Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

# Analysis of the EDHOC Lightweight Authenticated Key Exchange Protocol

Master Thesis

Marc Ilunga Tshibumbu Mukendi

August 2, 2022

Advisors: Prof. Dr. Kaveh Razavi
Prof. Dr. Kenny Paterson
Dr. Felix Günther

Applied Cryptography Group
Institute of Information Security
Department of Computer Science, ETH Zürich

## Abstract

A formal cryptographic analysis of the Ephemeral Diffie-Hellman Over COSE protocol in SIG-SIG mode is presented. More specifically, we analyze the protocol specification in draft 14 [SMP22a] to which we add minor changes that will be added to the protocol specification [Mat22]. EDHOC is a lightweight key exchange protocol for constrained environments, promising to be suitable for low-powered devices operating in restricted networks. The design of EDHOC is informed by the SIGMA [Kra03] family of key exchange protocols. Protocol participants authenticate to their peers using either a long-term signing key pair or a long-term Diffie-Hellman key. In this thesis, we analyze the EDHOC SIG-SIG variant wherein both the protocol initiator and the responder mutually authenticate with digital signatures. This protocol variant is an instantiation of the "MAc-then-SIGn" variant of SIGMA. We employ the Multi-Stage Key Exchange model [FG14, Gü18] to analyze the EDHOC SIG-SIG protocol. We pay special attention to the use of so-called "credential identifiers" that are short, not necessarily unique, identifiers of a user's long-term verification key and are used in the protocol to limit the bandwidth overhead. To this end, we design a tailored MSKE model for EDHOC; primarily inspired by analysis of TLS 1.3 [DFGS21] by Dowling *et al.*, we make use of further extensions to the MSKE [DDGJ22, DG21] to capture multiple security properties in a single game. Our analysis shows that EDHOC SIG-SIG is structurally sound and a secure instantiation of the "MAc-then-SIGn" protocol. Our proof shows that the EDHOC SIG-SIG protocol achieves the desired security properties including key indistinguishability, explicit authentication, and forward secrecy. Along the way, we gain insights of independent interest on the "MAc-then-SIGn" protocol and some of its shortcomings in contrast to other instantiations of the SIGMA protocol. Finally, we discuss our involvement with the LAKE working group at the IETF and our contributions to the EDHOC's specification resulting from the insights of our analysis.

## Acknowledgments

First and foremost, I thank my supervisor, Dr. Felix Günther. He patiently guided me during my thesis project for six months, even when the project stagnated for a long time. He was generous with his time and contributed many ideas, and his expertise in protocol analysis and key exchange protocol was invaluable. He supervised my interaction with the LAKE working group, and I cherish the freedom I had to learn and explore cryptography beyond the scope of the thesis. Moreover, he encouraged me to maintain a good life balance. This thesis would not have been possible without our shared passion for trains and working in exotic places.

I thank Prof. Kenny Paterson for the opportunity to do my thesis with the Applied Cryptography group; his applied cryptography lecture strengthened my existing passion for cryptography. I enjoyed my time in the group and the friendly chats. The weekly presentation by talented students and guests made the experience even more enjoyable.

I want to thank Prof. Kaveh Razavi for his supervision and facilitation of the administration at the ITET department.

I am grateful for the LAKE working group at the IETF and their openness and for welcoming our contributions.

My family has supported and encouraged me in virtually every way. My parents arranged for seventeen-year-old me to come to Switzerland to study science and technology. I wonder if they realized the stress my endeavor would cause them; they supported me through it all and gracefully. I thank my lovely partner Maya for her precious support during my studies. Thank you for putting up with me during these many semesters (I stopped counting) and accepting the inconvenient vacation schedule of ETH students. I love you! I am thankful to Rob and Helen Fielding for supporting Maya and me throughout my studies. My friends from C3 Zürich have been a great source of encouragement. Thank you, Eszter and Satya, for allowing me to take your new baby Dani with me on my thinking walks. The swing dancers at Langstrassenkultur helped me keep my mental sanity. The list of people I would like to thank is too large to fit on this page. Nevertheless, thank you to everyone who took an interest in my thesis, wanting to know what I was doing. I do, however, apologize for long rants on indifferentiability and how it is related to chopping onions, and in the end, I would still not have properly explained what I was working on. The opportunity to share one's passion sometimes comes with risks, but I shall work on improving that.

I am deeply indebted to the electrical engineering department at ETH Zürich for giving me a second chance to complete my Master's degree. Finally, thank you, Felix, Maya, Giacomo, and Helen, for proofreading this thesis.

*Act justly, love mercy, walk humbly.*

# Contents

Chapter 1

# Introduction

## 1.1 Overview and Motivation

### 1.1.1 Problem statement

**The proliferation of low-powered devices.** In the last decade, we have observed a tremendous proliferation of low-powered computing devices. Devices such as smart fridges or smart lightbulbs, colloquially referred to as "Internet of Things" (IoT), are becoming ubiquitous in many public spaces as well as private homes. Besides computational restrictions, IoT devices often operate in environments with network constraints. For instance, several IoT deployments operate on Low-Power Wide Area Networks such as LoRaWAN[1]. Cisco, a network equipment manufacturing company, forecasts in their 2020 annual report[2] that that by 2023 machine-to-machine communications will account for nearly half of the global internet connections.

**Poor security track record.** Although this increase in connectedness brings along numerous fascinating applications and a certain level of convenience, it also brings its share of challenges. Devices that operate in constrained environments are designed to offer a limited number of functionalities. With computational and even bandwidth constraints, the security of IoT applications is generally out of the picture. Hence, the famous saying *"the 'S' in IoT stands for security"*. Consequently, insecure IoT devices create newer and larger attack surfaces. For instance, applications for low-powered devices that transfer data over an insecure channel pose great dangers to privacy. Firmware updates over insecure channels can be exploited by network attackers to take control of the device. Because of their ubiquity, compromised

[1]LoRaWAN specification: `https://lora-alliance.org/about-lorawan/`.

[2]Cisco, annual report 2020: `https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html`.

devices give attackers access to further networks, compounding the initial and relatively low-cost compromise of the device.

**Towards securing communications in constrained environments.** Providing secure communications in restricted environments comes with two main challenges: first, the secure protocol designer is constrained to a limited set of efficient cryptographic primitives available on low-end devices. Second, the overhead of said protocols (bandwidth and round trips) must be kept at a minimum. A direct implication is a potential incompatibility with existing secure protocols; such as TLS 1.3 [Res18] due to the size of the message sent during the handshake; consequently, new tools and protocols that meet the numerous demands of constrained environments are needed. To this effect, the IETF set out to standardize protocols and efficient communications data formats for constrained devices. Of relevance is the Object Security for Constrained RESTful Environments protocol, (OSCORE) [SMPS19] protocol. Similar to the TLS 1.3 record protocol, the OSCORE protocol provides security of application data. OSCORE relies on the Concise Binary Object Representation (CBOR) [BH20] data format to keep the bandwidth overhead low. Moreover, the CBOR Object Signing and Encryption (COSE) [Sch17] provides a standard way to perform cryptographic operations atop CBOR objects. As a secure transport protocol, OSCORE requires that protocol participants have established a so-called security context. At a high level, this means the OSCORE requires an established session key. It stands then that an essential missing piece in this technological chain for securing communications in constrained devices is a lightweight and secure key exchange protocol.

**EDHOC, an effort to close the gap.** The Ephemeral Diffie–Hellman Over COSE protocol, or EDHOC, is an effort toward closing the gap due to a missing key exchange protocol adequate for constrained environments. EDHOC is a lightweight authenticated key exchange protocol (LAKE) for constrained environments. Currently, in draft 15, as of this writing, the LAKE working group at the Internet Engineering Task Force leads the development of EDHOC.

### 1.1.2 Existing alternative for EDHOC: TLS 1.3

**A robust, battle-tested protocol.** TLS 1.3 [Res18] is one of the most widely used protocols on the internet. The TLS 1.3 protocol decomposes into two sub-protocols: the *handshake protocol* is an authenticated key exchange, and the *record protocol* protects application data. TLS version 1.3 is a full do-over of previous versions plagued by several vulnerabilities. Similar to EDHOC, TLS 1.3 finds its foundation in the SIGMA protocol. Unlike EDHOC, TLS 1.3 is an older protocol that underwent numerous security analyses,

| | STATIC DH Keys | | Signature Keys | |
|---|---|---|---|---|
| | kid | x5t | kid | x5t |
| Message 1 | 37 | 37 | 37 | 37 |
| Message 2 | 45 | 58 | 102 | 115 |
| Message 3 | 19 | 33 | 77 | 90 |
| Total | 101 | 128 | 216 | 242 |

**Table 1.1:** Sizes of protocol message in EDHOC with static Diffie–Hellman and signature keys. CBOR Web Token credentials use its kid field as a credential identifier, whereas X.509 certificates use the x5t hash header.

both computational and using formal methods [DFGS21, DDGJ22, KMO+14, CHH+17]. Furthermore, the widespread nature of TLS 1.3 consequently gave rise to many optimized implementations, keeping the overhead incurred by TLS.13 on latency to a minimum [LKK21].

