}}$ $\Longrightarrow$ 3P-AKE$^{\mathsf{fs}}$

In this section we state and prove our second composition theorem. This result corresponds to the cryptographic core of full EAP.

Construction. From a 3P-AKE protocol Π_3 and a PSK-based 2P-AKE protocol Π_4, we construct the 3P-AKE protocol Π_5 shown in Figure 4.1. 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 key K_{Π_3}. Then sub-protocol Π_4 is run between A and B using K_{Π_3} as the their PSK. The session key derived in sub-protocol Π_4 becomes A and B's final session key in protocol Π_5.

Result. Our second composition result shows that protocol Π_5 is 3P-AKE secure if sub-protocol Π_3 is 3P-AKE$^{\mathsf{wfs}}$ secure and sub-protocol Π_4 is 2P-AKE$^{\mathsf{nfs}}$ 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. Moreover, while Π_4 does not necessarily achieve any 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 from weak forward secrecy to full forward secrecy.

For technical reasons, we additionally need to assume some minimal structure on sub-protocol Π_4 beyond it being a 2P-AKE protocol. In particular, we require that the probability that two sessions at the same party end up with the same local transcript τ in sub-protocol Π_4 should be "small". Formally, we demand that the probability of such a transcript collision should be statistically bounded by some function ϵ of the number of parties and sessions. This technical requirement is needed in order be able to apply a local partner function to the transcript of sub-protocol Π_4 (see the proof of Claim 4.18).

Theorem 4.12. *Let Π_5 be the 3P-AKE protocol constructed from the 3P-AKE protocol Π_3 and 2P-AKE protocol Π_4 as described in the construction above. Let f_3 and f_4 be partner functions, where f_4 is required to be local. Then, for any adversary $\mathcal{A}$ in security experiment AKE$^{\mathsf{fs}}$ against protocol Π_5, we can create a partner function f_5 and algorithms $\mathcal{B}_1$ and $\mathcal{B}_2$, such that*

$$\mathbf{Adv}^{\mathsf{3P\text{-}AKE}^{\mathsf{fs}}}_{\Pi_5,f_5}(\mathcal{A}) \leq 3n^2 \cdot \mathbf{Adv}^{\mathsf{3P\text{-}AKE}^{\mathsf{wfs}}}_{\Pi_3,f_3}(\mathcal{B}_1) + 2n^2 \cdot \mathbf{Adv}^{\mathsf{2P\text{-}AKE}^{\mathsf{nfs}}}_{\Pi_4,f_4}(\mathcal{B}_2) + 2\epsilon, \quad (4.21)$$

where $n = (n_\pi + 1) \cdot |\mathcal{I} \cup \mathcal{R}|$, n_π is the maximum number of sessions that $\mathcal{A}$ creates at each party, and $\epsilon = \epsilon(|\mathcal{I} \cup \mathcal{R}|, n_\pi)$ is a function that bounds the probability that two sessions at the same party get the same local transcript in protocol Π_4.

The idea of the proof is as follows. Using that sub-protocol Π_3 is 3P-AKE$^{\mathsf{wfs}}$ secure, we can replace the intermediate key K_{Π_3} coming out of Π_3 with a random key for the test-session. This then allows us to reduce the 3P-AKE$^{\mathsf{fs}}$ security of protocol Π_5 to the 2P-AKE$^{\mathsf{nfs}}$ of sub-protocol Π_4, since the now random intermediate key of the test-session is identically distributed with the PSKs used in Π_4. The partner function f_5 will be composed out of f_3 and f_4, so that two sessions are f_5-partners if and only if they are f_3-partners and f_4-partners.

The main difficulty of the proof lies in the first step, i.e., replacing the intermediate key of the test-session with a random key. The issue is that $\mathcal{A}$ plays in a security game that has *full* forward secrecy, whereas the reduction $\mathcal{B}_1$ to sub-protocol Π_3 plays in a security game with only *weak* forward secrecy. As such, $\mathcal{A}$ is allowed strictly more $\mathsf{Corrupt}$ queries than what $\mathcal{B}_1$ can do itself. The question is how $\mathcal{B}_1$ can simulate the 3P-AKE$^{\mathsf{fs}}$ security game for $\mathcal{A}$ while still keeping the test-session fresh in its own 3P-AKE$^{\mathsf{wfs}}$ security game.

This is where we use that sub-protocol Π_4 provides explicit entity authentication. Essentially, it guarantees that the test-session must have a partner in protocol Π_5. By definition of f_5, this implies that it must also have an f_3-partner in sub-protocol Π_3. But recall from Table 3.1 that when the test-session has a partner, then there is no difference between the AKE$^{\mathsf{fs}}$ and AKE$^{\mathsf{wfs}}$ models! Thus, as long as we can show that the test-session has a partner in protocol Π_5, we are fine. Consequently, we first prove as an initial lemma that protocol Π_5 provides explicit entity authentication.

Proof of Theorem 4.12. We begin by defining the partner function f_5 for protocol Π_5. We construct f_5 from the partner functions f_3 and f_4 given for sub-protocols Π_3 and Π_4 as follows:

$$f_{5,T_5}(\pi) = \pi' \iff (f_{3,T_3}(\pi) = \pi') \wedge (f_{4,T_4}(\pi) = \pi'), \tag{4.22}$$

where T_3 and T_4 are the transcripts one gets from T_5 by restricting to the messages pertaining to sub-protocols Π_3 and Π_4, respectively. The soundness of f_5 follows directly from the soundness of f_3 and f_4. Moreover, like in the proof of the first composition theorem (Theorem 4.2), we assume for simplicity that f_3 and f_4 have perfect soundness. It follows that f_5 has perfect soundness too.

4.3.1 Explicit entity authentication

Lemma 4.13. *With f_5 as defined above, and everything else as otherwise stated in Theorem 4.12, we have that*

$$\mathbf{Adv}^{\text{3P-AKE}^{\mathsf{fs}}\text{-}\,\mathsf{EA}}_{\Pi_5,f_5}(\mathcal{A}) \leq 2n^2 \cdot \mathbf{Adv}^{\text{3P-AKE}^{\mathsf{wfs}}}_{\Pi_3,f_3}(\mathcal{B}_1) + n^2 \cdot \mathbf{Adv}^{\text{2P-AKE}^{\mathsf{nfs}}\text{-}\,\mathsf{EA}}_{\Pi_4,f_4}(\mathcal{B}_2) + \epsilon.$$

Proof.

Game 0: This is the original 3P-AKE$^{\mathsf{fs}}$-EA security experiment, hence

$$\mathbf{Adv}^{\mathsf{G}_0\text{-}\,\mathsf{EA}}_{\Pi_5,f_5}(\mathcal{A}) = \mathbf{Adv}^{\mathsf{3P\text{-}AKE}^{\mathsf{fs}}\text{-}\,\mathsf{EA}}_{\Pi_5,f_5}(\mathcal{A}). \tag{4.23}$$

Game 1: In this game the challenger aborts if two sessions at the same party end up with the same local transcript τ in sub-protocol Π_4. By definition of the function ϵ this gives

$$\mathbf{Adv}^{\mathsf{G}_0\text{-}\,\mathsf{EA}}_{\Pi_5,f_5}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{G}_1\text{-}\,\mathsf{EA}}_{\Pi_5,f_5}(\mathcal{A}) + \epsilon. \tag{4.24}$$

Game 2: This game implements a selective security game where the adversary is required to commit to the session that will accept maliciously first. Specifically, at the beginning of the game the adversary must first choose a pair (U, i), with $i \in [1, n_\pi]$. The game then proceeds as in Game 1, except that if some session accepts maliciously before π^i_U, the challenger aborts the game and outputs 0 (i.e., $\mathcal{A}$ loses). In particular, this includes the possibility that $\mathcal{A}$ makes a query that renders π^i_U unfresh (which would preclude π^i_U from accepting maliciously).

Claim 4.14.

$$\mathbf{Adv}^{\mathsf{G}_1\text{-}\,\mathsf{EA}}_{\Pi_5,f_5}(\mathcal{A}) \leq n_\pi \cdot |\mathcal{I} \cup \mathcal{R}| \cdot \mathbf{Adv}^{\mathsf{G}_2\text{-}\,\mathsf{EA}}_{\Pi_5,f_5}(\mathcal{A}'). \tag{4.25}$$

Proof. The proof is essentially the same as for Game 2 in Theorem 4.2 (Claim 4.5), only that this time the selective security adversary guesses one session rather than two. ■

In the remaining games, let π^i_U denote the session that the adversary commits to in Game 2. Note that π^i_U is not necessarily the same as the test-session chosen by the adversary.

Game 3: This game extends the selective security requirement of Game 2 by demanding that the adversary also commits to the partner of π^i_U in sub-protocol Π_3 (if any). Specifically, at the beginning of the game the adversary must pick a pair (U, i) as in Game 2, but it must also pick a pair (V, j), with $j \in [0, n_\pi]$. Game 3 then proceeds as in Game 2, but it additionally aborts if π^i_U gets a different f_3-partner than π^j_V in sub-protocol Π_3. This includes the case that π^i_U *gets* an f_3-partner if $j = 0$.

Remark 4.15. Note that there is no contradiction between π_U^i accepting maliciously in protocol Π_5 according to partner function f_5, while simultaneously having an f_3-partner in sub-protocol Π_3. ▲

Claim 4.16.

$$\mathbf{Adv}^{\mathsf{G}_2\text{-}\,\mathsf{EA}}_{\Pi_5,f_5}(\mathcal{A}) \leq (n_\pi + 1) \cdot \max\{|\mathcal{I}|, |\mathcal{R}|\} \cdot \mathbf{Adv}^{\mathsf{G}_3\text{-}\,\mathsf{EA}}_{\Pi_5,f_5}(\mathcal{A}'). \tag{4.26}$$

Proof. Again, from an adversary $\mathcal{A}$ that plays against the *single* selective security requirement of Game 2, we can create an adversary $\mathcal{A}'$ against the *two* selective security requirements of Game 3. Basically, after $\mathcal{A}$ outputs its commitment to a pair (U, i), then $\mathcal{A}'$ guesses another pair (V, j) (conditioned on the role of U), and outputs (U, i) and (V, j) as its own commitments to Game 3. ■

In the remaining games, let π_V^j denote the (possibly empty) f_3-partner of π_U^i that the adversary commits to in Game 3 in addition to π_U^i.

Game 4: This game proceeds as the previous one, but it replaces the intermediate key K_{Π_3} of π_U^i and π_V^j in sub-protocol Π_3 with a random key $\widetilde{K}$.

Claim 4.17.

$$\mathbf{Adv}^{\mathsf{G}_3\text{-}\,\mathsf{EA}}_{\Pi_5,f_5}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{G}_4\text{-}\,\mathsf{EA}}_{\Pi_5,f_5}(\mathcal{A}) + \mathbf{Adv}^{\mathsf{3P\text{-}AKE}^{\mathsf{wfs}}}_{\Pi_3,f_3}(\mathcal{B}_1). \tag{4.27}$$

Proof. We show that if it is possible to distinguish Game 3 and Game 4, then we can create an algorithm $\mathcal{B}_1$ that breaks the 3P-AKE$^{\mathsf{wfs}}$ security of sub-protocol Π_3. Reduction $\mathcal{B}_1$ begins by choosing a random bit b_{sim}. It then runs $\mathcal{A}$ and implements all the abort conditions introduced so far. All of $\mathcal{A}$'s Send queries that pertain to the 3P-AKE sub-protocol Π_3, $\mathcal{B}_1$ forwards to its 3P-AKE$^{\mathsf{wfs}}$ security game. For all sessions different from π_U^i and π_V^j, $\mathcal{B}_1$ obtains their intermediate keys K_{Π_3} in sub-protocol Π_3 by making a corresponding Reveal query to its 3P-AKE$^{\mathsf{wfs}}$ game.

On the other hand, when the first session out of π_U^i and π_V^j accepts in sub-protocol Π_3, then $\mathcal{B}_1$ instead makes a Test query to obtain its intermediate key K_{Π_3} in protocol Π_5. Let k^* denote this key. When the second session out of π_U^i and π_V^j accepts in sub-protocol Π_3, it is assigned the same key k^* as its intermediate key in sub-protocol Π_3.

$\mathcal{B}_1$ simulates sub-protocol Π_4 itself using the intermediate keys it obtained for sub-protocol Π_3 as the PSKs for Π_4. To answer $\mathcal{A}$'s Test query, $\mathcal{B}_1$ uses the bit b_{sim} it drew in the beginning of the simulation. Finally, when π_U^i accepts in protocol Π_5, then $\mathcal{B}_1$ stops it simulation and outputs a 0 to its 3P-AKE$^{\mathsf{wfs}}$ game if π_U^i accepted maliciously, and a 1 otherwise.

Before analyzing $\mathcal{B}_1$'s advantage, we argue that if π_U^i accepts maliciously in $\mathcal{B}_1$'s simulation, then both π_U^i and π_V^j are valid test-targets in its 3P-AKE$^{\mathsf{wfs}}$

game, i.e., fresh according to predicate $\mathsf{Fresh}_{\mathsf{AKE}^{\mathsf{wfs}}}$. Recall that $\mathcal{B}_1$ selects its test-session based on which of π_U^i and π_V^j accepted first in sub-protocol Π_3. We consider three cases:

- $j = 0$: In this case π_U^i is chosen as the test-session, and $\mathcal{B}_1$ makes no Reveal query towards it in its 3P-AKE$^{\mathsf{wfs}}$ game because it uses the Test query to obtain its intermediate key. Since π_U^i does not have an f_3-partner ($j = 0$), there are of course no other Reveal queries that could have made π_U^i unfresh.

  We claim that $\mathcal{B}_1$ also never issued a $\mathsf{Corrupt}$ query to π_U^i's peers. To see this, note that if π_U^i is to accept maliciously in protocol Π_5, then it must be fresh according to predicate $\mathsf{Fresh}_{\mathsf{AKE}^{\mathsf{fs}}}$. In particular, this means that $\mathcal{A}$ cannot issue any $\mathsf{Corrupt}$ queries to π_U^i's peers *before* π_U^i accepted.[5] But $\mathcal{B}_1$ stops its simulation immediately once π_U^i accepts, so no $\mathsf{Corrupt}$ query will actually be forwarded to π_U^i's peers in $\mathcal{B}_1$'s 3P-AKE$^{\mathsf{wfs}}$ experiment in this case.

- $j \neq 0$ *and* π_U^i *chosen as test-session*: Again, $\mathcal{B}_1$ makes no Reveal query towards π_U^i or π_V^j in its 3P-AKE$^{\mathsf{wfs}}$ game, since they are both handled by the Test query. Moreover, since π_U^i has an f_3-partner ($j \neq 0$), it remains AKE$^{\mathsf{wfs}}$ fresh even if its peers are corrupted.

- $j \neq 0$ *and* π_V^j *chosen as test-session*: By symmetry of the f_3 partner function, π_V^j has π_U^i as its f_3-partner, and thus the argument is the same as for the above case.

Taken together, the above cases show that no-matter which one of π_U^i and π_V^i was selected as the test-session by $\mathcal{B}_1$, it will be fresh according to predicate $\mathsf{Fresh}_{\mathsf{AKE}^{\mathsf{wfs}}}$ in $\mathcal{B}_1$'s 3P-AKE$^{\mathsf{wfs}}$ game if π_U^i accepted maliciously.

Finally, we analyze $\mathcal{B}_1$'s advantage. Let b denote the challenge-bit used to answer $\mathcal{B}_1$'s Test query in its 3P-AKE$^{\mathsf{wfs}}$ game. If $b = 0$, then $\mathcal{B}_1$'s Test query is answered with a real key, and $\mathcal{B}_1$ simulates Game 3 perfectly for $\mathcal{A}$ up until the point when π_U^i accepts (in protocol Π_5). Thus:

$$\Pr[\mathbf{Exp}^{\mathsf{AKE}^{\mathsf{wfs}}}_{\Pi_3,\mathcal{Q}}(\mathcal{B}_1) \Rightarrow 1 \mid b = 0] = \mathbf{Adv}^{\mathsf{G}_3\text{-}\mathsf{EA}}_{\Pi_5,f_5}(\mathcal{A}). \tag{4.28}$$

On the other hand, if $b = 1$, meaning that $\mathcal{B}_1$'s Test query is answered with a random key, then $\mathcal{B}_1$ perfectly simulates Game 4. Since $\mathcal{B}_1$ only outputs a

[5] Recall that predicate $\mathsf{Fresh}_{\mathsf{AKE}^{\mathsf{fs}}}$ forbids any $\mathsf{Corrupt}$ query to a session's peers if (1) it does not have a partner, and (2) it has not accepted yet. This corresponds exactly to the setting we are in when a session accepts maliciously.

1 to its 3P-AKE$^{\mathsf{wfs}}$ security game if $\mathcal{A}$ loses in $\mathcal{B}_1$'s simulation, i.e., if π_U^i does *not* accept maliciously, we have

$$\Pr[\mathbf{Exp}^{\mathsf{AKE}^{\mathsf{wfs}}}_{\Pi_3,\mathcal{Q}}(\mathcal{B}_1) \Rightarrow 1 \mid b = 1] = 1 - \mathbf{Adv}^{\mathsf{G_4\text{-}\,EA}}_{\Pi_5,f_5}(\mathcal{A}). \tag{4.29}$$

Thus, $\mathcal{B}_1$'s advantage is

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{3P\text{-}AKE}^{\mathsf{wfs}}}_{\Pi_3,f_3}(\mathcal{B}_1) &= 2 \cdot \Pr[\mathbf{Exp}^{\mathsf{AKE}^{\mathsf{wfs}}}_{\Pi,\mathcal{Q}}(\mathcal{B}_1) \Rightarrow 1] - 1 && (4.30)\\
&= \Pr[\mathbf{Exp}^{\mathsf{AKE}^{\mathsf{wfs}}}_{\Pi,\mathcal{Q}}(\mathcal{B}_1) \Rightarrow 1 \mid b = 0] \\
&\quad + \Pr[\mathbf{Exp}^{\mathsf{AKE}^{\mathsf{wfs}}}_{\Pi,\mathcal{Q}}(\mathcal{B}_1) \Rightarrow 1 \mid b = 1] - 1 && (4.31)\\
&= \mathbf{Adv}^{\mathsf{G_3\text{-}\,EA}}_{\Pi_5,f_5}(\mathcal{A}) - \mathbf{Adv}^{\mathsf{G_4\text{-}\,EA}}_{\Pi_5,f_5}(\mathcal{A}), && (4.32)
\end{aligned}$$

as stated in the claim. ■

Since the intermediate key K_{Π_3} of π_U^i and π_V^j is replaced with an independent uniformly random key in Game 4, we can finally show that if π_U^i accepts maliciously in Game 4, then it must have accepted maliciously in sub-protocol Π_4.

Claim 4.18.

$$\mathbf{Adv}^{\mathsf{G_4\text{-}\,EA}}_{\Pi_5,f_5}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{2P\text{-}AKE}^{\mathsf{nfs}}\text{-}\,\mathsf{EA}}_{\Pi_4,f_4}(\mathcal{B}_2). \tag{4.33}$$

Proof. If $\mathcal{A}$ wins in Game 4, we can create the following algorithm $\mathcal{B}_2$ which breaks the explicit entity authentication of sub-protocol Π_4. Algorithm $\mathcal{B}_2$ begins by creating all the long-term keys for sub-protocol Π_3 and drawing a random bit b_{sim}. It then runs $\mathcal{A}$ and simulates sub-protocol Π_3 itself using the intermediate keys coming out of sub-protocol Π_3 as the PSKs for sub-protocol Π_4. Additionally, for all sessions different from π_U^i and π_V^j, $\mathcal{B}_2$ also simulates sub-protocol Π_4 itself. On the other hand, to simulate sub-protocol Π_4 for π_U^i and π_V^j, $\mathcal{B}_2$ instead forwards the corresponding queries to its own 2P-AKE$^{\mathsf{nfs}}$ security game. Once $\mathcal{A}$ stops and outputs a guess b', then $\mathcal{B}_2$ stops too.

For sessions different from π_U^i and π_V^j, $\mathcal{B}_2$ simulates Game 4 perfectly since it created all the keys. However, $\mathcal{B}_2$ also perfectly simulates π_U^i and π_V^j, since by the change in Game 4, their intermediate key K_{Π_3} from sub-protocol Π_3 is replaced with an independent uniformly random key $\widetilde{K}$. This is identically distributed to the long-term PSK used in $\mathcal{B}_2$'s 2P-AKE$^{\mathsf{nfs}}$ security game, so by forwarding the Send queries directed to π_U^i and π_V^j to its 2P-AKE$^{\mathsf{nfs}}$ game, $\mathcal{B}_2$ perfectly simulates these sessions too.

It remains to argue that if π_U^i accepts maliciously in Game 4, then it must also have accepted maliciously in $\mathcal{B}_2$'s 2P-AKE$^{\mathsf{nfs}}$ security game. First we claim that session π_V^j cannot be π_U^i's f_4-partner in sub-protocol Π_4.

- If $j = 0$, then $\mathcal{B}_2$ never creates a corresponding proxy session in its 2P-AKE$^{\mathsf{nfs}}$ security game, hence π_U^i cannot possibly have a partner there.

- If $j \neq 0$, then π_V^j by definition (Game 3) must be π_U^i's f_3-partner in sub-protocol Π_3. But if π_V^j was *also* the f_4-partner of π_U^i in sub-protocol Π_4, then by the construction of f_5 from f_3 and f_4, π_V^j would be π_U^i's f_5-partner—contradicting the fact that π_U^i was supposed to accept maliciously.

There is one subtlety with the arguments above: technically we need to show that π_U^i and π_V^i are f_4-partners in Game 4 if and only if they are f_4-partners in $\mathcal{B}_2$'s 3P-AKE$^{\mathsf{nfs}}$-EA security game. However, the T_4 transcript from Game 4 contains *many* sessions, while the transcript $T_{\mathcal{B}_2}$ from $\mathcal{B}_2$'s 2P-AKE$^{\mathsf{nfs}}$ security game contains at most two: π_U^i and π_V^j. In particular, transcript $T_{\mathcal{B}_2}$ is the restriction $T_4\big|_{\pi_U^i,\pi_V^i}$ of T_4. But evaluating the same partner function on these two transcripts does not necessarily have to yield the same answer. This is where we use the assumption that f_4 is a local partner function (see Definition 3.5). Namely, since f_4 is local, we have that π_U^i and π_V^i are f_4-partners based on T_4 if and only if they are f_4-partners based on the restriction $T_4\big|_{\pi_U^i,\pi_V^i}$—provided that T_4 is a unique transcript, i.e., no two sessions at the same party have the same local transcript τ. But this is exactly what the abort condition in Game 1 ensures.

Having shown that π_U^i does not accept with an f_4-partner in $\mathcal{B}_2$'s 2P-AKE$^{\mathsf{nfs}}$-EA security game, we only have to show that π_U^i is fresh according to predicate $\mathsf{Fresh}_{\mathsf{AKE}^{\mathsf{nfs}}}$. But this is true since $\mathcal{B}_2$ makes no $\mathsf{Corrupt}$ query at all in its 2P-AKE$^{\mathsf{nfs}}$ security game, and also makes no Reveal query to π_U^i. Thus π_U^i accepts maliciously in $\mathcal{B}_2$'s 2P-AKE$^{\mathsf{nfs}}$ security game whenever it accepts maliciously in Game 4, proving Claim 4.18. ■

Combining the bounds from Game 1 to Game 4 with Claim 4.18 yields Lemma 4.13. ■

4.3.2 AKE$^{\mathsf{fs}}$ security

Given Lemma 4.13, which shows that Π_5 provides explicit entity authentication, we can now proceed with the proof of Theorem 4.12.

Game 0: This is the real 3P-AKE$^{\mathsf{fs}}$ security game, hence

$$\mathbf{Adv}^{\mathsf{G}_0}_{\Pi_5,f_5}(\mathcal{A}) = \mathbf{Adv}^{\mathsf{3P\text{-}AKE}^{\mathsf{fs}}}_{\Pi_5,f_5}(\mathcal{A}).$$

Game 1: In this game, the challenger aborts if a session accepts maliciously in protocol Π_5, whence

$$\mathbf{Adv}^{\mathsf{G}_0}_{\Pi_5,f_5}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{G}_1}_{\Pi_5,f_5}(\mathcal{A}) + \mathbf{Adv}^{\mathsf{3P\text{-}AKE^{fs}\text{-}EA}}_{\Pi_5,f_5}(\mathcal{A}). \tag{4.34}$$

The remaining game hops are essentially the same as those of Lemma 4.13. Hence, we merely state their descriptions and corresponding bounds, but omit the proofs.

Game 2: In this game the challenger aborts if two sessions at the same party end up with the same local transcript τ in sub-protocol Π_4. By definition of the function ϵ this gives

$$\mathbf{Adv}^{\mathsf{G}_1}_{\Pi_5,f_5}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{G}_2}_{\Pi_5,f_5}(\mathcal{A}) + \epsilon. \tag{4.35}$$

Game 3: This game implements a selective security game where the adversary has to commit to a test-session and its partner (required to exist by Game 1). Specifically, at the beginning of the game the adversary must output two pairs (U,i) and (V,j). The game then proceeds as in Game 2, except that if either of the following events occur, then the challenger penalizes the adversary by outputting a random bit at the end.

(i) Neither π^i_U nor π^j_V were selected as the test-session by $\mathcal{A}$.

(ii) π^i_U and π^j_V did not get partnered to each other.

Claim 4.19.

$$\mathbf{Adv}^{\mathsf{G}_2}_{\Pi_5,f_5}(\mathcal{A}) \leq (n_\pi^2 \cdot |\mathcal{I}| \cdot |\mathcal{R}|)/2 \cdot \mathbf{Adv}^{\mathsf{G}_3}_{\Pi_5,f_5}(\mathcal{A}'). \tag{4.36}$$

In the remaining games, let π^i_U and π^j_V denote the two sessions the adversary commits to in the selective security game.

Game 4: This game proceeds as the previous one, but it replaces the intermediate key K_{Π_3} of π^i_U and π^j_V in sub-protocol Π_3 with a random key $\widetilde{K}$.

Claim 4.20.

$$\mathbf{Adv}^{\mathsf{G}_3}_{\Pi_5,f_5}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{G}_4}_{\Pi_5,f_5}(\mathcal{A}) + 2 \cdot \mathbf{Adv}^{\mathsf{3P\text{-}AKE^{wfs}}}_{\Pi_3,f_3}(\mathcal{B}_1). \tag{4.37}$$

Finally, any successful attack on protocol Π_5 in Game 4 can be transformed into a successful attack on sub-protocol Π_4.

Claim 4.21.

$$\mathbf{Adv}^{\mathsf{G}_4}_{\Pi_5,f_5}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{2P\text{-}AKE^{nfs}}}_{\Pi_4,f_4}(\mathcal{B}_2). \tag{4.38}$$

Concluding the proof of Theorem 4.12. Combining the bounds from Section 4.3.2 to Game 4 with Claim 4.21, we get

$$\begin{aligned}\mathbf{Adv}^{\mathsf{3P\text{-}AKE}^{\mathsf{fs}}}_{\Pi_5,f_5}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{3P\text{-}AKE}^{\mathsf{fs}}\text{-}\mathsf{EA}}_{\Pi_5,f_5}(\mathcal{A}) + 2n^2 \cdot \mathbf{Adv}^{\mathsf{3P\text{-}AKE}^{\mathsf{wfs}}}_{\Pi_3,f_3}(\mathcal{B}_1) \\ + n^2 \cdot \mathbf{Adv}^{\mathsf{2P\text{-}AKE}^{\mathsf{nfs}}}_{\Pi_4,f_4}(\mathcal{B}_2) + \epsilon.\end{aligned} \tag{4.39}$$

The concrete bound of Theorem 4.12 now follows by applying Lemma 4.13 to Equation (4.39). ■

4.4 Application to EAP

Our two composition theorems (Theorem 4.2 and Theorem 4.12) apply to the basic and full variants of EAP respectively, as defined in Section 4.1. Specifically, in Theorem 4.2 we identify sub-protocol Π_1 with the EAP method run between the client and the server, and sub-protocol Π_2 with the key-transport protocol between the server and the authenticator. By a suitable instantiation of these building blocks, and assuming that the EAP method provides channel binding, we immediately get from Theorem 4.2 that basic EAP (Π_3) is a secure 3P-AKE protocol in the weak forward secrecy model $\text{3P-AKE}^{\mathsf{wfs}}$. This result can then be combined with Theorem 4.12, where we identify sub-protocol Π_3 with basic EAP and sub-protocol Π_4 with the subsequent Security Association Protocol between the client and the authenticator. It follows immediately that full EAP (Π_5) is a secure 3P-AKE protocol in the full forward secrecy model AKE^{fs}.

In Chapter 5 we will show that the EAP method EAP-TLS [RFC5216] satisfies the requirements on sub-protocol Π_1 in Theorem 4.2. Likewise, in Chapter 6 we show that IEEE 802.11 [IEEE 802.11] satisfies the requirements on sub-protocol Π_4 in Theorem 4.12. Thus, it remains to demonstrate that the key-transport protocol between the server and the authenticator satisfies the requirements on sub-protocol Π_2 in Theorem 4.2.

As we argued in Section 4.1.2, the security properties provided by the MPPE encryption method employed by RADIUS are largely unknown. It is therefore difficult to assess whether RADIUS alone can safely instantiate our first composition theorem. On the other hand, RADIUS is commonly run on top of a secure channel protocol like TLS or IPsec. In this case the security reduces to that of the underlying secure channel protocol. Both TLS and IPsec are well-studied, and have received large amounts of formal analysis. In particular, a number of works have shown TLS to be a secure ACCE protocol [Jag+12, KPW13b, KSS13, Brz+13a, Li+14], so in Theorem 4.2 we can for instance set sub-protocol Π_2 to be RADIUS-over-TLS [RFC6614].

4.4.1 EAP without channel binding

Theorem 4.2 requires that the EAP method provides channel binding. Without it, EAP becomes vulnerable to the UKS attack described in Section 4.1.1. Unfortunately, many concrete EAP methods do not provide channel binding. Because the communication between the client and the server is normally routed through the authenticator, a malicious authenticator can trivially modify the information presented to the two sides. As a consequence, without channel binding it suffices to compromise a *single* authenticator in order to compromise an entire network. Since authenticators are typically low-protected devices, such as wireless access points, this represents a substantial attack vector on larger enterprise networks.

Interestingly, a very similar situation can be found in the UMTS and LTE mobile networks. UMTS and LTE employ a key exchange protocol called AKA which is structured almost identically to the EAP protocol: a mobile client that wants to connect to a base station first has to authenticate to its home operator. The home operator then transmits a list of so-called *authentication vectors* (which in particular includes a session key) to the base station in much the same way the server forwards the session key to the authenticator in EAP. Moreover, similar to many EAP methods, the AKA protocol lacks channel binding for its authentication vectors. In their analysis of the AKA protocol, Alt et al. [Alt+16, §5] noted this lack of channel binding, and suggested a fix which is almost identical to the key-derivation approach analyzed in this chapter.

4.4.2 Channel binding scope

Theorem 4.2 assumes that the channel binding includes the identity of the client and the authenticator in order to bind them cryptographically to the session key. However, this fine-grained scope of the channel binding might not be relevant all situations. For example, in a WLAN supported by many access points, the client does not actually 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 does not have to 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). Of course, by doing so the protection provided by the channel binding will be weaker. In particular, 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 WLAN. In this case, 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, can all be used as input to the channel binding (see [CH09] for a discussion of these possibilities). The specifications for channel binding within EAP [OPY06, RFC6677] leave open exactly what type of information should go into the binding, because the amount of information that will be available to both the client and the server may vary.

Chapter 5

Security of EAP-TLS

Contents

5.1 Motivation

In Chapter 4 we showed that EAP is a secure 3P-AKE protocol assuming, among other things, that the EAP method between the client and the server is a secure 2P-AKE protocol. Thus, in order to complete the picture on EAP, we need to establish that at least *some* EAP method satisfies the 2P-AKE security notion. Fortunately, such an EAP method already exists, namely EAP-IKEv2 [RFC5106]. In particular, since an EAP method is just a wrapper around some concrete AKE protocol, the security of EAP-IKEv2 reduces to that of IKEv2, which has been proven secure by Canetti and Krawczyk [CK02].

On the other hand, probably the most widely supported EAP method of all—EAP-TLS—has no such proof. In fact, as we explained in Section 3.4, TLS

in all versions up to TLS 1.2 is not a secure AKE protocol at all! The reason is that TLS encrypts some of the handshake messages using the session key itself, giving the adversary a trivial way of distinguishing the session key from random. Thus, it might appear that our results on EAP cannot be applied to the case when EAP-TLS is being used as the EAP method.

Fortunately, it turns out that within the context of EAP-TLS, TLS *can* be proven to be a secure AKE protocol. Recall from Section 2.1 that an EAP method is supposed to *export* a master session key MSK. Crucially, in EAP-TLS the MSK is *not* the ordinary session key of TLS which is used to protect the channel. Instead, the MSK is derived as a separate key from the master secret established during the TLS handshake. This fact makes it possible to prove that EAP-TLS is a secure AKE protocol, by considering the MSK as the session key.

Concretely, in this chapter we show that if one derives an additional *export key* from the TLS master secret—independent of the other handshake messages—then TLS constitutes a secure AKE protocol by taking this export key to be the session key. Furthermore, while our starting point is the TLS protocol, our result will in fact be much more general. Instead of focusing solely on TLS, we generalize to a wider class of protocols which we call *TLS-like ACCE protocols*. Roughly, TLS-like ACCE protocols are protocols that satisfy the ACCE security notion and, like TLS, establish a master secret during the handshake. Apart from this requirement, our result has no other dependencies on the specifics of the TLS protocol. In other words, our main result is a general theorem showing that any ACCE protocol which has a concept of a master secret can be turned into an AKE protocol.

An immediate corollary of this result is of course that EAP-TLS is a secure 2P-AKE protocol. However, it also applies more broadly to the general practice of exporting additional keys from the master secret in TLS, as has been formalized in RFC 5705: "Keying Material Exporters for Transport Layer Security (TLS)" [RFC5705] (which we call *TLS Key Exporters* from now on).

Motivation for our approach. For the moment, suppose we only wanted to show that EAP-TLS was a secure AKE protocol, leaving aside the possibility of further generalizations for now. One obvious approach would be to reuse one of the many existing security proofs which shows that TLS is a secure ACCE protocol (e.g., [Jag+12, KSS13, Li+14]). Specifically, in these proofs the master secret of a particular session is typically swapped out with a completely random value, allowing the rest of the proof to continue on the assumption that the master secret is completely hidden from the adversary. Due to the unpredictability of the master secret, the adversary will not be able to detect the switch. Using this truly random master secret, we could then extend the

proof with one additional step where we derive the export key using a random oracle. It would then follow that the derived export key is indistinguishable from random.

However, a big downside of such a result is that it could not be re-used across different TLS ciphersuites, nor would it hold for future versions of TLS. Indeed, for every variant of TLS one would have to redo the corresponding security proof and augment it accordingly to account for the extra export key. Besides being tedious, this approach is of course also inherently non-modular since it is tied to the innards of each particular proof. Still, it seems likely that most of these proofs would be fairly similar in terms of technique, and also reasonably independent of the specific details of the TLS protocol itself.

The question is whether we can isolate exactly those properties of the TLS protocol that these proofs rely on. If so, we could extract a generic proof of TLS key exporters that works across different versions unmodified. Moreover, it would be even better if we could have a result that is not tied to TLS at all, but rather one that targets an appropriate abstract security notion.

Essentially, this is what we do in this chapter: we identify certain features of the TLS protocol which, when added to a generic ACCE protocol, are sufficient to establish the indistinguishability of the export keys derived by the protocol. Note that, apart from the features that we identify, the result is completely independent of the internals of TLS. Below we describe these features.

Technical overview of our result. Surprisingly, the number of additional features that needs to be added beyond a generic ACCE protocol is rather minimal. They consist of the following three requirements.

(i) The handshake includes a random *nonce* from each session.

(ii) Each session maintains a value called the *master secret* during the handshake.

(iii) The session key is derived from the master secret, the nonces, and possibly some other public information using a key derivation function (KDF).

We call an ACCE protocol that satisfies these requirements *TLS-like*. Our result can now be more precisely formulated as follows: starting from an ACCE secure TLS-like protocol Π, we create an AKE secure protocol Π^+, where Π^+ consists of running protocol Π until a session accepts (according to Π), and then derives one additional key from the master secret and nonces of Π. This key—which is distinct from the session key in the underlying protocol Π—becomes the session key of Π^+. In our security proof the key derivation step will be modeled using a random oracle. The construction of Π^+ from Π precisely captures the definition used in TLS key exporters [RFC5705] and EAP-TLS [RFC5216].

Note that while we put no security requirements on the master secret of a TLS-like protocol, it is pivotal in our proof to relate the indistinguishability of the session keys in Π^+ to the ACCE security of Π. However, at first sight it does not seem like merely assuming the ACCE security of TLS will allow us to say anything about the *internal* variables of TLS, and in particular its master secret. Nevertheless, inspired by Morrissey, Smart, and Warinschi [MSW10], we can show that the ACCE security of TLS implies that the master secret is *unpredictable*, meaning that no adversary is able to output the full master secret of a fresh target session. If the master secret was predictable, then we would be able to break the security of the ACCE channel. This intuition lies at the heart of our proof, which uses the ACCE property of TLS in a (semi-)black-box way.

Specifically, Morrissey et al. [MSW10] proved that a secure AKE protocol can be built out of a secure *master key agreement protocol*, which has the much weaker security requirement of having unpredictable master secrets. In their security reduction they assumed to have access to a *key-checking oracle* $\mathcal{O}$ that answers whether a supplied value equals the master secret of a given session. Using the key-checking oracle $\mathcal{O}$, they could simulate the session key derivation of the AKE protocol as well the random oracle. Crucially, however, it required that $\mathcal{O}$ was *perfect*, meaning that it always answered correctly.

By contrast, our proof is complicated by the fact that there is no perfect key-checking oracle available. That is, given only a (TLS-like) ACCE protocol, there is no apparent mechanism for testing the master secret of a session with certainty. The main technical novelty of our proof is to show that we can still create an approximation of the key-checking oracle as long as we allow a (small) one-sided error probability. This emulated key-checking oracle suffices to simulate the AKE experiment of protocol Π^+ in our reduction to the ACCE security of Π.

To give some intuition for our key-checking oracle in the ACCE setting, suppose we want to test whether the value ms is the master secret of some session π. First, we use ms, the nonces π accepted with, and the KDF of Π (all available since Π is TLS-like) to derive a *guess* on π's session key *in* Π. Next, we obtain a ciphertext C of a random message under π's *actual* session key in Π using our access to the "left-or-right" LR query in the ACCE game. Finally, we *locally* decrypt C using the guessed session key of Π, i.e., we do not use the $\mathsf{Decrypt}$ query of the ACCE game. If the local decryption gives back the random message we started with, we guess that ms was the correct master secret of π; otherwise, we guess that it was incorrect.

In the above, we tacitly assumed that different master secrets derive different session keys (using the same nonces). Normally, this would follow directly from the pseudorandomness of the KDF used in Π. However, since we do not require the master secrets to be independent and uniformly distributed,

we cannot invoke this property of the KDF. Instead, we have to explicitly assume that different master secrets do not collide to the same session key. We expect this property to hold for most real-world KDFs. Particularly, we show in Theorem 5.14 (Section 5.3.2) that the HMAC-based KDF used in TLS 1.2 has this property, provided the underlying hash function in HMAC is collision-resistant.

5.1.1 Related work on EAP-TLS

The classic result of Canetti and Krawczyk [CK01] shows how to build secure channels from AKEs. Our result can be seen as a kind of dual: building AKEs from secure channels. Specifically, we create a *compiler* that on input a secure TLS-like ACCE protocol outputs a secure AKE protocol. There is a long tradition for generic compiler results like this in the literature [BCK98, KY03, Jag+10, BG11, CF12, Kra16].

On the specific topic of EAP-TLS we are not aware of any existing results. There are results on other EAP methods, however. For example, the already mentioned EAP-IKEv2 method is a secure AKE protocol (following directly from the corresponding result on IKEv2 [CK02]). Likewise, Küsters and Tuengerthal [KT11a] have shown that the EAP-PSK method (see Table 2.1) is secure in their IITM universal composability framework.

5.2 TLS-like ACCE $\Longrightarrow$ AKE

In this section we state and prove our generic result which will be used to establish the AKE security of EAP-TLS in Section 5.3. The protocols analyzed in this section are generic in the sense that they are not assumed to have any specific structure except for being TLS-like.

5.2.1 TLS-like protocols

Since the definition of TLS-like is motived by the structure of the TLS 1.2 protocol, we first give a brief description of it here. Figure 5.1 shows a simplified version of the TLS 1.2 handshake parameterized on a key encapsulation mechanism (KEM) KEM. This presentation is inspired by Krawczyk et al. [KPW13b] and allows us to treat all the main TLS handshake variants, TLS-RSA, TLS-DH, TLS-DHE, in a uniform manner by suitably instantiating the KEM. Note that in order to do so, the "Cert^+" notation captures more than just one side's certificate. For example, for TLS-DHE we have that Cert^+_C includes the client's ephemeral Diffie-Hellman share, a signature on the share, as well as the client certificate itself.

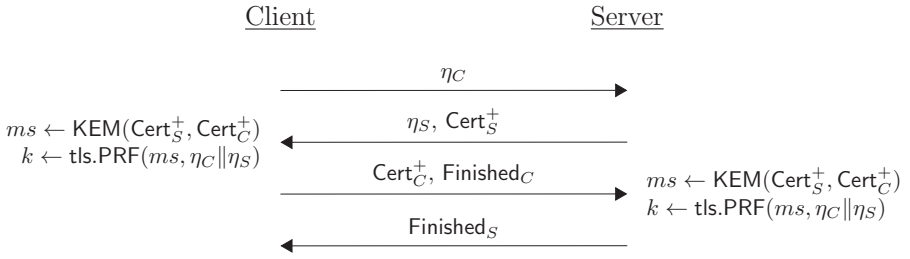


Figure 5.1: Simplified TLS 1.2 handshake.

The TLS handshake begins with the client sending a random nonce η_C. The server responds with its own random nonce η_S and its contribution to the KEM, denoted Cert_S^+. On receiving this message, the client generates its own KEM contribution Cert_C^+, and from the two KEM values derives a *master secret* key ms. From the master secret and the nonces, the client also derives the TLS session key k using the TLS key derivation function $\mathsf{tls.PRF}$. The client sends its KEM contribution together with a key confirmation message $\mathsf{Finished}_C$ to the server. On receiving the client's KEM contribution, the server derives the same ms and k as the client and checks the validity of $\mathsf{Finished}_C$. It ends the handshake by sending a key confirmation message $\mathsf{Finished}_S$ of its own.

Essentially, a TLS-like protocol abstracts from the TLS 1.2 handshake the idea of having random nonces, a master secret, and a session key derived from the master secret and the nonces using a KDF. In the definition below, recall that a session's local transcript τ consists of all the messages it has sent and received during the protocol run.

Definition 5.1 (TLS-like protocols). An ACCE protocol Π is *TLS-like* if:

(i) each session transmits a random *nonce value* $\eta \twoheadleftarrow \{0,1\}^\lambda$ during its run of the protocol,

(ii) each session holds a variable $ms \in \{0,1\}^\kappa \cup \{\perp\}$, called the *master secret*,

(iii) if η_1, η_2 are the two nonces on a session's local transcript τ, then the *session key* is derived as

$$k \leftarrow \mathsf{Kdf}(ms, \eta_1 \| \eta_2, F_\Pi(\tau)), \tag{5.1}$$

where $\mathsf{Kdf}\colon \{0,1\}^\kappa \times \{0,1\}^{2\lambda} \times \{0,1\}^* \to \{0,1\}^\kappa$ and $F_\Pi\colon \{0,1\}^* \to \{0,1\}^*$ are deterministic functions.

It should be clear from Figure 5.1 that TLS 1.2 is indeed TLS-like. But many other real-world protocols also belong to this class, like SSH, IKEv2, and

QUIC. The function F_Π is protocol-specific and meant to capture any auxiliary input that might be used to derive the session keys in addition to the nonces. For example in TLS 1.2, $F_\Pi(\tau)$ is the empty string, while in IKEv2, $F_\Pi(\tau)$ is the Security Parameter Index (SPI) of the initiator and the responder.

5.2.2 Construction

Let Π be a TLS-like ACCE protocol with key derivation function Kdf and let $G\colon \{0,1\}^\kappa \times \{0,1\}^{2\lambda} \times \{0,1\}^* \to \{0,1\}^\kappa$ be a random oracle. From Π and G we create an AKE protocol Π^+ as follows. Protocol Π^+ consists of first running protocol Π as usual until a session accepts, then it derives an additional key $ek \leftarrow G(ms, \eta, aux)$, where ms is the master secret of Π, η_C and η_S are the nonces, and $aux \in \{0,1\}^*$ is an (optional) string containing selected values from the session's local transcript τ. The key ek becomes the *session key* in protocol Π^+.

By construction, a session in Π^+ derives (at least) two keys: its "true" session key in the sense of the AKE model, i.e., the key ek derived from the random oracle G; and the "session key" derived in the underlying protocol Π using the KDF $\Pi.\mathsf{Kdf}$. To avoid confusion, we will call the former key the *export key*; and the latter key the *channel key* and denote it ck. In particular, in the AKE game the session key variable $\pi.k$ will store the export key ek, while the channel key ck will merely be one of π's other internal state variables, written $\pi.ck$. The reason why the export key ek needs to be derived using a random oracle will be explained below.

5.2.3 Main result

Informally, our main result shows that the construction described above transforms a secure TLS-like ACCE protocol Π into a secure AKE protocol Π^+. However, in our proof we need to make one additional assumptions besides that of ACCE security. We need to assume that the key derivation function $\Pi.\mathsf{Kdf}$ does not have *key collisions*, i.e., that two different master secrets produce the same output when given the same nonces and auxiliary data as input.

Definition 5.2 (KDF collision resistance). Let KDF be a function with the same domain and range as the function in Equation (5.1). Define the following advantage measure for an adversary $\mathcal{A}$:

$$\mathbf{Adv}^{\mathsf{kdfcoll}}_{\mathsf{KDF}}(\mathcal{A}) \stackrel{\text{def}}{=} \Pr\left[((ms, ms'), r, s) \leftarrow \mathcal{A} : \begin{array}{c}\mathsf{KDF}(ms, r, s) = \mathsf{KDF}(ms', r, s) \\ ms \neq ms'\end{array}\right].$$

A tuple $((ms, ms'), r, s)$ satisfying the criterion in the equation above is called a *key collision* for KDF.

Remark 5.3. Definition 5.2 is a variant of the more common notion of collision-resistant *hash functions*. The difference is that KDF collision resistance is about collisions in the *keys*, not the messages. ▲

Theorem 5.4. *Let Π^+ be the AKE protocol derived from a TLS-like ACCE protocol Π and a random oracle G using the construction described in Section 5.2.2. Let f be a partner function with perfect soundness. Then for any adversary $\mathcal{A}$ in the AKE security experiment against Π^+, we can create adversaries $\mathcal{B}$ and $\mathcal{C}$ such that:*

$$\mathbf{Adv}^{\mathsf{AKE}^{\mathsf{fs}}}_{\Pi^+,f}(\mathcal{A}) \leq 3 \cdot \mathbf{Adv}^{\mathsf{ACCE}}_{\Pi,f}(\mathcal{B}) + 3 \cdot \mathbf{Adv}^{\mathsf{KDFcoll}}_{\Pi.\mathsf{Kdf}}(\mathcal{C}) + \frac{6qn_Pn_\pi}{2^c} + \frac{(n_Pn_\pi)^2}{2^{\lambda+1}}, \quad (5.2)$$

where λ is the nonce length of protocol Π, n_P is the number of parties, n_π is the max number of sessions that $\mathcal{A}$ creates at each party, q is $\mathcal{A}$'s number of random oracle queries, and $c \in \mathbf{N}$ is an arbitrary constant.

The main idea behind the proof of Theorem 5.4 is to relate the security of the derived export keys to the security of the channel keys in the underlying ACCE protocol Π. Basically, by using the property that TLS-like protocols derive their channel keys from the master secret and nonces, we establish that two sessions derive the same export key if and only if they derive the same channel key (barring certain bad events which we bound). The reason why the export key needs to be derived using a random oracle is because the master secret is not guaranteed to be uniform and independently distributed. Because of this we cannot invoke the pseudorandomness of the KDF when deriving the export key from the master secret.

Like in the proofs of our compositions theorems in Chapter 4, we assume for simplicity that the partner function f has perfect soundness so that we can always take for granted that partners derive the same session key.

Proof. Let $\mathcal{A}$ be the adversary in the $\mathrm{AKE}^{\mathsf{fs}}$ security game against protocol Π^+. Our proof proceeds through a sequence of games, where each consecutive game aims to decrease the challenger's dependency on the sessions' master secrets and the random oracle G, in order to derive the export keys in protocol Π^+. Eventually, in the final game the random oracle G will have been completely replaced by a local list L_G, and the Π^+ export keys are derived independently of the sessions' master secrets. At this point we can construct an algorithm $\mathcal{B}$ against the ACCE security of the underlying protocol Π, since $\mathcal{B}$ will now be able to simulate the game.

Game 0: This is the original $\mathrm{AKE}^{\mathsf{fs}}$ security game for protocol Π^+:

$$\mathbf{Adv}^{\mathsf{AKE}^{\mathsf{fs}}}_{\Pi^+,f}(\mathcal{A}) = \mathbf{Adv}^{\mathsf{G}_0}_{\Pi^+}(\mathcal{A}). \quad (5.3)$$

Game 1: Game 1 proceeds like in Game 0, but aborts if two sessions generate the same nonce value. Since there are $n_P \cdot n_\pi$ generated nonces, the probability of there being at least one collision is bounded by $(n_P n_\pi)^2 \cdot 2^{-(\lambda+1)}$. By the Difference Lemma we have

$$\mathbf{Adv}^{\mathsf{G}_0}_{\Pi^+,f}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{G}_1}_{\Pi^+,f}(\mathcal{A}) + \frac{(n_P n_\pi)^2}{2^{\lambda+1}}. \tag{5.4}$$

The remaining games are aimed at removing the challenger's dependency on the random oracle and enabling it to derive the Π^+ export keys without knowing the sessions' master secrets. To this end, the challenger will begin to maintain a list L_G which it will use to simulate the random oracle G and derive the sessions' export keys. The entries of L_G are tuples of the form $(ms, \eta, aux, ek, [*])$, where $ms \in \{0,1\}^\kappa \cup \{\perp\}$, $\eta \in \{0,1\}^{2\lambda}$, $aux \in \{0,1\}^*$, $ek \in \{0,1\}^\kappa$, and $[*]$ denotes a list that contains zero or more session oracles. Specifically, we use the notation "$[\,]$" to denote an empty list, "$[\pi]$" for a list containing exactly π, "$[\pi, *]$" for a list containing π plus zero or more (unspecified) sessions, and "$[*]$" for a list containing zero or more (unspecified) sessions. L_G is initially empty and is filled out either in response to $\mathcal{A}$'s random oracle queries or when a session reaches the accepted state.

All the remaining games either change the way export keys are derived for newly accepted sessions (which we call the "Send-code"), or how they answer random oracle calls (which we call the "G-code"). The evolution of the Send-code from Game 2 through Game 6 is shown in Figure 5.2 on Page 102, while the corresponding G-code is shown in Figure 5.3 on Page 103. Here is how to read the Send-code. When a session π accepts with master secret ms, nonces $\eta = \eta_C \| \eta_S$, and auxiliary data aux, then we look for the existence of a tuple $t \in L_G$ that matches these variables. We use red color to indicate the variables that a given if/else-if clause uses to "pattern-match" against the corresponding variables of π. The G-code is read in a similar way.

We annotate the changes made in one game relative to the previous one using boxes. Note that some games make changes to both the Send-code and the G-code at the same time For the remainder of this proof we always use η to refer to the concatenation of the two nonces η_C, η_S that a session has received.

Game 2: This game introduces the list L_G. When a session π accepts with master secret ms, nonces $\eta = \eta_C \| \eta_{AP}$, and auxiliary data aux, the challenger uses the Send-code shown in the panel labeled "Game 2" in Figure 5.2 to derive its export key. It uses the G-code shown in the panel labeled "Game 2" in Figure 5.3 to answer the adversary's random oracle queries.

Claim 5.5.

$$\mathbf{Adv}^{\mathsf{G}_1}_{\Pi^+,f}(\mathcal{A}) = \mathbf{Adv}^{\mathsf{G}_2}_{\Pi^+,f}(\mathcal{A}). \tag{5.5}$$

Proof. Since the challenger considers all of the input values to the random oracle when answering from L_G in this game—in particular, it explicitly looks at the master secrets of the sessions—and because a random oracle always returns the same value when given the same input twice, the answers in Game 2 are distributed exactly like in Game 1. ■

In the remaining games, we define $ck\text{-}\mathsf{coll}_\mathsf{i}$ to be the event that during the run of Game i, the challenger calls the key derivation function $\Pi.\mathsf{Kdf}$ on two different master secrets $ms \neq ms'$, but with the same nonces η and additional input aux, such that $\Pi.\mathsf{Kdf}(ms, \eta, aux) = \Pi.\mathsf{Kdf}(ms', \eta, aux)$. We call event $ck\text{-}\mathsf{coll}_\mathsf{i}$ a *channel key collision*.

Game 3: In this game the Send-code is modified so that when a session accepts, the challenger first checks whether the session's partner is present in a tuple on L_G before deriving its export key (see the panel labeled "Game 3" in Figure 5.2). The G-code remains unchanged.

Claim 5.6.

$$\mathbf{Adv}^{\mathsf{G}_2}_{\Pi^+,f}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{G}_3}_{\Pi^+,f}(\mathcal{A}) + \Pr[ck\text{-}\mathsf{coll}_3]. \tag{5.6}$$

Proof. We claim that unless a channel key collision occurs, then Game 2 and Game 3 are identical. To see this, suppose the if-check at Line 2 of Game 3 matched two sessions π and π'. This means that $f_T(\pi) = \pi'$, which implies that they have the same channel key by our assumption of perfect soundness of f. Then our assumption that no key collision occurs further implies that they must also have the same master secret. Hence, the else-if check at Line 10 would also have matched π and π' in Game 2. This shows that Game 2 and Game 3 matches exactly the same sessions when no channel key collision occurs.

To bound $\Pr[ck\text{-}\mathsf{coll}_3]$ we create an algorithm $\mathcal{C}_1$ that finds key collisions in $\Pi.\mathsf{Kdf}$. Algorithm $\mathcal{C}_1$ emulates adversary $\mathcal{A}$ and the challenger in an execution of Game 3 by instantiating all the parties' long-term keys and running all the sessions according to the specification of the game. If event $ck\text{-}\mathsf{coll}_3$ happened during this run, say due to the calls $\Pi.\mathsf{Kdf}(ms, \eta, F_\Pi(\tau))$ and $\Pi.\mathsf{Kdf}(ms', \eta, F_\Pi(\tau))$, then algorithm $\mathcal{C}_1$ outputs $((ms, ms'), \eta, F_\Pi(\tau))$ as its collision for $\Pi.\mathsf{Kdf}$.

Since $\mathcal{C}_1$ holds all the keys, it can simulate Game 3 perfectly. In particular, it can correctly simulate the random oracle G in those places where it is called inside of Game 3 (i.e., Line 15 of the Send-code, and Line 11 of the G-code). Thus, the probability that $\mathcal{C}_1$ finds a collision in $\Pi.\mathsf{Kdf}$ is exactly the probability that event $ck\text{-}\mathsf{coll}_3$ occurs during its simulation of Game 3 for $\mathcal{A}$. ■

Remark 5.7. The reason we have to condition on there being no channel key collision in Game 3 is because we do not assume that being partners necessarily implies having equal master secrets. It is conceivable that two partner sessions might end up with the same channel key (and export key) even if their master secrets differ. This would lead to a discrepancy in how G queries are answered in Game 2 and Game 3. ▲

Game 4: In this game the Send-code is augmented by matching non-fresh sessions based on their channel keys (see Figure 5.2). That is, if two non-fresh sessions are found to have the same channel key (and the same nonces and auxiliary data), then they are given the same export key too.

Claim 5.8.

$$\mathbf{Adv}^{\mathsf{G}_3}_{\Pi^+,f}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{G}_4}_{\Pi^+,f}(\mathcal{A}) + \Pr[ck\text{-}\mathsf{coll}_4]. \tag{5.7}$$

Proof. Again, as long as a channel key collision does not occur (event ck-coll_4), then Game 3 and Game 4 are identical. To bound $\Pr[ck\text{-}\mathsf{coll}_4]$ we build an algorithm $\mathcal{C}_2$ against the collision resistance of $\Pi.\mathsf{Kdf}$ just like we created $\mathcal{C}_1$ in the proof of Claim 5.6. ■

Game 5: In this game the challenger replaces the calls to the random oracle (both in the Send-code and in the G-code) with strings drawn uniformly at random.

Claim 5.9.

$$\mathbf{Adv}^{\mathsf{G}_4}_{\Pi^+,f}(\mathcal{A}) = \mathbf{Adv}^{\mathsf{G}_5}_{\Pi^+,f}(\mathcal{A}). \tag{5.8}$$

Proof. We show that the challenger in Game 4 never repeats a call to the random oracle on the same input. Thus, replacing these calls with uniformly drawn strings in Game 5 yields exactly the same distribution on the export keys.

Suppose at some point during Game 4 the challenger made the random oracle call $G(ms, \eta, aux)$ for the first time (either due to a session accepting, or because $\mathcal{A}$ made this exact G query). Suppose the random oracle responded with ek, and let $t = (ms, \eta, aux, ek, [*])$ be the tuple that was added to L_G in response to the call.

If the adversary later makes a G query on the same values, i.e. a query of the form $G(ms, \eta, aux)$, then Line 2 of the G-code will be used to answer the query. Thus, the random oracle call on Line 11 of the G-code would never be made on the same values twice in Game 4.

Likewise, if a session π accepts with the same values, i.e., master secret ms, nonces $\eta = \eta_C \| \eta_S$, and auxiliary data aux, *after* the initial G query was made,

▷ Game 2

```
// look at the master secret
if ∃(ms, η, aux, ek, [∗]) ∈ L_G:
    π.k ← ek
    update (ms, η, aux, ek, [∗]) to (ms, η, aux, ek, [∗, π])
else
    // no match found – derive new key
    ek ← G(ms, η, aux)
    π.k ← ek
    L_G ← L_G ∪ (ms, η, aux, ek, [π])
```

▷ Game 3

```
// match partner sessions
if (∃(∗, η, aux, ek, [π′]) ∈ L_G) ∧ (f_T(π) = π′):
    π.k ← ek
    update (∗, η, aux, ek, [π′]) to (∗, η, aux, ek, [π′, π])
// look at the master secret
else if ∃(ms, η, aux, ek, [∗]) ∈ L_G:
    π.k ← ek
    update (ms, η, aux, ek, [∗]) to (ms, η, aux, ek, [∗, π])
else
    // no match found – derive new key
    ek ← G(ms, η, aux)
    π.k ← ek
    L_G ← L_G ∪ (ms, η, aux, ek, [π])
```

▷ Game 4

```
// match partner sessions
if (∃(∗, η, aux, ek, [π′]) ∈ L_G) ∧ (f_T(π) = π′):
    π.k ← ek
    update (∗, η, aux, ek, [π′]) to (∗, η, aux, ek, [π′, π])
// match non-fresh sessions on their channel keys
else if (∃(∗, η, aux, ek, [π′]) ∈ L_G)
        ∧ (π, π′ non-fresh) ∧ (π.ck = π′.ck):
    π.k ← ek
    update (∗, η, aux, ek, [π′]) to (∗, η, aux, ek, [π′, π])
// look at the master secret
else if ∃(ms, η, aux, ek, [∗]) ∈ L_G:
    π.k ← ek
    update (ms, η, aux, ek, [∗]) to (ms, η, aux, ek, [∗, π])
else
    // no match found – derive new key
    ek ← G(ms, η, aux)
    π.k ← ek
    L_G ← L_G ∪ (ms, η, aux, ek, [π])
```

▷ Game 5

```
// match partner sessions
if (∃(∗, η, aux, ek, [π′]) ∈ L_G) ∧ (f_T(π) = π′):
    π.k ← ek
    update (∗, η, aux, ek, [π′]) to (∗, η, aux, ek, [π′, π])
// match non-fresh sessions on their channel keys
else if (∃(∗, η, aux, ek, [π′]) ∈ L_G)
        ∧ (π, π′ non-fresh) ∧ (π.ck = π′.ck):
    π.k ← ek
    update (∗, η, aux, ek, [π′]) to (∗, η, aux, ek, [π′, π])
// look at the master secret
else if ∃(ms, η, aux, ek, [∗]) ∈ L_G:
    π.k ← ek
    update (ms, η, aux, ek, [∗]) to (ms, η, aux, ek, [∗, π])
else
    // no match found – derive new key
    ek ←$ {0,1}^κ
    π.k ← ek
    L_G ← L_G ∪ (ms, η, aux, ek, [π])
```

▷ Game 6

```
// match partner sessions
if (∃(∗, η, aux, ek, [π′]) ∈ L_G) ∧ (f_T(π) = π′):
    π.k ← ek
    update (∗, η, aux, ek, [π′]) to (∗, η, aux, ek, [π′, π])
// match non-fresh sessions on their channel keys
else if (∃(∗, η, aux, ek, [π′]) ∈ L_G)
        ∧ (π, π′ non-fresh) ∧ (π.ck = π′.ck):
    π.k ← ek
    update (∗, η, aux, ek, [π′]) to (∗, η, aux, ek, [π′, π])
// has G already been queried on an ms′ valid for π?
else if (∃(ms′, η, aux, ek, [∗]) ∈ L_G)
        ∧ (CheckKey(π, ms′) = true):
    π.k ← ek
    update (ms′, η, aux, ek, [∗]) to (ms′, η, aux, ek, [∗, π])
else
    // no match found – derive new key
    ek ←$ {0,1}^κ
    π.k ← ek
    L_G ← L_G ∪ (⊥, η, aux, ek, [π])
```

Figure 5.2: How to the derive the export key ek of a session π that accepted with master secret ms, nonces $\eta = \eta_C \| \eta_S$, and auxiliary data aux, in Game 2 to Game 6.

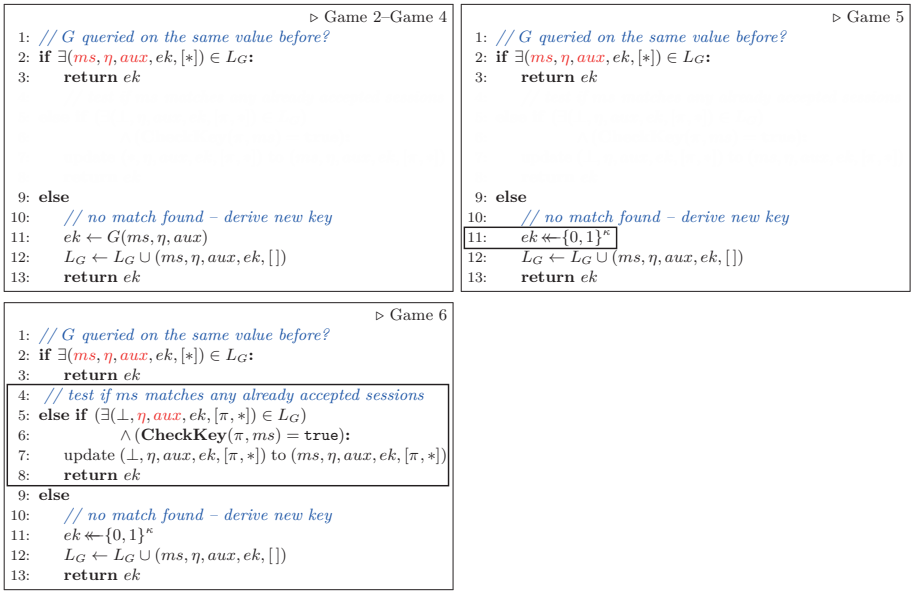


Figure 5.3: How G queries (RO calls) of the form $G(ms, \eta, aux)$ are answered in Game 2 to Game 6.

then the else-if check on Line 10 of the Send-code would match π to t. Thus, the random oracle call on Line 15 of the Send-code would not be made on the same values twice in Game 4 either. ■

In the final game hop the challenger will derive the sessions' export keys independently of their master secrets. To do this, the challenger will use a probabilistic *key-checking* procedure called **CheckKey** to test whether the adversary queried the random oracle at the correct master secret of a session. The **CheckKey** procedure is defined in Algorithm 1. The statements in blue boxes can be ignored for now.

The idea of **CheckKey** is to test the validity of a supplied master secret *indirectly* by checking whether it derives the same channel key as the one held by the session. Of course, this whole exercise might seem totally pointless, since the challenger has direct access to the session's master secrets. However, the purpose of the game hop is to prepare the ground for a subsequent reduction to the ACCE security of protocol Π. This reduction will *not* have direct access to the session's master secrets, hence it needs to be able to simulate this key-checking procedure.

The **CheckKey** procedure can be explained as follows. After being given a master secret ms and a session π, it first derives a guess on π's channel key, denoted ck' (Line 6 in Algorithm 1). If π is non-fresh, then **CheckKey** simply compares ck' with $\pi.ck$ directly (Line 10). On the other hand, when π is fresh, then **CheckKey** tests the validity of ck' by trying to decrypt a ciphertext C that was legitimately created with the actual channel key of π'.

Unfortunately, this stage is complicated by the fact that the stAE scheme is stateful. Recall that a stAE scheme maintains two counters st_{E} and st_{D} for encryption and decryption, respectively. Before attempting to decrypt C, **CheckKey** first needs to recreate a valid decryption state st_{D}. This is shown at Lines 16 through 18. Basically, starting from the initial state of the stAE scheme, **CheckKey** chronologically decrypts each encrypted message output by the session during the handshake (if any). Then it decrypts all ciphertext messages created in prior calls to **CheckKey** (because these advance the session's encrypt state st_{E}). If the correct channel key was used, then this process is guaranteed to generate a decryption state st'_D that "matches"[1] the encrypt state st_E which was used to create the ciphertext C. Finally, **CheckKey** attempts the decryption of C (Line 21).

If **CheckKey** was called on the correct master secret of a session π, then the above shows that it will always return `true` since the derived channel key ck' will equal $\pi.ck$ because Π.Kdf is deterministic. Conversely, if **CheckKey** was

[1] We write "matches" since the recreated state st'_D will not necessarily be *equal* to the decryption state held by π, only that it has the property of yielding a valid decryption.

Algorithm 1 CheckKey(π, ms)

Note: The procedure is parameterized by some integer $c \in \mathbf{N}$. Calls on the same input always return the same value, i.e. **CheckKey** caches its results for every input combination. To simplify the presentation this is not shown below. Statements shown in blue boxes are only executed by reduction algorithm $\mathcal{B}$.

Precondition: To every session π, **CheckKey** associates a random bit d, written $\pi.d$. Let $L_\pi = \{(C_1, H_1), \ldots, (C_r, H_r), (C_{r+1}, \varepsilon), \ldots, (C_s, \varepsilon)\}$ be the list of all the encrypted handshake messages (if any) output by π during the run of Π^+, as well as all the ciphertexts produced by previous calls to **CheckKey**$(\pi, *)$.

$x, y \twoheadleftarrow \{0,1\}^c$
$m_0 \leftarrow 0 \| x$
$m_1 \leftarrow 1 \| y$

// $\eta = \eta_C \| \eta_S$ are the nonces π accepted with, and $aux \leftarrow F_\Pi(\pi.\tau)$
$ck' \leftarrow \Pi.\mathsf{Kdf}(ms, \eta, aux)$

if π is non-fresh**:**
    $ck \leftarrow \pi.ck$ $\quad ck \leftarrow \mathsf{Reveal}(\pi)$
    return $ck \stackrel{?}{=} ck'$
else
    // obtain an encryption of $m_{\pi.d}$ under $\pi.ck$
    $(C, st_{\mathrm{E}}) \leftarrow \Lambda.\mathsf{Enc}(\pi.ck, m_{\pi.d}, \varepsilon, st_{\mathrm{E}})$ $\quad C \leftarrow \mathsf{LR}(\pi, m_0, m_1, \varepsilon)$

    // recreate a decrypt state st'_{D}
    $(*, st'_D) \leftarrow \Lambda.\mathsf{Init}$
    for all $(C', H') \in L_\pi$**:**
        $(*, st'_D) \leftarrow \Lambda.\mathsf{Dec}(ck', C', H', st'_{\mathrm{D}})$

    // decrypt C using ck' and st'_D
    $(m', *) \leftarrow \Lambda.\mathsf{Dec}(ck', C, \varepsilon, st'_D)$

    return $m' \stackrel{?}{\in} \{m_0, m_1\}$

called on a wrong master secret, then it is possible that it incorrectly returns `true`. Namely, if the derivation of the channel key ck' at Line 6 in Algorithm 1 yields the *same* channel key as π, then **CheckKey** will erroneously return `true` both when π is fresh and when it is non-fresh. Moreover, even if the derived channel key was wrong, there is still a possibility of error when π is fresh: by pure chance the decryption at Line 21 of Algorithm 1 could return one of the messages m_0 or m_1 even with the wrong key.

Thus, **CheckKey** has a one-sided error probability. Let $\mathsf{CKerror}$ denote the event that a call to **CheckKey** erroneously returns `true`.

Game 6: The challenger in Game 6 proceeds as in Game 5, except that it starts using the **CheckKey** procedure as indicated on Line 10 of the Send-code and Line 6 of the G-code (Figure 5.2 and Figure 5.3 respectively). Additionally, if a session accepts without a match on L_G, then Game 6 omits its master secret from the tuple that gets added to L_G in the Send-code (Line 17).