**Using TLS 1.3 in constrained environments.** Besides a shared theoretical underpinning (SIGMA), TLS 1.3 also supports most primitives needed for EDHOC, including 1) X25519 for Diffie–Hellman, 2) AES-GCM, AES-CCM, and Chacha20/Poly1305 for authenticated encryption 3) Ed25519 and ECDSA over NIST curves for digital signatures. The three classes of primitives listed before are the most commonly used and are enough to cover all modes of authentication in EDHOC. This begs the question, why not simply use TLS 1.3 in EDHOC?

**TLS 1.3 is not lightweight.** A major drawback of TLS 1.3 in constrained environments is its bandwidth overhead. Mattsson *et al.* [MPV20] compared the bandwidth overhead of EDHOC to that of TLS 1.3 and DTLS.3. Their experiments show that EDHOC with a minimum total bandwidth usage as low as 101 bytes outperforms both competitors (D)TLS1.3. For instance, in comparison to DTLS 1.3, EDHOC messages can use up to six times less bandwidth[3]. Experimental measurements of bandwidth overhead are shown in Table 1.1.

### 1.1.3 Overview of EDHOC's design

**Authentication methods.** As already discussed, EDHOC in SIG-SIG mode is an instantiation of the SIGMA protocol in its "MAc-then-SIGn" incarnation, similarly to the IKEv2 protocol [CH98]. Whereas the "original" SIGMA protocol sends both a signature and a MAC tag, "MAc-then-SIGn" puts the

[3]EDHOC draft 15, section 1.2: https://www.ietf.org/archive/id/draft-ietf-lake-edhoc-15.html#section-1.2.

MAC "under" the signature, thereby saving a few bytes on the wire. Additionally, EDHOC provides an authentication method based on long-term Diffie–Hellman keys known as STAT-STAT. Finally, authentication methods may be mixed, thereby giving the two additional modes STAT-SIG and SIG-STAT.

**Key usages.** The primary use case is establishing a security context for the OSCORE protocol. However, EDHOC can be used for authentication only where no application data needs to be sent. What's more, it provides an *exporter* mechanism to derive further keys if needed in other applications. It also provides a lightweight *key update* mechanism to limit the lifetime of key material without re-running the protocol again.

**Long-term secrets and credentials.** Each protocol participant in EDHOC is associated with a long-term key pair (for Diffie–Hellman or signing) called a *credential*. It is a data structure containing a unique identifier and the corresponding DH share or verification key. In contrast to TLS 1.3, EDHOC assumes that any device typically has local access credentials of other devices. Hence, further bandwidth savings is achieved by sending a short *credential identifier* rather than the entire credential itself, contrary to TLS 1.3, where certificates are always sent by at least one protocol participant.

A close look at the specification for credential identifiers shows that such an identifier *need not be unique* and may reference multiple credentials belonging to potentially different users. Hence, in EDHOC, the receiver of a signature (both initiators and responders) *must* try all possible public keys associated with the received credential identifier before determining the identity of the peer. Therefore, it is reasonable to investigate the implicit assumption that signature schemes must provide, as we did in this thesis project.

## 1.2 Contributions

In this thesis, we performed a cryptographic analysis of the EDHOC protocol in SIG-SIG mode for authentication. Our main contributions are as follows:

- We study the EDHOC protocol in SIG-SIG mode and specify a suitable abstraction of its cryptographic core.
- We design a code-based multi-stage key exchange model [FG14, Gü18] for the EDHOC protocol. Following previous work on TLS 1.3 [DFGS21, DDGJ22], we capture key secrecy, explicit authentication, and forward secrecy.
- Due to non-unique credential identifiers in EDHOC, we extend the MSKE model to capture explicit authentication in a scenario allowing

adversarial registration of long-term secrets following by the work of Boyd *et al.* [BCF$^+$13]. Our conservative approach allows considering additional attack scenarios that are not necessarily a concern in protocols like TLS 1.3.

- We give a game-based security proof for the EDHOC protocol in SIG-SIG mode. Our analysis shows that the EDHOC protocol in SIG-SIG mode is structurally sound and that all protocol components contribute to the protocol's security.

- We describe our interaction with the IETF, including the findings we shared with the Lake working group and several improvements we suggested to the EDHOC protocol. The changes we suggested to draft 14 have an official pull request awaiting to be merged to the main draft [Mat22]. Our suggestions strengthen explicit authentication, require minor changes, and were also independently verified by a team performing a symbolic analysis of EDHOC. We are therefore confident that our suggestion will be soon integrated into the main protocol specification.

## 1.3 Related and Concurrent Work

Krawczyk introduced the SIGMA [Kra03] family of authenticated key exchange protocols. SIGMA is the basis of the IKEv2 protocol [CH98], the TLS 1.3 handshake protocol [Res18] and the EDHOC SIG-SIG protocol [SMP22a]. Numerous works have analyzed the security of SIGMA protocols; in the context of TLS 1.3, Dowling *et al.* [DFGS21] analyzed the TLS 1.3 handshake protocol. Additionally, Davis and Günther [DG21] provided tighter security proofs for both SIGMA and TLS 1.3 handshake. Concurrently, Diemert and Jager [DJ21] also provided a tight security proof for TLS 1.3. Later, Davis *et al.* [DDGJ22] extended the results to the PSK modes, thereby closing the gaps between theoretical analysis and practical deployment of TLS 1.3.

When it comes to EDHOC, Bruni *et al.* [BSJGPS18] performed a formal verification of EDHOC using ProVerif; however, their analysis is only limited to draft 8. Norman *et al.* [NSB21] extended the work of Bruni to cover newly included mixed authentication modes. However, their work does not cover the current draft version. More recently, Cheval *et al.* [CJKK22] used their formal analysis tool chain, SAPIC+, to analyze all four authentication methods in EDHOC. Concurrent to our computational analysis, Cottier and Pointcheval [Cot22] worked on a tight computational analysis for EDHOC STAT-STAT.

## 1.4 Outline

The remaining of this document proceeds as follows. In Chapter 2, we recap some fundamental cryptographic notions and introduce the notation we use in this thesis. In Chapter 3, we describe the EDHOC protocol in SIG-SIG mode. In Chapter 4 we introduce our MSKE model in preparation for the analysis of the EDHOC protocol. Our analysis and proof is given in Chapter 5. Finally, in Chapter 6, we discuss various aspects of the standardization process and their interplay with this project; we conclude by discussing limitations and proposing directions for future work.

Chapter 2

# Preliminaries

In this chapter, we introduce the notation we will use in the rest of the thesis; we also summarize the syntax and assumptions of the cryptographic primitives used in EDHOC. We refer to [BS] for a detailed treatment of the cryptographic primitives.

## 2.1 Notation

**Pseudocode.** We use the notation $x \leftarrow a$ to indicate that the value $a$ is assigned to the variable $x$. $x \xleftarrow{\$} \mathcal{X}$ denotes sampling $x$ at random from a domain $\mathcal{X}$; Unless otherwise specified, the sampling is assumed to be efficient and follows a uniform distribution on $\mathcal{X}$.

**Sequences, tuples, and arrays.** We use the notation $(v_1, v_2, \ldots, v_n)$ to denote a sequence of $n$ elements each from an arbitrary domain; we use the terms "sequence", "tuple", and "arrays" interchangeably. For a domain $\mathcal{X}$, $x^n$ refers to a sequence of $n$ elements from $\mathcal{X}$; given a sequence $x$, $x_i$ refers to the $i$'th element of the sequence. Most cryptographic primitives, such as hash functions, operate on bit strings, whereas the EDHOC specification operates on sequences of objects encoded with the Concise Binary Object Representation (CBOR [BH20]). CBOR *unambiguously* encode objects and sequences into bit strings; hence for each function called on a tuple, we implicitly assume that internally, the function first uses an encoding function $\xi()$ to map sequences or single CBOR objects to bit strings[1]. For a given sequence $x^n$, $|x^n|$ denotes the length of the bit string encoding of $x^n$, i.e., $|x^n|$ is the length of $\xi(x^n)$. $x \| y$ denotes the concatenation of the bitstrings $x$ and $y$.

[1]In the remaining of this text, we do not explicitly show this encoding for conciseness.

**Groups and Diffie–Hellman.** Let $(\mathbb{G}, +)$ be a group, we consider cyclic groups of prime order $q$, i.e., $\mathbb{G} = \langle G \rangle$ is the set $\{x * G : x \in \mathbb{Z}_q\}$ generated by $G$. Throughout this document, we use the *additive* notation to reflect that EDHOC only utilizes a (sub)group of an elliptic curve over a finite field. For $x \in \mathbb{Z}_q$ and $Y \in \mathbb{G}$, $\mathsf{DH}(x, Y)$ denotes the Diffie–Hellman function that computes the shared secret $S = x * Y$.