Claim 5.10.

$$\mathbf{Adv}^{\mathsf{G5}}_{\Pi^+,f}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{G6}}_{\Pi^+,f}(\mathcal{A}) + \Pr[\mathsf{CKerror}]. \tag{5.9}$$

Proof. By inspecting the Send-code and G-code of Game 5 and Game 6, one sees that they proceed identically unless event $\mathsf{CKerror}$ occurs. In particular, provided **CheckKey** does not make a mistake, then the else-if clause on Line 10 in the Send-code of Game 6 matches π with a tuple on L_G if and only the tuple contains the correct master secret of π (plus of course all the other input to the KDF, which we ignore here). But this is exactly the same as what the else-if clause on Line 12 in the Send-code of Game 5 does too. Similarly, in the G-code of Game 6, the else-if clause on Line 6 assigns a master secret ms to a tuple of L_G if and only it matches the master secret of those sessions contained in the tuple. Combined with the preceding argument for the Send-code of Game 6, this means that no more sessions gets matched to tuples in L_G in the Send-code of Game 6 than in Game 5.

Hence, provided **CheckKey** does not make a mistake, Game 5 and Game 6 proceed identically and the claim follows. ■

It remains to bound the right-hand side of Equation (5.9). Recall that $\mathsf{CKerror}$ represents the event that **CheckKey** erroneously returns `true` on a wrong master secret. Note that this can happen both with a fresh session and with a non-fresh session. Let fresh denote that **CheckKey** was called on a fresh session π according to predicate $\mathsf{Fresh}_{\mathsf{ACCE}}$, and let $\mathsf{non\text{-}fresh}$ denote that **CheckKey** was called on a non-fresh session. Then we have:

$$\Pr[\mathsf{CKerror}] \leq \Pr[\mathsf{CKerror} \wedge \mathsf{fresh}] + \Pr[\mathsf{CKerror} \wedge \mathsf{non\text{-}fresh}]. \tag{5.10}$$

In the case of a non-fresh session, **CheckKey** can by design only make a mistake if the there is a key collision, so the next claim follows at once.

Claim 5.11.

$$\Pr[\mathsf{CKerror} \wedge \mathsf{non\text{-}fresh}] \leq \mathbf{Adv}^{\mathsf{kdfcoll}}_{\Pi.\mathsf{Kdf}}(\mathcal{C}_3). \tag{5.11}$$

Consequently, we are left to bound $\Pr[\mathsf{CKerror} \wedge \mathsf{fresh}]$. To this end, we define the following event:

$$Q : \textbf{CheckKey} \text{ returns } \texttt{true} \text{ when called on a fresh session.} \tag{5.12}$$

We stress that if event Q happened, say due to a call $\textbf{CheckKey}(\pi, ms')$, then this does not necessarily imply that $\pi.ms = ms'$. Event Q also includes those cases where **CheckKey** erroneously returns `true`. We will later show that $\mathcal{A}$ has zero advantage in guessing the Test-challenge correctly unless Q happens (Claim 5.13). The probability $\Pr[\mathsf{CKerror} \wedge \mathsf{fresh}]$ can now be bounded in terms of the occurrence of event Q.

Claim 5.12.

$$\Pr[\mathsf{CKerror} \wedge \mathsf{fresh}] \leq 2 \cdot \Pr[Q]. \tag{5.13}$$

Proof. Event $\mathsf{CKerror} \wedge \mathsf{fresh}$ only occurs if the decryption of C at Line 21 of Algorithm 1 returned one of the two messages m_0 and m_1. We write correctDec for the event that C got decrypted to m_d, and wrongDec for the event that it got decrypted to $m_{\overline{d}}$, where d is the bit associated to the session π in the **CheckKey** procedure.[2] The events correctDec and wrongDec are mutually exclusive, so

$$\Pr[\mathsf{CKerror} \wedge \mathsf{fresh}] = \Pr[\mathsf{correctDec}] + \Pr[\mathsf{wrongDec}]. \tag{5.14}$$

Finally, within the context of **CheckKey**, both correctDec and wrongDec are sub-events of Q, hence, $\Pr[\mathsf{correctDec}] + \Pr[\mathsf{wrongDec}] \leq 2 \cdot \Pr[Q]$. ■

The next claims shows that unless Q happens in Game 6, then $\mathcal{A}$ has zero advantage in answering the Test-challenge correctly.

Claim 5.13. *Suppose that $\mathcal{A}$ output b' as its answer to the* Test*-challenge in Game 6. Then,*

$$\Pr[b' = b \mid \overline{Q}] = \frac{1}{2}. \tag{5.15}$$

[2]Note that event correctDec can happen both legitimately ($\pi.ms = ms'$) and due to an error ($\pi.ms \neq ms'$). Event wrongDec can only happen because of an error.

Proof. If event Q did not happen, then **CheckKey** never returned `true` for any fresh session during Game 6. Since **CheckKey** is always correct when rejecting a key, i.e., when outputting `false`, this implies that $\mathcal{A}$ never queried the random oracle on the correct master secret of any fresh session. In particular, this means that the derived export key of the test-session in Game 6 is distributed exactly like that of a random key. Thus, the bit b is independent of the derived export key from $\mathcal{A}$'s point of view. ■

Claim 5.13 implies that it is sufficient to bound the probability of event Q to bound $\mathcal{A}$'s advantage in Game 6. Furthermore, Claim 5.12 showed that the probability of event $\mathsf{CKerror} \wedge \mathsf{fresh}$ is also bounded in terms of Q. Thus, the only thing that remains in order to bound the right-hand side of Equation (5.9) is to bound $\Pr[Q]$. To this end, we construct an ACCE adversary $\mathcal{B}$ against protocol Π such that

$$\Pr[Q] \leq \mathbf{Adv}^{\mathsf{ACCE}}_{\Pi,f}(\mathcal{B}) + \frac{2qn_P n_\pi}{2^c}, \tag{5.16}$$

where q is the number of random oracle calls made by $\mathcal{A}$ and $c \in \mathbf{N}$ is the free parameter value of the **CheckKey** procedure.

Description of algorithm $\mathcal{B}$. Algorithm $\mathcal{B}$ plays in an ACCE security experiment against protocol Π. It simulates Game 6 for $\mathcal{A}$ by using the sessions in its own ACCE experiment to represent the sessions in Game 6. Basically, $\mathcal{B}$ forwards all of $\mathcal{A}$'s queries to its own ACCE game (to simulate the Test query, $\mathcal{B}$ draws a mock bit b_{sim}). To simulate the sessions' export keys in Game 6, $\mathcal{B}$ maintains the list L_G which it fills out, and answers from, according to the Send-code and G-code of Game 6. However, $\mathcal{B}$ implements the **CheckKey** procedure slightly different from what the challenger in Game 6 does.

Specifically, at Line 9 and Line 13 in Algorithm 1, $\mathcal{B}$ executes the statements shown in blue boxes instead of the respective statements at those lines. The blue boxes represents queries to $\mathcal{B}$'s ACCE game. To compare the key ck' with the real channel key of π when π is non-fresh, $\mathcal{B}$ uses the Reveal query. To obtain a valid ciphertext under π's real channel key when π is fresh, $\mathcal{B}$ uses the "left-or-right" LR query.

Finally, $\mathcal{B}$ stops and outputs a guess (π, b') in its ACCE game if one of the following events happen.

- *Two sessions generated the same nonce*: select π arbitrarily among the fresh sessions and draw b' randomly.
- $\mathcal{A}$ *outputs a guess for the* Test*-challenge*: select π arbitrarily among the fresh sessions and draw b' randomly.

- *A call to* **CheckKey**(π, ms) *returned* `true` *for a fresh session* π:

  This means that the decryption at Line 21 of Algorithm 1 either returned m_0 or m_1. If the result was m_0 then $\mathcal{B}$ outputs $(\pi, 0)$ to its ACCE game. If the result was m_1 then $\mathcal{B}$ outputs $(\pi, 1)$ to its ACCE game.

Analysis of $\mathcal{B}$. Note that the only thing that differs between $\mathcal{B}$'s simulation and Game 6 is $\mathcal{B}$'s implementation of the **CheckKey** procedure. However, $\mathcal{B}$'s usage of Reveal and LR queries perfectly simulates the respective lines in Algorithm 1. Particularly, the secret bit of a session π in $\mathcal{B}$'s ACCE game, i.e., $\pi.b$, simulates the bit associated to π in the **CheckKey** procedure, i.e., $\pi.d$. Thus, if event Q happens, then $\mathcal{B}$'s output in its ACCE experiment will be directly related to the value of $\pi.b$. On the other hand, if Q does *not* happen, then $\mathcal{B}$ by design wins in its ACCE game with probability 1/2.

Formally, suppose $\mathcal{B}$ output (π, b') for some fresh session π. The probability that $\mathcal{B}$ wins in its ACCE security game is then:

$$\Pr[\pi.b = b'] = \Pr[\pi.b = b' \mid Q] \cdot \Pr[Q] + \Pr[\pi.b = b' \mid \overline{Q}] \cdot \Pr[\overline{Q}] \tag{5.17}$$

$$\stackrel{(a)}{=} \Pr[\pi.b = b' \mid Q] \cdot \Pr[Q] + \frac{1}{2}(1 - \Pr[Q]) \tag{5.18}$$

$$\stackrel{(b)}{=} \Big(\overbrace{\Pr[\pi.b = b' \mid Q \wedge \mathsf{correctDec}]}^{=1} \cdot \Pr[\mathsf{correctDec} \mid Q] + \overbrace{\Pr[\pi.b = b' \mid Q \wedge \mathsf{wrongDec}]}^{=0} \cdot \Pr[\mathsf{wrongDec} \mid Q] \Big) \cdot \Pr[Q] + \frac{1}{2}(1 - \Pr[Q]) \tag{5.19}$$

$$= \Pr[\mathsf{correctDec} \mid Q] \cdot \Pr[Q] + \frac{1}{2}(1 - \Pr[Q]) \tag{5.20}$$

$$= \Pr[\mathsf{correctDec} \wedge Q] - \frac{1}{2} \cdot \Pr[Q] + \frac{1}{2} \tag{5.21}$$

$$\stackrel{(c)}{=} \big(\Pr[Q] - \Pr[\mathsf{wrongDec} \wedge Q] \big) - \frac{1}{2} \Pr[Q] + \frac{1}{2} \tag{5.22}$$

$$= \frac{1}{2} \Pr[Q] - \Pr[\mathsf{wrongDec} \wedge Q] + \frac{1}{2} \tag{5.23}$$

$$= \frac{1}{2} \Pr[Q] - \Pr[\mathsf{wrongDec}] + \frac{1}{2} \tag{5.24}$$

$$\geq \frac{1}{2} \Pr[Q] - \frac{q n_P n_\pi}{2^c} + \frac{1}{2}. \tag{5.25}$$

In (a) we used the fact that $\mathcal{B}$ outputs a random bit when Q does not happen,

while (b) and (c) used that event Q is the union of the mutually exclusive events $\mathsf{correctDec}$ and $\mathsf{wrongDec}$. The final inequality is proved as follows.

Let $\overline{b} = 1 - \pi.b$ and let (m_0, m_1) be the two messages associated to the pair (π, ms) in **CheckKey**. Since $m_{\overline{b}}$ is independent of the ciphertext C produced at Line 13 of Algorithm 1, the probability that C decrypts to $m_{\overline{b}}$ at Line 21 is statistically bounded by 2^{-c} for any key k. By taking the union bound over all parties, the number of sessions per party, and the number of random oracle calls, we get that $\Pr[\mathsf{wrongDec}] \leq qn_P n_\pi / 2^c$.

Solving (5.25) and (5.17) for $\Pr[Q]$ yields

$$\Pr[Q] \leq \mathbf{Adv}^{\mathsf{ACCE}}_{\Pi,f}(\mathcal{B}) + \frac{2qn_P n_\pi}{2^c}, \tag{5.26}$$

which is what we wanted to prove.

Concluding the proof of Theorem 5.4. Applying Claim 5.11, Claim 5.12, Claim 5.13 and Equation (5.26) to Equation (5.9), we get

$$\mathbf{Adv}^{\mathsf{G}_5}_{\Pi^+,f}(\mathcal{A}) \leq 3 \cdot \mathbf{Adv}^{\mathsf{ACCE}}_{\Pi,f}(\mathcal{B}) + \frac{6qn_P n_\pi}{2^c} + \mathbf{Adv}^{\mathsf{kdfcoll}}_{\Pi.\mathsf{Kdf}}(\mathcal{C}_3). \tag{5.27}$$

Collecting all the probabilities from Section 5.2.3 to Game 5 we get

$$\mathbf{Adv}^{\mathsf{AKE}^{\mathsf{fs}}}_{\Pi^+,f}(\mathcal{A}) \leq 3 \cdot \mathbf{Adv}^{\mathsf{ACCE}}_{\Pi,f}(\mathcal{B}) + \frac{6qn_P n_\pi}{2^c} + \frac{(n_P n_\pi)^2}{2^{\lambda+1}} + \sum_{i=1}^{3} \mathbf{Adv}^{\mathsf{kdfcoll}}_{\Pi.\mathsf{Kdf}}(\mathcal{C}_i) \tag{5.28}$$

Let $\mathcal{C}$ be the algorithm that with probability $1/3$ implements one of the $\mathcal{C}_i$'s, then

$$\sum_{i=1}^{3} \mathbf{Adv}^{\mathsf{kdfcoll}}_{\Pi.\mathsf{Kdf}}(\mathcal{C}_i) \leq 3 \cdot \mathbf{Adv}^{\mathsf{kdfcoll}}_{\Pi.\mathsf{Kdf}}(\mathcal{C}), \tag{5.29}$$

and Theorem 5.4 follows. ■

5.3 Application to EAP-TLS

In EAP-TLS [RFC5216] the export key ek is derived as follows:

$$ek \stackrel{\text{def}}{=} \mathsf{tls.PRF}(ms, \text{“}\texttt{client EAP encryption}\text{”}, \eta_C \| \eta_S). \tag{5.30}$$

More generally, RFC 5705: “Keying Material Exporters for Transport Layer Security (TLS)” [RFC5705] defines how export keys should be derived from the

TLS handshake for any type of application. In particular, the export key in RFC 5707 is derived as follows:

$$ek \stackrel{\text{def}}{=} \mathsf{tls.PRF}(ms, \text{“}\texttt{label}\text{”}, \eta_C \| \eta_S, aux), \tag{5.31}$$

where `label` is some application dependent label, and aux is an optional auxiliary input that can be added into the key derivation together with the nonces.

Thus, we can apply Theorem 5.4 to EAP-TLS and TLS Key Exporters by in the theorem setting $\Pi = \mathsf{TLS}$ and $\Pi^+ = \mathsf{EAP\text{-}TLS}$ or $\Pi^+ = \mathsf{TLS\text{-}EXPORT}$, with F_Π being ε or aux, respectively. However, we still have to argue that TLS is in fact a secure TLS-like ACCE protocol and that the TLS KDF is key collision resistant. In the following two sections we address these questions.

5.3.1 TLS security

There is a large body of existing analysis on TLS. Here we only focus on a small sample of these results, based on how relevant they are to our current analysis on EAP-TLS. The first thing that should be acknowledged is that each study of TLS comes with its own unique security model. Our own work is no exception in this regard. As a consequence, most of the existing results on TLS cannot be applied verbatim to our setting, but will need some reinterpretation within our formal models. Fortunately, most of the differences are quite minor, concerning superficial things like choice of notation and so on. But there are also some differences that are more substantial and which we feel are worthy to point out. Mainly, these have to do with the choice of corruption model and partnering mechanism. Below we survey a few of the existing results on TLS and discuss how they pertain to our result on EAP-TLS.

Jager, Kohlar, Schäge, and Schwenk (JKKS) [Jag+12]. JKKS were the first to conduct an analysis of the unmodified TLS 1.2 protocol, looking specifically at the TLS-DHE ciphersuite. They showed that TLS-DHE constitutes a secure ACCE protocol. The security model used by JKKS largely mirrors our own, but with some slight differences. First, their corruption model is a little weaker than the AKE^{fs} model we have used in Theorem 5.4. Specifically, in our AKE^{fs} model the adversary is allowed to corrupt the long-term keys of the test-session's peers *before* it accepted provided it has a partner. On the other hand, in the model of JKKS these corruptions always need to happen *after* the test-session accepts. Theorem 5.4 allows us to conclude that $\Pi^+ =$ EAP-TLS-DHE achieves the stronger security guarantees of the AKE^{fs} model only by making a similarly strong assumption on $\Pi =$ TLS-DHE. However, Theorem 5.4 can be modified to work with the weaker model of JKKS as well, but then with a correspondingly weaker conclusion for Π^+.

Second, JKKS used matching conversations in their analysis of TLS, while we use partner functions. Since matching conversations can be recast in terms of partner functions in a straightforward manner this is not a big issue.[3] However, there is a subtle technical difference between the ACCE model defined in this thesis and the ACCE model defined by JKSS, stemming from the difference in choice of partnering mechanism. Specifically, in JKSS's definition of ACCE [Jag+12, Def. 11] one must forbid the adversary from issuing a Reveal query towards the server after it sent out its last message, but *before* the client to which it has a matching conversation received it. This is to avoid a trivial attack whereby the adversary can re-encrypt the final message towards the client, getting it to accept maliciously.

By contrast, the definition of ACCE used in this thesis allows all Reveal queries. It should be noted that the trivial attack in JKSS's model does not imply any obvious weakness in TLS, but rather highlights a peculiarity of using matching conversations as the partnering mechanism when defining ACCE.

Krawczyk, Paterson, and Wee (KPW) [KPW13b]. As mentioned in Section 5.1, KPW showed that all the main handshake variants of TLS, i.e., TLS-DHE, TLS-DH, and TLS-RSA, satisfy a security notion on its key encapsulation mechanism (KEM) called IND-CCCA [HK07]. Additionally, starting from the assumption that the TLS KEM is IND-CCCA secure, they generically proved that the full TLS protocol is a secure ACCE protocol, Hence, by combining these two sets of results, KPW could show that TLS is a secure ACCE protocol for all the handshake variants TLS-DHE, TLS-DH and TLS-RSA.

Remarks similar to those we made for JKKS [Jag+12] also apply to KPW regarding their result's applicability to our analysis of EAP-TLS. Specifically, the ACCE model of KPW is mostly the same as JKKS's, except for one major difference: KPW do not treat forward secrecy at all. Thus, since KPW only allow us to assume (ACCE) security of TLS in our weakest corruption model, we can correspondingly only conclude with (AKE) security of EAP-TLS in our weakest corruption model $\mathsf{AKE}^{\mathsf{nfs}}$ as well.

Kohlar et al. [KSS13] and Li et al. [Li+14]. In parallel work to KPW [KPW13b], Kohlar et al. [KSS13] also proved that the TLS variants TLS-RSA and TLS-DH satisfy the ACCE notion. However, their result is less modular, essentially following the original approach of JKKS [Jag+12]. Later, Li et al. [Li+14] complemented the work of JKKS and Kohlar et al. [KSS13], by conducting an analysis of the pre-shared key variants of TLS, i.e., they showed that the TLS-DHE-PSK, TLS-DH-PSK, TLS-RSA-PSK and TLS-PSK ciphersuites

[3] See Section 6.2.2 for how to recast SIDs as partner functions. The procedure for matching conversations is completely analogous.

all satisfy the ACCE security notion (in appropriate corruption models). As before, the results of Kohlar et al. [KSS13] and Li et al. [Li+14] can be used to instantiate our Theorem 5.4, again given the relevant caveats on the assumed security of the underlying TLS variant and the corresponding conclusion we can draw for EAP-TLS. Note that the result of Li et al. [Li+14] is also particularly interesting from the viewpoint of our first composition theorem (Theorem 4.2), seeing as RADIUS is based on symmetric shared secrets. In particular, if one wants migrate from plain RADIUS to RADIUS-over-TLS [RFC6614], shared secrets can still be supported by using one of the TLS-PSK ciphersuites.

Brzuska et al. [Brz+13a]. Given that the above results can be applied more or less directly to our Theorem 5.4 in order to obtain a result on EAP-TLS, it is interesting to discuss another result where this *cannot* be done. Specifically, Brzuska et al. [Brz+13a] developed a generic composition framework which allowed them to show that the TLS variants TLS-DHE, TLS-DH, and TLS-RSA, all satisfy the ACCE security notion. However, in their analysis—which used SIDs as the partnering mechanism—Brzuska et al. [Brz+13a] defined the SID to consist of the parties' nonces, identities, *and* the TLS pre-master secret. Basing the SID upon secret values does not in general allow for public partnering. For instance, if the KEM used in the TLS handshake was a *re-randomizable* encryption scheme [CKN03, PR07], then the choice of Brzuska et al. [Brz+13a] would not allow for public partnering (see also [Brz+11] for further details). Unfortunately, this also means that we cannot use Brzuska et al.'s result within our Theorem 5.4 since partner functions are defined on public data.

On TLS 1.3. The IETF is currently in the process of standardizing a new version of TLS, denoted TLS 1.3 [Res17]. Unlike the prior versions, TLS 1.3. does not use the derived session key within the handshake itself, and so it avoids the issue that prevented the other versions from being a secure AKE protocol. In fact, several preliminary analyses have already proven that different draft versions of TLS 1.3 satisfy the AKE security notion [Dow+15, KW15, Dow+16]. Moreover, in the context of exporting keys from the TLS handshake, TLS 1.3 even defines a dedicated exporter key much like how EAP-TLS and TLS Key Exporters [RFC5705] do it. Thus, our generic result for turning an ACCE protocol into an AKE is unnecessary for TLS 1.3. Nevertheless, TLS 1.3 *is* still a TLS-like protocol, and so our result could in principle be applied to TLS 1.3 as well, albeit redundantly. Interestingly, however, the existing analysis of TLS 1.3 would not work for this purpose, because of an issue similar to the one we had with the analysis of Brzuska et al. [Brz+13a]. That is, a new feature of TLS 1.3 is that many of the handshake messages are encrypted with a temporary handshake key. But the analyses in [Dow+15] and [Dow+16],

define the SID over the *unencrypted* messages. Thus, in trying to use these results in our Theorem 5.4, we would run into the same problems with public partnering as we had with the analysis of Brzuska et al. [Brz+13a]. The reason why Dowling et al. [Dow+15, Dow+16] are still able to carry out their analysis, is because they leverage the fact that TLS 1.3 provides so-called *multi-stage* security [FG14], where different stage keys are computationally independent. By modifying Theorem 5.4 to assume a "TLS 1.3-like" structure on protocol Π, and by incorporating the multi-stage assumption, we could potentially be able to obtain a similar black-box result for EAP-TLSv1.3 as we have for EAP-TLS(v1.2).

Other results on TLS. Two other works that also analyze the TLS (1.2) protocol are [Koh+15] and [Bha+14b]. However, the models used in these analyses are significantly different from ours, making their use in Theorem 5.4 difficult. On a different note, Bhargavan et al. [Bha+14b] showed that the *full* TLS protocol, including resumption and renegotiation, is vulnerable to an unknown key-share attack [BM99]. The attack allows an adversary to synchronize the master secret and nonces of two non-partnered sessions, leading them to derive the same channel key. While the attack carries over to EAP-TLS, it does not invalidate our results, since our model does not consider resumption and renegotiation. However, it should be noted that this has been done for the sake of simplicity, not because of an essential limitation in our analysis. Our result can be extended to incorporate features like renegotiation, resumption or ciphersuite and version negotiation, either by using the *multi-phase* ACCE model of Giesen et al. [GKS13] or the *multi-ciphersuite* ACCE model of Bergsma et al. [Ber+14]. The former has been used to prove results on TLS with renegotiation [Ber+14], while the latter has been used to prove results on SSH and TLS with ciphersuite and version negotiation [Ber+14, DS15]. Since our proof uses the underlying ACCE protocol in an almost black-box way, by adopting one of the above models we could inherit their corresponding results for EAP-TLS as well.

Alternatives to the ACCE security notion. The main reason for using the ACCE security notion in our analysis is that is has proved to be a very useful model for studying real-world protocols that intermix the key exchange stage with the channel stage. Since our result applies to *any* ACCE protocol that is TLS-like, it can be applied to all of these protocols in a nearly black-box manner. In particular, we can plug in any existing ACCE result without having to re-do any of the steps carried out in the (ACCE) proof itself. For example, our result applies unmodified to every ciphersuite version of TLS for which there exist an ACCE proof. Moreover, we can even apply our theorem to

future versions of TLS, as long as these continue to be TLS-like and derive their channel keys using a key collision resistant KDF. Even so, in the specific case of TLS, one might ask whether another approach could have given a simpler, yet equally modular proof of the same result, namely that EAP-TLS constitutes a secure AKE protocol.

Krawczyk, Paterson, and Wee (KPW) [KPW13b] showed that all the major handshake variants of TLS satisfy a security notion on its key encapsulation mechanism (KEM) called IND-CCCA [HK07]. If we could reduce the AKE security of EAP-TLS to the IND-CCCA security of the TLS KEM, then the results of [KPW13b] would automatically give us a corresponding result on EAP-TLS for all the major TLS ciphersuites.

Unfortunately, it is not obvious how such a result could be obtained in a black-box manner from the KEM defined by KPW. Technically, in order to reduce the AKE security of EAP-TLS to the IND-CCCA security of the TLS KEM, we need to be able to simulate the key derivation step in the AKE game of EAP-TLS. This requires knowledge about the sessions' master secrets. However, the KEM defined by KPW does not contain the TLS master secret. This means that an adversary against the TLS-KEM in the IND-CCCA game cannot simulate the Test-challenge for some adversary playing in the AKE game against EAP-TLS. Moreover, as remarked by KPW [KPW13b, Remark 4], if the KEM key *was* actually defined to be the TLS master secret, then the resulting scheme would be insecure for TLS-RSA, provided that RSA PKCS#1v1.5 is re-randomizable. On the other hand, Bhargavan et al. [Bha+14b] conjecture that re-randomizing RSA PKCS#1v1.5 is infeasible, allowing the master secret to be used as the KEM key in TLS-RSA too. We forgo the whole issue by not reducing to the KEM-security of TLS at all.

We stress that the KEM used to explain the TLS handshake in Figure 5.1 is only meant for illustratory purposes, and is *not* the same as the KEM used by KPW [KPW13b].

5.3.2 On the key collision resistance of the TLS KDF

The TLS key derivation function tls.PRF is an iterated construction based on the HMAC [RFC2104] function, which itself is based on some underlying hash function H. Let $\overline{H}$ denote the HMAC function using H as its underlying hash function, that is,

$$\overline{H}(K, M) \stackrel{\text{def}}{=} H\left((K \oplus \texttt{opad}) \| H\left((K \oplus \texttt{ipad}) \| M\right)\right), \tag{5.32}$$

where ipad and opad are distinct constants.

The TLS 1.2 KDF is defined as follows, where the variable t depends on

how much keying material is needed:

$$\mathsf{tls.PRF}(K, M) \stackrel{\text{def}}{=} \Big|\Big|_{i=1}^{t} \overline{H}(K, A(i) \| \text{“}\texttt{key expansion}\text{”} \| M), \tag{5.33}$$

with

$$\begin{aligned} A(1) &= \overline{H}(K, \text{“}\texttt{key expansion}\text{”} \| M) \\ A(i) &= \overline{H}(K, A(i-1)). \end{aligned}$$

In TLS, $M = \eta_C \| \eta_S$ is the concatenation of the client and server nonce. Note that $\mathsf{tls.PRF}$ does not take any auxiliary input.

Theorem 5.14. *A key collision in* $\mathsf{tls.PRF}$ *implies a collision in* H*.*

Proof. Suppose $\mathsf{tls.PRF}(K, M) = \mathsf{tls.PRF}(K', M)$, with $K \neq K'$, and let $S =$ “$\texttt{key expansion}$”$\|M$. By (5.33) we have in particular that

$$\overline{H}(K, A(1) \| S) = \overline{H}(K', A'(1) \| S), \tag{5.34}$$

where $A'(1) = \overline{H}(K', S)$. Expanding (5.34) using (5.32) we get:

$$\begin{gathered} H\left(K \oplus \texttt{opad} \| H\left(K \oplus \texttt{ipad} \| A(1) \| S\right)\right) \\ = \\ H\left(K' \oplus \texttt{opad} \| H\left(K' \oplus \texttt{ipad} \| A'(1) \| S\right)\right). \end{gathered} \tag{5.35}$$

Letting $X = H\left(K \oplus \texttt{ipad} \| A(1) \| S\right)$ and $Y = H\left(K' \oplus \texttt{ipad} \| A'(1) \| S\right)$ denote the “inner” hash function values, (5.35) becomes:

$$H(K \oplus \texttt{opad} \| X) = H(K' \oplus \texttt{opad} \| Y). \tag{5.36}$$

Since $K \oplus \texttt{opad} \neq K' \oplus \texttt{opad}$, it follows that $(K \oplus \texttt{opad} \| X, K' \oplus \texttt{opad} \| Y)$ constitute a collision in H. ■

Remark 5.15. The construction of $\mathsf{tls.PRF}$ in TLS 1.0/1.1 is different from the one in TLS 1.2 (shown in Equation (5.33)). In versions prior to TLS 1.2, $\mathsf{tls.PRF}$ is defined as $P_{\text{MD5}} \oplus P_{\text{SHA1}}$, where P_{MD5} and P_{SHA1} are equal to the right-hand side of Equation (5.33) with $\overline{H}$ using MD5 and SHA1, respectively. Theorem 5.14 only applies to the construction used in TLS 1.2. ▲

Remark 5.16. It is interesting to note that HMAC in general is *not* key collision resistant. As observed by Dodis et al. [Dod+12], HMAC has two large classes of so-called *weak keys* with exactly the property that $\text{HMAC}(K, M) = \text{HMAC}(K', M)$. These weak keys arise due to an ambiguity in how HMAC

handle different-length keys. For example, if d is the block size of the underlying hash function used in HMAC and $|K| < d$, then K and $K' = K\|0$ lead to a key collision. Similarly, if $|K| > d$ and $K' = H(K)$ we also get a key collision. On the other hand, the way HMAC is used within TLS does not lead to key collisions since TLS only uses fixed-length keys. ▲

Chapter 6

Security of IEEE 802.11

Contents

In Chapter 5 we proved that EAP-TLS is a secure 2P-AKE protocol. By the first composition theorem this means that basic EAP using EAP-TLS as its EAP method is a secure 3P-AKE protocol with weak forward secrecy. To complete the picture on full EAP having full forward, we need to establish that there is a link-layer protocol which satisfies the requirements of the second composition theorem. In this chapter we do exactly that for the IEEE 802.11 protocol. IEEE 802.11 is also of independent interest outside of its use in EAP, since it is the most widely used standard for creating wireless LANs.

6.1 Summary of the IEEE 802.11 protocol

IEEE 802.11 is a link-layer protocol, aiming to secure the wireless link between a client and an access point. As explained in Section 2.2.3, IEEE 802.11 consists of two main security protocols for this purpose: the 4-Way-Handshake (4WHS) protocol used to authenticate and establish a session key between the client and access point; and CCMP used to secure the actual application data.

The 4WHS is based on a symmetric *pairwise master key (PMK)* shared between the client and the access point. The PMK can either be a pre-shared key (PSK) or distributed through some other means, for instance using EAP. The first alternative is most typically found in home networks where a static PMK is manually configured at the access point and at every connecting device.[1] 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. The protocol used to access the authentication server is normally EAP.

In Section 6.2 we will analyze the PSK variant of the 4WHS protocol, and in Section 6.2.4 we describe how this result can be combined with the composition theorems of Chapter 4 to also get a result for the enterprise variant of IEEE 802.11. In Section 6.3 we analyze the CCMP algorithm. Finally, in Section 6.4 we informally discuss how our results on the 4WHS protocol can be extended to also cover multi-ciphersuite and negotiation security.

6.1.1 Related work on IEEE 802.11

As explained in Section 2.2.2, IEEE 802.11 has been subject to a large amount of cryptanalysis, especially against WEP and TKIP. Here we only discuss related work as it pertains to the formal analysis of IEEE 802.11. In the symbolic setting, He et al. [He+05] have conducted a formal analysis of the 4WHS protocol using their Protocol Compositional Logic. In the computational setting, Küsters and Tuengerthal [KT11b, KT11a] have analyzed both the 4WHS protocol and CCMP in their universal composability framework called IITM. In the game-based setting the only work we are aware of that attempts to analyze the 4WHS protocol is [ZMM05]. However, this work is quite rudimentary; security definitions and theorems are only outlined and it provides no proofs nor

[1] The PMK is usually not configured directly, but instead derived from a password using the Password Based Key Derivation Function 2 (PBKDF2) [RFC8018]. We ignore this detail here.

proof sketches. To the best of our knowledge, there is no existing analysis of CCMP in the game-based setting.

6.2 Analyzing the 4-Way Handshake

6.2.1 Formal modeling

The 4WHS protocol was described in detail in Section 2.2.3, and our formal modeling of it is shown in Figure 6.1. The 4WHS depends on a pseudorandom function PRF and a MAC scheme $\Sigma = (\mathsf{MAC}, \mathsf{Vrfy})$. We use the notation $[x]_k \stackrel{\text{def}}{=} x \| \sigma$ to denote a message x together with its MAC tag $\sigma \leftarrow \mathsf{MAC}(k, x)$.

An IEEE 802.11 network is identified by its SSID. In the PSK setting each SSID is associated with a single 256 bit pairwise master key (PMK). However, the same SSID can be broadcasted by multiple different access points. This could happen either by chance if independent networks unknowingly configuring the same SSID, or deliberately if multiple access points are combined to form an extended service set (ESS) in order increase the coverage of the network. In the former case, the PMK will (usually) be different, while in the latter case the same PMK will be shared by all the access points of the ESS. Technically speaking, if two independent networks configure the same SSID *and* PMK, then they are in fact part of the same ESS.

An access point can also broadcast multiple SSIDs at the same time, and hence belong to more than one ESS (using different PMKs). For simplicity we are going to assume that every ESS has a unique SSID. In the PSK setting all clients connecting to the same ESS will share the same PMK.

We are mostly going to ignore the details of the IEEE 802.11 frame format used in the real 4WHS protocol. For our purposes it is sufficient to model the four handshake messages as consisting of a nonce plus some value $p_i = i \| x$ which uniquely determines each message m_i. If a received message does not match the expected format it is silently discarded.

For p_1 in particular we moreover assume that x is a constant, which means that p_1 itself is a constant. Thus, although the first handshake message lacks integrity protection, an attacker can in effect only modify the nonce value because a client will always check that it matches the excepted format of "$\eta_{AP} \| 1 \| x$". Of course, a real IEEE 802.11 frame consists of many bit fields, but for message m_1 they all have pre-determined values except for the nonce field. So modeling p_1 as a constant faithfully represents the real IEEE 802.11 frame.

For the other three handshake messages there *are* variable bit fields that an attacker could potentially influence. But since these messages are protected by a MAC, the adversary will be unable to modify them (as we will show).

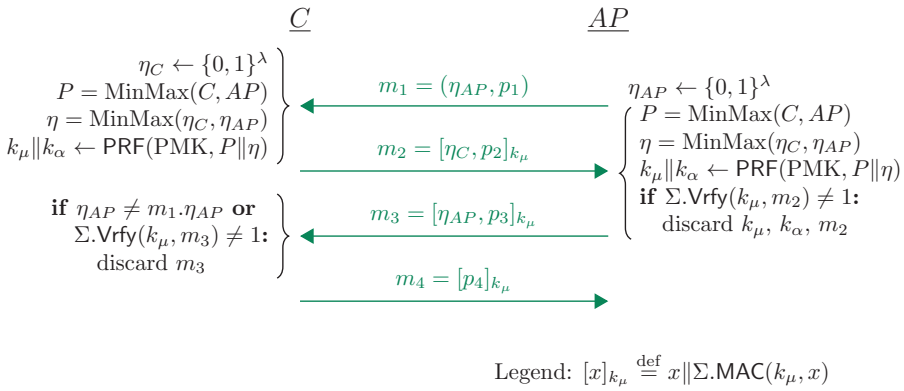


Figure 6.1: Our formal model of the IEEE 802.11 4-Way Handshake protocol. The client C and access point AP share a symmetric key PMK.

Recall from Section 2.2.3 that prior to the 4WHS there is a negotiation phase where the client and access point agree upon the ciphersuite to use. This includes the choice of PRF and Σ. In this section we assume that there is a single fixed ciphersuite being used. The topic of multi-ciphersuite and negotiation security will be treated in Section 6.4.

Identities in the 4WHS protocol are based on the parties' 48 bit link-layer addresses. The functions min and max compute the minimum and maximum of two link-layer addresses when treated as 48 bit unsigned integers. In the following, let

$$\text{MinMax}(X, Y) = \min\{X, Y\} || \max\{X, Y\}. \tag{6.1}$$

The 4WHS protocol proceeds as follows.

1. The exchange begins with the access point AP sending the message $m_1 = \eta_{AP} \| p_1$ to the client C, where η_{AP} is a 256 bit nonce and p_1 is a constant.

2. On receiving $m_1 = \eta_{AP} \| p_1$, the client C generates its own 256 bit nonce η_C and derives a *pairwise transient key (PTK)* as

$$\text{PTK} \stackrel{\text{def}}{=} k_\mu \| k_\epsilon \| k_\alpha \leftarrow \mathsf{PRF}(\text{PMK}, P \| \eta), \tag{6.2}$$

   where $P \leftarrow \text{MinMax}(C, AP)$ and $\eta \leftarrow \text{MinMax}(\eta_C, \eta_{AP})$.

   The sub-key k_α will be the session key eventually output by the client in the 4WHS. The sub-key k_μ is the MAC key used to protect the handshake messages. The sub-key k_ϵ is an encryption key used to protect a group

key GTK transmitted from AP to C. Since we do not model any group aspect of IEEE 802.11 in this thesis, we ignore k_ϵ and set it to be the empty string ε.

After deriving PTK, C creates and sends the next protocol message $m_2 = [\eta_C \| p_2]_{k_\mu}$.

3. On receiving $m_2 = [\eta_C \| p_2]_{k_\mu}$, the access point AP derives the pairwise transient key $\text{PTK} = k_\mu \| k_\alpha \leftarrow \mathsf{PRF}(\text{PMK}, P \| \eta)$ according to Equation (6.2). With the sub-key k_μ it verifies the MAC tag on m_2.

   If the verification succeeds, then AP stores $\text{PTK} \leftarrow k_\mu \| k_\alpha$ as its PTK and sends out the third protocol message $m_3 = [\eta_{AP} \| p_3]_{k_\mu}$. Additionally, AP, or rather the corresponding session at AP, sets the accept state to $\alpha = \mathsf{accepted}$ (since the 4WHS does not consist of any sub-protocols, we simplify the accept vector $\vec{\alpha}$ to a single value $\alpha = \alpha_F$).

   If the verification fails, then AP silently discards m_2, as well as the derived PTK, and continues running.

4. On receiving $m_3 = [\eta_{AP} \| p_3]_{k_\mu}$, the client C checks that η_{AP} is the same as the nonce it received in message m_1 (denoted "$m_1.\eta_{AP}$" in Figure 6.1) and verifies that the MAC tag on message m_3 is valid.

   If either of these checks fail, then C silently discards m_3 and continues running.

   Otherwise, C sends out the final handshake message $m_4 = [p_4]_{k_\mu}$. Additionally, it sets its own acceptance state to $\alpha = \mathsf{accepted}$.

5. On receiving m_4, the access point AP verifies the MAC with the key k_μ.

   If the verification succeeds, then the 4WHS is over and AP is ready to receive encrypted messages under the key k_α.

   If the verification fails, then AP silently discards the message and continues running.

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

Note also that the error handling semantics of the 4WHS is different from protocols like TLS and SSH. Specifically, rather than rejecting immediately on receiving a bad message, a session will instead silently discard it. Combined with the fact that the key used to verify the handshake messages (k_μ) is derived from the handshake messages themselves, modeling the error handling semantics of the 4WHS protocol will make our analysis a little more complicated (specifically the proof of explicit entity authentication in Section 6.2.3).

6.2.2 AKE$^{\mathsf{nfs}}$ security

We begin by proving that the 4WHS constitutes a secure 2P-AKE protocol in the AKE$^{\mathsf{nfs}}$ model, when all PMKs are pre-shared keys. Remember that we have assumed that each SSID belongs to a unique ESS, which is potentially served by multiple access points all sharing the same PMK. All clients connecting to this ESS will also use the same PMK. Since a client might share multiple PMKs with the same access point if the latter serves multiple ESSs, we slightly change the syntax of the $\mathsf{Corrupt}$ query to instead take an SSID as input, identifying the PMK of a specific ESS. Since the access point has the initiator role and the client has the responder role in the 4WHS, we write $\mathcal{P}_{AP} = \mathcal{I}$ and $\mathcal{P}_C = \mathcal{R}$.

Theorem 6.1. *For any adversary $\mathcal{A}$ in security experiment AKE$^{\mathsf{nfs}}$ against the 4WHS protocol as described above, we can create a partner function f and an algorithm $\mathcal{D}$, such that*

$$\mathbf{Adv}^{\mathsf{2P\text{-}AKE}^{\mathsf{nfs}}}_{\mathsf{4WHS},f}(\mathcal{A}) \leq 2 \cdot n_{\mathrm{SSID}} \cdot \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{PRF}}(\mathcal{D}) + \frac{(n_P n_\pi)^2}{2^{\lambda+1}}, \tag{6.3}$$

where n_{SSID} is the number of unique SSIDs, n_π is the number of sessions that $\mathcal{A}$ creates at each party, λ is the length of the nonces, and $n_P = |\mathcal{P}_C| + |\mathcal{P}_{AP}|$.

By our assumption above, n_{SSID} corresponds to the number of ESSs and thus also gives an upper bound on the number of PMKs in the system. Moreover, by assuming that no access point belongs to more than c different ESSs, then $c \cdot |\mathcal{P}_{AP}|$ is an upper bound on n_{SSID}.

Proof.

Defining the partner function f. For the analysis of the 4HWS it would be natural to use session identifiers as the partnering mechanism. Namely, the session identifier of a session π would be the string $\mathsf{sid} = P\|\eta$ that π input to its PRF in order to create the session key (see Equation (6.2)). However, because our security model is phrased in terms of partner functions, we instead synthetically encode the session identifier as a partner function by saying that π's partner is the *first* session—different from π—that sets the same session identifier as π. Taking the first such session is important in order to make the partner function well-defined.

In more detail, suppose π^i_C is a client session and π^i_{AP} is an access point session. For the purposes of this description, let us associate an extra variable sid to each session. Session π^i_C sets its value of sid to be the string that it input to the PRF after having received the first handshake message. Session π^i_{AP} also sets its value of sid to be the input to the PRF, but it only sets its value *after*

it has verified the MAC of the second handshake message. Partner function f can now be defined as follows.

- *Definition of f*: π_C^i and π_{AP}^j are partners if and only if
  1. $\pi_C^i.\mathsf{sid} = \pi_A^j.\mathsf{sid}$, and
  2. π_C^i and π_A^i where the *first* sessions at C and AP, respectively, for which Item 1 holds.

Note that since the party identities of the session's intended peers are included in the sid string, we do not need to include agreement on peers as an explicit requirement. The soundness of f is immediate from the value of sid and PRF being a deterministic function. In fact, f has perfect soundness and is also a local partner function (Definition 3.5). This will be important when we look at IEEE 802.11 combined with upper-layer authentication in Section 6.2.4.

Game 0: This is the real 2P-AKE$^{\mathsf{nfs}}$ security game, hence

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

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

$$\mathbf{Adv}^{\mathsf{G}_0}_{\mathsf{4WHS},f}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{G}_1}_{\mathsf{4WHS},f}(\mathcal{A}) + \frac{(n_P n_\pi)^2}{2^{\lambda+1}}. \tag{6.4}$$

Game 2: This game implements a selective AKE security game where at the beginning of the game the adversary has to commit to the ESS which the test-session will be connected to. Specifically, at the beginning of the game the adversary has to output an SSID and the game aborts if the test-session was not connected to the ESS having this SSID.

Claim 6.2.

$$\mathbf{Adv}^{\mathsf{G}_1}_{\mathsf{4WHS},f}(\mathcal{A}) \leq n_{\mathrm{SSID}} \cdot \mathbf{Adv}^{\mathsf{G}_2}_{\mathsf{4WHS},f}(\mathcal{A}'). \tag{6.5}$$

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. $\mathcal{A}'$ randomly selects an SSID (and thus an ESS) and outputs this 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. If the test-session selected by $\mathcal{A}$ does not belong to the SSID network guessed by $\mathcal{A}'$, then $\mathcal{A}'$ stops its simulation an outputs a random bit. Else, it outputs the same bit as $\mathcal{A}$. Algorithm $\mathcal{A}'$ perfectly simulates Game 1 for $\mathcal{A}$, and since its guess is correct with probability at least $1/n_{\mathrm{SSID}}$ the claim follows. ■

In the remainder of the proof, let SSID^* denote the SSID that the adversary commits to in Game 2, and let PMK^* denote the corresponding PMK used in the ESS identified by SSID^*. Note that by the requirements of the $\mathsf{Fresh}_{\mathsf{AKE}^{\mathsf{nfs}}}$ predicate (Figure 3.2), PMK^* cannot be exposed if the test-session is to be fresh. In particular, this means that the adversary cannot make a $\mathsf{Corrupt}(\text{SSID}^*)$ query.

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

Claim 6.3.

$$\mathbf{Adv}^{\mathsf{G}_2}_{\mathsf{4WHS},f}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{G}_3}_{\mathsf{4WHS},f}(\mathcal{A}) + 2 \cdot \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{PRF}}(\mathcal{D}). \tag{6.6}$$

Proof. If it is possible to distinguish between Game 2 and Game 3, then we can create a distinguisher algorithm $\mathcal{D}$ against the PRF security of the function PRF. Algorithm $\mathcal{D}$ has access to an oracle $\mathcal{O}$, which either implements the function $\mathsf{PRF}(\widetilde{\text{PMK}}, \cdot)$ for some independent and uniformly distributed key $\widetilde{\text{PMK}}$, or it implements a truly random function $\$(\cdot)$. Algorithm $\mathcal{D}$ begins by choosing a random bit b_{sim} and creating all the PMKs for all ESSs different from the one identified by SSID^*. It then runs $\mathcal{A}$, answering its queries as follows.

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

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

Concluding the proof of Theorem 6.1. Suppose the test-session in Game 3 accepted with the sid variable set to $P\|\eta$. Since all nonces in the game are unique by Game 1, the only sessions that evaluated the pseudorandom function on sid was the test-session and possibly its partner. However, by Game 3 the PRF is now a truly random function, so the PTK derived by the test-session (and possibly its partner) is a truly random string $\widetilde{\text{PTK}} = \widetilde{k_\mu}\|\widetilde{k_\alpha} \leftarrow \{0,1\}^{2\kappa}$. Moreover, $\widetilde{k_\alpha}$ is independent of all other values. Thus, $\mathbf{Adv}^{\mathsf{G}_3}_{\mathsf{4WHS},f}(\mathcal{A}) = 0$, and Theorem 6.1 follows. ■

6.2.3 Explicit entity authentication

We now prove that the 4WHS protocol additionally provides explicit entity authentication. The proof of this fact follows the same outline as for the key-indistinguishability part of the proof, using essentially the same game hops. However, instead of bounding the key-indistinguishability advantage of the adversary in the final game, we instead bound the probability that a session will accept maliciously. Intuitively, we can translate this event into a forgery for the MAC algorithm Σ since the adversary will either have to forge an m_2 message to the access point or an m_3 message to the client in order for a malicious accept to happen.

Alas, the proof is complicated by the aforementioned error handling semantics of the 4WHS protocol. That is, when a session receives a bad message it silently discards it and continues running the protocol instead of immediately rejecting. This means that the adversary can make many attempts at getting an access point to accept an m_2 message or a client to accept an m_3 message. The first case is especially subtle to deal with since the access point will derive a new PTK for each received m_2 message. To better align our reduction to the possibility of an adversary making many attempts at the m_2 and m_3 messages, we reduce to a variant of SUF-CMA security that allows multiple verification attempts; see Appendix A.2 for the formal definition.

While single-verification and multi-verification are not equivalent in the traditional UF-CMA setting, they *are* equivalent in the stronger SUF-CMA setting; see [BGM04]. Moreover, for message authentication *codes*—as opposed to message authentication schemes in general—UF-CMA security implies SUF-CMA security. Since the IEEE 802.11 4WHS protocol only uses MACs and not general message authentication schemes, the multi-verification SUF-CMA assumption is justified provided the MAC scheme is UF-CMA secure. The security of the HMAC [RFC2104] algorithm and the CMAC [FIPS:SP-800-38B] algorithm used by the IEEE 802.11 standard is well-studied; see [GPR14, Bel15, IK03a, IK03b].

Theorem 6.4. *For any adversary $\mathcal{A}$ in security experiment AKE$^{\mathsf{nfs}}$-EA against the 4WHS, we can create algorithms $\mathcal{D}$ and $\mathcal{F}$, such that*

$$\begin{aligned}\mathbf{Adv}^{\mathsf{2P\text{-}AKE^{nfs}\text{-}EA}}_{\mathsf{4WHS},f}(\mathcal{A}) \leq 2 \cdot n_{\mathrm{SSID}} \cdot \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{PRF}}(\mathcal{D}) + \frac{(n_P n_\pi)^2}{2^{\lambda+1}} \\ + 2n'_\pi \cdot (q+1) \cdot n_{\mathrm{SSID}} \cdot \mathbf{Adv}^{\mathsf{SUF\text{-}CMA}}_{\Sigma}(\mathcal{F}),\end{aligned} \tag{6.7}$$

where f, n_{SSID}, n_π, n_P and λ are the same as in Theorem 6.1, and where n'_π is the maximum number of sessions $\mathcal{A}$ creates in each ESS, and q is the maximum number of m_2 messages that $\mathcal{A}$ sends to an access point session.

Proof. The initial part of the proof proceeds through three game hops that are completely analogous to the first three game hops of the proof of Theorem 6.1.

Game 0: This is the real explicit entity authentication security game, hence

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

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

$$\mathbf{Adv}^{\mathsf{G}_0}_{\mathsf{4WHS},f}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{G}_1}_{\mathsf{4WHS},f}(\mathcal{A}) + \frac{(n_P n_\pi)^2}{2^{\lambda+1}}. \tag{6.8}$$

Game 2: This game implements a selective security game where at the beginning of the game the adversary has to commit to the ESS which the first session that accepts maliciously connects to. Just like for Game 2 of Theorem 6.1 (Claim 6.2), we have

$$\mathbf{Adv}^{\mathsf{G}_1}_{\mathsf{4WHS},f}(\mathcal{A}) \leq n_{\mathrm{SSID}} \cdot \mathbf{Adv}^{\mathsf{G}_2}_{\mathsf{4WHS},f}(\mathcal{A}'). \tag{6.9}$$

In the remainder of the proof, let SSID^* denote the SSID that the adversary commits to in Game 2, let ESS^* denote the ESS identified by SSID^*, and let PMK^* denote the corresponding PMK used in ESS^*.

Game 3: In this game the challenger replaces the pseudorandom function PRF with a random function $\$(\cdot)$ in all evaluations involving PMK^*. That is, calls of the form $\mathsf{PRF}(\mathrm{PMK}^*, \cdot)$ are instead answered by $\$(\cdot)$. By the same arguments as for Game 3 of Theorem 6.1 (Claim 6.3), we have

$$\mathbf{Adv}^{\mathsf{G}_2}_{\mathsf{4WHS},f}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{G}_3}_{\mathsf{4WHS},f}(\mathcal{A}) + 2 \cdot \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{PRF}}(\mathcal{D}). \tag{6.10}$$

In the next game the adversary additionally has to commit to the session (in ESS^*) that will accept maliciously first.

Game 4: This game implements a selective security game where at the beginning of the game the adversary has to commit to the session in ESS^* which accepts maliciously first. With the same type of reduction as for Game 2, we have

$$\mathbf{Adv}^{\mathsf{G}_3}_{\mathsf{4WHS},f}(\mathcal{A}) \leq n'_\pi \cdot \mathbf{Adv}^{\mathsf{G}_4}_{\mathsf{4WHS},f}(\mathcal{A}'). \tag{6.11}$$

In the following let $\pi^i_{U^*}$ denote the session that $\mathcal{A}$ commits to in Game 4. We conclude the proof of Theorem 6.4 by showing that if $\mathcal{A}$ gets $\pi^i_{U^*}$ to accept maliciously in Game 4, then we can construct an algorithm $\mathcal{F}$ that creates forgeries for the MAC algorithm Σ.

Claim 6.5.

$$\mathbf{Adv}^{\mathsf{G_4\text{-}EA}}_{\mathsf{4WHS},f}(\mathcal{A}) \leq 2(q+1) \cdot \mathbf{Adv}^{\mathsf{SUF\text{-}CMA}}_{\Sigma}(\mathcal{F}). \tag{6.12}$$

Proof. From an adversary $\mathcal{A}$ in Game 4 we create an algorithm $\mathcal{F}$ against the SUF-CMA security of MAC algorithm Σ. The algorithm $\mathcal{F}$ has access to two oracles $\mathcal{O}^{\mathsf{MAC}}(\cdot)$ and $\mathcal{O}^{\mathsf{Vrfy}}(\cdot,\cdot)$ which implements the functions $\Sigma.\mathsf{MAC}(\widetilde{k},\cdot)$ and $\Sigma.\mathsf{Vrfy}(\widetilde{k},\cdot,\cdot)$ for some independent uniformly distributed key $\widetilde{k}$. The idea of $\mathcal{F}$ is to embed the oracles $\mathcal{O}^{\mathsf{MAC}}$ and $\mathcal{O}^{\mathsf{Vrfy}}$ into computations that would normally involve using the PTK of the target session $\pi^i_{U^*}$.

Algorithm $\mathcal{F}$ begins by drawing a random bit b_{sim} and waits for $\mathcal{A}$ to commit to a pair $(\mathrm{SSID}^*, \pi^i_{U^*})$ according to Game 2 and Game 4. If $\pi^i_{U^*}$ belongs to a client, say $U^* = C$, let AP be its intended peer, i.e., $\pi^i_{U^*}.\mathsf{peers} = \{C, AP\}$ (remember that even if an ESS potentially contains many clients and access points, a connection will nonetheless be between two specific parties in ESS). Conversely, if $\pi^i_{U^*}$ belongs to an access point AP, let C be the client it wants to talk to.

For sessions not pertaining to network ESS^*, i.e., those not using PMK^* as their long-term key, $\mathcal{F}$ simulates everything itself by creating their PMKs. For sessions in ESS^*, $\mathcal{F}$ still mostly simulates everything itself, but this time by implementing a random function $\$(\cdot)$ rather than the function $\mathsf{PRF}(\mathrm{PMK}^*, \cdot)$. This can be done using lazy sampling [BR04, Section 4.3]. However, for certain specific computations which we describe below, $\mathcal{F}$ will embed its MAC oracles $\mathcal{O}^{\mathsf{MAC}}$ and $\mathcal{O}^{\mathsf{Vrfy}}$. We split our proof into two cases, depending on whether $\pi^i_{U^*}$ belongs to the client C or the access point AP.

Case $U^* = C$. Assume that $\pi^i_{U^*}$ belongs to the client C. In this case $\mathcal{F}$ uses its tagging oracle $\mathcal{O}^{\mathsf{MAC}}$ to create the m_2 message of $\pi^i_{U^*}$. Similarly, when $\pi^i_{U^*}$ receives an m_3 message, then $\mathcal{F}$ uses the oracle $\mathcal{O}^{\mathsf{Vrfy}}$ to verify it. If the verification succeeds, then we will argue below that $\mathcal{F}$ has created a forgery in its SUF-CMA security game. If the verification fails, then $\mathcal{F}$ discards the message and continues the simulation.

Besides this, there is one additional place where $\mathcal{B}$ embeds its $\mathcal{O}^{\mathsf{Vrfy}}$ oracle. Namely, suppose the nonce $\pi^i_{U^*}$ received in the first handshake message, say η_{AP}, was created by a session π^j_{AP} at the access point AP. If $\mathcal{A}$ forwards $\pi^i_{U^*}$'s m_2 message back to π^j_{AP}, or at least the nonce η_C contained in it, then $\pi^i_{U^*}$ and π^j_{AP} will derive the same PTK since the they will input the same nonces η_C, η_{AP} to $\$(\cdot)$. Thus, in order for the simulation to be consistent, $\mathcal{F}$ needs to embed the verification oracle $\mathcal{O}^{\mathsf{Vrfy}}$ into π^j_{AP}'s verification of this m_2 message. If the verification fails, then $\mathcal{F}$ discards the message and continues the simulation. Else, $\mathcal{F}$ aborts with a failure. The procedure labeled SimC in Figure 6.2 on Page 129 makes this high-level description precise.

$\mathsf{SimC}(\pi^i_{U^*})$:

100: Initialization:
101: $\eta^*_C, \eta^*_{AP} \leftarrow \perp$
102: $\pi^*_{AP} \leftarrow \perp$

200: On receiving m_1:
201: parse m_1 as (η_{AP}, p_1)
202: $\eta^*_{AP} \leftarrow \eta_{AP}$
203: **if** session π^j_{AP} created η_{AP}**:**
204: $\pi^*_{AP} \leftarrow \pi^j_{AP}$
205: $\eta^*_C \twoheadleftarrow \{0,1\}^\lambda$
206: $m_2 \leftarrow \mathcal{O}^{\mathsf{MAC}}(\eta^*_C \| p_2)$
207: send m_2

300: On receiving an m_3 message:
301: parse m_3 as $\eta_{AP} \| p_3 \| \tau$
302: **if** $\eta_{AP} \neq \eta^*_{AP}$**:**
303: discard m_3
304: continue simulation
305: **else**
306: $d \leftarrow \mathcal{O}^{\mathsf{Vrfy}}(\eta_{AP} \| p_3, \tau)$
307: **if** $d = 1$**:**
308: stop simulation
309: **else**
310: discard m_3
311: continue simulation

400: On forwarding η^*_C to π^*_{AP}:
401: parse m_2 attempt as $\eta^*_C \| p_2 \| \tau$
402: $d \leftarrow \mathcal{O}^{\mathsf{Vrfy}}(\eta^*_C \| p_2, \tau)$
403: **if** $d = 1$**:**
404: abort with failure
405: **else**
406: discard m_2
407: continue simulation

$\mathsf{SimAP}(\pi^i_{U^*})$:

100: Initialization:
101: $\eta^*_{AP} \leftarrow \perp$
102: $q^* \twoheadleftarrow [1, q]$
103: $\mathsf{distinct} \leftarrow 0$
104: $\vec{\mathsf{N}} \leftarrow \emptyset$
105: $\mathsf{Fwd} \leftarrow \emptyset$

200: Creating m_1:
201: $\eta^*_{AP} \twoheadleftarrow \{0,1\}^\lambda$
202: $m_1 \leftarrow \eta^*_{AP} \| p_1$
203: send m_1

300: On receiving an m_2 message:
301: parse m_2 attempt as $\eta_C \| p_2 \| \tau$
302:
303: **if** $\eta_C \notin \vec{\mathsf{N}}$**:**
304: $\mathsf{distinct} \leftarrow \mathsf{distinct} + 1$
305: $\vec{\mathsf{N}}[\mathsf{distinct}] \leftarrow \eta_C$
306:
307: $d \leftarrow 0$
308: **if** $(\eta_C = \vec{\mathsf{N}}[q^*]) \wedge (\eta_C \notin \mathsf{Fwd})$**:**
309: $d \leftarrow \mathcal{O}^{\mathsf{Vrfy}}(\eta_C \| p_2, \tau)$
310: **else**
311: $k_\mu \| k_\alpha \leftarrow \$(AP, C, \eta^*_{AP}, \eta_C)$
312: $d \leftarrow \Sigma.\mathsf{Vrfy}(k_\mu, \eta_C \| p_2, \tau)$
313:
314: **if** $d = 1$**:**
315: stop simulation
316: **else**
317: discard m_2
318: continue simulation

400: On forwarding η^*_{AP} to any π^j_C:
401: create π^j_C's response message
402: $m_2 \leftarrow \eta_C \| p_2 \| \tau$ as normal
403: $\mathsf{Fwd} \leftarrow \mathsf{Fwd} \cup \{\eta_C\}$

Figure 6.2: Description of $\mathcal{F}$'s simulation in the proof of Claim 6.5. The simulation depends on whether $U^* = C$ (shown in SimC) or $U^* = AP$ (shown in SimAP). The random function that $\mathcal{F}$ implements for key derivation between AP and C is denoted by $\$(\cdot)$. In both SimC and SimAP it is assumed that if the parsing of a received message fails, then the message is silently dropped and the simulation continues; for simplicity this behavior is omitted from the code.

Algorithm $\mathcal{F}$'s simulation of Game 4 is perfect. We argue that if $\pi^i_{U^*}$ accepted maliciously, then $\mathcal{F}$ must also have made a valid forgery in its SUF-CMA game. By definition, for $\pi^i_{U^*}$ to have accepted maliciously it must have received an m_3 message that verified correctly, and no session at AP can have set the same sid as $\pi^i_{U^*}$. Since $\mathcal{F}$ uses its $\mathcal{O}^{\mathsf{Vrfy}}$ oracle to verify m_3 messages (Line 306 in the SimC procedure), and because $\mathcal{F}$ never asks for a tag on an m_3 message to its $\mathcal{O}^{\mathsf{MAC}}$ oracle (since $p_2 \neq p_3$); this implies that whenever $\pi^i_{U^*}$ accepts maliciously, then $\mathcal{F}$ creates a valid forgery in its SUF-CMA game (Line 308 in SimC).

Note that $\mathcal{F}$ aborts with failure at Line 404 in SimC only if $\mathcal{A}$ failed. That is, $\mathcal{F}$ embeds its $\mathcal{O}^{\mathsf{Vrfy}}$ oracle also when verifying m_2 messages delivered to $\pi^*_{AP} = \pi^j_{AP}$ that contains $\pi^i_{U^*}$'s nonce η^*_C (Line 402 in the SimC procedure). However, if such a verification were to succeed, then π^j_{AP} would accept and store the nonces η^*_C, η^*_{AP} in $\pi^j_{AP}.\mathsf{sid}$. But this would yield the same sid as that of $\pi^i_{U^*}$. So if $\pi^i_{U^*}$ were to accept on receiving an m_3 message, $\pi^i_{U^*}$ and π^j_{AP} would be partners, contradicting the fact that $\pi^i_{U^*}$ was supposed to accept maliciously.

Case $U^* = AP$. Now suppose $\pi^i_{U^*}$ belongs to the access point AP. For $\pi^i_{U^*}$ to accept maliciously, it must receive some m_2 message that verifies correctly. Again, $\mathcal{F}$ will embed its MAC oracles into some of $\pi^i_{U^*}$'s computations, in particular the $\mathcal{O}^{\mathsf{Vrfy}}$ oracle. However, $\mathcal{F}$ cannot verify *every* m_2 message with $\mathcal{O}^{\mathsf{Vrfy}}$. The problem is that $\mathcal{A}$ might forward $\pi^i_{U^*}$'s initial message, containing the nonce η_{AP}, to *multiple* sessions at C. Since these sessions will all generate their own unique nonces η_C, they will also derive distinct[2] PTKs which they use to create their m_2 messages. But the MAC oracles only represent a *single* key k_α, so it would not be correct to embed the $\mathcal{O}^{\mathsf{Vrfy}}$ oracle to verify *all* these m_2 messages. Moreover, some m_2 messages may not even have been created by sessions at client C at all because $\mathcal{A}$ could have forged the nonces itself or taken them from sessions at other clients. So which m_2 messages should be verified with $\mathcal{O}^{\mathsf{Vrfy}}$?

Since each nonce η_C combined with $\pi^i_{U^*}$'s own nonce η_{AP} determines a single PTK, $\mathcal{F}$ must guess one nonce and use its $\mathcal{O}^{\mathsf{Vrfy}}$ oracle to verify all m_2 messages that contain this nonce. For all other m_2 messages, $\mathcal{F}$ will instead derive a PTK using $\$(\cdot)$, and use the MAC algorithm $\Sigma.\mathsf{Vrfy}$ to "locally" verify the message without calling $\mathcal{O}^{\mathsf{Vrfy}}$. This strategy is described in the SimAP procedure shown in Figure 6.2.

In SimAP, the value q represents the maximum number of *unique* nonces that $\mathcal{A}$ will ever send to an access point session. It is upper bounded by the

[2] At least with high probability.

number of Send queries made by $\mathcal{A}$. Algorithm $\mathcal{F}$ makes a guess $q^* \twoheadleftarrow [1, q]$ and hopes that m_2 messages that contain the q^*-th unique nonce will lead $\pi^i_{U^*}$ to accept maliciously. We emphasize that this does not mean that $\pi^i_{U^*}$ must necessarily accept maliciously after receiving the q^*-th m_2 message in total (since $\mathcal{A}$ could make repeated attempts with some of the earlier nonces first); nor does it mean that $\pi^i_{U^*}$ must necessarily accept maliciously after receiving an m_2 message containing the q^*-th unique nonce for the *first* time (since $\mathcal{A}$ can make many m_2 attempts with this particular nonce).

The counter $\mathsf{distinct}$ is used to keep track of how many unique nonces $\pi^i_{U^*}$ have received so far. The array $\vec{\mathsf{N}}$ stores all the distinct nonces. In particular, the nonce in $\vec{\mathsf{N}}[q^*]$ is the one for which $\mathcal{F}$ will embed the $\mathcal{O}^{\mathsf{Vrfy}}$ oracle. Additionally, $\mathcal{F}$ maintains a list Fwd which is used to record situations where $\pi^i_{U^*}$ cannot accept malicously (discussed below). The variable η^*_{AP} stores the nonce created by $\pi^i_{U^*}$.

We first argue that $\mathcal{F}$ perfectly simulates Game 4. Looking at the SimAP procedure, it is clear that we only need to focus on the verification of m_2 messages. If $\pi^i_{U^*}$ receives an m_2 message which contains a nonce η_C which is different from the q^*-th nonce $\vec{\mathsf{N}}[q^*]$, or if η_C has been forwarded from a session at C which first received $\pi^i_{U^*}$'s nonce η^*_{AP} (meaning $\eta_C \in \mathsf{Fwd}$), then $\mathcal{F}$ derives the PTK itself and verifies with the MAC algorithm $\Sigma.\mathsf{Vrfy}$ (Line 311 and Line 312 in SimAP). This gives a correct simulation.

For the remaining m_2 messages, η_C is equal to the q^*-th unique nonce and $\eta_C \notin \mathsf{Fwd}$, so $\mathcal{F}$ embeds its $\mathcal{O}^{\mathsf{Vrfy}}$ oracle (Line 309). The condition $\eta_C \notin \mathsf{Fwd}$ implies that no session at C have input both η^*_{AP} and η_C to $\$(\cdot)$. This implies that the MAC keys used by the sessions at C are independent from the MAC key (if any) used to produce these specific m_2 messages. Consequently, using oracle $\mathcal{O}^{\mathsf{Vrfy}}$ to verify these m_2 messages lead to answers that are identically distributed to those one would get if $\mathcal{F}$ derived PTK from $\$(\cdot)$ itself and verified "locally" with $\Sigma.\mathsf{Vrfy}$.

It remains to analyze $\mathcal{F}$'s probability of making a valid forgery in its SUF-CMA game whenever $\pi^i_{U^*}$ accepts maliciously in Game 4. For $\pi^i_{U^*}$ to have accepted maliciously it must have successfully verified an m_2 message, so at some point we must have had $d = 1$ in SimAP (Line 315). Moreover, by the same arguments as for the "abort with failure" condition in the SimC procedure, the nonce η_C which $\pi^i_{U^*}$ accepted on cannot have been produced by a session π^j_C which received $\pi^i_{U^*}$'s nonce η^*_{AP}. In other words, we must have $\eta_C \notin \mathsf{Fwd}$.

Thus, if $\mathcal{F}$'s guess of q^* was correct, $\mathcal{F}$ will have used the oracle $\mathcal{O}^{\mathsf{Vrfy}}$ to verify the $m_2 = \eta_C \| p_2 \| \tau$ message on which $\pi^i_{U^*}$ accepted maliciously. Furthermore, since $\mathcal{F}$ never makes a query to its tagging oracle $\mathcal{O}^{\mathsf{MAC}}$, if it happens that $\mathcal{O}^{\mathsf{Vrfy}}(\eta_C \| p_2, \tau) = 1$ then this must necessarily be a valid forgery in the SUF-CMA experiment. On the other hand, if $\mathcal{F}$'s guess of q^* was wrong, then

it will not have used $\mathcal{O}^{\mathsf{Vrfy}}$ to calculate d, in which case it clearly does not win in its SUF-CMA game.

Since $\mathcal{F}$'s simulation of Game 4 is perfect, and since q^* was drawn independently of $\mathcal{A}$, $\mathcal{F}$'s guess was correct with probability q^{-1}. Hence $\mathcal{F}$ winning probability in its SUF-CMA experiment is at least q^{-1} times $\mathcal{A}$'s winning probability in Game 4.