**Games, adversaries, and advantages.** We carry out the analysis of EDHOC in the code-based game-playing framework for provable security [BR06]. The security of a cryptographic scheme or protocol $\Pi$ is captured via a game $\mathbf{G}(\Pi)$ played by an adversary (or distinguisher) $\mathcal{A}$ that interacts with the game through several named oracles. We will sometimes refer to the collection of game oracles as *challenger*. In particular, each game provides two oracles $\mathcal{O}_{Initialize}$ and $\mathcal{O}_{Finalize}$; a winning condition is defined and the oracle $\mathcal{O}_{Finalize}$ outputs 1 if the condition is satisfied and 0 otherwise. We restrict our adversaries to the class $\mathcal{A}_{t,q}$ of efficient adversaries that run in time at most $t$ and make at most $q$ queries to its oracles. We do not give fully concrete security bounds; hence, we will not explicitly state any other of the adversary's resources. For any adversary $\mathcal{A} \in \mathcal{A}_{t,q}$, the advantage of $\mathcal{A}$ in winning the game $\mathbf{G}(\Pi)$, denoted by $\mathsf{Adv}^{\mathbf{G}}_{\mathcal{A}}(\Pi)$, captures the performance of $\mathcal{A}$ which is the probability that $\mathcal{O}_{Finalize}$ outputs 1 i.e.:

$$\mathsf{Adv}^{\mathbf{G}}_{\mathcal{A}}(\Pi) = \Pr[\mathbf{G}(\Pi) \to 1].$$

When the context is unambiguous, we simply write $\mathbf{G}$ and $\mathsf{Adv}^{\mathbf{G}}_{\mathcal{A}}$ instead of $\mathbf{G}(\Pi)$ and $\mathsf{Adv}^{\mathbf{G}}_{\mathcal{A}}(\Pi)$ respectively. We similarly omit the input to game oracles. For short games, we inline the winning condition in the advantage statement and do not explicitly show the game.

## 2.2 Cryptographic Primitives

The main cryptographic primitives in EDHOC are a cryptographic hash function, a pseudo-random function, a message authentication code, a digital signature scheme, a key derivation function, and an authenticated encryption scheme.

### 2.2.1 Hash function

A hash function deterministically computes a short fingerprint for inputs of arbitrary length. In the context of EDHOC, the hash function's primary usage is to hash the communication transcript, which is used to assert the authenticity of the key exchange and the peer. We will concern ourselves with cryptographic hash functions only and require that they are collision-resistant.

**The game $\mathbf{G}^{PRF}(F)$**

```
$\mathcal{O}_{Initialize}()$
    1 : $b \xleftarrow{\$} \{0,1\}$
    2 : $f \xleftarrow{\$} Funcs[\mathcal{X},\mathcal{Y}]$
    3 : $k \xleftarrow{\$} \mathcal{K}$

$\mathcal{O}_{prf}(x)$
    1 : $y_0 = F(k,x)$
    2 : $y_1 = f(x)$
    3 : **return** $y_b$

$\mathcal{O}_{Finalize}(b')$
    1 : **return** $b' = b$
```

**Figure 2.1:** The PRF security game for $F$

**Definition 2.1** *(Collision-resistant hash function) A hash function $H : \{0,1\}^* \to \{0,1\}^\lambda, \lambda \in \mathbb{N}$ is a deterministic mapping $H(m) = h_m \in \{0,1\}^\lambda, \forall m \in \{0,1\}^*$. The relevant security notion is collision resistance (*CR*), namely the (in)feasibility of producing values $m \neq m'$ such that $H(m) = H(m')$ for a given adversary $\mathcal{A}$. More precisely, collision resistance is captured by a game $\mathbf{G}^{\mathsf{CR}}_{\mathcal{A}}(H)$. The advantage of $\mathcal{A}$ is defined as:*

$$\mathsf{Adv}^{\mathsf{CR}}_{\mathcal{A}}(H) = \Pr\Big[(m, m') \xleftarrow{\$} \mathcal{A} : H(m) = H(m') \wedge m \neq m'\Big].$$

**Hash functions in EDHOC.** Each ciphersuite specifies a hash function to be used. We generically refer to the hash function as H. In each ciphersuite, H is instantiated with either of SHA2, Shake128 or Shake256. Shake128 and Shake256 are sponge-based extendable output functions (XOF [KCP16]).

### 2.2.2 Pseudo-Random Functions

A common task in EDHOC is to generate multiple random keys from an assumed (pseudo)random key. This is achieved using a pseudo-random function.

**Definition 2.2** *(Pseudo-Random Function) A pseudo-random function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is a deterministic function that takes as input a key $k$ and a value $x$ and outputs a value $y$. Intuitively, $F$ is a PRF if for a randomly chosen key $k$, it is computationally indistinguishable from a function $f : \mathcal{X} \to \mathcal{Y}$ chosen uniformly at random from $Funcs[\mathcal{X},\mathcal{Y}]$, the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$.*

Throughout the document, we unashamedly use the term “PRF” to refer to a function defined in this section and the associated security notion that we formalize next.

**Security of PRFs.** Let $\mathcal{A}$ be any efficient distinguisher for $F$ in the sense of the game $\mathbf{G}^{\mathsf{PRF}}$ (Figure 2.1). We define $\mathcal{A}$'s advantage against the PRF security of $F$ as follows:

$$\mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{A}}(F) = \Pr\Big[\mathbf{G}^{\mathsf{PRF}}(F) \to 1\Big].$$

**EXPAND as a PRF in EDHOC.** The Expand module in EDHOC defined as $\mathsf{Expand}(k, (label, context, len), len)$ (Section 2.2.3) is a *variable output length* PRF that outputs bit strings of length *len* when queried on the key $k$ and input *label*.

### 2.2.3 Key Derivation Functions

To derive the final key and other keys used during the key exchange, EDHOC uses a key derivation function. The key derivation in EDHOC closely follows the design of HKDF [Kra10], it is realized via two modules. The first $\mathsf{Extract}(s, \mathsf{ikm})$ that extracts a pseudo-random key from $\mathsf{ikm}$. The second module is the function $\mathsf{Expand}(\mathsf{prk}, info, len)$ that generates from the pseudo-random key $\mathsf{prk}$ another length-*len* pseudo-random key.

#### The extract module

The output of the anonymous Diffie–Hellman key exchange protocol is a group element, and its bitstring representation is usually not a uniform random variable. Hence, one needs a so-called *extractor* to extract a uniform random key. We note that Krawczyk described assumptions required for the function HKDF.Extract in [Kra10]. In our analysis of EDHOC, we will rely on the PRF-ODH assumption described in Section 2.2.7.

**Extract in EDHOC.** Whenever the hash function is SHA2, $\mathsf{Extract}(s, \mathsf{ikm})$ is instanciated with HKDF.Extract , i.e., $\mathsf{Extract}(s, \mathsf{ikm}) = \mathsf{HMAC}(s, \mathsf{ikm})$. If the hash algorithm is either Shake128 or Shake256, then $\mathsf{Extract}(s, \mathsf{ikm}) = \mathsf{KMAC}(s, \mathsf{ikm}, len, \texttt{""})$. Here, $s$ is used as the key for KMAC [KCP16].

#### The Expand module

Once a pseudo-random key prk is obtained from the extractor, one wishes to use it to generate other keys. The $\mathsf{Expand}(\mathsf{prk}, (label, context, len), len)$ module is used for this purpose. We assume that Expand is a PRF as defined in Section 2.2.2.

**Expand in EDHOC.** $\mathsf{Expand}(\mathsf{prk}, info, len) = \mathsf{HKDF.Expand}(\mathsf{prk}, info, len)$ (an iterated application of HMAC [BCK96]) whenever the hash function is SHA2. If the hash function is any of the sponge-based XOFs, $\mathsf{Expand}(\mathsf{prk}, info, len) = \mathsf{KMAC}(\mathsf{prk}, info, len, \texttt{""})$.

### 2.2.4 Message Authentication Codes

Asserting the authenticity of a received message given a shared secret key can be accomplished using a message authentication code (MAC).

```
Game $\mathsf{G}^{\text{EUF-CMA}}(\mathfrak{M})$

$\mathcal{O}_{Initialize}()$
    1 : $k \leftarrow \mathfrak{M}.\mathsf{KGen}$
    2 : $\mathcal{M} \leftarrow \emptyset$

$\mathcal{O}_{mac}(m)$
    1 : $\tau \xleftarrow{\$} \mathfrak{M}.\mathsf{Tag}(k, m)$
    2 : $\mathcal{M} \leftarrow \mathcal{M} \cup \{m\}$
    3 : return $\tau$

$\mathcal{O}_{Finalize}(m, \tau)$
    1 : return $\begin{pmatrix} \mathfrak{M}.\mathsf{Vf}(k, m, \tau) = 1 \\ \wedge\, m \notin \mathcal{M} \end{pmatrix}$
```

**Figure 2.2:** The MAC security game

**Definition 2.3** *A message authentication code (MAC) scheme $\mathfrak{M}$ is a triple of efficiently computable algorithms* $(\mathsf{KGen}, \mathsf{Tag}, \mathsf{Vf})$ *where:*

- $\mathsf{KGen}$ *is a probabilistic algorithm that generates a (uniform) random MAC key $k \in \mathcal{K}$.*
- $\mathsf{Tag} : \mathcal{K} \times \mathcal{M} \to \mathcal{T}$ *is a (possibly probabilistic) algorithm that on input a MAC key $k$ and a message $m$ computes a tag $\tau \xleftarrow{\$} \mathsf{Tag}(k, m)$.*
- $\mathsf{Vf} : \mathcal{K} \times \mathcal{M} \times \mathcal{T} \to \{0, 1\}$ *is a deterministic algorithm that takes as input a key $k$, a message $m$, a tag $\tau$ and outputs a bit $b \leftarrow \mathsf{Vf}(k, m, \tau), b \in \{0, 1\}$. The output is* 1 *when the tag is valid and* 0 *otherwise.*

**Correctness.** We demand that $\forall k, m \in \mathcal{K} \times \mathcal{M} : \Pr[\mathsf{Vf}(k, m, \mathsf{Tag}(k, m))] = 1$.

**Security of MAC schemes.** The standard security notion for MAC scheme is *Existential Unforgeability under Chosen-Message Attack* (EUF-CMA); this notion states that it is infeasible for an attacker $\mathcal{A}$ to produce a tag $\tau$ on a *new* message $m$ that was not mac'ed before, even given access to an oracle $\mathcal{O}_{mac}$ that computes and returns tags $\tau_i$ on adversarially chosen messages $m_i$ (See Figure 2.2).

### PRFs as MAC scheme

It is well-known that any pseudo-random function (Section 2.2.2) with a "large" output range is a secure deterministic MAC scheme. In EDHOC, MAC tags are computed exclusively with the Expand function, which is assumed to be a PRF. Although the MAC functionality is syntactically used in EDHOC for authentication, the MAC tag is placed "under" the signature (not sent) and is only implicitly verified. Therefore, the MAC in EDHOC is mainly used as a PRF.

## 2.2.5 Digital Signatures

A digital signature scheme allows a message sender (and only them) to produce publicly verifiable proof that the message is authentic. In EDHOC,

**Games $\mathsf{G}^{\text{EUF-CMA}}, \mathsf{G}^{\text{SUF-CMA}}, \mathsf{G}^{\text{S-UEO}}$**

```
$\mathcal{O}_{Initialize}()$
  1 : $(sk, pk) \leftarrow \mathfrak{S}.\mathsf{KGen}$
  2 : $\mathcal{M} \leftarrow \emptyset$
  3 : return $pk$

$\mathcal{O}_{Finalize}((m, \sigma))$
  1 : return $\begin{pmatrix} \mathfrak{S}.\mathsf{Vf}(pk, m, \sigma) = 1 \\ \wedge\, m \notin \mathcal{M} \end{pmatrix}$

$\mathcal{O}_{Sign}(m)$
  1 : $\mathcal{M} \leftarrow \mathcal{M} \cup \{m\}$
  2 : return $\mathfrak{S}.\mathsf{Sign}(sk, m)$
```

```
$\mathcal{O}_{Initialize}(m)$
  1 : $(sk, pk) \leftarrow \mathfrak{S}.\mathsf{KGen}$
  2 : $\mathcal{M} \leftarrow \emptyset$
  3 : return $pk$

$\mathcal{O}_{Finalize}(m, \sigma)$
  1 : return $\begin{pmatrix} \mathfrak{S}.\mathsf{Vf}(pk, m, \sigma) = 1 \\ \wedge\, (m, \sigma) \notin \mathcal{M} \end{pmatrix}$

$\mathcal{O}_{Sign}(m)$
  1 : $\sigma \xleftarrow{\$} \mathfrak{S}.\mathsf{Sign}(sk, m)$
  2 : $\mathcal{M} \leftarrow \mathcal{M} \cup \{(m, \sigma)\}$
  3 : return $\sigma$
```

```
$\mathcal{O}_{Initialize}()$
  1 : $(sk, pk) \leftarrow \mathfrak{S}.\mathsf{KGen}$
  2 : $\mathcal{M} \leftarrow \emptyset$
  3 : return $pk$

$\mathcal{O}_{Finalize}(m, m', \sigma, pk')$
  1 : return $\begin{pmatrix} (m, \sigma) \in \mathcal{M} \\ \wedge\, pk \neq pk' \\ \wedge\, \mathfrak{S}.\mathsf{Vf}(pk', m', \sigma) = 1 \end{pmatrix}$

$\mathcal{O}^{*}_{Sign}(m)$
  1 : $\sigma \xleftarrow{\$} \mathfrak{S}.\mathsf{Sign}(sk, m)$
  2 : $\mathcal{M} \leftarrow \mathcal{M} \cup \{(m, \sigma)\}$
  3 : return $\sigma$
```

**Figure 2.3:** The EUF-CMA game (top), the SUF-CMA game (middle) and the S-UEO game (bottom).

signatures are used to authenticate the peers.

**Definition 2.4** *A digital signature scheme $\mathfrak{S}$ is a triple of efficiently computable algorithms* $(\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ *where:*

- $\mathsf{KGen}$ *is a probabilistic algorithm that generates a signature key pair $(sk, pk) \in \mathcal{K}_{sk} \times \mathcal{K}_{pk}$.*
- $\mathsf{Sign} : \mathcal{K}_{sk} \times \mathcal{M} \to \Sigma$ *is a (possibly probabilistic) algorithm that on input a signature key $sk$ and a message $m$ computes a signature $\sigma \xleftarrow{\$} \mathsf{Sign}(sk, m)$.*
- $\mathsf{Vf} : \mathcal{K}_{pk} \times \mathcal{M} \times \Sigma \to \{0, 1\}$ *is a deterministic algorithm that takes as input a public key $pk$, and a message $m$, and a signature $\sigma$ and outputs a bit $b = \mathsf{Vf}(pk, m, \sigma)$. The output is $1$ when the signature is valid and $0$ otherwise.*

**Correctness.** $\forall (sk, pk) \xleftarrow{\$} \mathsf{KGen}, m \in \mathcal{M} : \Pr[\mathsf{Vf}(pk, m, \mathsf{Sign}(sk, m))] = 1.$

### Security of Digital Signatures Schemes

**Definition 2.5 (Existential Unforgeability under Chosen-Message Attack)** *For a signature scheme $\mathfrak{S}$ and an efficient adversary $\mathcal{A}$,* Existential Unforgeability under Chosen Message Attack *(EUF-CMA) is a security notion capturing $\mathcal{A}$'s* *success in forging signatures for* new *messages given access to a signing oracle (See Figure 2.3). The advantage of $\mathcal{A}$ is defined by:*

$$\mathsf{Adv}_{\mathcal{A}}^{\text{EUF-CMA}}(\mathfrak{S}) = \Pr\left[\mathbf{G}^{\text{EUF-CMA}}(\mathfrak{S}) \to 1\right].$$

**Definition 2.6 (Strong Unforgeability under Chosen Message Attack)** *For a signature scheme $\mathfrak{S}$ and an efficient adversary $\mathcal{A}$,* Strong Unforgeability under Chosen Message Attack *(*SUF-CMA*) is a security notion that captures $\mathcal{A}'$ success in forging a* new *message-signature pair given access to a signing oracle (see Figure 2.3). We define $\mathcal{A}$'s advantage as follows:*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{SUF\text{-}CMA}}(\mathfrak{S}) = \Pr\left[\mathbf{G}^{\mathsf{SUF\text{-}CMA}}(\mathfrak{S}) \to 1\right].$$

### Exclusive Ownership of Signatures

To analyze EDHOC, we additionally employ the *Strong Universal Exclusive Ownership* security notion put forward by [BCJZ20].

**Definition 2.7 (Strong Universal Exclusive Ownership)** *A signature scheme $\mathfrak{S}$ is said to provide* Strong Universal Exclusive Ownership *(*S-UEO*) against an adversary $\mathcal{A}$ if: given a public key pk and access to the signing oracle with the corresponding signing key sk, it is infeasible for an adversary to create a different public key $pk' \neq pk$ and a message $m'$ such that $\mathsf{Vf}(pk', m', \sigma) = 1$ for $\sigma$ that was previously obtained from the signing oracle (see Figure 2.3). $\mathcal{A}$'s advantage is defined as follows:*

$$\mathsf{Adv}_{\mathcal{A}}^{\text{S-UEO}}(\mathfrak{S}) = \Pr\left[\mathbf{G}^{\text{S-UEO}}(\mathfrak{S}) \to 1\right].$$

The S-UEO notion implies two related and weaker notions; namely, *Strong Conservative Exclusive Ownership* (S-CEO) and *Strong Destructive Exclusive Ownership* (S-DEO). The former corresponds to the case where the adversary must have queried the signing oracle to obtain a signature for $m'$. This scenario captures, for instance, duplicate signature key selection attacks (DSKS) [MS04]. The latter encodes that $m'$ must *not* have been queried to the signing oracle. Conversely, S-CEO and S-DEO jointly imply S-UEO. We refer to [CDF+21] for further details.

**Digital signatures in EDHOC.** The signature scheme used in EDHOC, hereafter referred to as Sig, is instantiated with either Ed25519 [BDL+12] or ECDSA [JMV01].

We note here that Ed25519 was studied by Brendel *et al.* [BCJZ20] and was shown to be SUF-CMA and S-UEO secure. In contrast, ECDSA is only

EUF-CMA secure and fails to meet either SUF-CMA[2] nor S-DEO[3] (hence is not S-UEO).