Concluding the proof of Claim 6.5. Up to a factor of q^{-1}, we see that $\mathcal{F}$ successfully forges whenever $\pi^i_{U^*}$ accepts maliciously in Game 4 regardless of whether $U^* = C$ or $U^* = AP$. This proves Claim 6.5. ■

Concluding the proof of Theorem 6.4. Combining all the bounds from Game 0 to Game 4 with Claim 6.5 yields Theorem 6.4. ■

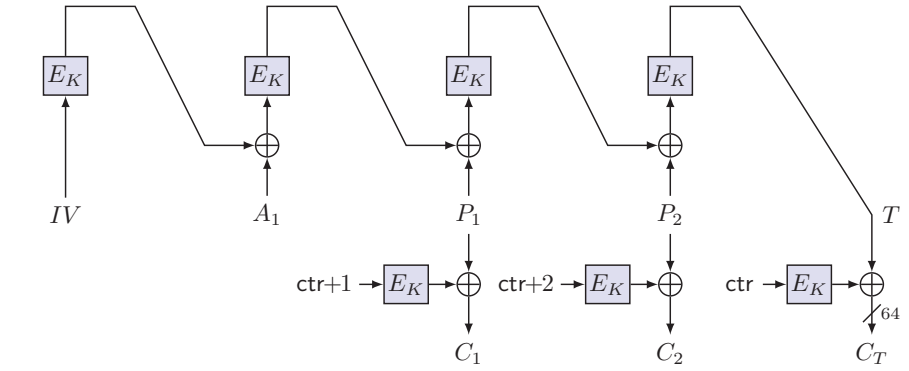
6.2.4 Security of IEEE 802.11 with upper-layer authentication

Theorem 6.1 and Theorem 6.4 apply to the WPA2-PSK variant of IEEE 802.11. To also address the security of IEEE 802.11 in its WPA2-Enterprise variant, we need to analyze the setting where the PMK is provided by some upper-layer authentication protocol. Technically, IEEE 802.11 can be combined with any type of upper-layer authentication protocol, but in practice the *de facto* standard is EAP. Consequently, our second composition result from Chapter 4 can immediately be applied to obtain a result on IEEE 802.11 in its WPA2-Enterprise variant as well.

More precisely, by setting Π_3 = basic EAP and Π_4 = 4WHS in Theorem 4.12, we get that the combination $\Pi_5 = \text{EAP} + \text{4WHS}$ is a secure 3P-AKE protocol in our strongest security model AKE^{fs}. However, technically speaking, in order to apply Theorem 4.12 we also need to show that the probability that two sessions end up with the same local transcript in the 4WHS protocol is small. Fortunately, this is trivial since each side in the 4WHS protocol creates a random 256 bit nonce. In detail, the function ϵ required by Theorem 4.12 is in the 4WHS bounded by the probability of a nonce collision among the sessions at a specific party, namely

$$\epsilon \leq \frac{|\mathcal{I} \cup \mathcal{R}| \cdot n_\pi^2}{2^\lambda} = \frac{n_P \cdot n_\pi^2}{2^{256}}. \tag{6.13}$$

Note that the bound is proportional to $n_P \cdot n_\pi^2$ and not $(n_P \cdot n_\pi)^2$, since the collision needs to happen at a specific party.



$IV = \mathsf{flags}_1||N||\mathsf{length}_{16}(A + P)$

$\mathsf{ctr} = \mathsf{flags}_2||N||0^{16}$

Figure 6.3: The CCM mode of operation.

6.3 Analyzing CCMP

While this chapter is primarily about the AKE security of the 4WHS key exchange protocol, for completeness we also include an analysis of the CCMP algorithm used to protect the IEEE 802.11 application data. CCMP is a stateful authenticated encryption (stAE) scheme built out of the CCM mode of operation [RFC3610] using AES as its underlying block cipher. Since CCMP is only defined within the context of IEEE 802.11, we specialize all of our descriptions to this setting, including that of CCM.

6.3.1 Description of CCMP

An IEEE 802.11 frame consists of a *header* $A = A_1\|A_2\|\cdots\|A_r$ which will be integrity protected but not encrypted, and a *plaintext* message $P = P_1\|P_2\|\cdots\|P_s$ which will be both integrity protected and encrypted. Each block of A and P is 128 bits, except possibly for the final blocks A_r, P_s which might be shorter.

CCM. The CCM mode of operation is shown in Figure 6.3 with one header block A_1 and two plaintext blocks $P_1\|P_2$. CCM combines a CBC-MAC with CTR mode encryption in the style of MAC-then-encrypt, and can be summarized as follows. On input a key K, a message $A\|P$, and a 104 bit nonce N; CCM first derives the initial value IV needed by CBC-MAC, and the initial counter value ctr needed by CTR mode, from the nonce N and two distinct 8 bit

CCMP.Enc(K, P, A):

1: $\mathsf{sent} \leftarrow \mathsf{sent} + 1$
2: $\widehat{U} \leftarrow \mathsf{Address}(A)$
3: $N \leftarrow \mathsf{flags} \| \widehat{U} \| \mathsf{sent}$
4:
5: $C \leftarrow \mathsf{CCM.Enc}(K, N, P, A)$
6:
7: **return** (sent, C)

CCMP.Dec$(K, \mathsf{sent} \| C, A)$:

1: $\widehat{U} \leftarrow \mathsf{Address}(A)$
2: $N \leftarrow \mathsf{flags} \| \widehat{U} \| \mathsf{sent}$
3:
4: $P \leftarrow \mathsf{CCM.Dec}(K, N, C, A)$
5:
6: **if** $P = \perp$**:**
7: **return** $\perp$
8:
9: **if** $\mathsf{sent} \leq \mathsf{rcvd}$**:**
10: **return** $\perp$
11:
12: $\mathsf{rcvd} \leftarrow \mathsf{sent}$
13:
14: **return** P

Figure 6.4: The CCMP encryption and decryption procedures.

flags flags_1 and flags_2. Then, CBC-MAC is computed over the whole message $A \| P$ to produce a tag T. Next, the plaintext message P is encrypted using CTR mode to produce a ciphertext C_P. Finally, the tag T is encrypted with a single counter block to produce a ciphertext C_T. The combination $C \leftarrow C_P \| C_T$ is the output of CCM. Decryption works in the obvious manner.

CCMP. The CCMP encrypt and decrypt procedures are shown in Figure 6.4. The two main responsibilities of CCMP are to create the nonce N that will be used as input to CCM, and to ensure replay protection for the IEEE 802.11 frames. CCMP achieves both by maintaining a 48 bit counter sent, which is incremented for each sent IEEE 802.11 frame; and a 48 bit counter rcvd, which is (potentially) updated for each received IEEE 802.11 frame.[3] The sent counter is initialized to 1 and the rcvd counter is initialized to 0.

In order to encrypt an IEEE 802.11 frame consisting of a header A and a plaintext P, CCMP first increments the sent counter and creates the 104 bit nonce N as

$$N \leftarrow \mathsf{flags} \| \widehat{U} \| \mathsf{sent}, \tag{6.14}$$

where $\widehat{U}$ is the 48 bit link-layer address of the sender, and flags is an 8 bit value

[3]The sent counter is called the *packet number* in the IEEE 802.11 standard [IEEE 802.11], while the rcvd counter is called the *replay counter*.

encoding various IEEE 802.11 settings. We treat it as a constant. The link-layer address $\widehat{U}$ is always part of the header A, so in Figure 6.4 we use a function $\mathsf{Address}$ to indicate the process of extracting $\widehat{U}$ from A. Given the nonce N, CCMP then encrypts the IEEE 802.11 frame consisting of header data A and plaintext P using the CCM mode of operation to produce the ciphertext C. The output of CCMP is the concatenation $\mathsf{sent}\|C$.

Remember that CCM will add additional elements to the nonce N in order to create the CBC-MAC IV and the CTR mode initial counter as indicated in Figure 6.3. Particularly, the IV and initial counter for CCM when used in the context of CCMP are the following two 128 bit values.

$$IV \leftarrow \mathsf{flags}_1\|\mathsf{flags}\|\widehat{U}\|\mathsf{sent}\|\mathsf{length}_{16}(A+P) \tag{6.15}$$

$$\mathsf{ctr} \leftarrow \mathsf{flags}_2\|\mathsf{flags}\|\widehat{U}\|\mathsf{sent}\|0^{16} \tag{6.16}$$

Here, $\mathsf{length}_{16}(A+P)$ is the length of A plus P in bytes, encoded as 16 bits. Note that 16 bits is sufficient to accommodate the length of the maximum size IEEE 802.11 frame.

To decrypt an IEEE 802.11 frame $Z = (\mathsf{sent}\|C, A)$, CCMP first recreates the nonce N from A and sent, and then decrypts C with CCM. If the decryption fails, then CCMP outputs $\perp$. Else, it checks that the value sent contained in Z is strictly greater than the internally maintained rcvd counter. If not, then CCMP outputs $\perp$ again. Otherwise, it updates the value of rcvd to match that of the received sent, and returns the plaintext P.

6.3.2 Analysis of CCMP

Jonsson [Jon03] has shown that the CCM mode of operation is a secure authenticated encryption scheme. He proved that CCM satisfies the two separate security notions of indistinguishability under chosen-plaintext attacks (IND-CPA) and integrity of ciphertexts (INT-CTXT); see [BN00] for their formal definitions. In Appendix A.3, we show that this is equivalent to our all-in-one definition of AE security. Thus, in order to prove the stateful AE security of CCMP, it is sufficient to reduce to the (stateless) AE security of CCM.

However, on closer inspection, we cannot, in fact, prove that CCMP is a secure stAE scheme according to Definition A.5. The reason is that the security experiment used to define stAE security in Appendix A.4, targets a different *integrity semantic* than what is provided by CCMP. Namely, the security experiment in Figure A.4 is adapted from [Jag+12], which presented the definition in the context of analyzing TLS. But the integrity guarantees provided by the TLS Record Layer protocol are stronger than those provided by CCMP.

Specifically, in terms of integrity, the TLS Record Layer is supposed to provide protection against:

1. forgeries,
2. replays,
3. reordering, and
4. dropping of messages.

However, CCMP does *not* provide protection against messages being dropped. To see this, suppose a sender transmits as its first encrypted CCMP messages, the frames Z_1, Z_2 and Z_3, thus having corresponding packet numbers 2, 3 and 4 (remember that the sent counter starts at 1). Now suppose an attacker drops messages Z_1 and Z_2, but delivers Z_3 to the receiver. Since Z_3 is a validly created CCMP frame, the CCM decryption at Line 4 of the $\mathsf{CCMP.Dec}$ procedure will succeed. Moreover, when the $\mathsf{CCMP.Dec}$ procedure continues to check whether Z_3 is a replay at Line 9, then this will also succeed, since the receiver's rcvd counter is set to 0 and so we have $\mathsf{rcvd} < \mathsf{sent}$. Thus, Z_3 will be accepted by the receiver even though Z_1 and Z_2 were lost.

By contrast, the TLS Record Layer demands that the decryption process happens in exactly the same order as the ciphertexts were created by the encryption process. Thus, it does not accept any messages being dropped. Other integrity semantics are also possible by considering different combinations of the four properties listed above. Boyd et al. [Boy+16] have analyzed this question in depth.

To summarize, CCMP cannot be proven secure according to the stAE security definition given in Appendix A.4 because it implies a *stronger* integrity semantics than what CCMP achieves. Consequently, we have to consider a *weakening* of the stAE model that allows for messages to be dropped. Particularly, at Line 7 of the Dec oracle in Figure A.4, we change the condition from

$$\textbf{if } (C, A) \neq S[\mathsf{rcvd}]\textbf{:} \quad \texttt{in-sync} \leftarrow \texttt{false} \qquad \text{to} \qquad \textbf{if } (C, A) \notin S[\mathsf{rcvd} \ldots \mathsf{sent}]\textbf{:} \quad \texttt{in-sync} \leftarrow \texttt{false}$$

where $S[i \ldots j] = \{S[i], S[i+1], \ldots, S[j]\}$ if $i \leq j$, and $\emptyset$ otherwise. Notice that this widens the scope of which messages are considered in-sync, from one message $S[\mathsf{rcvd}]$, to potentially a large range $S[\mathsf{rcvd} \ldots \mathsf{sent}]$. Hence, this effectively weakens the model since it restricts the adversary more.

Let $\mathbf{Adv}^{\mathsf{stAE\text{-}d}}_{\Pi}(\mathcal{A})$ denote the stAE advantage of an adversary $\mathcal{A}$ against some stAE scheme Π within this weakened model. We then have the following result.

Theorem 6.6. *Let $\mathcal{A}$ be an adversary against the stAE security of CCMP, then we can create an adversary $\mathcal{B}$ against the AE security of CCM, such that*

$$\mathbf{Adv}^{\mathsf{stAE\text{-}d}}_{\mathsf{CCMP}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{AE}}_{\mathsf{CCM}}(\mathcal{B}). \tag{6.17}$$

Proof. From an adversary $\mathcal{A}$ that breaks the stAE security of CCMP where message drops are allowed, we can construct an algorithm $\mathcal{B}$ that breaks the CCM mode of operation. Specifically, algorithm $\mathcal{B}$ creates and maintains the counters sent and rcvd of the CCMP scheme, as well as the variables sent, rcvd, $S[\cdot]$, and $\texttt{in-sync}$ of security game $\mathbf{Exp}^{\mathsf{stAE\text{-}d}}_{\mathsf{CCMP}}(\mathcal{A})$. We write $\mathsf{sent}_{\mathsf{CCMP}}$, $\mathsf{rcvd}_{\mathsf{CCMP}}$ for the counters that $\mathcal{B}$ maintains for the CCMP scheme, and $\mathsf{sent}_{\mathbf{Exp}}$, $\mathsf{rcvd}_{\mathbf{Exp}}$ for the counters that $\mathcal{B}$ maintains for the stAE experiment.

When $\mathcal{A}$ makes an encryption query (M_0, M_1, A), $\mathcal{B}$ first increments $\mathsf{sent}_{\mathsf{CCMP}}$ and creates a nonce N from $\mathsf{sent}_{\mathsf{CCMP}}$ according Equation (6.14). It then queries the Enc oracle in its own AE security game on (N, M_0, M_1, A). $\mathcal{B}$ returns the resulting ciphertext C together with $\mathsf{sent}_{\mathsf{CCMP}}$ to $\mathcal{A}$, and updates the variables $\mathsf{sent}_{\mathbf{Exp}}$ and $S[\mathsf{sent}_{\mathbf{Exp}}]$ accordingly.

When $\mathcal{A}$ makes a decryption query $(\mathsf{sent}'\|C, A)$, $\mathcal{B}$ first increments the value of the counter $\mathsf{rcvd}_{\mathbf{Exp}}$ by 1 and then proceeds as follows.

- If $(\mathsf{sent}'\|C, A) \in S[\mathsf{rcvd}_{\mathbf{Exp}} \ldots \mathsf{sent}']$, then $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$ (since this query is in-sync).

- Else, $\mathcal{B}$ creates the nonce $N \leftarrow \mathsf{flags}\|\widehat{U}\|\mathsf{sent}'$, and queries (N, C, A) to the Dec oracle in its own AE security game to produce a message M. If $M \neq \perp$ then $\mathcal{B}$ stops its simulation and outputs 1 to its AE security game. Otherwise, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$.

Finally, when $\mathcal{A}$ stops with some output b', then $\mathcal{B}$ stops and outputs the same bit to its AE security game.

Notice that when the secret bit in $\mathcal{B}$'s own AE security game is 0, then $\mathcal{B}$ perfectly simulates the "left-world" of experiment $\mathbf{Exp}^{\mathsf{stAE\text{-}d}}_{\mathsf{CCMP}}(\mathcal{A})$, i.e., where the encryption query returns the encryption of M_0 and the decryption query always returns $\perp$. This is because in this scenario $\mathcal{B}$'s own AE decryption oracle always returns $\perp$, and so $\mathcal{B}$ answers all of $\mathcal{A}$'s decryption queries with $\perp$ too. Moreover, $\mathcal{B}$'s simulation of $\mathcal{A}$'s encryption queries is perfect no matter what the value of the secret bit in its own AE security game is. This is because $\mathcal{B}$ properly creates the nonce N from the counter $\mathsf{sent}_{\mathsf{CCMP}}$, which it creates and maintains itself.

Thus, it only remains to argue that $\mathcal{B}$ perfectly simulates the decryption query of the "right-world" of experiment $\mathbf{Exp}^{\mathsf{stAE\text{-}d}}_{\mathsf{CCMP}}(\mathcal{A})$ when the secret bit in $\mathcal{B}$'s own AE security game is 1. To this end, it is sufficient to note that $\mathcal{B}$ forwards the decryption query to its own AE security game exactly when the query is out-of-sync according to the requirements of game $\mathbf{Exp}^{\mathsf{stAE\text{-}d}}_{\mathsf{CCMP}}(\mathcal{A})$. This is so because $\mathcal{B}$ itself has created and maintains the counter $\mathsf{rcvd}_{\mathbf{Exp}}$ in accordance with $\mathbf{Exp}^{\mathsf{stAE\text{-}d}}_{\mathsf{CCMP}}(\mathcal{A})$. Moreover, if $\mathcal{B}$'s own decryption oracle returns something other than $\perp$, which for instance could happen if $\mathcal{A}$ replays an old

ciphertext but with a new counter sent' such that $\mathsf{rcvd}_{\mathbf{Exp}} < \mathsf{sent}'$, then $\mathcal{B}$ stops and immediately wins in its AE security game.[4]

To summarize: $\mathcal{B}$ perfectly simulates the $\mathbf{Exp}^{\mathsf{stAE\text{-}d}}_{\mathsf{CCMP}}(\mathcal{A})$ game for $\mathcal{A}$, and wins in its own AE security against CCM with at least the same probability as $\mathcal{A}$. ■

Since CCMP maintains a 48 bit counter sent in order to create the nonces for CCM, it can technically be used to encrypt up to 2^{48} IEEE 802.11 frames. The maximum allowable IEEE 802.11 frame size is around 2^{13} bytes = 8 kB, so a total of 2^{48+13} bytes can be encrypted under the same key. However, the CCM security bound proven by Jonsson [Jon03] includes a "birthday bound" term of the form $c \cdot \ell^2 \cdot 2^{-\kappa_b}$, where c is a small constant, ℓ is the number of invocations of the underlying block cipher, and κ_b is the block length. In CCMP the block cipher is AES, so $\kappa_b = 128$. Moreover, since CCM makes roughly 2 block cipher calls per input block, this implies that an IEEE 802.11 frame of B bytes will involve around $2B/16 = B/8$ block cipher calls. If we set $c = 1$, then in order for $\ell^2 \cdot 2^{-\kappa_b}$ to stay below ϵ, we need $(B/8)^2 \cdot 2^{-128} \leq \epsilon$. In other words, no more than $2^{67} \cdot B^{-1} \cdot \epsilon^{1/2}$ frames of B bytes can be encrypted under the same key.

For example, if we set the frame size to be $B = 2^{10}$ bytes = 1 kB, and we want $\epsilon < 2^{-60}$, then Jonsson's bound only justifies up to 2^{27} IEEE 802.11 frames being encrypted with the same key, or roughly $2^{37} \approx 137$ GB of data. Alternatively, if the requirement is reduced to $\epsilon < 2^{-50}$, then we get the more tolerable bound of ≈ 4 TB of data; while if the requirement is increased to $\epsilon < 2^{-70}$, we get virtually no guarantees (≈ 4 GB of data).

6.4 Multi-ciphersuite and negotiation security of IEEE 802.11

In Section 6.2 we analyzed the 4WHS protocol under the assumption that there is only a single version of it. However, as explained in Section 2.2.3, IEEE 802.11 actually supports several different *ciphersuites*, leading to slightly different instances of the 4WHS protocol. A ciphersuite in IEEE 802.11 essentially determines five things:

- Whether to use upper-layer authentication or not.

[4] Note that aborting early is a tiny optimization which is not strictly necessary in order to bound $\mathcal{A}$'s winning probability in terms of $\mathcal{B}$'s. Without it, $\mathcal{B}$ would additionally have to maintain the $\mathsf{rcvd}_{\mathsf{CCMP}}$ counter and properly check sent' against $\mathsf{rcvd}_{\mathsf{CCMP}}$ before answering $\mathcal{A}$. The benefit of the optimization is that it makes the description of $\mathcal{B}$ simpler, and can only strictly increase $\mathcal{B}$'s winning chances.

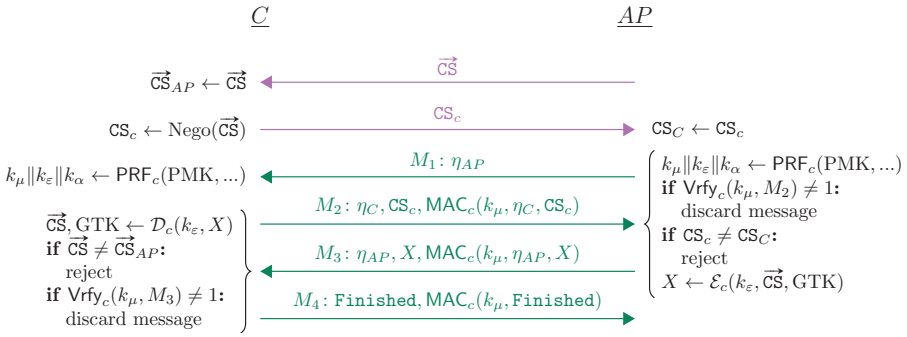


Figure 6.5: Ciphersuite negotiation in the 4WHS protocol.

- Which KDF to use inside the 4WHS. There are two possible options: a PRF based on HMAC-SHA1 and a PRF based on HMAC-SHA256.
- Which MAC algorithm to use inside the 4WHS. There are three possible options: HMAC-MD5 [RFC2104] (deprecated), HMAC-SHA1-128 [RFC2104], and AES-CMAC-128 [FIPS:SP-800-38B].
- Which encryption algorithm to use inside the 4WHS. There are two possible options: RC4 (deprecated) and AES Key Wrap [RFC3394].
- Which encryption algorithm to use for the IEEE 802.11 application data. There are two possible options: TKIP (deprecated) and CCMP.

Although in principle this gives a total number of $2^4 \cdot 3 = 48$ ciphersuites, the actual number is smaller since some options need to be used together. The ciphersuite to use is determined by a *negotiation protocol* run prior to the 4WHS. In particular, recall from Section 2.2.3 that during the association phase of the IEEE 802.11 protocol, the access point will send the client a list of supported ciphersuites $\overrightarrow{\mathtt{CS}}$. From this list the client will chose a single preferred ciphersuite $\mathtt{CS}_c$ based on some negotiation function Nego. Leaving out the irrelevant messages from Figure 2.2, this gives the simplified protocol shown in Figure 6.5. Compared to Figure 6.1, we have made the values of the constants $p_1, \ldots, p_4$ explicit in order to reflect the presence of ciphersuite-specific information in the 4WHS.

In particular, in Figure 6.5 the chosen ciphersuite $\mathtt{CS}_c$ determines a specific KDF PRF_c, a MAC algorithm $\Sigma_c = (\mathsf{MAC}_c, \mathsf{Vrfy}_c)$, a handshake encryption algorithm $\mathcal{E}_c = (\mathcal{E}_c, \mathcal{D}_c)$, and an application data encryption algorithm $\Lambda_c = (\mathsf{Init}_c, \mathsf{Enc}_c, \mathsf{Dec}_c)$ (not shown). Both the client and the access points repeat their respective messages from the negotiation protocol inside the 4WHS

(integrity protected by the MAC algorithm Σ_c). If the received values $\mathtt{CS}_c$ and $\overrightarrow{\mathtt{CS}}$ are not identical to what the client and access point received during the negotiation protocol, they abort the 4WHS. Note that the access point encrypts its ciphersuite list $\overrightarrow{\mathtt{CS}}$ with the encryption algorithm $\mathcal{E}_c$ in the third message of the 4WHS (which also includes a group key GTK).

Given that IEEE 802.11 supports multiple ciphersuites, there are two related questions to ask. The first is whether the 4WHS protocol is *multi-ciphersuite secure*, meaning that it is still secure as an AKE protocol when multiple different ciphersuites can be run in parallel. The second is whether the negotiation protocol in combination with the 4WHS achieves *negotiation security*, meaning that the client and access point end up with the ciphersuite prescribed by their initial configurations.

In the sections below we examine both of these issues and indicate how one could prove the multi-ciphersuite and negotiation security of IEEE 802.11. We stress that unlike the other sections of this chapter, we do not provide any formal theorem statements or proofs, but only keep the discussion at an informal level.

6.4.1 Multi-ciphersuite security

If a multi-ciphersuite protocol uses different long-term keys for each ciphersuite, then the multi-ciphersuite security of the protocol follows trivially from the security of the individual ciphersuites. However, if the same long-term key is used across different ciphersuites, then this does not necessarily have to be the case. Chatterjee et al. [CMU09] give several examples of attacks that are possible when long-term keys are reused across different protocols. Since the same PSK can be reused across different ciphersuites in IEEE 802.11, we cannot automatically conclude that the 4WHS achieves multi-ciphersuite security.

Generic composition. Seemingly, a possible approach to proving the multi-ciphersuite security of the 4WHS protocol would be to adapt the multi-ciphersuite framework of Bergsma et al. (BDKSS) [Ber+14]. In their framework a multi-ciphersuite protocol $\mathtt{NP}\|\overrightarrow{\Pi}$ is the composition of a negotiation protocol $\mathtt{NP}$, followed by a sub-protocol $\Pi_c \in \overrightarrow{\Pi}$, corresponding to specific ciphersuite c. Importantly, BDKSS do not demand that a different long-term key be used for each sub-protocol Π_c. One of the central results of BDKSS is a generic composition theorem that tries to recover as much as possible of the intuition that the security of $\mathtt{NP}\|\overrightarrow{\Pi}$ should follow from the security of the individual sub-protocols.

The main issue in reducing the security of $\mathtt{NP}\|\overrightarrow{\Pi}$ to the security of a sub-protocol Π_c, is that the reduction also needs to be able to simulate the *other*

sub-protocols Π_d that use the same long-term key as Π_c. BDKSS solve this problem by considering an extension to the single-ciphersuite security game where the adversary additionally gets access to an *auxiliary oracle* that runs a certain operation $\mathrm{Aux}(sk, \cdot)$ based on a private key sk.[5] For instance, if sk was the private key for some signature scheme, then $\mathrm{Aux}(sk, \cdot)$ could be a signing operation under sk. The idea of introducing the auxiliary oracle is that it can make it possible to simulate the other ciphersuites which uses the same private key sk. This is the central ingredient of the composition theorem of BDKSS: if the auxiliary oracle allows simulation of other ciphersuites that use the same private key as Π_c, then the multi-ciphersuite security of $\mathtt{NP}\|\vec{\Pi}$ can be reduced to the single-ciphersuite security of Π_c, in the extended auxiliary oracle model.

Still, this instead shifts the problem to showing that Π_c satisfies single-ciphersuite security in the possibly stronger auxiliary oracle model. In particular, if the operation $\mathrm{Aux}(sk, \cdot)$ is very liberal in terms of what function it computes with sk, then it will be easy to simulate the other ciphersuites, but at the cost of making it more difficult to show security of Π_c in the (single-ciphersuite) auxiliary oracle model. To counteract this, BDKSS additionally introduced the notion of a *constraint predicate* Φ. If the adversary is to win in the extended single-ciphersuite security game, then it cannot make an auxiliary oracle query that satisfies Φ. The stricter Φ is, the easier it becomes to prove the security of Π_c in the single-ciphersuite setting. On the other hand, a stricter Φ could also make it harder to simulate the other ciphersuites, which is needed by the composition theorem.

Let $\mathtt{NP}$ be the negotiation protocol in Figure 6.5 and let $\vec{\Pi}$ be the collection of all the different 4WHS ciphersuite variants. To prove the multi-ciphersuite security of $\mathtt{NP}\|\vec{\Pi}$ using the composition theorem of BDKSS [Ber+14], we have to come up with a suitable auxiliary oracle and constraint predicate Φ. Although BDKSS's original formulation was in the public-key setting, we can adapt it to the PSK setting in a straightforward manner. However, what is not so straightforward is constructing the PSK operation $\mathrm{Aux}(\mathrm{PMK}, \cdot)$ and the corresponding constraint predicate Φ. In order for an adversary against a ciphersuite Π_c to simulate another ciphersuite Π_d sharing the same PMK, it needs to be able to create the application keys k_α of Π_d. For simplicity, suppose ciphersuites c and d share the same KDF PRF. If we let $\mathrm{Aux}(\mathrm{PMK}, x)$ return $\mathsf{PRF}(\mathrm{PMK}, x)$ then we can certainly simulate protocol Π_d. The problem is that having access to this operation also makes it trivial to break protocol Π_c in the single-ciphersuite setting. Unfortunately, there does not seem to be a way to constrain the inputs to $\mathrm{Aux}(\mathrm{PMK}, \cdot)$ using Φ which simultaneously protects Π_c and at the same time enables simulation of sub-protocol Π_d. This

[5] Technically, BDKSS only considered multi-ciphersuite security for ACCE protocols, but their results can easily be adapted to other protocol types as well.

is because the chosen ciphersuite is not cryptographically bound to the derived keys through the KDF input (in the sense of channel binding). Hence there is no difference between how the key k_α is derived in Π_c and Π_d. In the end, we do not see how the composition theorem of BDKSS [Ber+14] can be used to prove the multi-ciphersuite security of IEEE 802.11.

Agile security. An alternative to using the modular composition theorem of BDKSS [Ber+14] is to prove the multi-ciphersuite security of IEEE 802.11 directly. Essentially, a proof of multi-ciphersuite AKE security or multi-ciphersuite explicit entity authentication would mostly mirror the corresponding single-ciphersuite proofs in Section 6.2.2 and Section 6.2.3, respectively. However, there is one major difference: the multi-ciphersuite proofs would have to rely on a so-called cryptographic *agility* assumption [Aca+10, Bha+14b]. Cryptographic agility refers to a setting where the same key is being used across multiple members of the same type of function, e.g., a PRF or a MAC.

In IEEE 802.11, the same PMK is used in two different PRFs: one based on HMAC-SHA1, and one based on HMAC-SHA256. A key agility assumption would then say that each of these schemes is secure (as a PRF), even when the adversary has oracle access to the other scheme under the same key. Additionally, it is also possible that the same MAC key could be used across the different MAC algorithms supported by IEEE 802.11, namely HMAC-MD5, HMAC-SHA1, and AES-CMAC. This could happen if an attacker in the negotiation protocol (see Figure 6.5) replaced the client's choice of ciphersuite $\texttt{CS}_c$ with another ciphersuite $\texttt{CS}'_c$ that selects a different MAC algorithm. Thus, we would also need an agility assumption on the MAC security of the collective $\{\text{HMAC-MD5}, \text{HMAC-SHA1}, \text{AES-CMAC}\}$.

In more detail, during a multi-ciphersuite security proof of the 4WHS protocol, the PRF agility assumption would be invoked in the game hop where we replace the KDF with a random function (Game 3 in both proofs), and the MAC agility assumption would be invoked in the analysis of the final game in the proof of explicit entity authentication (Claim 6.5 in Theorem 6.4). Except for the added assumptions of agility security, we expect the proofs to be straightforward extensions of those presented in Section 6.2.

Incidentally, the 4WHS protocol is quite similar to the PSK based variant of the IKEv1 [RFC2409] protocol in "aggressive" mode, which Bhargavan et al. [Bha+16] have conducted a multi-ciphersuite security analysis of.

6.4.2 Negotiation security

Intuitively, the goal of negotiation security is that no attacker should be able to get two parties to successfully agree upon a worse ciphersuite than the best

ciphersuite they mutually support. If an adversary succeeds in getting the parties to use a worse ciphersuite then what is prescribed by their mutual configurations, it is said to have performed a *downgrade* attack. As mentioned in Section 2.2.3, if WEP is supported alongside Robust Security Network (RSN) ciphersuites, then IEEE 802.11 cannot provide any protection against downgrade attacks. This is because when WEP is used, the 4WHS protocol is not run at all, and hence there is no way for the client and access point to verify that a downgrade attack has occurred. The IEEE 802.11 standard [IEEE 802.11] requires that WEP and RSN should not be enabled together. In the remainder we only discuss the negotiation security of IEEE 802.11 when using RSN ciphersuites exclusively.

Similar to the modular composition theorem of BDKSS [Ber+14] for the multi-ciphersuite security of a protocol $\mathtt{NP}\|\vec{\Pi}$, Dowling and Stebila [DS15] proposed a generic composition theorem that relates the negotiation security of $\mathtt{NP}\|\vec{\Pi}$ to the authentication security of the individual sub-protocols Π_c. However, their theorem assumes that different sub-protocols use different long-term keys, and so cannot be applied to IEEE 802.11.

In contrast, Bhargavan et al. [Bha+16] formulate negotiation security in the setting of key reuse across ciphersuites. They also prove a generic theorem that allows the negotiation security of $\mathtt{NP}\|\vec{\Pi}$ to be lifted from a simpler *core negotiation protocol* $\mathtt{NP}'$ extracted from $\mathtt{NP}$ and $\vec{\Pi}$. Thus, it is sufficient to focus on the negotiation security of protocol $\mathtt{NP}'$. Using their generic theorem, Bhargavan et al. [Bha+16] proved the negotiation security of the SSH [RFC4253] protocol under agile assumptions on its cryptographic primitives.

Admittedly, the value of applying the composition theorem of Bhargavan et al. [Bha+16] to IEEE 802.11 is rather limited, since the core negotiation protocol one can extract for IEEE 802.11 is almost the same as the whole protocol itself; essentially corresponding to the protocol we have shown in Figure 6.5. A proof of negotiation security for IEEE 802.11 would thus proceed in more or less the same way as the proofs of multi-ciphersuite security—which themselves are essentially the same as our proofs of single-ciphersuite AKE security (Theorem 6.1) and explicit entity authentication (Theorem 6.4). But again, it would require agile security assumptions on the KDFs and MACs.

As mentioned in Section 6.4.1, Bhargavan et al. [Bha+16] also analyzed the negotiation security of IKEv1-PSK in aggressive mode, which is very similar to the 4WHS protocol. However, for simplicity they assumed that only a single KDF and a single MAC algorithm was being used in order to rely on more traditional non-agile security assumptions.

Chapter 7

Conclusions

Contents

In recent years there has been a great interest in formally analyzing the TLS protocol. Almost every aspect of TLS have been scrutinized, ranging from the security of its core handshake protocol [MSW10, Jag+12, KPW13b, Brz+13a, KSS13, Li+14, Bha+14b] and Record Layer Protocol [Kra01, PRS11, Boy+16], to its multi-ciphersuite and (re-)negotiation (in)security [GKS13, Ber+14, DS15, Beu+15]. Even for the unfinished upcoming TLS 1.3 standard, there have already been multiple results [Bad+15b, Dow+15, KW15, Dow+16, FG17]. This analysis has greatly increased our understanding of TLS. At the same time, there are other important real-world protocols that have received much less attention. EAP, EAP-TLS and IEEE 802.11 are all examples of this. In this thesis we have tried to remedy this state-of-affairs by conducting a formal security analysis of the three aforementioned protocols in the game-based setting. We have concentrated on the central cryptographic operations of each of these protocols.

Beginning with EAP in Chapter 4, we showed through two generic composition theorems how the security of the overall EAP framework is sound. More specifically, these composition theorems give sufficient security requirements on the components that make up EAP in order to securely instantiate the framework. In this sense, they make precise the security requirements that were only informally described in [RFC5247]. A particularly interesting observation is the

importance of channel binding in the EAP methods. This has so far lacked any formal justification in the EAP standard, only being argued based on ad-hoc examples. Recall that our first composition theorem showed that basic EAP is a secure 3P-AKE protocol. Without channel binding this would not be true.

In Chapter 5 we analyzed the EAP-TLS protocol, which is one of the most widely supported EAP methods. We showed that it is a secure 2P-AKE protocol. However, more interesting than this result on its own, is how it was established. Namely, our result on EAP-TLS follows as a corollary of a more general theorem that shows how almost any secure channel protocol (i.e., TLS-like ACCE protocols), can be transformed into a secure AKE protocol. Although intuitively straightforward, the proof of this was non-trivial.

Finally, in Chapter 6 we analyzed the IEEE 802.11 protocol. We first looked at the setting found in most home networks, where a common key is manually installed at all connecting devices. In this scenario we proved that the 4WHS protocol is a secure 2P-AKE with no forward secrecy, and that it additionally provides explicit entity authentication. These are also exactly the properties needed by our second composition theorem, which when combined with our 4WHS result, culminated in a first security proof of EAP + IEEE 802.11 in the computational setting. Besides our results on the 4WHS handshake, we also proved that the IEEE 802.11 channel protocol CCMP is a secure stateful authenticated encryption scheme.

7.1 Limitations of our results

There are several caveats to our results. First and foremost, all of our results are based on simplifications of the actual real-world protocols. This is always necessary in order to make any analysis feasible. Nevertheless, it opens up the possibility that our modeling does not accurately reflect the protocols as they really are. Unfortunately, this is a fundamental problem for which there is no easy solution. A recent trend in the analysis of TLS has been to introduce machine-checkable proofs, and even derive models from executable code itself, so as to mirror the real TLS protocol as closely as possible [Bha+14b, Beu+15]. But even these approaches make simplifications compared to the real protocol. Moreover, it is not clear how this approach can be applied to the kind of generic theorems we have proven, which inherently *have no* implementations. In the end, we have tried to model the protocols as faithfully as we can, but ultimately there are no guarantees that our models completely match the real-world protocols.

7.1.1 Things not covered by our analysis

There are several things not covered by our analysis due to assumptions made in our security model (Chapter 3). Below we discuss some of them.

PKI. We have not considered the question of PKI in this thesis. Instead, we have assumed that all long-term keys are honestly generated, and that all parties have authentic copies of every public-key in the system. Since PKI is highly non-trivial, this certainly simplifies our model. At the same time, we argue that the PKI needed by most EAP use-cases is much simpler than the PKI needed for the public Internet. With the exception of eduroam, EAP is typically used within the scope of a single organization. Thus, all administration of users and long-term credentials is fully controlled by the organization itself. Moreover, in the current design of eduroam, the authentication of long-term keys is only relevant between users and servers belonging to the *same* institution, thus reducing to the single organization scenario (see Section 7.2 below for further discussion on eduroam).

Long-term key configurations. Our security model only considered three classes of protocols: (i) two-party protocols based solely on public keys, (ii) two-party protocols based solely on PSKs, and (iii) three-party protocols where the client and servers have public keys, and the servers and authenticators share PSKs. The reason for choosing to focus exclusively on these three classes of protocols was that they correspond to the way long-term keys are being used in EAP-TLS, IEEE 802.11, and EAP(-TLS) + IEEE 802.11, respectively. Still, this choice was mostly done for ease of exposition. All our results should be largely orthogonal to the type of long-term keys being used.

Side-channel attacks. Traditionally, security models have only considered adversaries that attack a cryptographic algorithm in a black-box way. That is, the adversary only acts based on the input/output behavior of the algorithm. However, this misses a large class of real-world attacks known as *side-channel* attacks. These are attacks where the adversary exploits some implementation specific detail about an algorithm in order to learn its secret key. Side-channel attacks can be based on observations of an algorithm's power usage, its memory usage, its running time, or its behavior in faulty running environments. Being able to observe details about an algorithm's run-time characteristics is a powerful capability, and many algorithms that are secure in traditional security models can nonetheless be broken when given access to this information.

Although protection against side-channel attacks is important, and even though security models that try to capture this exist [ASB14], we have consid-

ered it out of scope for this thesis. Besides the additional complexity it would add to our models, we also feel that it would distract from the overall theme of our results, which are mostly generic and do not focus on any particular implementation.

7.1.2 Tightness of security reductions

A security reduction which transforms an adversary $\mathcal{A}$ breaking protocol Π, into an algorithm $\mathcal{B}$ solving a problem P, is said to be *tight* if $\mathcal{B}$'s success probability and computational cost is essentially the same as that of $\mathcal{A}$. A security reduction which is not tight is said to be *non-tight*. The value of a tight reduction is that it allows to transfer confidence in the hardness of problem P into a similar confidence level for the security of protocol Π. Ostensibly, this means that one can also determine safe parameters for Π based on confidence in the hardness of problem P. On the other hand, if the security reduction is non-tight, then one would have to compensate for the difference by choosing larger parameters for Π. Generally, this leads to less efficient protocols.

Our composition results in Chapter 4 are non-tight since they incur a factor of n^2, where n is the total number of sessions created by adversary $\mathcal{A}$. Unfortunately, this seems to be inevitable for these types of generic composition results. Although Bader et al. [Bad+15a] have shown how to construct an AKE protocol with a tight security reduction, they required a special construction. Our composition results on the other hand, uses its protocol building blocks in a black-box manner. Moreover, the protocol of Bader et al. [Bad+15a] is currently the only known protocol with a tight reduction—all other protocols comes with a non-tight reduction, black-box or not. The recent impossibility result of Jager et al. [Jag+17], which shows that non-tight security reduction are necessary for multi-key authenticated encryption schemes when corruption is allowed, suggests that the same should be true for protocols that are largely generic in nature.

7.2 Future work and open problems

There are several possible avenues for future work based on the results of this thesis. One possibility is further specialization, aiming to include more details about the concrete protocols into the analysis. One example could be a more detailed analysis of the eduroam network. In particular, eduroam does not currently employ a global PKI. In order to facilitate roaming between different institutions, eduroam employs a *hierarchy* of RADIUS servers, organized at an institutional, national, and global level, somewhat similar to DNS. When transferring the MSK from the home RADIUS server to the access point of

the visiting network to which the client is currently roaming, eduroam will send the MSK through a chain of RADIUS servers within the hierarchy. Every pairwise RADIUS servers in this hierarchy share a symmetric secret. It could be interesting to factor this type of authentication server structure into the analysis of EAP. At the same time, eduoram is in fact planning a transition away from this PSK-based RADIUS hierarchy, moving instead towards a global PKI [RFC7593]. This would make it possible for an access point to establish a secure channel directly with the RADIUS server of a roaming user's home institution. This could also be interesting to analyze and would require PKIs to be incorporated into the security model, similar to [Boy+13].

Another example of further specialization could be to look at the MPPE algorithm described in Section 4.1.2, used by RADIUS to encrypt the MSK. As mentioned in the corresponding related work section (Section 4.1.4), Horst et al. [Hor+16] have successfully cryptanalyzed MPPE within the context of PPTP. However, establishing the precise security guarantees provided by MPPE as it is used within RADIUS is an open problem.

The alternative to a more detailed analysis is further generalization. A straightforward generalization would be to consider more general protocol classes in our security models by allowing arbitrary configurations of long-term keys, as well as protocols having $N > 3$ roles. Another generalization would be to incorporate multi-ciphersuite and negotiation security into our composition theorems.

Beyond the dichotomy of further specialization or generalization, there is also the question of *applying* our results in settings outside of EAP, EAP-TLS and IEEE 802.11. For instance, the AKA protocol used within 3G and 4G mobile networks is very similar to the architecture of the EAP framework. Thus, it could be possible to apply the composition theorems of Chapter 4 to the AKA protocol as well. This would provide an alternative, and perhaps more modular, proof to the one that was recently given by Alt et al. [Alt+16].

Finally, it is an open problem whether the tightness of our security reductions for the generic composition theorems can be improved, or if the n^2 security loss is essentially optimal. If the n^2 loss is inherent, then it might be possible to prove this using meta-reduction techniques similar to those of Jager et al. [Jag+17], who showed that reductions from multi-key security to single-key security must be non-tight for authenticated encryption schemes when keys can be corrupted.

Appendix A

Additional definitions

Contents

A.1 Pseudorandom functions

A *pseudorandom function (PRF)* is a family of functions $F\colon \{0,1\}^\kappa \times \{0,1\}^\ell \to \{0,1\}^L$, having key length κ, 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 experiment shown in Figure A.1.

Definition A.1 (PRF security). Let F be a PRF. The *PRF advantage* of an adversary $\mathcal{A}$ is

$$\mathbf{Adv}^{\mathsf{prf}}_F(\mathcal{A}) \stackrel{\text{def}}{=} 2 \cdot \Pr[\mathbf{Exp}^{\mathsf{prf}}_F(\mathcal{A}) \Rightarrow 1] - 1 \tag{A.1}$$

$$= \Pr[\mathcal{A}^{F_K(\cdot)} \Rightarrow 1] - \Pr[\mathcal{A}^{\$(\cdot)} \Rightarrow 1]. \tag{A.2}$$

Equation (A.1) is the definition. Equation (A.2) is an equivalent formulation, where $F_K = F(K, \cdot)$ for a random key $K \twoheadleftarrow \{0,1\}^\kappa$, and $\$ \twoheadleftarrow \mathsf{Func}(\ell, L)$.

$\underline{\mathbf{Exp}_F^{\mathsf{prf}}(\mathcal{A})}$:
1: $b \leftarrow \{0,1\}$
2: $K \twoheadleftarrow \{0,1\}^\kappa$
3: $f_0 \leftarrow F(K,\cdot)$
4: $f_1 \twoheadleftarrow \mathsf{Func}(\ell, L)$
5:
6: $b' \leftarrow \mathcal{A}^{f_b(\cdot)}$
7: **return** $(b' = b)$

Figure A.1: Experiment defining PRF security.

A.2 Message authentication codes

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

- $\mathsf{MAC}\colon \{0,1\}^\kappa \times \{0,1\}^* \to \{0,1\}^*$ is a deterministic *tag-generation* algorithm that takes in a key $K \in \{0,1\}^\kappa$, a *message* $m \in \{0,1\}^*$ and returns a *tag* $\tau \in \{0,1\}^*$.
- $\mathsf{Vrfy}\colon \{0,1\}^\kappa \times \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ is a deterministic *verification-algorithm* that takes in a key $K \in \{0,1\}^\kappa$, a message $m \in \{0,1\}^*$ and a candidate tag $\tau \in \{0,1\}^*$; and produces a *decision* $d \in \{0,1\}$. Algorithm $\mathsf{Vrfy}(K,\cdot,\cdot)$ is uniquely determined by algorithm $\mathsf{MAC}(K,\cdot)$ as follows:

$$\mathsf{Vrfy}(K,m,\tau) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } \mathsf{MAC}(K,m) = \tau, \\ 0 & \text{otherwise.} \end{cases} \tag{A.3}$$

The security of a MAC is defined by the experiment shown in Figure A.2.

Definition A.2 (SUF-CMA security). Let $\Sigma = (\mathsf{MAC}, \mathsf{Vrfy})$ be a MAC. The *SUF-CMA advantage* of an adversary $\mathcal{A}$ is

$$\mathbf{Adv}_\Sigma^{\mathsf{SUF\text{-}CMA}}(\mathcal{A}) \stackrel{\text{def}}{=} \Pr[\mathbf{Exp}_\Sigma^{\mathsf{SUF\text{-}CMA}}(\mathcal{A}) \Rightarrow 1]. \tag{A.4}$$

A.3 Authenticated encryption

An *authenticated encryption (AE)* scheme is a tuple $\Lambda = (\mathsf{Enc}, \mathsf{Dec})$ consisting of two algorithms.[1]

[1] We omit the property of *length-hiding* in our treatment of AE, stAE (Appendix A.4) and ACCE (Section 3.4). This omission is immaterial for the results established in this thesis.

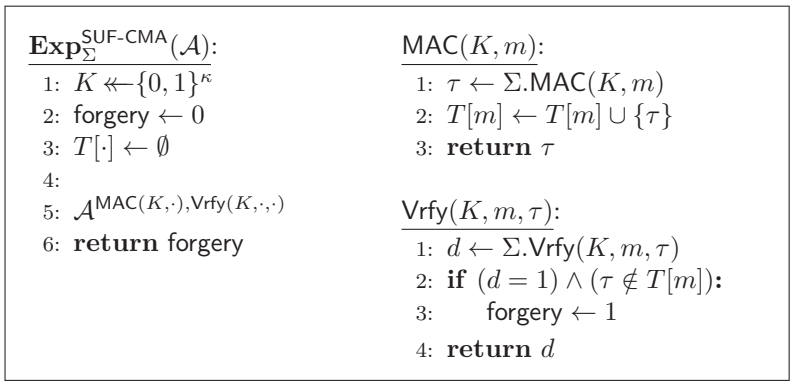


Figure A.2: Experiment defining SUF-CMA security.

- $\mathsf{Enc}\colon \{0,1\}^{\kappa} \times \{0,1\}^{\lambda} \times \{0,1\}^{*} \times \{0,1\}^{*} \to \{0,1\}^{*}$ is an *encryption algorithm* that takes as input a key $K \in \{0,1\}^{\kappa}$, a *nonce* $N \in \{0,1\}^{\lambda}$, a *message* $M \in \{0,1\}^{*}$, and *associated data* $A \in \{0,1\}^{*}$; and outputs a *ciphertext* $C \in \{0,1\}^{*}$.
- $\mathsf{Dec}\colon \{0,1\}^{\kappa} \times \{0,1\}^{\lambda} \times \{0,1\}^{*} \times \{0,1\}^{*} \to \{0,1\}^{*} \cup \{\perp\}$ is a *decryption algorithm* that takes as input a key $K \in \{0,1\}^{\kappa}$, a nonce N, a *ciphertext* $C \in \{0,1\}^{*}$, and associated data A; and either outputs a message $M \in \{0,1\}^{*}$ or a distinguished failure symbol $\perp$.

Correctness of an AE scheme demands that

$$\mathsf{Dec}(K, N, \mathsf{Enc}(K, N, M, A), A) = M \tag{A.5}$$

for all $K \in \{0,1\}^{\kappa}$, $N \in \{0,1\}^{\lambda}$, and $M, A \in \{0,1\}^{*}$.

The security of an AE scheme is defined by the experiment shown in Figure A.3. This is a nonce-based variant of the definition used in [PRS11]. We require that the adversary is *nonce-respecting*, which means that it never queries its encryption oracle with the same nonce twice. However, rather than only quantifying over nonce-respecting adversaries, we instead enforce the requirement directly in the encryption oracle itself.

Definition A.3 (AE security). Let $\Lambda = (\mathsf{Enc}, \mathsf{Dec})$ be an AE scheme. The *AE advantage* of an adversary $\mathcal{A}$ is

$$\mathbf{Adv}^{\mathsf{AE}}_{\Lambda}(\mathcal{A}) \stackrel{\text{def}}{=} 2 \cdot \Pr[\mathbf{Exp}^{\mathsf{AE}}_{\Lambda}(\mathcal{A}) \Rightarrow 1] - 1 \tag{A.6}$$

$$= \Pr[\mathbf{Exp}^{\mathsf{AE}}_{\Lambda}(\mathcal{A}) \Rightarrow 1 \mid b = 0] - \Pr[\mathbf{Exp}^{\mathsf{AE}}_{\Lambda}(\mathcal{A}) \Rightarrow 0 \mid b = 1]. \tag{A.7}$$

Equation (A.6) is the definition, while (A.7) is an equivalent formulation.

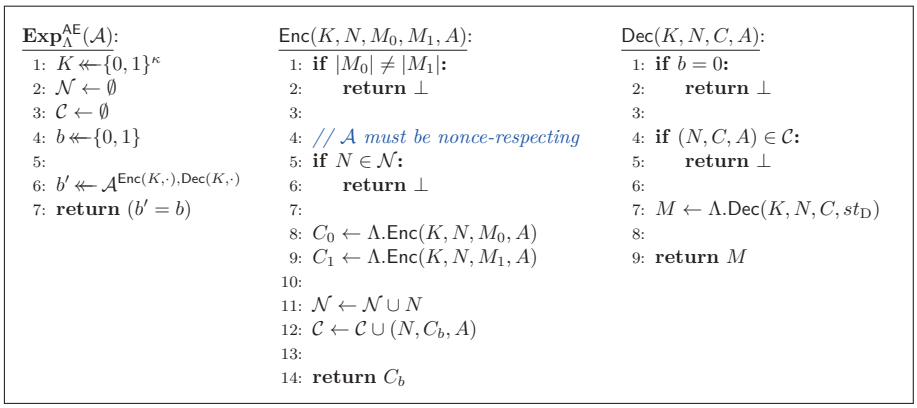


Figure A.3: Experiment defining security for an AE scheme $\Lambda = (\mathsf{Enc}, \mathsf{Dec})$.

Equivalence with other formulations. In Section 6.3.2 we reduce the security of CCMP to the AE security of the CCM mode of operation. Jonsson [Jon03] have shown that CCM satisfies the two separate security notions of IND-CPA and INT-CTXT. Furthermore, Rogaway and Shrimpton [RS06a] have shown that this is equivalent to an all-in-one formulation of AE security. However, the AE definition used by Rogaway and Shrimpton is slightly different from the all-in-one definition we have given above. Specifically, Rogaway and Shrimpton [RS06a] use a "Real-vs-Ideal" formulation of AE security, whereas we (and [PRS11]) use a "Left-or-Right" formulation for encryption combined with a "Real-vs-Ideal" formulation for decryption.

In more detail, let $\mathrm{AE}_{\mathrm{RvI}}$ denote the variant of AE security in [RS06a] and, for the remainder of this section, let $\mathrm{AE}_{\mathrm{LR}}$ denote the formulation of AE security we have used in Definition A.3. The $\mathrm{AE}_{\mathrm{RvI}}$ advantage of an adversary $\mathcal{A}$ against some AE scheme Λ is

$$\mathbf{Adv}^{\mathsf{AE\text{-}RvI}}_{\Lambda}(\mathcal{A}) \stackrel{\mathrm{def}}{=} \Pr[\mathcal{A}^{\mathcal{E}_K(\cdot,\cdot,\cdot),\mathcal{D}_K(\cdot,\cdot,\cdot)} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathcal{E}_K(\cdot,\$,\cdot),\perp(\cdot,\cdot,\cdot)} \Rightarrow 1], \quad \text{(A.8)}$$

where $\mathcal{E}_K(\cdot,\cdot,\cdot)$ and $\mathcal{D}_K(\cdot,\cdot,\cdot)$ are oracles for the real encryption and decryption algorithms of Λ, while $\mathcal{E}_K(\cdot,\$,\cdot)$ is an oracle that returns encryption of random bits $\$$ equal in length to the input message and $\perp(\cdot,\cdot,\cdot)$ is an oracle that always returns $\perp$. Just like in $\mathrm{AE}_{\mathrm{LR}}$, it is required that $\mathcal{A}$ is nonce-respecting and does not forward the output of its encryption oracles to its decryption oracles. Additionally, $\mathcal{A}$ is forbidden from making the same encryption query twice.

Theorem A.4 (Informal). *The following three notions of AE security are all equivalent:*

(i) AE_{LR},

(ii) AE_{RvI},

(iii) *IND-CPA + INT-CTXT.*

Proof sketch.

$(i) \implies (ii)$: Suppose we have an adversary $\mathcal{A}$ against the $\mathrm{AE}_{\mathrm{RvI}}$ security of some AE scheme Λ. From $\mathcal{A}$ we construct the following adversary $\mathcal{B}$ against the $\mathrm{AE}_{\mathrm{LR}}$ security of Λ. To answer $\mathcal{A}$'s encrypt queries (N, M, A), $\mathcal{B}$ queries its left-or-right Enc oracle on $(N, \$, M, A)$, where $\$$ is a random bit string of the same length as M. To answer $\mathcal{A}$'s decrypt queries, $\mathcal{B}$ forwards to its own decryption oracle Dec. When $\mathcal{A}$ outputs a bit b', then $\mathcal{B}$stops and outputs the same bit b'.

Note that when the secret bit b in $\mathcal{B}$'s $\mathrm{AE}_{\mathrm{LR}}$ security game is 0, then $\mathcal{B}$ perfectly simulates the "Ideal" world for $\mathcal{A}$. On the other hand, when $b = 1$, then $\mathcal{B}$ perfectly simulates the "Real" world. Hence, $\mathcal{B}$ wins with the same probability as $\mathcal{A}$.

$(ii) \implies (iii)$: This follows by an adaption of the proof of Proposition 9 in [RS06b] to the setting of nonce-based AE schemes.

$(iii) \implies (i)$: Let $\mathcal{A}$ be an adversary against the $\mathrm{AE}_{\mathrm{LR}}$ security of Λ. Let G_0 be the original $\mathrm{AE}_{\mathrm{LR}}$ security game, and let G_1 be the game where all Dec queries are answered by $\perp$ irregardless of the secret bit b. Game G_0 and G_1 are identical unless $\mathcal{A}$ makes a forgery. Let F denote this event. The probability $\Pr[F]$ is bounded by the following INT-CTXT adversary $\mathcal{F}$. To answer $\mathcal{A}$'s left-or-right encryption queries (N, M_0, M_1, A), $\mathcal{F}$ simulates the secret bit b of the $\mathrm{AE}_{\mathrm{LR}}$ security game itself by drawing a random bit b_{sim}. It then queries its own (proper) encryption oracle $\mathcal{E}_K(\cdot,\cdot,\cdot)$ on $(N, M_{b_{sim}}, A)$. To simulate $\mathcal{A}$'s decryption queries, $\mathcal{F}$ forwards to its own decryption oracle $\mathcal{D}_K(\cdot,\cdot,\cdot)$. In this way $\mathcal{F}$ perfectly simulates game G_0 and wins exactly when event F occurs.

To bound $\mathcal{A}$'s advantage in game G_1, we create an IND-CPA adversary $\mathcal{B}$ which forwards $\mathcal{A}$'s left-or-right encryption queries to its own left-or-right encryption oracle $\mathcal{E}_K(\cdot,\cdot,\cdot,\cdot)$, and answers all of $\mathcal{A}$'s decryption queries by $\perp$. ■

A.4 Stateful authenticated encryption

A *stateful authenticated encryption (stAE)* scheme is a tuple $\Lambda = (\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$ consisting of three algorithms.

- Init is a deterministic *initialization* algorithm, that takes no input and produces two states $(st_{\mathrm{E}}, st_{\mathrm{D}}) \in \{0,1\}^* \times \{0,1\}^*$; one for encryption and one for decryption.
- $\mathsf{Enc}\colon \{0,1\}^\kappa \times (\{0,1\}^*)^3 \to \{0,1\}^* \times \{0,1\}^*$ is an *encryption algorithm* that takes as input a key $K \in \{0,1\}^\kappa$, a *message* $M \in \{0,1\}^*$, *associated data* $A \in \{0,1\}^*$, and an encryption state st_{E}; and produces a *ciphertext* $C \in \{0,1\}^*$ and a new encryption state st'_{E}.
- $\mathsf{Dec}\colon \{0,1\}^\kappa \times (\{0,1\}^*)^3 \to \{0,1\}^* \times \{0,1\}^*$ is a deterministic *decryption algorithm* that takes as input a key $K \in \{0,1\}^\kappa$, a *ciphertext* $C \in \{0,1\}^*$, associated data $A \in \{0,1\}^*$, and a decryption state st_{D}. It then either produces a message $m \in \{0,1\}^*$ or distinguished failure symbol $\perp$, together with a new decryption state st'_{D}.

Correctness of a stAE scheme demands that for all $K \in \{0,1\}^\kappa$, if the states $(st^0_{\mathrm{E}}, st^0_{\mathrm{D}})$ were produced by running algorithm Init, and the sequence of encryptions $(C_i, st^{i+1}_{\mathrm{E}}) \leftarrow \Lambda.\mathsf{Enc}(K, M_i, A_i, st^i_{\mathrm{E}})$ is such that $C_i \neq \perp$ for all $i \geq 0$; then the sequence of decryptions $(M'_i, st^{i+1}_{\mathrm{D}}) \leftarrow \Lambda.\mathsf{Dec}(K, C_i, A_i, st^i_{\mathrm{D}})$ is such that $M'_i = M_i$ for each $i \geq 0$.

Following [PRS11] and [Jag+12], the security of an stAE scheme is defined by the experiment shown in Figure A.4. Note that we have $S[i] = \perp$ for all $i \notin [1, \mathsf{sent}]$.

Definition A.5 (stAE security). Let $\Lambda = (\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$ be a stAE scheme. The *stAE advantage* of an adversary $\mathcal{A}$ is

$$\mathbf{Adv}^{\mathsf{stAE}}_{\Lambda}(\mathcal{A}) \stackrel{\mathrm{def}}{=} 2 \cdot \Pr[\mathbf{Exp}^{\mathsf{stAE}}_{\Lambda}(\mathcal{A}) \Rightarrow 1] - 1 \tag{A.9}$$

$$\begin{aligned} &= \Pr[\mathbf{Exp}^{\mathsf{stAE}}_{\Lambda}(\mathcal{A}) \Rightarrow 1 \mid b = 0] \\ &\quad - \Pr[\mathbf{Exp}^{\mathsf{stAE}}_{\Lambda}(\mathcal{A}) \Rightarrow 0 \mid b = 1]. \end{aligned} \tag{A.10}$$

Equation (A.9) is the definition, while (A.10) is an equivalent formulation.

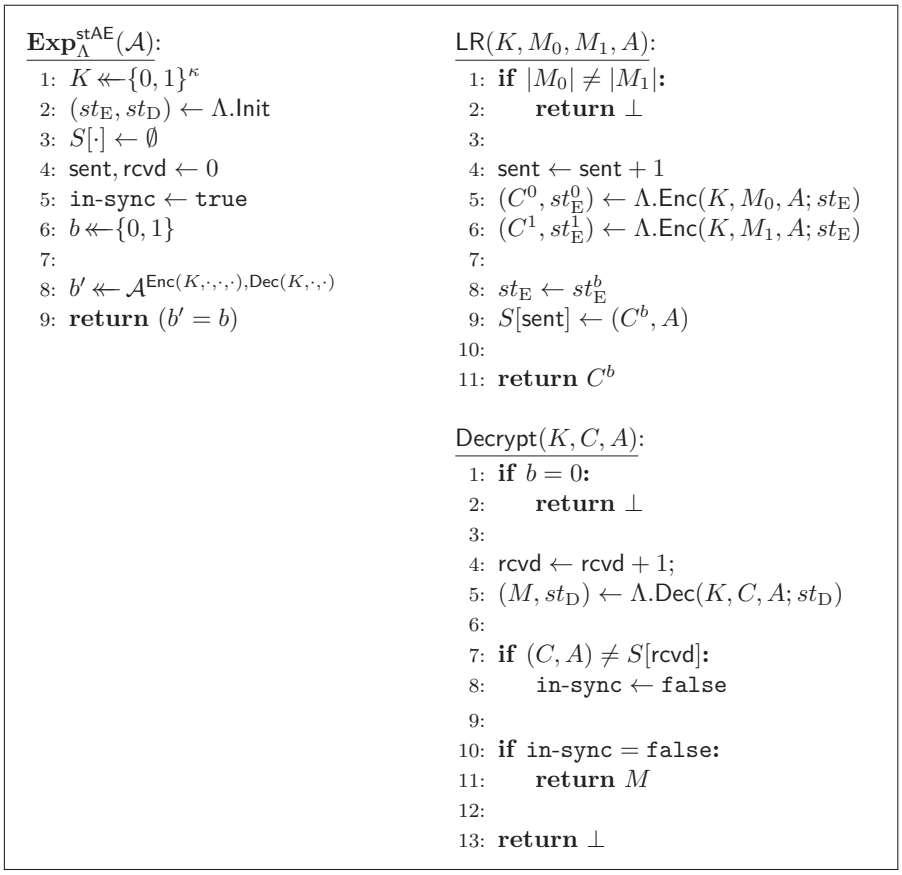


Figure A.4: Experiment defining security for a stAE scheme $\Lambda = (\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$.

Appendix B

Transcript parsing rules

Let T_3 be a transcript produced by running experiment $\mathbf{Exp}_{\Pi_3,\mathcal{Q}}(\mathcal{A})$, where Π_3 is the protocol described in Section 4.2. Table B.1 defines how to extract from T_3 two other transcripts, T_1 and T_2, corresponding to runs of $\mathbf{Exp}_{\Pi_1,\mathcal{Q}}(\mathcal{A}')$, and $\mathbf{Exp}_{\Pi_2,\mathcal{Q}}(\mathcal{A}'')$, respectively. Essentially, T_1 and T_2 are extracted from T_3 as follows.

- For each initiator session on T_3, create a corresponding initiator session on T_1.
- For each responder session on T_3, create a corresponding responder session on T_2.
- For each server session on T_3, create *two* sessions: one *responder* session on T_1; and one *initiator* session on T_2. However, the latter is only created if the server session reached the accept state in sub-protocol Π_1 on T_3.
- For each Send message on T_3 directed to an initiator session, copy the Send message to the corresponding session on T_1.
- For each Send message on T_3 directed to a responder session, copy the Send message to the corresponding session on T_2.
- For each Send message on T_3 directed to a server session which has *not* accepted in sub-protocol Π_1, copy the Send message to the corresponding session on T_1.
- For each Send message on T_3 directed to a server session which *has* accepted in sub-protocol Π_1, copy the Send message to the corresponding session on T_2.

Table B.1: Parsing rules for extracting transcripts T_1 and T_2 from a transcript T_3 generated by an execution of experiment $\mathbf{Exp}_{\Pi_3,\mathcal{Q}}(\mathcal{A})$, where Π_3 is the protocol defined in Section 4.2. The table assumes that $A \in \mathcal{I}$, $B \in \mathcal{R}$ and $S \in \mathcal{S}$ in protocol Π_3. Parsing is done as follows. For each entry in T_3, look up the row in the column marked "T_3" that matches this query. From this row, use the corresponding queries in the columns marked "T_1" and "T_2" to create the entries on T_1 and T_2 respectively ("$-$" means that no query should be created).

T_3	T_1	T_2
$(\mathsf{NewSession}(A, B, S), \pi_A^i, m)$	$(\mathsf{NewSession}(A, S), \pi_A^i, m)$	$-$
$(\mathsf{NewSession}(B, A, S), \pi_B^j, \perp)$	$-$	$(\mathsf{NewSession}(B, S), \pi_B^j, \perp)$
$(\mathsf{NewSession}(S, A, B), \pi_S^k, \perp)$	$(\mathsf{NewSession}(S, A), \pi_S^k, \perp)$	$-$
$(\mathsf{Send}(\pi_A^i, m), m^*, (\mathsf{running}, \mathsf{running}, \mathsf{running}))$	$(\mathsf{Send}(\pi_A^i, m), m^*, (\mathsf{running}))$	$-$
$(\mathsf{Send}(\pi_A^i, m), m^*, (\mathsf{accepted}, \mathsf{accepted}, \mathsf{accepted}))$	$(\mathsf{Send}(\pi_A^i, m), m^*, (\mathsf{accepted}))$	$-$
$(\mathsf{Send}(\pi_A^i, m), \perp, (\mathsf{rejected}, \mathsf{rejected}, \mathsf{rejected}))$	$(\mathsf{Send}(\pi_A^i, m), \perp, (\mathsf{rejected}))$	$-$
$(\mathsf{Send}(\pi_B^j, m), m^*, (\mathsf{accepted}, \mathsf{running}, \mathsf{running}))$	$-$	$(\mathsf{Send}(\pi_B^j, m), m^*, (\mathsf{running}))$
$(\mathsf{Send}(\pi_B^j, m), m^*, (\mathsf{accepted}, \mathsf{accepted}, \mathsf{running}))$	$-$	$(\mathsf{Send}(\pi_B^j, m), m^*, (\mathsf{accepted}))$
$(\mathsf{Send}(\pi_B^j, m), \perp, (\mathsf{accepted}, \mathsf{rejected}, \mathsf{rejected}))$	$-$	$(\mathsf{Send}(\pi_B^j, m), \perp, (\mathsf{rejected}))$
$(\mathsf{Send}(\pi_B^j, C_{\mathrm{key}}), \perp, (\mathsf{accepted}, \mathsf{accepted}, \mathsf{accepted}))$	$-$	$-$
$(\mathsf{Send}(\pi_B^j, C'_{\mathrm{key}}), \perp, (\mathsf{accepted}, \mathsf{accepted}, \mathsf{rejected}))$	$-$	$-$
$(\mathsf{Send}(\pi_S^k, m), m^*, (\mathsf{running}, \mathsf{running}, \mathsf{running}))$	$(\mathsf{Send}(\pi_S^k, m), m^*, (\mathsf{running}))$	$-$
$(\mathsf{Send}(\pi_S^k, m), \perp, (\mathsf{rejected}, \mathsf{rejected}, \mathsf{rejected}))$	$(\mathsf{Send}(\pi_S^k, m), \perp, (\mathsf{rejected}))$	$-$
$(\mathsf{Send}(\pi_S^k, m), m^*, (\mathsf{accepted}, \mathsf{running}, \mathsf{running}))^\dagger$	$(\mathsf{Send}(\pi_S^k, m), \perp, (\mathsf{accepted}))$	$(\mathsf{NewSession}(S, B), \pi_S^k, m^*)$
$(\mathsf{Send}(\pi_S^k, m), m^*, (\mathsf{accepted}, \mathsf{running}, \mathsf{running}))$	$-$	$(\mathsf{Send}(\pi_S^k, m), m^*, (\mathsf{running}))$
$(\mathsf{Send}(\pi_S^k, m), C_{\mathrm{key}}, (\mathsf{accepted}, \mathsf{accepted}, \mathsf{accepted}))$	$-$	$(\mathsf{Send}(\pi_S^k, m), \perp, (\mathsf{accepted}))$
$(\mathsf{Send}(\pi_S^k, m), \perp, (\mathsf{accepted}, \mathsf{rejected}, \mathsf{rejected}))$	$-$	$(\mathsf{Send}(\pi_S^k, m), \perp, (\mathsf{rejected}))$

†This rule only applies if $\pi_S^k.\vec{\alpha} = (\mathsf{running}, \mathsf{running}, \mathsf{running})$ for all prior Send queries to π_S^k, i.e., if receiving message m caused session π_S^k to accept in sub-protocol Π_1.

Bibliography

[Aca+10] Tolga Acar, Mira Belenkiy, Mihir Bellare, and David Cash. "Cryptographic Agility and Its Relation to Circular Encryption". In: *Advances in Cryptology – EUROCRYPT 2010*. Ed. by Henri Gilbert. Vol. 6110. Lecture Notes in Computer Science. French Riviera: Springer, Heidelberg, Germany, May 2010, pp. 403–422 (Cited on page 142).

[ADW09] Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. "Leakage-Resilient Public-Key Cryptography in the Bounded-Retrieval Model". In: *Advances in Cryptology – CRYPTO 2009*. Ed. by Shai Halevi. Vol. 5677. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2009, pp. 36–54 (Cited on page 53).

[AFP05] Michel Abdalla, Pierre-Alain Fouque, and David Pointcheval. "Password-Based Authenticated Key Exchange in the Three-Party Setting". In: *PKC 2005: 8th International Workshop on Theory and Practice in Public Key Cryptography*. Ed. by Serge Vaudenay. Vol. 3386. Lecture Notes in Computer Science. Les Diablerets, Switzerland: Springer, Heidelberg, Germany, Jan. 2005, pp. 65–84 (Cited on pages 39, 65).