### 2.2.6 Authenticated Encryption

An authenticated encryption scheme with associated data(AEAD) is an encryption scheme in which, given a message $m$ and additional data $ad$, the scheme ensures integrity for both $m$ and $ad$. Still, secrecy is guaranteed for $m$ only. In EDHOC, AEAD is used to encrypt part of the handshake, thereby providing identity protection.

Throughout this document, we use the nonce-based syntax for AEAD[Rog02].

**Definition 2.8 (Nonce-based Authenticated Encryption with Associated Data)** *A nonce-based authenticated encryption scheme with additional data $\mathfrak{E}$ is a triple of efficiently computable algorithms $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ where:*

- $\mathsf{KGen}$ *is a probabilistic algorithm that generates a random key $k$.*
- $\mathsf{Enc} : \mathcal{K} \times \mathcal{M} \times \mathcal{A}_d \times \mathcal{N} \to \mathcal{C} = \{0,1\}^*$ *is a deterministic function that takes a key $k \in \mathcal{K}$, a message $m \in \mathcal{M}$, an additional data $ad \in \mathcal{A}_d$, a nonce $n \in \mathcal{N}$ and returns a ciphertext $c = \mathfrak{E}.\mathsf{Enc}(k, m, ad, n) \in \mathcal{C}$.*
- $\mathsf{Dec} : \mathcal{K} \times \mathcal{C} \times \mathcal{A}_d \times \mathcal{N} \to \mathcal{M} \cup \{\perp\}$ *is a deterministic function that takes a key $k \in \mathcal{K}$, a ciphertext $c \in \mathcal{C}$, an additional data $ad \in \mathcal{A}_d$, a nonce $n \in \mathcal{N}$ and returns a message $m \in \mathcal{M}$ or a distinguished error symbol $\perp$.*

**Correctness** We demand that

$$\forall k, m, ad, n : \mathsf{Dec}(k, \mathsf{Enc}(k, m, ad, n), ad, n) = m.$$

**Definition 2.9 (Nonce-based AE security (nAE))** *Let $\mathfrak{E}$ be an AEAD scheme and let $\mathcal{A}$ be an efficient adversary, the security of $\mathfrak{E}$ against $\mathcal{A}$ is captured by the game $\mathbf{G}^{\mathrm{nAE}}$ (Figure 2.4), which is is the* all-in-one *game put forward in [Shr04]. The performance of $\mathcal{A}$ in the game is characterized by $\mathcal{A}$'s advantage, which we defined as follows :*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}^{\mathrm{nAE}}}(\mathfrak{E}) = \Pr\left[\mathbf{G}^{\mathrm{nAE}}(\mathfrak{E}) \to 1\right].$$

**AEADs in EDHOC.** We refer to the scheme used in EDHOC by AEAD. Depending on the ciphersuite it is instantiated by ChaCha20/Poly1305, or AES-GCM or AES-CCM.

[2]For a ECDSA signature $\sigma = (r, s) \in \mathbb{F}_q^2$, on $m$, $(r, -s)$ is also a valid signature on $m$.
[3]See https://cronokirby.com/posts/2022/02/on-the-malleability-of-ecdsa/.

**Game $\mathsf{G}^{\mathrm{nAE}}(\mathfrak{E})$**

```
$\mathcal{O}_{Initialize}()$
  1 : $k \xleftarrow{\$}$ KGen
  2 : $b \xleftarrow{\$} \{0,1\}$
  3 : $\mathcal{N} \leftarrow \emptyset$

$\mathcal{O}_{Finalize}(b')$
  1 : return $b = b'$

$\mathcal{O}_{Enc}(m, ad, n)$
  1 : $c_0 \leftarrow \mathfrak{E}.\mathsf{Enc}(k, m, ad, n)$
  2 : $\mathcal{N} \leftarrow \mathcal{N} \cup \{n\}$
  3 : $c_1 \xleftarrow{\$} \mathcal{C}^{|c_0|}(|c_1| = |c_0|)$
  4 : if $n \in \mathcal{N}$ : return $\perp$
  5 : return $c_b$

$\mathcal{O}_{Dec}(c, ad, n)$
  1 : $m_0 \leftarrow \mathfrak{E}.\mathsf{Dec}(k, c, ad, n)$
  2 : $m_1 \leftarrow \perp$
  3 : return $m_b$
```

**Figure 2.4:** The *all-in-one* nAE security game for nonce-based AEAD.

**Game $\mathsf{G}^{\mathrm{snPRF\text{-}ODH}}(F)$**

```
$\mathcal{O}_{Initialize}()$
  1 : $b \xleftarrow{\$} \{0,1\}$
  2 : $(u, v) \xleftarrow{\$} \mathbb{Z}_q^2$
  3 : blocked $\leftarrow$ false

$\mathcal{O}_{Chall}(x^*)$
  1 : $y_0^* \leftarrow F(uv * G, x^*)$
  2 : $y_1^* \xleftarrow{\$} \mathcal{Y}$
  3 : return $y_b^*$

$\mathcal{O}_u(T, x)$
  1 : if blocked : return $\perp$
  2 : if $T \notin \mathbb{G}$ : return $\perp$
  3 : if $(T, x) = (v * G, x^*)$ :
  4 :     return $\perp$
  5 : $y \leftarrow F(u * T, x)$
  6 : blocked $\leftarrow$ true
  7 : return $y$

$\mathcal{O}_v(T, x)$
  1 : return $\perp$

$\mathcal{O}_{Finalize}(b')$
  1 : return $b = b'$
```

**Figure 2.5:** The snPRF-ODH game. The $\mathcal{O}_{Chall}$ returns either output of $F$ or a random value. The oracle $\mathcal{O}_v$ is shown to return $\perp$ because no queries are allowed.

### 2.2.7 Pseudo-Random Function Diffie–Hellman Oracle Assumption

The PRF-ODH [BFGJ17] assumption has been introduced and used to analyze DH-based key exchange protocols. In DH-based protocols, participants exchange DH shares $x * G$, $y * G$ and compute the shared secret $ss = \mathsf{DH}(x, y * G)$, which is further processed into a session key $k$ with a key derivation function and other auxiliary inputs. The assumption arises naturally in such protocols in the presence of an active adversary who may, for instance, obtain one or more values $ss' = \mathsf{DH}(v, x * G)$ for an adversarially chosen $v$. Therefore, by the PRF-ODH assumption, we can consider the final session key $k$ to be an independent pseudo-random value even though $ss$ and $ss'$ are related in a nontrivial manner. In EDHOC, we will rely on the snPRF-ODH security of the Extract function.

**Definition 2.10 (The snPRF-ODH assumption)** *Let $\mathbb{G} = \langle G \rangle$ be a cyclic group* *of order $q$, let $F : \mathbb{G} \times \mathcal{X} \to \mathcal{Y}$ be a PRF (see Section 2.2.2) that takes a key $k \in \mathbb{G}$, an input $x \in \mathcal{X}$ and outputs a value $y = F(k, x) \in \mathcal{Y}$. The* snPRF-ODH *assumption essentially states that $F(k, \cdot)$ is a* PRF *keyed with $k = uv * G$ for $(u, v) \xleftarrow{\$} \mathbb{Z}_q^2$. Similiarly to the usual* PRF *security notion, an adversary is given access to an oracle $\mathcal{O}_{Chall}$ that returns either the output of $F$ or a uniform random value. However, for* PRF-ODH*, the adversary is additionally given $u * G$, $v * G$ and access (with restriction) to oracles $\mathcal{O}_v$ and $\mathcal{O}_u$ defined by $\mathcal{O}_{u,v}(T, x) = F(T^{u,v}, x)$. The former allows a **s**ingle query whereas $\mathcal{O}_v$ allows **n**o query. The security notion is made more formal in the game $\mathbf{G}^{\text{snPRF-ODH}}$ (Figure 2.5). The advantage of an adversary $\mathcal{A}$ is defined as:*

$$\mathsf{Adv}_{\mathcal{A}}^{\text{snPRF-ODH}}(F) = \Pr\Big[\mathbf{G}^{\text{snPRF-ODH}}(F) \to 1\Big].$$

The PRF-ODH assumption was studied in [BFGJ17], and the authors showed that in the random oracle model, the strongest variant of the PRF-ODH assumption under the strong Diffie–Hellman assumption.

**PRF-ODH security of Extract.** Brendel *et al.* [BFGJ17] showed that HMAC is snPRF-ODH-secure, i.e., it is a PRF $F(k, x) = \mathsf{HMAC}(x, k)$. The authors remark that the results will likely apply if a sponge-based construction replaces the underlying hash function. However, in EDHOC, sponge-based hashes are not used within the HMAC construction. Instead, EDHOC uses KMAC for MACing and Shake128 or Shake256 for hashing and as a XOFs. Therefore, it seems to be an open question whether we can also assume the use of KMAC in Extract snPRF-ODH-secure. In our analysis, we assume that this is the case.