[Alt+16] Stéphanie Alt, Pierre-Alain Fouque, Gilles Macario-Rat, Cristina Onete, and Benjamin Richard. "A Cryptographic Analysis of UMTS/LTE AKA". In: *ACNS 16: 14th International Conference on Applied Cryptography and Network Security*. Ed. by Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider. Vol. 9696. Lecture Notes in Computer Science. Guildford, UK: Springer, Heidelberg,

Germany, June 2016, pp. 18–35. DOI: 10.1007/978-3-319-39555-5_2 (Cited on pages 89, 148).

[AN96] Martín Abadi and Roger M. Needham. "Prudent Engineering Practice for Cryptographic Protocols". In: *IEEE Trans. Software Eng.* 22.1 (1996), pp. 6–15. DOI: 10.1109/32.481513. (Cited on page 1).

[ANN03] N. Asokan, Valtteri Niemi, and Kaisa Nyberg. "Man-in-the-Middle in Tunnelled Authentication Protocols". In: *Security Protocols, 11th International Workshop, Cambridge, UK, April 2-4, 2003, Revised Selected Papers*. Ed. by Bruce Christianson, Bruno Crispo, James A. Malcolm, and Michael Roe. Vol. 3364. Lecture Notes in Computer Science. Springer, 2003, pp. 28–41. DOI: 10.1007/11542322_6. (Cited on page 65).

[AR00] Martín Abadi and Phillip Rogaway. "Reconciling Two Views of Cryptography (The Computational Soundness of Formal Encryption)". In: *Theoretical Computer Science, Exploring New Frontiers of Theoretical Informatics, International Conference IFIP TCS 2000, Sendai, Japan, August 17-19, 2000, Proceedings*. Ed. by Jan van Leeuwen, Osamu Watanabe, Masami Hagiya, Peter D. Mosses, and Takayasu Ito. Vol. 1872. Lecture Notes in Computer Science. Springer, 2000, pp. 3–22. DOI: 10.1007/3-540-44929-9_1. (Cited on page 2).

[ASB14] Janaka Alawatugoda, Douglas Stebila, and Colin Boyd. "Modelling after-the-fact leakage for key exchange". In: *ASIACCS 14: 9th ACM Symposium on Information, Computer and Communications Security*. Ed. by Shiho Moriai, Trent Jaeger, and Kouichi Sakurai. Kyoto, Japan: ACM Press, June 2014, pp. 207–216 (Cited on pages 53, 146).

[ASB15] Janaka Alawatugoda, Douglas Stebila, and Colin Boyd. "Continuous After-the-Fact Leakage-Resilient eCK-Secure Key Exchange". In: *15th IMA International Conference on Cryptography and Coding*. Ed. by Jens Groth. Vol. 9496. Lecture Notes in Computer Science. Oxford, UK: Springer, Heidelberg, Germany, Dec. 2015, pp. 277–294. DOI: 10.1007/978-3-319-27239-9_17 (Cited on page 53).

[Bad+15a] Christoph Bader, Dennis Hofheinz, Tibor Jager, Eike Kiltz, and Yong Li. "Tightly-Secure Authenticated Key Exchange". In: *TCC 2015: 12th Theory of Cryptography Conference, Part I*. Ed. by Yevgeniy Dodis and Jesper Buus Nielsen. Vol. 9014. Lecture Notes in Computer Science. Warsaw, Poland: Springer, Heidelberg, Germany, Mar. 2015, pp. 629–658. DOI: 10.1007/978-3-662-46494-6_26 (Cited on page 147).

[Bad+15b] Christian Badertscher, Christian Matt, Ueli Maurer, Phillip Rogaway, and Björn Tackmann. "Augmented Secure Channels and the Goal of the TLS 1.3 Record Layer". In: *ProvSec 2015: 9th International Conference on Provable Security*. Ed. by Man Ho Au and Atsuko Miyaji. Vol. 9451. Lecture Notes in Computer Science. Kanazawa, Japan: Springer, Heidelberg, Germany, Nov. 2015, pp. 85–104. DOI: 10.1007/978-3-319-26059-4_5 (Cited on page 144).

[Bar+13] Gilles Barthe, François Dupressoir, Benjamin Grégoire, César Kunz, Benedikt Schmidt, and Pierre-Yves Strub. "EasyCrypt: A Tutorial". In: *Foundations of Security Analysis and Design VII - FOSAD 2012/2013 Tutorial Lectures*. Ed. by Alessandro Aldini, Javier Lopez, and Fabio Martinelli. Vol. 8604. Lecture Notes in Computer Science. Springer, 2013, pp. 146–166. DOI: 10.1007/978-3-319-10082-1_6. (Cited on page 2).

[BCK98] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. "A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols (Extended Abstract)". In: *30th Annual ACM Symposium on Theory of Computing*. Dallas, TX, USA: ACM Press, May 1998, pp. 419–428 (Cited on page 95).

[Bel15] Mihir Bellare. "New Proofs for NMAC and HMAC: Security without Collision Resistance". In: *Journal of Cryptology* 28.4 (Oct. 2015), pp. 844–878. DOI: 10.1007/s00145-014-9185-x (Cited on pages 62, 126).

[Ber+14] Florian Bergsma, Benjamin Dowling, Florian Kohlar, Jörg Schwenk, and Douglas Stebila. "Multi-Ciphersuite Security of the Secure Shell (SSH) Protocol". In: *ACM CCS 14: 21st Conference on Computer and Communications Security*. Ed. by Gail-Joon Ahn, Moti Yung, and Ninghui

Li. Scottsdale, AZ, USA: ACM Press, Nov. 2014, pp. 369–381 (Cited on pages 39, 54, 114, 140–144).

[Beu+15] Benjamin Beurdouche, Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cédric Fournet, Markulf Kohlweiss, Alfredo Pironti, Pierre-Yves Strub, and Jean Karim Zinzindohoue. "A Messy State of the Union: Taming the Composite State Machines of TLS". In: *2015 IEEE Symposium on Security and Privacy*. San Jose, CA, USA: IEEE Computer Society Press, May 2015, pp. 535–552. DOI: 10.1109/SP.2015.39 (Cited on pages 144, 145).

[BG11] Colin Boyd and Juanma González Nieto. "On Forward Secrecy in One-Round Key Exchange". In: *13th IMA International Conference on Cryptography and Coding*. Ed. by Liqun Chen. Vol. 7089. Lecture Notes in Computer Science. Oxford, UK: Springer, Heidelberg, Germany, Dec. 2011, pp. 451–468 (Cited on page 95).

[BGM04] Mihir Bellare, Oded Goldreich, and Anton Mityagin. *The Power of Verification Queries in Message Authentication and Authenticated Encryption*. Cryptology ePrint Archive, Report 2004/309. http://eprint.iacr.org/2004/309. 2004 (Cited on page 126).

[BGW01] Nikita Borisov, Ian Goldberg, and David Wagner. "Intercepting mobile communications: the insecurity of 802.11". In: *MOBICOM 2001, Proceedings of the seventh annual international conference on Mobile computing and networking, Rome, Italy, July 16-21, 2001*. Ed. by Christopher Rose. ACM, 2001, pp. 180–189. DOI: 10.1145/381677.381695. (Cited on page 16).

[Bha+13] Karthikeyan Bhargavan, Cédric Fournet, Markulf Kohlweiss, Alfredo Pironti, and Pierre-Yves Strub. "Implementing TLS with Verified Cryptographic Security". In: *2013 IEEE Symposium on Security and Privacy*. Berkeley, CA, USA: IEEE Computer Society Press, May 2013, pp. 445–459 (Cited on page 2).

[Bha+14a] Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cédric Fournet, Alfredo Pironti, and Pierre-Yves Strub. "Triple Handshakes and Cookie Cutters: Breaking and Fixing Authentication over TLS". In: *2014 IEEE Symposium on Security and Privacy*. Berkeley, CA, USA: IEEE Computer

Society Press, May 2014, pp. 98–113. DOI: 10.1109/SP.2014.14 (Cited on page 65).

[Bha+14b] Karthikeyan Bhargavan, Cédric Fournet, Markulf Kohlweiss, Alfredo Pironti, Pierre-Yves Strub, and Santiago Zanella Béguelin. "Proving the TLS Handshake Secure (As It Is)". In: *Advances in Cryptology – CRYPTO 2014, Part II*. Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8617. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2014, pp. 235–255. DOI: 10.1007/978-3-662-44381-1_14 (Cited on pages 114, 115, 142, 144, 145).

[Bha+16] Karthikeyan Bhargavan, Christina Brzuska, Cédric Fournet, Matthew Green, Markulf Kohlweiss, and Santiago Zanella Béguelin. "Downgrade Resilience in Key-Exchange Protocols". In: *2016 IEEE Symposium on Security and Privacy*. San Jose, CA, USA: IEEE Computer Society Press, May 2016, pp. 506–525. DOI: 10.1109/SP.2016.37 (Cited on pages 142, 143).

[BHK15] Mihir Bellare, Dennis Hofheinz, and Eike Kiltz. "Subtleties in the Definition of IND-CCA: When and How Should Challenge Decryption Be Disallowed?" In: *Journal of Cryptology* 28.1 (Jan. 2015), pp. 29–48. DOI: 10.1007/s00145-013-9167-4 (Cited on page 48).

[BHL06] Andrea Bittau, Mark Handley, and Joshua Lackey. "The Final Nail in WEP's Coffin". In: *2006 IEEE Symposium on Security and Privacy*. Berkeley, CA, USA: IEEE Computer Society Press, May 2006, pp. 386–400 (Cited on page 16).

[BJ17] Chris Brzuska and Håkon Jacobsen. "A Modular Security Analysis of EAP and IEEE 802.11". In: *PKC 2017: 20th International Conference on Theory and Practice of Public Key Cryptography, Part II*. Ed. by Serge Fehr. Vol. 10175. Lecture Notes in Computer Science. Amsterdam, The Netherlands: Springer, Heidelberg, Germany, Mar. 2017, pp. 335–365 (Cited on page 7).

[BJS16] Christina Brzuska, Håkon Jacobsen, and Douglas Stebila. "Safely Exporting Keys from Secure Channels: On the Security of EAP-TLS and TLS Key Exporters". In: *Advances*

in Cryptology – EUROCRYPT 2016, Part I. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9665. Lecture Notes in Computer Science. Vienna, Austria: Springer, Heidelberg, Germany, May 2016, pp. 670–698. DOI: 10.1007/978-3-662-49890-3_26 (Cited on page 7).

[BKR94] Mihir Bellare, Joe Kilian, and Phillip Rogaway. "The Security of Cipher Block Chaining". In: *Advances in Cryptology – CRYPTO'94*. Ed. by Yvo Desmedt. Vol. 839. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 1994, pp. 341–358 (Cited on page 26).

[BL13] Daniel J. Bernstein and Tanja Lange. "Non-uniform Cracks in the Concrete: The Power of Free Precomputation". In: *Advances in Cryptology – ASIACRYPT 2013, Part II*. Ed. by Kazue Sako and Palash Sarkar. Vol. 8270. Lecture Notes in Computer Science. Bengalore, India: Springer, Heidelberg, Germany, Dec. 2013, pp. 321–340. DOI: 10.1007/978-3-642-42045-0_17 (Cited on page 27).

[Bla08] Bruno Blanchet. "A Computationally Sound Mechanized Prover for Security Protocols". In: *IEEE Trans. Dependable Sec. Comput.* 5.4 (2008), pp. 193–207. DOI: 10.1109/TDSC.2007.1005. (Cited on page 2).

[Bla16] Bruno Blanchet. "Modeling and Verifying Security Protocols with the Applied Pi Calculus and ProVerif". In: *Foundations and Trends® in Privacy and Security* 1.1-2 (2016), pp. 1–135. ISSN: 2474-1558. DOI: 10.1561/3300000004. (Cited on page 2).

[BM97] Simon Blake-Wilson and Alfred Menezes. "Entity Authentication and Authenticated Key Transport Protocols Employing Asymmetric Techniques". In: *Security Protocols, 5th International Workshop, Paris, France, April 7-9, 1997, Proceedings*. Ed. by Bruce Christianson, Bruno Crispo, T. Mark A. Lomas, and Michael Roe. Vol. 1361. Lecture Notes in Computer Science. Springer, 1997, pp. 137–158. DOI: 10.1007/BFb0028166. (Cited on pages 4, 27, 35, 39, 49).

[BM99] Simon Blake-Wilson and Alfred Menezes. "Unknown Key-Share Attacks on the Station-to-Station (STS) Protocol". In: *PKC'99: 2nd International Workshop on Theory and Practice in Public Key Cryptography*. Ed. by Hideki Imai and Yuliang Zheng. Vol. 1560. Lecture Notes in Computer Science. Kamakura, Japan: Springer, Heidelberg, Germany, Mar. 1999, pp. 154–170 (Cited on page 114).

[BN00] Mihir Bellare and Chanathip Namprempre. "Authenticated Encryption: Relations among notions and analysis of the generic composition paradigm". In: *Advances in Cryptology – ASIACRYPT 2000*. Ed. by Tatsuaki Okamoto. Vol. 1976. Lecture Notes in Computer Science. Kyoto, Japan: Springer, Heidelberg, Germany, Dec. 2000, pp. 531–545 (Cited on page 135).

[Bol+07] Alexandra Boldyreva, Marc Fischlin, Adriana Palacio, and Bogdan Warinschi. "A Closer Look at PKI: Security and Efficiency". In: *PKC 2007: 10th International Conference on Theory and Practice of Public Key Cryptography*. Ed. by Tatsuaki Okamoto and Xiaoyun Wang. Vol. 4450. Lecture Notes in Computer Science. Beijing, China: Springer, Heidelberg, Germany, Apr. 2007, pp. 458–475 (Cited on page 30).

[Boy+13] Colin Boyd, Cas Cremers, Michele Feltz, Kenneth G. Paterson, Bertram Poettering, and Douglas Stebila. "ASICS: Authenticated Key Exchange Security Incorporating Certification Systems". In: *ESORICS 2013: 18th European Symposium on Research in Computer Security*. Ed. by Jason Crampton, Sushil Jajodia, and Keith Mayes. Vol. 8134. Lecture Notes in Computer Science. Egham, UK: Springer, Heidelberg, Germany, Sept. 2013, pp. 381–399. DOI: 10.1007/978-3-642-40203-6_22 (Cited on pages 30, 53, 148).

[Boy+16] Colin Boyd, Britta Hale, Stig Frode Mjølsnes, and Douglas Stebila. "From Stateless to Stateful: Generic Authentication and Authenticated Encryption Constructions with Application to TLS". In: *Topics in Cryptology – CT-RSA 2016*. Ed. by Kazue Sako. Vol. 9610. Lecture Notes in Computer Science. San Francisco, CA, USA: Springer, Heidelberg, Germany, Feb. 2016, pp. 55–71. DOI: 10.1007/978-3-319-29485-8_4 (Cited on pages 136, 144).

[BPR00] Mihir Bellare, David Pointcheval, and Phillip Rogaway. "Authenticated Key Exchange Secure against Dictionary Attacks". In: *Advances in Cryptology – EUROCRYPT 2000*. Ed. by Bart Preneel. Vol. 1807. Lecture Notes in Computer Science. Bruges, Belgium: Springer, Heidelberg, Germany, May 2000, pp. 139–155 (Cited on pages 4, 27, 32, 38–40, 43, 51, 52).

[BR04] Mihir Bellare and Phillip Rogaway. *Code-Based Game-Playing Proofs and the Security of Triple Encryption*. Cryptology ePrint Archive, Report 2004/331. `http://eprint.iacr.org/2004/331`. 2004 (Cited on pages 25, 69, 128).

[BR94] Mihir Bellare and Phillip Rogaway. "Entity Authentication and Key Distribution". In: *Advances in Cryptology – CRYPTO'93*. Ed. by Douglas R. Stinson. Vol. 773. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 1994, pp. 232–249 (Cited on pages 3, 4, 27, 37, 39).

[BR95] Mihir Bellare and Phillip Rogaway. "Provably Secure Session Key Distribution: The Three Party Case". In: *27th Annual ACM Symposium on Theory of Computing*. Las Vegas, NV, USA: ACM Press, May 1995, pp. 57–66 (Cited on pages 4, 27, 33, 38–40, 43, 44, 57).

[Brz+11] Christina Brzuska, Marc Fischlin, Bogdan Warinschi, and Stephen C. Williams. "Composability of Bellare-Rogaway key exchange protocols". In: *ACM CCS 11: 18th Conference on Computer and Communications Security*. Ed. by Yan Chen, George Danezis, and Vitaly Shmatikov. Chicago, Illinois, USA: ACM Press, Oct. 2011, pp. 51–62 (Cited on pages 39, 41, 45, 54, 113).

[Brz+13a] Christina Brzuska, Marc Fischlin, Nigel P. Smart, Bogdan Warinschi, and Stephen C. Williams. "Less is more: relaxed yet composable security notions for key exchange". English. In: *International Journal of Information Security* 12.4 (2013), pp. 267–297. ISSN: 1615-5262. DOI: `10.1007/s10207-013-0192-y`. (Cited on pages 39, 88, 113, 114, 144).

[Brz+13b] Christina Brzuska, Nigel P. Smart, Bogdan Warinschi, and Gaven J. Watson. "An analysis of the EMV channel establishment protocol". In: *ACM CCS 13: 20th Conference on Computer and Communications Security*. Ed. by Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung. Berlin, Germany: ACM Press, Nov. 2013, pp. 373–386 (Cited on page 39).

[Cam+03] Nancy Cam-Winget, Russell Housley, David Wagner, and Jesse Walker. "Security flaws in 802.11 data link protocols". In: *Commun. ACM* 46.5 (2003), pp. 35–39. DOI: 10.1145/769800.769823. (Cited on page 16).

[Can01] Ran Canetti. "Universally Composable Security: A New Paradigm for Cryptographic Protocols". In: *42nd Annual Symposium on Foundations of Computer Science*. Las Vegas, NV, USA: IEEE Computer Society Press, Oct. 2001, pp. 136–145 (Cited on page 4).

[Cas+13] Aldo Cassola, William K. Robertson, Engin Kirda, and Guevara Noubir. "A Practical, Targeted, and Stealthy Attack Against WPA Enterprise Authentication". In: *ISOC Network and Distributed System Security Symposium – NDSS 2013*. San Diego, CA, USA: The Internet Society, Feb. 2013 (Cited on page 17).

[CCG16] Katriel Cohn-Gordon, Cas J. F. Cremers, and Luke Garratt. "On Post-compromise Security". In: *IEEE 29th Computer Security Foundations Symposium, CSF 2016, Lisbon, Portugal, June 27 - July 1, 2016*. IEEE Computer Society, 2016, pp. 164–178. DOI: 10.1109/CSF.2016.19. (Cited on pages 39, 53).

[CF12] Cas J. F. Cremers and Michele Feltz. "Beyond eCK: Perfect Forward Secrecy under Actor Compromise and Ephemeral-Key Reveal". In: *ESORICS 2012: 17th European Symposium on Research in Computer Security*. Ed. by Sara Foresti, Moti Yung, and Fabio Martinelli. Vol. 7459. Lecture Notes in Computer Science. Pisa, Italy: Springer, Heidelberg, Germany, Sept. 2012, pp. 734–751 (Cited on pages 37, 39, 50, 95).

[CH05] Kim-Kwang Raymond Choo and Yvonne Hitchcock. "Security Requirements for Key Establishment Proof Models: Revisiting Bellare-Rogaway and Jeong-Katz-Lee Protocols". In: *ACISP 05: 10th Australasian Conference on*

Information Security and Privacy. Ed. by Colin Boyd and Juan Manuel González Nieto. Vol. 3574. Lecture Notes in Computer Science. Brisbane, Queensland, Australia: Springer, Heidelberg, Germany, July 2005, pp. 429–442 (Cited on page 38).

[CH09] 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*. IEEEXplore, Mar. 2009, pp. 1–5. DOI: 10.1109/SARNOF.2009.4850319 (Cited on pages 65, 90).

[Cho+05] Kim-Kwang Raymond Choo, Colin Boyd, Yvonne Hitchcock, and Greg Maitland. "On Session Identifiers in Provably Secure Protocols: The Bellare-Rogaway Three-Party Key Distribution Protocol Revisited". In: *SCN 04: 4th International Conference on Security in Communication Networks*. Ed. by Carlo Blundo and Stelvio Cimato. Vol. 3352. Lecture Notes in Computer Science. Amalfi, Italy: Springer, Heidelberg, Germany, Sept. 2005, pp. 351–366 (Cited on page 38).

[CK01] Ran Canetti and Hugo Krawczyk. "Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels". In: *Advances in Cryptology – EUROCRYPT 2001*. Ed. by Birgit Pfitzmann. Vol. 2045. Lecture Notes in Computer Science. Innsbruck, Austria: Springer, Heidelberg, Germany, May 2001, pp. 453–474 (Cited on pages 4, 27, 36, 38, 39, 53, 54, 95).

[CK02] Ran Canetti and Hugo Krawczyk. "Security Analysis of IKE's Signature-based Key-Exchange Protocol". In: *Advances in Cryptology – CRYPTO 2002*. Ed. by Moti Yung. Vol. 2442. Lecture Notes in Computer Science. http://eprint.iacr.org/2002/120/. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2002, pp. 143–161 (Cited on pages 36, 38, 39, 91, 95).

[CKN03] Ran Canetti, Hugo Krawczyk, and Jesper Buus Nielsen. "Relaxing Chosen-Ciphertext Security". In: *Advances in Cryptology – CRYPTO 2003*. Ed. by Dan Boneh. Vol. 2729. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2003, pp. 565–582 (Cited on page 113).

[CM12] Cas Cremers and Sjouke Mauw. *Operational Semantics and Verification of Security Protocols*. Information Security and Cryptography. ISBN 978-3-540-78636-8. Springer, 2012. DOI: 10.1007/978-3-540-78636-8. (Cited on page 57).

[CMU09] Sanjit Chatterjee, Alfred Menezes, and Berkant Ustaoglu. "Reusing Static Keys in Key Agreement Protocols". In: *Progress in Cryptology - INDOCRYPT 2009: 10th International Conference in Cryptology in India*. Ed. by Bimal K. Roy and Nicolas Sendrier. Vol. 5922. Lecture Notes in Computer Science. New Delhi, India: Springer, Heidelberg, Germany, Dec. 2009, pp. 39–56 (Cited on page 140).

[Cre08] Cas J. F. Cremers. "The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols". In: *Computer Aided Verification, 20th International Conference, CAV 2008, Princeton, NJ, USA, July 7-14, 2008, Proceedings*. Ed. by Aarti Gupta and Sharad Malik. Vol. 5123. Lecture Notes in Computer Science. Springer, 2008, pp. 414–418. DOI: 10.1007/978-3-540-70545-1_38. (Cited on page 2).

[Cre09] Cas J.F. Cremers. *Formally and Practically Relating the CK, CK-HMQV, and eCK Security Models for Authenticated Key Exchange*. Cryptology ePrint Archive, Report 2009/253. http://eprint.iacr.org/2009/253. 2009 (Cited on page 29).

[Cre11a] Cas Cremers. "Examining indistinguishability-based security models for key exchange protocols: the case of CK, CK-HMQV, and eCK". In: *ASIACCS 11: 6th ACM Symposium on Information, Computer and Communications Security*. Ed. by Bruce S. N. Cheung, Lucas Chi Kwong Hui, Ravi S. Sandhu, and Duncan S. Wong. Hong Kong, China: ACM Press, Mar. 2011, pp. 80–91 (Cited on page 29).

[Cre11b] Cas J. F. Cremers. "Key Exchange in IPsec Revisited: Formal Analysis of IKEv1 and IKEv2". In: *ESORICS 2011: 16th European Symposium on Research in Computer Security*. Ed. by Vijay Atluri and Claudia Dıaz. Vol. 6879. Lecture Notes in Computer Science. Leuven, Belgium: Springer, Heidelberg, Germany, Sept. 2011, pp. 315–334 (Cited on page 29).

[Dah+17] Thorsten Dahm, Andrej Ota, Douglas C. Medway Gash, and David Carrel. *The TACACS+ Protocol draft-ietf-opsawg-tacacs-06*. Internet-draft. RFC Editor, Feb. 2017, pp. 1–41. URL: https://tools.ietf.org/html/draft-ietf-opsawg-tacacs-06 (Cited on page 62).

[Dod+12] Yevgeniy Dodis, Thomas Ristenpart, John P. Steinberger, and Stefano Tessaro. "To Hash or Not to Hash Again? (In)Differentiability Results for H^2 and HMAC". In: *Advances in Cryptology – CRYPTO 2012*. Ed. by Reihaneh Safavi-Naini and Ran Canetti. Vol. 7417. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2012, pp. 348–366 (Cited on page 116).

[Dow+15] Benjamin Dowling, Marc Fischlin, Felix Günther, and Douglas Stebila. "A Cryptographic Analysis of the TLS 1.3 Handshake Protocol Candidates". In: *ACM CCS 15: 22nd Conference on Computer and Communications Security*. Ed. by Indrajit Ray, Ninghui Li, and Christopher Kruegel: Denver, CO, USA: ACM Press, Oct. 2015, pp. 1197–1210 (Cited on pages 39, 50, 113, 114, 144).

[Dow+16] Benjamin Dowling, Marc Fischlin, Felix Günther, and Douglas Stebila. *A Cryptographic Analysis of the TLS 1.3 draft-10 Full and Pre-shared Key Handshake Protocol*. Cryptology ePrint Archive, Report 2016/081. http://eprint.iacr.org/2016/081. 2016 (Cited on pages 113, 114, 144).

[DS15] Benjamin Dowling and Douglas Stebila. "Modelling Ciphersuite and Version Negotiation in the TLS Protocol". In: *ACISP 15: 20th Australasian Conference on Information Security and Privacy*. Ed. by Ernest Foo and Douglas Stebila. Vol. 9144. Lecture Notes in Computer Science. Wollongong, NSW, Australia: Springer, Heidelberg, Germany, June 2015, pp. 270–288. DOI: 10.1007/978-3-319-19962-7_16 (Cited on pages 114, 143, 144).

[Eia09] Martin Eian. "Fragility of the Robust Security Network: 802.11 Denial of Service". In: *ACNS 09: 7th International Conference on Applied Cryptography and Network Security*. Ed. by Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud. Vol. 5536. Lecture Notes in Computer Science. Paris-Rocquencourt, France:

Springer, Heidelberg, Germany, June 2009, pp. 400–416 (Cited on page 17).

[Eia10] Martin Eian. "A Practical Cryptographic Denial of Service Attack against 802.11i TKIP and CCMP". In: *CANS 10: 9th International Conference on Cryptology and Network Security*. Ed. by Swee-Huay Heng, Rebecca N. Wright, and Bok-Min Goi. Vol. 6467. Lecture Notes in Computer Science. Kuala Lumpur, Malaysia: Springer, Heidelberg, Germany, Dec. 2010, pp. 62–75 (Cited on page 17).

[EM12] Martin Eian and Stig Frode Mjølsnes. "A formal analysis of IEEE 802.11w deadlock vulnerabilities". In: *Proceedings of the IEEE INFOCOM 2012, Orlando, FL, USA, March 25-30, 2012*. Ed. by Albert G. Greenberg and Kazem Sohraby. IEEE, 2012, pp. 918–926. DOI: 10.1109/INFCOM.2012.6195841. (Cited on page 17).

[FC14] Michèle Feltz and Cas Cremers. *On the Limits of Authenticated Key Exchange Security with an Application to Bad Randomness*. Cryptology ePrint Archive, Report 2014/369. http://eprint.iacr.org/2014/369. 2014 (Cited on pages 37, 53).

[FG14] Marc Fischlin and Felix Günther. "Multi-Stage Key Exchange and the Case of Google's QUIC Protocol". In: *ACM CCS 14: 21st Conference on Computer and Communications Security*. Ed. by Gail-Joon Ahn, Moti Yung, and Ninghui Li. Scottsdale, AZ, USA: ACM Press, Nov. 2014, pp. 1193–1204 (Cited on pages 32, 39, 114).

[FG17] Marc Fischlin and Felix Günther. "Replay Attacks on Zero Round-Trip Time: The Case of the TLS 1.3 Handshake Candidates". In: *2017 IEEE European Symposium on Security and Privacy, EuroS&P 2017, Paris, France, April 26-28, 2017*. IEEE, 2017, pp. 60–75. DOI: 10.1109/EuroSP.2017.18. (Cited on page 144).

[FIPS:197-2001] *Advanced Encryption Standard (AES)*. Tech. rep. Gaithersburg, MD, United States: National Institute of Standards & Technology, Nov. 2001. DOI: 10.6028/NIST.FIPS.197 (Cited on page 23).

[FIPS:SP-800-38B] Morris J. Dworkin. *SP 800-38B. Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication*. Tech. rep. Gaithersburg, MD, United States: National Institute of Standards & Technology, Oct. 2016. DOI: 10.6028/NIST.SP.800-38B (Cited on pages 21, 126, 139).

[Fis+16] 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*. San Jose, CA, USA: IEEE Computer Society Press, May 2016, pp. 452–469. DOI: 10.1109/SP.2016.34 (Cited on page 57).

[FMS01] Scott R. Fluhrer, Itsik Mantin, and Adi Shamir. "Weaknesses in the Key Scheduling Algorithm of RC4". In: *SAC 2001: 8th Annual International Workshop on Selected Areas in Cryptography*. Ed. by Serge Vaudenay and Amr M. Youssef. Vol. 2259. Lecture Notes in Computer Science. Toronto, Ontario, Canada: Springer, Heidelberg, Germany, Aug. 2001, pp. 1–24 (Cited on page 16).

[FW09] Pooya Farshim and Bogdan Warinschi. "Certified Encryption Revisited". In: *AFRICACRYPT 09: 2nd International Conference on Cryptology in Africa*. Ed. by Bart Preneel. Vol. 5580. Lecture Notes in Computer Science. Gammarth, Tunisia: Springer, Heidelberg, Germany, June 2009, pp. 179–197 (Cited on page 30).

[GKS13] Florian Giesen, Florian Kohlar, and Douglas Stebila. "On the security of TLS renegotiation". In: *ACM CCS 13: 20th Conference on Computer and Communications Security*. Ed. by Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung. Berlin, Germany: ACM Press, Nov. 2013, pp. 387–398 (Cited on pages 114, 144).

[GM84] Shafi Goldwasser and Silvio Micali. "Probabilistic Encryption". In: *J. Comput. Syst. Sci.* 28.2 (1984), pp. 270–299. DOI: 10.1016/0022-0000(84)90070-9. (Cited on pages 3, 26).

[GPR14] Peter Gaži, Krzysztof Pietrzak, and Michal Rybár. "The Exact PRF-Security of NMAC and HMAC". In: *Advances in Cryptology – CRYPTO 2014, Part I*. Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8616. Lecture Notes in

Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2014, pp. 113–130. DOI: 10.1007/978-3-662-44371-2_7 (Cited on page 126).

[GR13] Wesley George and Charles Rackoff. *Rethinking Definitions of Security for Session Key Agreement*. Cryptology ePrint Archive, Report 2013/139. http://eprint.iacr.org/2013/139. 2013 (Cited on pages 39, 41, 48).

[Gup+15] Sourav Sen Gupta, Subhamoy Maitra, Willi Meier, Goutam Paul, and Santanu Sarkar. "Dependence in IV-Related Bytes of RC4 Key Enhances Vulnerabilities in WPA". In: *Fast Software Encryption – FSE 2014*. Ed. by Carlos Cid and Christian Rechberger. Vol. 8540. Lecture Notes in Computer Science. London, UK: Springer, Heidelberg, Germany, Mar. 2015, pp. 350–369. DOI: 10.1007/978-3-662-46706-0_18 (Cited on page 16).

[Hal+09] Finn Michael Halvorsen, Olav Haugen, Martin Eian, and Stig Frode Mjølsnes. "An Improved Attack on TKIP". In: *Identity and Privacy in the Internet Age, 14th Nordic Conference on Secure IT Systems, NordSec 2009, Oslo, Norway, 14-16 October 2009. Proceedings*. Ed. by Audun Jøsang, Torleiv Maseng, and Svein J. Knapskog. Vol. 5838. Lecture Notes in Computer Science. Springer, 2009, pp. 120–132. DOI: 10.1007/978-3-642-04766-4_9. (Cited on page 16).

[Har08] Dan Harkins. "Simultaneous Authentication of Equals: A Secure, Password-Based Key Exchange for Mesh Networks". In: *Proceedings of the 2008 Second International Conference on Sensor Technologies and Applications, Cap Esterel, France, 25-31 August 2008*. SENSORCOMM'08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 839–844. ISBN: 978-0-7695-3330-8. DOI: 10.1109/SENSORCOMM.2008.131. (Cited on page 17).

[HC07] Katrin Hoeper and Lidong Chen. "Where EAP security claims fail". In: *4th International ICST Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness, QSHINE 2007, Vancouver, Canada, August 14-17, 2007*. Ed. by Victor Leung and Sastri Kota. ACM, 2007, p. 46. DOI: 10.1145/1577222.1577285. (Cited on page 65).

[HC10] Katrin Hoeper and Lidong Chen. "An inconvenient truth about tunneled authentications". In: *The 35th Annual IEEE Conference on Local Computer Networks, LCN 2010, 10-14 October 2010, Denver, Colorado, USA, Proceedings*. IEEE Computer Society, 2010, pp. 416–423. DOI: 10.1109/LCN.2010.5735754. (Cited on page 65).

[He+05] Changhua He, Mukund Sundararajan, Anupam Datta, Ante Derek, and John C. Mitchell. "A Modular Correctness Proof of IEEE 802.11i and TLS". In: *ACM CCS 05: 12th Conference on Computer and Communications Security*. Ed. by Vijayalakshmi Atluri, Catherine Meadows, and Ari Juels. Alexandria, Virginia, USA: ACM Press, Nov. 2005, pp. 2–15 (Cited on pages 65, 119).

[HK07] Dennis Hofheinz and Eike Kiltz. "Secure Hybrid Encryption from Weakened Key Encapsulation". In: *Advances in Cryptology – CRYPTO 2007*. Ed. by Alfred Menezes. Vol. 4622. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2007, pp. 553–571 (Cited on pages 112, 115).

[HM04] Changhua He and John C. Mitchell. "Analysis of the 802.11i 4-way handshake". In: *Proceedings of the 2004 ACM Workshop on Wireless Security, Philadelphia, PA, USA, October 1, 2004*. Ed. by Markus Jakobsson and Adrian Perrig. ACM, 2004, pp. 43–50. DOI: 10.1145/1023646.1023655. (Cited on page 17).

[HM05] Changhua He and John C. Mitchell. "Security Analysis and Improvements for IEEE 802.11i". In: *ISOC Network and Distributed System Security Symposium – NDSS 2005*. San Diego, CA, USA: The Internet Society, Feb. 2005 (Cited on pages 17, 19).

[Hor+16] Matthias Horst, Martin Grothe, Tibor Jager, and Jörg Schwenk. "Breaking PPTP VPNs via RADIUS Encryption". In: *CANS 16: 15th International Conference on Cryptology and Network Security*. Ed. by Sara Foresti and Giuseppe Persiano. Vol. 10052. Lecture Notes in Computer Science. Milan, Italy: Springer, Heidelberg, Germany, Nov. 2016, pp. 159–175. DOI: 10.1007/978-3-319-48965-0_10 (Cited on pages 66, 148).

[HS15] Dennis Hofheinz and Victor Shoup. "GNUC: A New Universal Composability Framework". In: *Journal of Cryptology* 28.3 (July 2015), pp. 423–508. DOI: 10.1007/s00145-013-9160-y (Cited on page 4).

[IEEE 802.11] "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". In: *IEEE Std 802.11-2012* (Mar. 2012), pp. 1–2793. DOI: 10.1109/IEEESTD.2012.6178212 (Cited on pages 5, 13, 15–18, 88, 134, 143).

[IEEE 802.11i] "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: Amendment 6: Medium Access Control (MAC) Security Enhancements". In: *IEEE Std 802.11i-2004* (July 2004), pp. 1–190. DOI: 10.1109/IEEESTD.2004.94585 (Cited on page 17).

[IEEE 802.11r] "IEEE Standard for Information technology—Local and metropolitan area networks—Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 2: Fast Basic Service Set (BSS) Transition". In: *IEEE Std 802.11r-2008 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008)* (July 2008), pp. 1–126. DOI: 10.1109/IEEESTD.2008.4573292 (Cited on page 17).

[IEEE 802.11s] "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 Amendment 10: Mesh Networking". In: *IEEE Std 802.11s-2011 (Amendment to IEEE Std 802.11-2007 as amended by IEEE 802.11k-2008, IEEE 802.11r-2008, IEEE 802.11y-2008, IEEE 802.11w-2009, IEEE 802.11n-2009, IEEE 802.11p-2010, IEEE 802.11z-2010, IEEE 802.11v-2011,*

and IEEE 802.11u-2011) (Sept. 2011), pp. 1–372. DOI: 10.1109/IEEESTD.2011.6018236 (Cited on page 17).

[IEEE 802.11w] "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 Amendment 4: Protected Management Frames". In: *IEEE Std 802.11w-2009 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, and IEEE Std 802.11y-2008)* (Sept. 2009), pp. 1–111. DOI: 10.1109/IEEESTD.2009.5278657 (Cited on page 17).

[IEEE 802.1X] "IEEE Standard for Local and metropolitan area networks – Port-Based Network Access Control". In: *IEEE Std 802.1X-2010 (Revision of IEEE Std 802.1X-2004)* (Feb. 2010), pp. C1–205. DOI: 10.1109/IEEESTD.2010.5409813 (Cited on page 11).

[IK03a] Tetsu Iwata and Kaoru Kurosawa. "OMAC: One-Key CBC MAC". In: *Fast Software Encryption – FSE 2003*. Ed. by Thomas Johansson. Vol. 2887. Lecture Notes in Computer Science. Lund, Sweden: Springer, Heidelberg, Germany, Feb. 2003, pp. 129–153 (Cited on page 126).

[IK03b] Tetsu Iwata and Kaoru Kurosawa. "Stronger Security Bounds for OMAC, TMAC, and XCBC". In: *Progress in Cryptology - INDOCRYPT 2003: 4th International Conference in Cryptology in India*. Ed. by Thomas Johansson and Subhamoy Maitra. Vol. 2904. Lecture Notes in Computer Science. New Delhi, India: Springer, Heidelberg, Germany, Dec. 2003, pp. 402–415 (Cited on page 126).

[IM15] Ryoma Ito and Atsuko Miyaji. "New Linear Correlations Related to State Information of RC4 PRGA Using IV in WPA". In: *Fast Software Encryption – FSE 2015*. Ed. by Gregor Leander. Vol. 9054. Lecture Notes in Computer Science. Istanbul, Turkey: Springer, Heidelberg, Germany, Mar. 2015, pp. 557–576. DOI: 10.1007/978-3-662-48116-5_27 (Cited on page 16).

[Jag+10] Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk. "Generic Compilers for Authenticated Key Exchange". In: *Advances in Cryptology – ASIACRYPT 2010*. Ed. by Masayuki Abe. Vol. 6477. Lecture Notes in Computer Science. Singapore: Springer, Heidelberg, Germany, Dec. 2010, pp. 232–249 (Cited on page 95).

[Jag+12] Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk. "On the Security of TLS-DHE in the Standard Model". In: *Advances in Cryptology – CRYPTO 2012*. Ed. by Reihaneh Safavi-Naini and Ran Canetti. Vol. 7417. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2012, pp. 273–293 (Cited on pages 4, 29, 39, 53, 54, 88, 92, 111, 112, 135, 144, 154).

[Jag+17] Tibor Jager, Martijn Stam, Ryan Stanley-Oakes, and Bogdan Warinschi. *Multi-Key Authenticated Encryption with Corruptions: Reductions are Lossy*. Cryptology ePrint Archive, Report 2017/495. `http://eprint.iacr.org/2017/495`. 2017 (Cited on pages 147, 148).

[JKL04] Ik Rae Jeong, Jonathan Katz, and Dong Hoon Lee. "One-Round Protocols for Two-Party Authenticated Key Exchange". In: *ACNS 04: 2nd International Conference on Applied Cryptography and Network Security*. Ed. by Markus Jakobsson, Moti Yung, and Jianying Zhou. Vol. 3089. Lecture Notes in Computer Science. Yellow Mountain, China: Springer, Heidelberg, Germany, June 2004, pp. 220–232 (Cited on page 39).

[Joh+15] Tyler Johnson, Daniel Roggow, Phillip H. Jones, and Joseph Zambreno. "An FPGA Architecture for the Recovery of WPA/WPA2 Keys". In: *Journal of Circuits, Systems, and Computers* 24.7 (2015). DOI: `10.1142/S0218126615501054`. (Cited on page 17).

[Jon03] Jakob Jonsson. "On the Security of CTR + CBC-MAC". In: *SAC 2002: 9th Annual International Workshop on Selected Areas in Cryptography*. Ed. by Kaisa Nyberg and Howard M. Heys. Vol. 2595. Lecture Notes in Computer Science. St. John's, Newfoundland, Canada: Springer, Heidelberg, Germany, Aug. 2003, pp. 76–93 (Cited on pages 135, 138, 152).

[JV96] Mike Just and Serge Vaudenay. "Authenticated Multi-Party Key Agreement". In: *Advances in Cryptology – ASIACRYPT'96*. Ed. by Kwangjo Kim and Tsutomu Matsumoto. Vol. 1163. Lecture Notes in Computer Science. Kyongju, Korea: Springer, Heidelberg, Germany, Nov. 1996, pp. 36–49 (Cited on pages 35, 49).

[Kam+16] Markus Kammerstetter, Markus Muellner, Daniel Burian, Christian Kudera, and Wolfgang Kastner. "Efficient High-Speed WPA2 Brute Force Attacks Using Scalable Low-Cost FPGA Clustering". In: *Cryptographic Hardware and Embedded Systems – CHES 2016*. Ed. by Benedikt Gierlichs and Axel Y. Poschmann. Vol. 9813. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2016, pp. 559–577. DOI: 10.1007/978-3-662-53140-2_27 (Cited on page 17).

[Koh+15] Markulf Kohlweiss, Ueli Maurer, Cristina Onete, Björn Tackmann, and Daniele Venturi. "(De-)Constructing TLS 1.3". In: *Progress in Cryptology - INDOCRYPT 2015: 16th International Conference in Cryptology in India*. Ed. by Alex Biryukov and Vipul Goyal. Vol. 9462. Lecture Notes in Computer Science. Bangalore, India: Springer, Heidelberg, Germany, Dec. 2015, pp. 85–102. DOI: 10.1007/978-3-319-26617-6_5 (Cited on page 114).

[KPW13a] Hugo Krawczyk, Kenneth G. Paterson, and Hoeteck Wee. *On the Security of the TLS Protocol: A Systematic Analysis*. Cryptology ePrint Archive, Report 2013/339. http://eprint.iacr.org/2013/339. 2013 (Cited on page 70).

[KPW13b] Hugo Krawczyk, Kenneth G. Paterson, and Hoeteck Wee. "On the Security of the TLS Protocol: A Systematic Analysis". In: *Advances in Cryptology – CRYPTO 2013, Part I*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8042. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2013, pp. 429–448. DOI: 10.1007/978-3-642-40041-4_24 (Cited on pages 39, 54, 88, 95, 112, 115, 144).

[Kra01] Hugo Krawczyk. "The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?)" In: *Advances in Cryptology – CRYPTO 2001*. Ed. by Joe Kilian. Vol. 2139. Lecture Notes in Computer

Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2001, pp. 310–331 (Cited on page 144).

[Kra03] Hugo Krawczyk. "SIGMA: The "SIGn-and-MAc" Approach to Authenticated Diffie-Hellman and Its Use in the IKE Protocols". In: *Advances in Cryptology – CRYPTO 2003*. Ed. by Dan Boneh. Vol. 2729. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2003, pp. 400–425 (Cited on page 57).

[Kra05a] Hugo Krawczyk. *HMQV: A High-Performance Secure Diffie-Hellman Protocol*. Cryptology ePrint Archive, Report 2005/176. `http://eprint.iacr.org/2005/176`. 2005 (Cited on page 51).

[Kra05b] Hugo Krawczyk. "HMQV: A High-Performance Secure Diffie-Hellman Protocol". In: *Advances in Cryptology – CRYPTO 2005*. Ed. by Victor Shoup. Vol. 3621. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2005, pp. 546–566 (Cited on pages 29, 39, 51, 64).

[Kra16] Hugo Krawczyk. "A Unilateral-to-Mutual Authentication Compiler for Key Exchange (with Applications to Client Authentication in TLS 1.3)". In: *ACM CCS 16: 23rd Conference on Computer and Communications Security*. Ed. by Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi. Vienna, Austria: ACM Press, Oct. 2016, pp. 1438–1450 (Cited on pages 50, 95).

[KSS09] Kazukuni Kobara, SeongHan Shin, and Mario Strefler. "Partnership in key exchange protocols". In: *ASIACCS 09: 4th ACM Symposium on Information, Computer and Communications Security*. Ed. by Wanqing Li, Willy Susilo, Udaya Kiran Tupakula, Reihaneh Safavi-Naini, and Vijay Varadharajan. Sydney, Australia: ACM Press, Mar. 2009, pp. 161–170 (Cited on pages 39, 41, 43).

[KSS13] 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. `http://eprint.iacr.org/2013/367`. 2013 (Cited on pages 54, 88, 92, 112, 113, 144).

[KT11a] Ralf Küsters and Max Tuengerthal. "Composition theorems without pre-established session identifiers". In: *ACM CCS 11: 18th Conference on Computer and Communications Security*. Ed. by Yan Chen, George Danezis, and Vitaly Shmatikov. Chicago, Illinois, USA: ACM Press, Oct. 2011, pp. 41–50 (Cited on pages 95, 119).

[KT11b] Ralf Küsters and Max Tuengerthal. "Ideal Key Derivation and Encryption in Simulation-Based Security". In: *Topics in Cryptology – CT-RSA 2011*. Ed. by Aggelos Kiayias. Vol. 6558. Lecture Notes in Computer Science. San Francisco, CA, USA: Springer, Heidelberg, Germany, Feb. 2011, pp. 161–179 (Cited on page 119).

[KT13] Ralf Kuesters and Max Tuengerthal. *The IITM Model: a Simple and Expressive Model for Universal Composability*. Cryptology ePrint Archive, Report 2013/025. `http://eprint.iacr.org/2013/025`. 2013 (Cited on page 4).

[KW15] Hugo Krawczyk and Hoeteck Wee. *The OPTLS Protocol and TLS 1.3*. Cryptology ePrint Archive, Report 2015/978. `http://eprint.iacr.org/2015/978`. 2015 (Cited on pages 113, 144).

[KY03] Jonathan Katz and Moti Yung. "Scalable Protocols for Authenticated Group Key Exchange". In: *Advances in Cryptology – CRYPTO 2003*. Ed. by Dan Boneh. Vol. 2729. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2003, pp. 110–125 (Cited on page 95).

[Law+03] Laurie Law, Alfred Menezes, Minghua Qu, Jerome A. Solinas, and Scott A. Vanstone. "An Efficient Protocol for Authenticated Key Agreement". In: *Des. Codes Cryptography* 28.2 (2003), pp. 119–134 (Cited on page 29).

[Li+14] Yong Li, Sven Schäge, Zheng Yang, Florian Kohlar, and Jörg Schwenk. "On the Security of the Pre-shared Key Ciphersuites of TLS". In: *PKC 2014: 17th International Conference on Theory and Practice of Public Key Cryptography*. Ed. by Hugo Krawczyk. Vol. 8383. Lecture Notes in Computer Science. Buenos Aires, Argentina: Springer, Heidelberg, Germany, Mar. 2014, pp. 669–684. DOI: `10.1007/978-3-642-54631-0_38` (Cited on pages 54, 88, 92, 112, 113, 144).

[LLM07] Brian A. LaMacchia, Kristin Lauter, and Anton Mityagin. "Stronger Security of Authenticated Key Exchange". In: *ProvSec 2007: 1st International Conference on Provable Security*. Ed. by Willy Susilo, Joseph K. Liu, and Yi Mu. Vol. 4784. Lecture Notes in Computer Science. Wollongong, Australia: Springer, Heidelberg, Germany, Nov. 2007, pp. 1–16 (Cited on pages 4, 39, 51, 53).

[LM06] Kristin Lauter and Anton Mityagin. "Security Analysis of KEA Authenticated Key Exchange Protocol". In: *PKC 2006: 9th International Conference on Theory and Practice of Public Key Cryptography*. Ed. by Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin. Vol. 3958. Lecture Notes in Computer Science. New York, NY, USA: Springer, Heidelberg, Germany, Apr. 2006, pp. 378–394 (Cited on page 39).

[Lyc+15] Robert Lychev, Samuel Jero, Alexandra Boldyreva, and Cristina Nita-Rotaru. "How Secure and Quick is QUIC? Provable Security and Performance Analyses". In: *2015 IEEE Symposium on Security and Privacy*. San Jose, CA, USA: IEEE Computer Society Press, May 2015, pp. 214–231. DOI: 10.1109/SP.2015.21 (Cited on page 54).

[Mei+13] Simon Meier, Benedikt Schmidt, Cas Cremers, and David A. Basin. "The TAMARIN Prover for the Symbolic Analysis of Security Protocols". In: *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*. Ed. by Natasha Sharygina and Helmut Veith. Vol. 8044. Lecture Notes in Computer Science. Springer, 2013, pp. 696–701. DOI: 10.1007/978-3-642-39799-8_48. (Cited on page 2).

[Mis+04] Arunesh Mishra, Nick L. Petroni Jr., William A. Arbaugh, and Timothy Fraser. "Security issues in IEEE 802.11 wireless local area networks: a survey". In: *Wireless Communications and Mobile Computing* 4.8 (2004), pp. 821–833. DOI: 10.1002/wcm.257. (Cited on page 16).

[MO11] Daisuke Moriyama and Tatsuaki Okamoto. "Leakage resilient eCK-secure key exchange protocol without random oracles (Short Paper)". In: *ASIACCS 11: 6th ACM Symposium on Information, Computer and Communications Security*. Ed. by Bruce S. N. Cheung, Lucas Chi Kwong

Hui, Ravi S. Sandhu, and Duncan S. Wong. Hong Kong, China: ACM Press, Mar. 2011, pp. 441–447 (Cited on page 53).

[MRH04] Vebjørn Moen, Håvard Raddum, and Kjell J. Hole. "Weaknesses in the Temporal Key Hash of WPA". In: *SIGMOBILE Mob. Comput. Commun. Rev.* 8.2 (Apr. 2004), pp. 76–83. ISSN: 1559-1662. DOI: 10.1145/997122.997132. (Cited on page 16).

[MSW10] Paul Morrissey, Nigel P. Smart, and Bogdan Warinschi. "The TLS Handshake Protocol: A Modular Analysis". In: *Journal of Cryptology* 23.2 (Apr. 2010), pp. 187–223 (Cited on pages 94, 144).

[MT11] Masakatu Morii and Yosuke Todo. "Cryptanalysis for RC4 and Breaking WEP/WPA-TKIP". In: *IEICE Transactions* 94-D.11 (2011), pp. 2087–2094. URL: http://search.ieice.org/bin/summary.php?id=e94-d_11_2087 (Cited on page 16).

[MU06] Alfred Menezes and Berkant Ustaoglu. "On the Importance of Public-Key Validation in the MQV and HMQV Key Agreement Protocols". In: *Progress in Cryptology - INDOCRYPT 2006: 7th International Conference in Cryptology in India*. Ed. by Rana Barua and Tanja Lange. Vol. 4329. Lecture Notes in Computer Science. Kolkata, India: Springer, Heidelberg, Germany, Dec. 2006, pp. 133–147 (Cited on page 30).

[MU08] Alfred Menezes and Berkant Ustaoglu. "Security arguments for the UM key agreement protocol in the NIST SP 800-56A standard". In: *ASIACCS 08: 3rd ACM Symposium on Information, Computer and Communications Security*. Ed. by Masayuki Abe and Virgil Gligor. Tokyo, Japan: ACM Press, Mar. 2008, pp. 261–270 (Cited on page 39).

[MvV96] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. Boca Raton, Florida: CRC Press, 1996 (Cited on page 35).

[OPY06] Yoshihiro Ohba, Mohan Parthasarathy, and Mayumi Yanagiya. *Channel Binding Mechanism based on Parameter Binding in Key Derivation*. Internet-draft. RFC Editor, Dec. 2006, pp. 1–18. URL: https://tools.ietf.org/

html/draft-ohba-eap-channel-binding-02 (Cited on page 90).

[PEAPv2] Dan Simon, Glen Zorn, Simon Josefsson, Hao Zhou, and Joseph Salowey. *Protected EAP Protocol (PEAP) Version 2*. Internet-draft. RFC Editor, Oct. 2004, pp. 1–87. URL: https://tools.ietf.org/html/draft-josefsson-pppext-eap-tls-eap-10 (Cited on pages 12, 65).

[PPS15] Kenneth G. Paterson, Bertram Poettering, and Jacob C. N. Schuldt. "Plaintext Recovery Attacks Against WPA/TKIP". In: *Fast Software Encryption – FSE 2014*. Ed. by Carlos Cid and Christian Rechberger. Vol. 8540. Lecture Notes in Computer Science. London, UK: Springer, Heidelberg, Germany, Mar. 2015, pp. 325–349. DOI: 10.1007/978-3-662-46706-0_17 (Cited on page 16).

[PR07] Manoj Prabhakaran and Mike Rosulek. "Rerandomizable RCCA Encryption". In: *Advances in Cryptology – CRYPTO 2007*. Ed. by Alfred Menezes. Vol. 4622. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2007, pp. 517–534 (Cited on page 113).

[PRS11] Kenneth G. Paterson, Thomas Ristenpart, and Thomas Shrimpton. "Tag Size Does Matter: Attacks and Proofs for the TLS Record Protocol". In: *Advances in Cryptology – ASIACRYPT 2011*. Ed. by Dong Hoon Lee and Xiaoyun Wang. Vol. 7073. Lecture Notes in Computer Science. Seoul, South Korea: Springer, Heidelberg, Germany, Dec. 2011, pp. 372–389 (Cited on pages 144, 151, 152, 154).

[Res17] Eric Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3 draft-ietf-tls-tls13-21*. Internet-draft. RFC Editor, July 2017, pp. 1–143. URL: https://tools.ietf.org/html/draft-ietf-tls-tls13-21 (Cited on page 113).

[RFC2104] Hugo Krawczyk, Mihir Bellare, and Ran Canetti. *HMAC: Keyed-Hashing for Message Authentication*. Internet Requests for Comments. RFC 2104. Feb. 1997. URL: https://tools.ietf.org/html/rfc2104 (Cited on pages 21, 115, 126, 139).

[RFC2409] Dan Harkins and Dave Carrel. *The Internet Key Exchange (IKE)*. RFC 2409. RFC Editor, Nov. 1998, pp. 1–41. URL: https://tools.ietf.org/html/rfc2409 (Cited on page 142).

[RFC2548] Glen Zorn. *Microsoft Vendor-specific RADIUS Attributes*. Internet Requests for Comments. RFC 2548. Mar. 1999. URL: https://tools.ietf.org/html/rfc2548 (Cited on pages 14, 62).

[RFC2759] Glen Zorn. *Microsoft PPP CHAP Extensions, Version 2*. RFC 2759. RFC Editor, Jan. 2000, pp. 1–20. URL: https://tools.ietf.org/html/rfc2759 (Cited on page 12).

[RFC2865] Allan C. Rubens, William Allen Simpson, and Steve Willens. *Remote Authentication Dial In User Service (RADIUS)*. RFC 2865. RFC Editor, June 2000, pp. 1–76. URL: https://tools.ietf.org/html/rfc2865 (Cited on pages 12, 62).

[RFC3394] Jim Schaad and Russell Housley. *Advanced Encryption Standard (AES) Key Wrap Algorithm*. RFC 3394. RFC Editor, Sept. 2002, pp. 1–41. URL: https://tools.ietf.org/html/rfc3394 (Cited on pages 22, 139).

[RFC3579] Bernard Aboba and Pat R. Calhoun. *RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)*. RFC 3579. RFC Editor, Sept. 2003, pp. 1–46. URL: https://tools.ietf.org/html/rfc3579 (Cited on page 62).

[RFC3610] Doug Whiting, Russell Housley, and Niels Ferguson. *Counter with CBC-MAC (CCM)*. RFC 3610. RFC Editor, Sept. 2003, pp. 1–26. URL: https://tools.ietf.org/html/rfc3610 (Cited on pages 23, 133).

[RFC3748] Bernard Aboba, Larry J. Blunk, John R. Vollbrecht, James Carlson, and Henrik Levkowetz. *Extensible Authentication Protocol (EAP)*. RFC 3748. RFC Editor, June 2004, pp. 1–67. URL: https://tools.ietf.org/html/rfc3748 (Cited on pages 5, 9, 10, 12, 16, 59, 60, 63, 65).

[RFC4187] Jari Arkko and Henry Haverinen. *Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA)*. RFC 4187. RFC Editor, Jan. 2006, pp. 1–79. URL: https://tools.ietf.org/html/rfc4187 (Cited on page 12).

[RFC4253] Tatu Ylonen and Chris Lonvick. *The Secure Shell (SSH) Transport Layer Protocol*. RFC 4253. RFC Editor, Jan. 2006, pp. 1–32. URL: https://tools.ietf.org/html/rfc4253 (Cited on page 143).

[RFC4764] Florent Bersani and Hannes Tschofenig. *The EAP-PSK Protocol: A Pre-Shared Key Extensible Authentication Protocol (EAP) Method*. RFC 4764. RFC Editor, Jan. 2007, pp. 1–64. URL: https://tools.ietf.org/html/rfc4764 (Cited on page 12).

[RFC5106] Hannes Tschofenig, Dirk Kroeselberg, Andreas Pashalidis, Yoshihiro Ohba, and Florent Bersani. *The Extensible Authentication Protocol-Internet Key Exchange Protocol version 2 (EAP-IKEv2) Method*. RFC 5106. RFC Editor, Feb. 2008, pp. 1–33. URL: https://tools.ietf.org/html/rfc5106 (Cited on pages 12, 91).

[RFC5216] Dan Simon, Bernard Aboba, and Ryan Hurst. *The EAP-TLS Authentication Protocol*. RFC 5216. RFC Editor, Mar. 2008, pp. 1–34. URL: https://tools.ietf.org/html/rfc5216 (Cited on pages 6, 11, 12, 60, 88, 93, 110).

[RFC5246] Tim Dierks and Eric Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.2*. RFC 5246. RFC Editor, Aug. 2008, pp. 1–104. URL: https://tools.ietf.org/html/rfc5246 (Cited on pages 5, 53).

[RFC5247] Bernard Aboba, Dan Simon, and Pasi Eronen. *Extensible Authentication Protocol (EAP) Key Management Framework*. RFC 5247. RFC Editor, Aug. 2008, pp. 1–79. URL: https://tools.ietf.org/html/rfc5247 (Cited on pages 60, 63, 144).

[RFC5281] Paul Funk and Simon Blake-Wilson. *Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0)*. RFC 5281. RFC Editor, Aug. 2008, pp. 1–51. URL: https://tools.ietf.org/html/rfc5281 (Cited on pages 12, 65).

[RFC5297] Dan Harkins. *Synthetic Initialization Vector (SIV) Authenticated Encryption Using the Advanced Encryption Standard (AES)*. RFC 5297. RFC Editor, Oct. 2008, pp. 1–26. URL: https://tools.ietf.org/html/rfc5297 (Cited on page 17).

[RFC5705] Eric Rescorla. *Keying Material Exporters for Transport Layer Security (TLS)*. RFC 5705. RFC Editor, Mar. 2010, pp. 1–7. URL: https://tools.ietf.org/html/rfc5705 (Cited on pages 92, 93, 110, 113).

[RFC6614] Stefan Winter, Mike McCauley, Stig Venaas, and Klaas Wierenga. *Transport Layer Security (TLS) Encryption for RADIUS*. RFC 6614. RFC Editor, May 2012, pp. 1–22. URL: https://tools.ietf.org/html/rfc6614 (Cited on pages 62, 88, 113).

[RFC6677] Sam Hartman (editor), T. Charles Clancy, and Katrin Hoeper. *Channel-Binding Support for Extensible Authentication Protocol (EAP) Methods*. RFC 6677. RFC Editor, July 2012, pp. 1–31. URL: https://tools.ietf.org/html/rfc6677 (Cited on pages 61, 90).

[RFC6733] Victor Fajardo, Jari Arkko, John Loughney, and Glen Zorn. *Diameter Base Protocol*. RFC 6733. RFC Editor, Oct. 2012, pp. 1–152. URL: https://tools.ietf.org/html/rfc6733 (Cited on pages 12, 62).

[RFC7296] Charlie Kaufman, Paul Hoffman, Yoav Nir, Pasi Eronen, and Tero Kivinen. *Internet Key Exchange Protocol Version 2 (IKEv2)*. RFC 7296. RFC Editor, Oct. 2014, pp. 1–142. URL: https://tools.ietf.org/html/rfc7296 (Cited on page 12).

[RFC7593] Klaas Wierenga, Stefan Winter, and Tomasz Wolniewicz. *The eduroam Architecture for Network Roaming*. RFC 7593. RFC Editor, Sept. 2015, pp. 1–37. URL: https://tools.ietf.org/html/rfc7593 (Cited on page 148).

[RFC8018] Kathleen M. Moriarty, Burt Kaliski, and Andreas Rusch. *PKCS #5: Password-Based Cryptography Specification Version 2.1*. RFC 8018. RFC Editor, Jan. 2017, pp. 1–40. URL: https://tools.ietf.org/html/rfc8018 (Cited on pages 20, 119).

[RLM06] Floriano De Rango, Dionigi Cristian Lentini, and Salvatore Marano. "Static and Dynamic 4-Way Handshake Solutions to Avoid Denial of Service Attack in Wi-Fi Protected Access and IEEE 802.11i". In: *EURASIP J. Wireless Comm. and Networking* 2006 (2006). DOI: 10.1155/WCN/2006/47453. (Cited on page 17).

[Rog04] Phillip Rogaway. "On the of Role of Definitions in and Beyond Cryptography". In: *ASIAN*. Vol. 3321. Lecture Notes in Computer Science. Springer, 2004, pp. 13–32 (Cited on pages 38, 57).

[Rog06] Phillip Rogaway. "Formalizing Human Ignorance". In: *Progress in Cryptology - VIETCRYPT 06: 1st International Conference on Cryptology in Vietnam*. Ed. by Phong Q. Nguyen. Vol. 4341. Lecture Notes in Computer Science. Hanoi, Vietnam: Springer, Heidelberg, Germany, Sept. 2006, pp. 211–228 (Cited on page 27).

[RS06a] Phillip Rogaway and Thomas Shrimpton. "A Provable-Security Treatment of the Key-Wrap Problem". In: *Advances in Cryptology – EUROCRYPT 2006*. Ed. by Serge Vaudenay. Vol. 4004. Lecture Notes in Computer Science. St. Petersburg, Russia: Springer, Heidelberg, Germany, May 2006, pp. 373–390 (Cited on page 152).

[RS06b] Phillip Rogaway and Thomas Shrimpton. *Deterministic Authenticated-Encryption: A Provable-Security Treatment of the Key-Wrap Problem*. Cryptology ePrint Archive, Report 2006/221. `http://eprint.iacr.org/2006/221`. 2006 (Cited on page 153).

[RS09] Phillip Rogaway and Till Stegers. "Authentication without Elision: Partially Specified Protocols, Associated Data, and Cryptographic Models Described by Code". In: *Proceedings of the 22nd IEEE Computer Security Foundations Symposium, CSF 2009, Port Jefferson, New York, USA, July 8-10, 2009*. IEEE Computer Society, 2009, pp. 26–39. DOI: `10.1109/CSF.2009.23`. (Cited on page 39).

[Sep+14] Pouyan Sepehrdad, Petr Susil, Serge Vaudenay, and Martin Vuagnoux. "Smashing WEP in a Passive Attack". In: *Fast Software Encryption – FSE 2013*. Ed. by Shiho Moriai. Vol. 8424. Lecture Notes in Computer Science. Singapore: Springer, Heidelberg, Germany, Mar. 2014, pp. 155–178. DOI: `10.1007/978-3-662-43933-3_9` (Cited on page 16).

[Sho04] Victor Shoup. *Sequences of games: a tool for taming complexity in security proofs*. Cryptology ePrint Archive, Report 2004/332. `http://eprint.iacr.org/2004/332`. 2004 (Cited on pages 25, 69).

[Sho99] Victor Shoup. *On Formal Models for Secure Key Exchange*. Cryptology ePrint Archive, Report 1999/012. http://eprint.iacr.org/1999/012. 1999 (Cited on page 4).

[SIR02] Adam Stubblefield, John Ioannidis, and Aviel D. Rubin. "Using the Fluhrer, Mantin, and Shamir Attack to Break WEP". In: *ISOC Network and Distributed System Security Symposium – NDSS 2002*. San Diego, CA, USA: The Internet Society, Feb. 2002 (Cited on page 16).

[SIR04] Adam Stubblefield, John Ioannidis, and Aviel D. Rubin. "A key recovery attack on the 802.11b wired equivalent privacy protocol (WEP)". In: *ACM Trans. Inf. Syst. Secur.* 7.2 (2004), pp. 319–332. DOI: 10.1145/996943.996948. (Cited on page 16).

[SR96] Victor Shoup and Aviel D. Rubin. "Session Key Distribution Using Smart Cards". In: *Advances in Cryptology – EUROCRYPT'96*. Ed. by Ueli M. Maurer. Vol. 1070. Lecture Notes in Computer Science. Saragossa, Spain: Springer, Heidelberg, Germany, May 1996, pp. 321–331 (Cited on page 39).

[SVV11] Pouyan Sepehrdad, Serge Vaudenay, and Martin Vuagnoux. "Statistical Attack on RC4 - Distinguishing WPA". In: *Advances in Cryptology – EUROCRYPT 2011*. Ed. by Kenneth G. Paterson. Vol. 6632. Lecture Notes in Computer Science. Tallinn, Estonia: Springer, Heidelberg, Germany, May 2011, pp. 343–363 (Cited on page 16).

[TB09] Erik Tews and Martin Beck. "Practical Attacks Against WEP and WPA". In: *Proceedings of the Second ACM Conference on Wireless Network Security*. WiSec '09. Zurich, Switzerland: ACM, 2009, pp. 79–86. ISBN: 978-1-60558-460-7. DOI: 10.1145/1514274.1514286. (Cited on page 16).

[Tew07] Erik Tews. *Attacks on the WEP protocol*. Cryptology ePrint Archive, Report 2007/471. http://eprint.iacr.org/2007/471. 2007 (Cited on page 16).

[Tod+12] Yosuke Todo, Yuki Ozawa, Toshihiro Ohigashi, and Masakatu Morii. "Falsification Attacks against WPA-TKIP in a Realistic Environment". In: *IEICE Transactions* 95-D.2 (2012), pp. 588–595. URL: http://search.

ieice.org/bin/summary.php?id=e95-d_2_588 (Cited on page 16).

[TWP08] Erik Tews, Ralf-Philipp Weinmann, and Andrei Pyshkin. "Breaking 104 Bit WEP in Less Than 60 Seconds". In: *WISA 07: 8th International Workshop on Information Security Applications*. Ed. by Sehun Kim, Moti Yung, and Hyung-Woo Lee. Vol. 4867. Lecture Notes in Computer Science. Jeju Island, Korea: Springer, Heidelberg, Germany, Aug. 2008, pp. 188–202 (Cited on page 16).

[VP13] Mathy Vanhoef and Frank Piessens. "Practical verification of WPA-TKIP vulnerabilities". In: *ASIACCS 13: 8th ACM Symposium on Information, Computer and Communications Security*. Ed. by Kefei Chen, Qi Xie, Weidong Qiu, Ninghui Li, and Wen-Guey Tzeng. Hangzhou, China: ACM Press, May 2013, pp. 427–436 (Cited on page 16).

[VP15] Mathy Vanhoef and Frank Piessens. "All Your Biases Belong to Us: Breaking RC4 in WPA-TKIP and TLS". In: *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015*. Ed. by Jaeyeon Jung and Thorsten Holz. USENIX Association, 2015, pp. 97–112. URL: https://www.usenix.org/system/files/conference/usenixsecurity15/sec15-paper-vanhoef.pdf (Cited on page 16).

[VP16] Mathy Vanhoef and Frank Piessens. "Predicting, Decrypting, and Abusing WPA2/802.11 Group Keys". In: *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*. Ed. by Thorsten Holz and Stefan Savage. USENIX Association, 2016, pp. 673–688. URL: https://www.usenix.org/system/files/conference/usenixsecurity16/sec16_paper_vanhoef.pdf (Cited on page 17).

[Wal00] Jesse Walker. "Unsafe at any key size; An analysis of the WEP encapsulation". Oct. 2000. URL: https://www.csee.umbc.edu/courses/graduate/CMSC691A/Spring04/papers/wep-problems.pdf (Cited on page 16).

[Woo04] Avishai Wool. "A note on the fragility of the "Michael" message integrity code". In: *IEEE Trans. Wireless Communications* 3.5 (2004), pp. 1459–1462. DOI: 10.1109/TWC.2004.833470. (Cited on page 16).

[Woo08] Mark Wooding. *New proofs for old modes*. Cryptology ePrint Archive, Report 2008/121. http://eprint.iacr.org/2008/121. 2008 (Cited on page 62).

[ZMM05] Fan Zhang, Jianfeng Ma, and Sang-Jae Moon. "The Security Proof of a 4-Way Handshake Protocol in IEEE 802.11i". In: *Computational Intelligence and Security, International Conference, CIS 2005, Xi'an, China, December 15-19, 2005, Proceedings, Part II*. Ed. by Yue Hao, Jiming Liu, Yuping Wang, Yiu-ming Cheung, Hujun Yin, Licheng Jiao, Jianfeng Ma, and Yong-Chang Jiao. Vol. 3802. Lecture Notes in Computer Science. Springer, 2005, pp. 488–493. DOI: 10.1007/11596981_72. (Cited on page 119).