Chapter 3

# The EDHOC SIG-SIG Protocol

In this chapter, we describe the EDHOC protocol in SIG-SIG mode as described in draft version 14 by the IETF LAKE working group [SMP22b]. The scope of this thesis is limited to the SIG-SIG protocol. The protocol we describe here is slightly different from the one described in draft 14. In our description, we include a few modifications we suggested to the working group regarding the computation of transcript hashes [Mat22]. As we discussed before, the changes are expected to be included in the upcoming drafts. We highlight said modifications at the relevant places in this chapter. The current draft 15 introduces minor changes, including optional message padding, the possibility for longer messages, and changes to the key schedule[1]. The analysis presented in the following chapter should carry over to draft 15 with minor changes.

## 3.1 Overview

**Overview.** EDHOC protocol is a *lightweight authenticated key exchange* (LAKE). It is designed to be a secure and efficient authenticated key exchange protocol for constrained environments. Namely, the envisioned beneficiaries of the protocol are low-end devices such as Internet of Things (IoT). Additionally, devices operating in low-bandwidth environments, such as LoRaWAN[2], may take advantage of small-sized protocol messages needed to execute the protocol.

**Use cases.** The primary use case of the EDHOC protocol is to establish a security context for the Object Security for Constrained RESTful Environ-

[1]Overview of changes in draft 15: https://datatracker.ietf.org/meeting/114/materials/slides-114-lake-edhoc-traces-01.

[2]https://lora-alliance.org/about-lorawan/.

ments (OSCORE) protocol [SMPS19]. The EDHOC protocol is also used to exchange keys for other purposes.

**Design rationale.** The design of the EDHOC protocol in SIG-SIG mode draws from the design SIGMA [Kra03]. The "SIGn-and-MAc" family of authenticated key exchange protocols has been used as the basis of the widely deployed TLS 1.3 [Res18] and the Internet Key Exchange protocol[CH98].

## 3.2 Identities and Long-Term Keys

**Credentials.** In EDHOC, long-term verification keys are stored in so-called credentials. It is assumed that in a typical deployment, credentials will be locally accessible and don't need to be transmitted. Furthermore, one or multiple certification authorities attest of the binding between a credential and a long-term verification key.

**Credential identifiers.** Given that the credentials are not transmitted, EDHOC uses credential identifiers instead to allow retrieval of the credentials. These identifiers are typically short-sized and hence beneficial to keep overhead usage low. Credential identifiers need not be unique; they can typically refer to a set of credentials. In scenarios where a credential is not locally available, credentials may also be embedded in the credential identifier.

## 3.3 Details of the SIG-SIG mode

The EDHOC protocol consists of three mandatory messages exchanged between the initiator $I$ and the responder $R$. We hereafter refer to these messages as $\mathsf{msg}_1$, $\mathsf{msg}_2$ and $\mathsf{msg}_3$. Additionally, the responder may send a fourth message to the initiator if EDHOC is used for authentication only and no application data is exchanged. The fourth message is optional, and we omit it from our analysis of the EDHOC protocol. Protocol participants additionally agree on the authentication method and ciphersuite to be used by exchanging protocol messages, aborting the protocol run if agreement cannot be reached on the chosen authentication method or ciphersuite. We illustrate the protocol details in Figure 3.1.

The scope of our analysis is constrained to the SIG-SIG mode, and we do not model negotiation of ciphersuites. Therefore, in the following description of the protocol messages, we consider the authentication method to be a fixed value $M$. Similarly, the ciphersuite is the fixed value S.

### EDHOC Message 1

The initiator chooses the authentication method $M$, the ciphersuite $\mathsf{S}$, a connection identifier $\mathsf{C}_I$ and an optional external authorization data $ead_1$. Additionally, the initiator selects an ephemeral Diffie-Hellman secret $x$ and computes the DH share $G_x = x * G$. The initiator then composes $\mathsf{msg}_1$ as the CBOR sequence $(M, \mathsf{S}, G_x, \mathsf{C}_I, ead_1)$ and sends $\mathsf{msg}_1$ to the responder.

**Processing of message 1 by the responder.** Upon receiving $\mathsf{msg}_1$ from the initiator and parsing it, the responder prepares the next protocol message. In the first phase, the responder completes an unauthenticated[3] Diffie-Hellman key exchange and then computes elements need to authenticate themselves to the initiator. For the unauthenticated Diffie-Hellman exchange, the responder generates an ephemeral Diffie-Hellman secret $y$ and computes the Diffie-Hellman share $G_y = y * G$. The responder computes the shared secret $G_{xy} = y * G_x$ and the shared pseudo-random key $\mathsf{PRK}_{2e} = \mathsf{Extract}(\texttt{""}, G_{xy})$.

### EDHOC Message 2

The responder proceeds to select a connection identifier $\mathsf{C}_R$, an optional external authorization data $ead_2$. The responder computes the partial transcript hash $\mathsf{th}_2 = \mathsf{H}(G_y, \mathsf{C}_R, \mathsf{H}((M, \mathsf{S}, G_x, \mathsf{C}_I, ead_1)))$. The responder computes a MAC tag $\tau_2 = \mathsf{Expand}(\mathsf{PRK}_{2e}, (2, \mathsf{context}_2, \mathsf{taglen}_2), \mathsf{taglen}_2)$. Where $\mathsf{context}_2$ is the string $\mathsf{kid}_R \| \mathsf{th}_2 \| \mathsf{cred}_R \| ead_2$. In other words, the responder computes a MAC tag with shared secret $\mathsf{PRK}_{2e}$ over the responder's credential identifier $\mathsf{kid}_R$, the transcript hash $\mathsf{th}_2$ and the responder's credential $\mathsf{cred}_R$. Note that the $\mathsf{info}$ parameter used in the $\mathsf{Expand}$ to compute $\tau_2$ also includes a unique label 2 and the tag length $\mathsf{taglen}_2$. Now the responder computes a signature $\sigma_2$ over the sequence $s = (\mathsf{l}_{\mathsf{sig}}, \mathsf{kid}_R, \mathsf{th}_2 \| \mathsf{cred}_R \| ead_2, \tau_2)$ i.e. $\sigma_2 = \mathsf{Sig.Sign}(sk_R, s)$. In the preceding, $\mathsf{l}_{\mathsf{sig}}$ is a unique label used when signing messages.

Finally, the responder composes $\mathsf{ptxt}_2 = (\mathsf{kid}_R, \sigma_2, ead_2)$. It derives a key stream $\mathsf{K}_2$ to encrypt $\mathsf{ptxt}_2$ by xoring $\mathsf{ptxt}_2$ with $\mathsf{K}_2$, creating the ciphertext $\mathsf{ctxt}_2 = \mathsf{ptxt}_2 \oplus \mathsf{K}_2$. The key stream is computed as $\mathsf{K}_2 = \mathsf{Expand}(\mathsf{PRK}_{2e}, (0, \mathsf{th}_2, |\mathsf{ptxt}_2|), |\mathsf{ptxt}_2|)$. At last, the responder sends $\mathsf{msg}_2$ which is the CBOR sequence $(G_y, \mathsf{ctxt}_2, \mathsf{C}_R)$.

**Processing of message two by the initiator.** Upon receiving and parsing $\mathsf{msg}_2$, the initiator decrypts $\mathsf{ctxt}_2$, obtaining $\mathsf{ptxt}_2$, and verifies the signature $\sigma_2$. We note here that the $\mathsf{kid}_R$ contained in $\mathsf{ptxt}_2$ may reference multiple credentials. Therefore, the initiator may have to verify the signature against

[3]At this point, neither the initiator nor the responder has authenticated themselves to the other party yet. Hence, the derived DH share is unauthenticated.

multiple public keys until at least one verifies. The draft 14 does not currently define a tie breaking mechanism in case multiple credentials would be valid during the verification process. We assume here that a protocol participant will always choose the first public key that verifies the signature. A potential consequence is that the initiator may accept with a wrong or malicious peer. We analyze the possibility of ambiguity on the responder identity in Chapter 5.

**EDHOC Message 3** If all the required verification steps of $\mathsf{msg}_2$ are successful, the initiator composes $\mathsf{msg}_3$. To do so, the initiator first computes the partial transcript hash[4] $\mathsf{th}_3 = \mathsf{H}(\mathsf{th}_2, \mathsf{ptxt}_2, \mathsf{cred}_R)$. Analogously to how the responder processes $\mathsf{msg}_2$, the initiator then computes the MAC tag $\tau_3 = \mathsf{Expand}(\mathsf{PRK}_{2e}, (6, \mathsf{context}_3, \mathsf{taglen}_3), \mathsf{taglen}_3)$. Where $\mathsf{context}_3$ is the string with value $\mathsf{kid}_I \| \mathsf{th}_3 \| \mathsf{cred}_I \| ead_3$. Furthermore, the initiator computes the signature $\sigma_3$ over the sequence $s = (\mathsf{l}_{\mathsf{sig}}, \mathsf{kid}_I, \mathsf{th}_3 \| \mathsf{cred}_I \| ead_3, \tau_3)$. More precisely, $\sigma_3 = \mathsf{Sig.Sign}(pk_I, s)$.

Next, the initiator computes $\mathsf{msg}_3 = (\mathsf{ctxt}_3)$ which is the sequence containing the ciphertext $\mathsf{ctxt}_3$. Using an AEAD scheme, the initiator produces $\mathsf{ctxt}_3$ by encrypting $\mathsf{ptxt}_3 = (\mathsf{kid}_I, \sigma_3, ead_3)$ with the key $\mathsf{K}_3$ and IV $\mathsf{IV}_3$. In the preceding $\mathsf{K}_3 = \mathsf{Expand}(\mathsf{PRK}_{2e}, (3, \mathsf{th}_3, \mathsf{klen}_3), \mathsf{klen}_3)$ and, the IV is computed as $\mathsf{IV}_3 = \mathsf{Expand}(\mathsf{PRK}_{2e}, (4, \mathsf{th}_3, \mathsf{ivlen}_3), \mathsf{ivlen}_3)$. Let $\mathsf{ad}_3 = (\mathsf{l}_{\mathsf{aead}}, "", \mathsf{th}_3)$ denote associated data, then $\mathsf{ctxt}_3 = \mathsf{AEAD.Enc}(\mathsf{K}_3, \mathsf{ptxt}_3, \mathsf{ad}, \mathsf{IV}_3)$. At last, the responder sends $\mathsf{msg}_3$ to the responder.

**Processing of message 3 by the responder.** Upon receiving and parsing $\mathsf{msg}_3$, the responder decrypts the received ciphertext $\mathsf{ctxt}_3$ and verifies the signature $\sigma_3$. Similar to the initiator's processing of $\mathsf{msg}_2$, the responder may have to verify the signature against multiple public keys referenced by $\mathsf{kid}_I$ until at least one verifies. At this point, the responder knows the identity of the peer. The responder aborts the protocol if any of the verification steps fails.

**Session key.** Upon successful exchange of the three protocol messages, the initiator, and the responder derive a session key $\mathsf{PRK}_{out}$ as follows. First, they both compute the partial transcript hash $\mathsf{th}_4 = \mathsf{H}(\mathsf{th}_3, \mathsf{ptxt}_3, \mathsf{cred}_I)$. Then, they compute $\mathsf{PRK}_{out} = \mathsf{Expand}(\mathsf{PRK}_{2e}, (9, \mathsf{th}_4, \mathsf{klen}_{out}), \mathsf{klen}_{out})$.

**EDHOC Message 4** If EDHOC is used for authentication only and no application data is sent, then the responder must send a fourth message. This

[4]As mentioned before, the value of $\mathsf{th}_3$ described in this text is different from what draft 14 currently does. We suggested to the working group this modification to strengthen explicit authentication in EDHOC, as discussed in Chapter 5.

message is AEAD encrypted with the key $\mathsf{K}_4$ and initialization vector $\mathsf{IV}_4$. This message is optional, and we do not consider it further in this thesis. The key schedule of EDHOC SIG-SIG is shown in Figure 3.2.

## 3.4 Key Schedule of EDHOC SIG-SIG

We summarize the key schedule of EDHOC SIG-SIG, including all elements that enter the key schedule and how they are computed. The computation of $\mathsf{th}_3$ and $\mathsf{th}_4$ shown is performed according to our proposed improvements [Mat22].

**Transcript hashes.** We summarize the partial transcript hashes $\mathsf{th}_2$, $\mathsf{th}_3$ and $\mathsf{th}_4$ computed throughout the protocol in the table below.

| Partial transcript hash | Value |
|---|---|
| $\mathsf{th}_2$ | $\mathsf{H}(G_y, \mathsf{C}_R, \mathsf{H}((M, \mathsf{S}, G_x, \mathsf{C}_I, ead_1)))$ |
| $\mathsf{th}_3$ | $\mathsf{H}(\mathsf{th}_2, \mathsf{ptxt}_2, \mathsf{cred}_R)$ |
| $\mathsf{th}_4$ | $\mathsf{H}(\mathsf{th}_3, \mathsf{ptxt}_3, \mathsf{cred}_I)$ |

**Table 3.1:** The partial transcript hashes computed in EDHOC SIG-SIG.

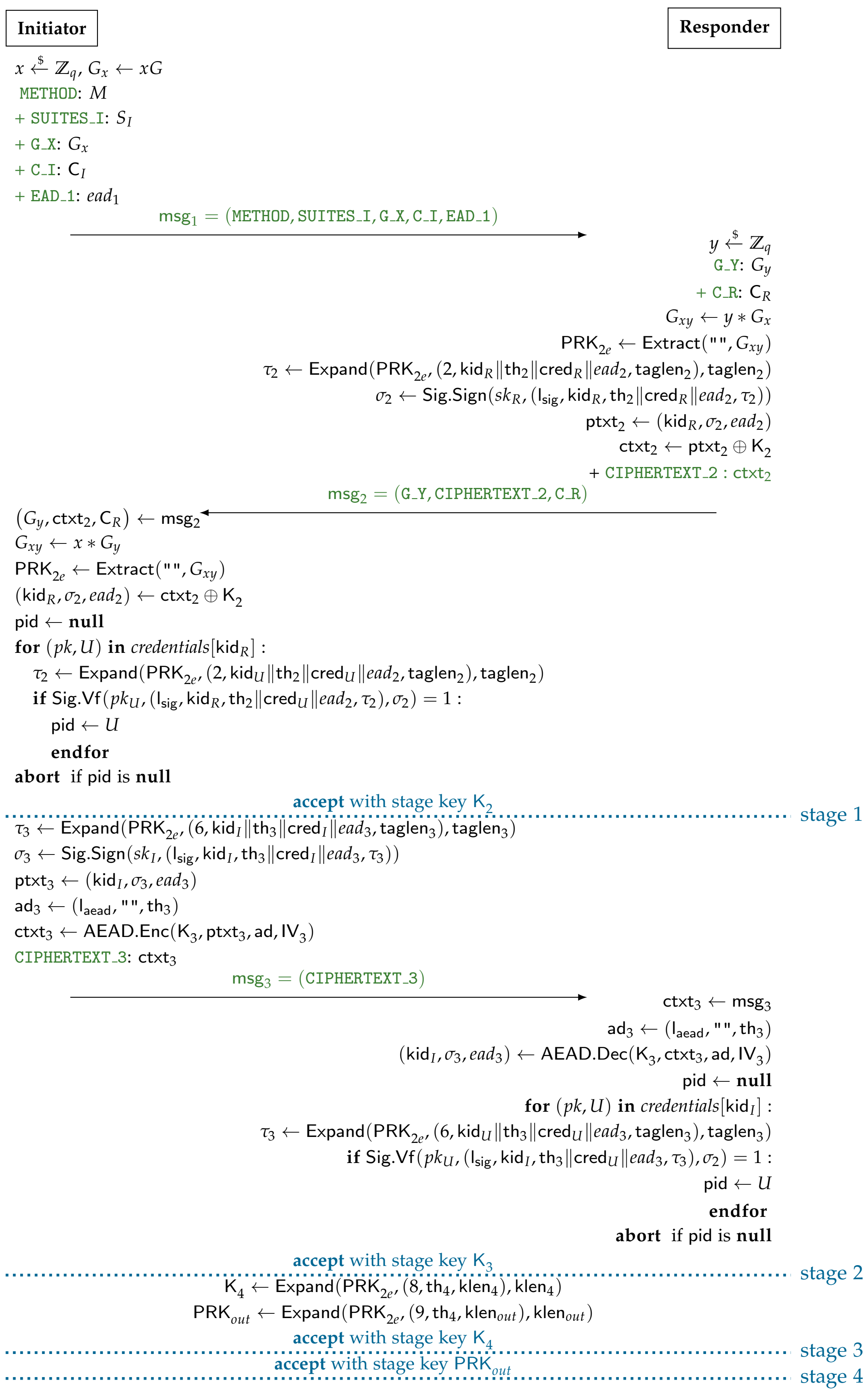
**Protocol keys.** We summarize the keys, IVs, and MAC tags computed throughout the protocol in the table below.

| Key | Value |
|---|---|
| $\mathsf{K}_2$ | $\mathsf{Expand}(\mathsf{PRK}_{2e}, (0, \mathsf{th}_2, \lvert\mathsf{ptxt}_2\rvert), \lvert\mathsf{ptxt}_2\rvert)$ |
| $\mathsf{K}_3$ | $\mathsf{Expand}(\mathsf{PRK}_{2e}, (3, \mathsf{th}_3, \mathsf{klen}_3), \mathsf{klen}_3)$ |
| $\mathsf{K}_4$ | $\mathsf{Expand}(\mathsf{PRK}_{2e}, (8, \mathsf{th}_4, \mathsf{klen}_4), \mathsf{klen}_4)$ |
| $\mathsf{PRK}_{out}$ | $\mathsf{Expand}(\mathsf{PRK}_{2e}, (9, \mathsf{th}_4, \mathsf{klen}_{out}), \mathsf{klen}_{out})$ |

| initialization vector | Value |
|---|---|
| $\mathsf{IV}_3$ | $\mathsf{Expand}(\mathsf{PRK}_{2e}, (4, \mathsf{th}_3, \mathsf{ivlen}_3), \mathsf{ivlen}_3)$ |
| $\mathsf{IV}_4$ | $\mathsf{Expand}(\mathsf{PRK}_{2e}, (7, \mathsf{th}_4, \mathsf{ivlen}_4), \mathsf{ivlen}_4)$ |

| MAC tag | Value |
|---|---|
| $\tau_2$ | $\mathsf{Expand}(\mathsf{PRK}_{2e}, (2, \mathsf{kid}_R \Vert \mathsf{th}_2 \Vert \mathsf{cred}_R \Vert ead_2, \mathsf{taglen}_2), \mathsf{taglen}_2)$ |
| $\tau_3$ | $\mathsf{Expand}(\mathsf{PRK}_{3e2m}, (6, \mathsf{kid}_I \Vert \mathsf{th}_3 \Vert \mathsf{cred}_I \Vert ead_3, \mathsf{taglen}_3), \mathsf{taglen}_3)$ |

**Table 3.2:** The protocol (top), IVs (middle), and MAC tags (bottom) in EDHOC SIG-SIG. $\lvert\mathsf{ptxt}_2\rvert$ is the length of the second protocol message.



```
Initiator                                                                     Responder
$x \xleftarrow{\$} \mathbb{Z}_q$, $G_x \leftarrow xG$
METHOD: $M$
+ SUITES_I: $S_I$
+ G_X: $G_x$
+ C_I: $\mathsf{C}_I$
+ EAD_1: $ead_1$
                    $\mathsf{msg}_1$ = (METHOD, SUITES_I, G_X, C_I, EAD_1)
                    ----------------------------------------------------->
                                                                  $y \xleftarrow{\$} \mathbb{Z}_q$
                                                                  G_Y: $G_y$
                                                                  + C_R: $\mathsf{C}_R$
                                                                  $G_{xy} \leftarrow y * G_x$
                                                                  $\mathsf{PRK}_{2e} \leftarrow \mathsf{Extract}("", G_{xy})$
        $\tau_2 \leftarrow \mathsf{Expand}(\mathsf{PRK}_{2e}, (2, \mathsf{kid}_R \| \mathsf{th}_2 \| \mathsf{cred}_R \| ead_2, \mathsf{taglen}_2), \mathsf{taglen}_2)$
        $\sigma_2 \leftarrow \mathsf{Sig.Sign}(sk_R, (\mathsf{l}_{\mathsf{sig}}, \mathsf{kid}_R, \mathsf{th}_2 \| \mathsf{cred}_R \| ead_2, \tau_2))$
                                                                  $\mathsf{ptxt}_2 \leftarrow (\mathsf{kid}_R, \sigma_2, ead_2)$
                                                                  $\mathsf{ctxt}_2 \leftarrow \mathsf{ptxt}_2 \oplus \mathsf{K}_2$
                                                                  + CIPHERTEXT_2 : $\mathsf{ctxt}_2$
                    $\mathsf{msg}_2$ = (G_Y, CIPHERTEXT_2, C_R)
$(G_y, \mathsf{ctxt}_2, \mathsf{C}_R) \leftarrow \mathsf{msg}_2$ <-----------------------------------
$G_{xy} \leftarrow x * G_y$
$\mathsf{PRK}_{2e} \leftarrow \mathsf{Extract}("", G_{xy})$
$(\mathsf{kid}_R, \sigma_2, ead_2) \leftarrow \mathsf{ctxt}_2 \oplus \mathsf{K}_2$
$\mathsf{pid} \leftarrow$ null
for $(pk, U)$ in $credentials[\mathsf{kid}_R]$ :
  $\tau_2 \leftarrow \mathsf{Expand}(\mathsf{PRK}_{2e}, (2, \mathsf{kid}_U \| \mathsf{th}_2 \| \mathsf{cred}_U \| ead_2, \mathsf{taglen}_2), \mathsf{taglen}_2)$
  if $\mathsf{Sig.Vf}(pk_U, (\mathsf{l}_{\mathsf{sig}}, \mathsf{kid}_R, \mathsf{th}_2 \| \mathsf{cred}_U \| ead_2, \tau_2), \sigma_2) = 1$ :
    $\mathsf{pid} \leftarrow U$
    endfor
abort if pid is null
                    accept with stage key $\mathsf{K}_2$
.............................................................................. stage 1
$\tau_3 \leftarrow \mathsf{Expand}(\mathsf{PRK}_{2e}, (6, \mathsf{kid}_I \| \mathsf{th}_3 \| \mathsf{cred}_I \| ead_3, \mathsf{taglen}_3), \mathsf{taglen}_3)$
$\sigma_3 \leftarrow \mathsf{Sig.Sign}(sk_I, (\mathsf{l}_{\mathsf{sig}}, \mathsf{kid}_I, \mathsf{th}_3 \| \mathsf{cred}_I \| ead_3, \tau_3))$
$\mathsf{ptxt}_3 \leftarrow (\mathsf{kid}_I, \sigma_3, ead_3)$
$\mathsf{ad}_3 \leftarrow (\mathsf{l}_{\mathsf{aead}}, "", \mathsf{th}_3)$
$\mathsf{ctxt}_3 \leftarrow \mathsf{AEAD.Enc}(\mathsf{K}_3, \mathsf{ptxt}_3, \mathsf{ad}, \mathsf{IV}_3)$
CIPHERTEXT_3: $\mathsf{ctxt}_3$
                    $\mathsf{msg}_3$ = (CIPHERTEXT_3)
                    ----------------------------------------------------->
                                                                  $\mathsf{ctxt}_3 \leftarrow \mathsf{msg}_3$
                                                                  $\mathsf{ad}_3 \leftarrow (\mathsf{l}_{\mathsf{aead}}, "", \mathsf{th}_3)$
                              $(\mathsf{kid}_I, \sigma_3, ead_3) \leftarrow \mathsf{AEAD.Dec}(\mathsf{K}_3, \mathsf{ctxt}_3, \mathsf{ad}, \mathsf{IV}_3)$
                                                                  $\mathsf{pid} \leftarrow$ null
                                                                  for $(pk, U)$ in $credentials[\mathsf{kid}_I]$ :
        $\tau_3 \leftarrow \mathsf{Expand}(\mathsf{PRK}_{2e}, (6, \mathsf{kid}_U \| \mathsf{th}_3 \| \mathsf{cred}_U \| ead_3, \mathsf{taglen}_3), \mathsf{taglen}_3)$
                    if $\mathsf{Sig.Vf}(pk_U, (\mathsf{l}_{\mathsf{sig}}, \mathsf{kid}_I, \mathsf{th}_3 \| \mathsf{cred}_U \| ead_3, \tau_3), \sigma_2) = 1$ :
                                                                  $\mathsf{pid} \leftarrow U$
                                                                  endfor
                                                                  abort if pid is null
                    accept with stage key $\mathsf{K}_3$
.............................................................................. stage 2
                    $\mathsf{K}_4 \leftarrow \mathsf{Expand}(\mathsf{PRK}_{2e}, (8, \mathsf{th}_4, \mathsf{klen}_4), \mathsf{klen}_4)$
              $\mathsf{PRK}_{out} \leftarrow \mathsf{Expand}(\mathsf{PRK}_{2e}, (9, \mathsf{th}_4, \mathsf{klen}_{out}), \mathsf{klen}_{out})$
                    accept with stage key $\mathsf{K}_4$
.............................................................................. stage 3
                    accept with stage key $\mathsf{PRK}_{out}$
.............................................................................. stage 4
```

**Figure 3.1:** The EDHOC SIG-SIG protocol. The fourth optional message $\mathsf{msg}_4$ is omitted. The computation of the partial transcript hashes $\mathsf{th}_i$ and protocol keys are summarized in Table 3.1

**Key Exporter and OSCORE context.** Once the session key $\mathsf{PRK}_{out}$ is established, the initiator and responder can derive further application keys when needed. The exporter mechanism is used to archive that. To do so, first an exporter key $\mathsf{PRK}_{exp}$ is derived using $\mathsf{PRK}_{out}$ and the Expand function as follows $\mathsf{PRK}_{exp} = \mathsf{Expand}(\mathsf{PRK}_{out}, (10, \texttt{""}, \mathsf{klen}_{exp}), \mathsf{klen}_{exp})$.

An application key $\mathsf{K}$ of length $\mathsf{klen}$ is then derived by computing $\mathsf{K} = \mathsf{Expand}(\mathsf{PRK}_{exp}, (\mathsf{label}, \mathsf{context}, \mathsf{klen}), \mathsf{klen})$. The pair $(\mathsf{label}, \mathsf{context})$ must be unique per application.

To establish and OSCORE context, a master key and mast salt are computed using the exporter mechanism. Namely, the exporter mechanism is queried with the following label-context $(0, \texttt{""})$ for the master secret and $(1, \texttt{""})$ for the master salt.

**Lightweight key update.** To limit the lifetime of keying material, EDHOC specifies a lightweight key update mechanism, in the sense that it does not require a re-run of the EDHOC protocol. Instead, the initiator and responder can update the session key $\mathsf{PRK}_{out}$ by computing the new key as follows: $\mathsf{PRK}_{out} = \mathsf{Expand}(\mathsf{PRK}_{out}, (11, \mathsf{context}, \mathsf{klen}_{out}), \mathsf{klen}_{out})$. The context input is used to bind the key update to a specific event that triggered the key update.

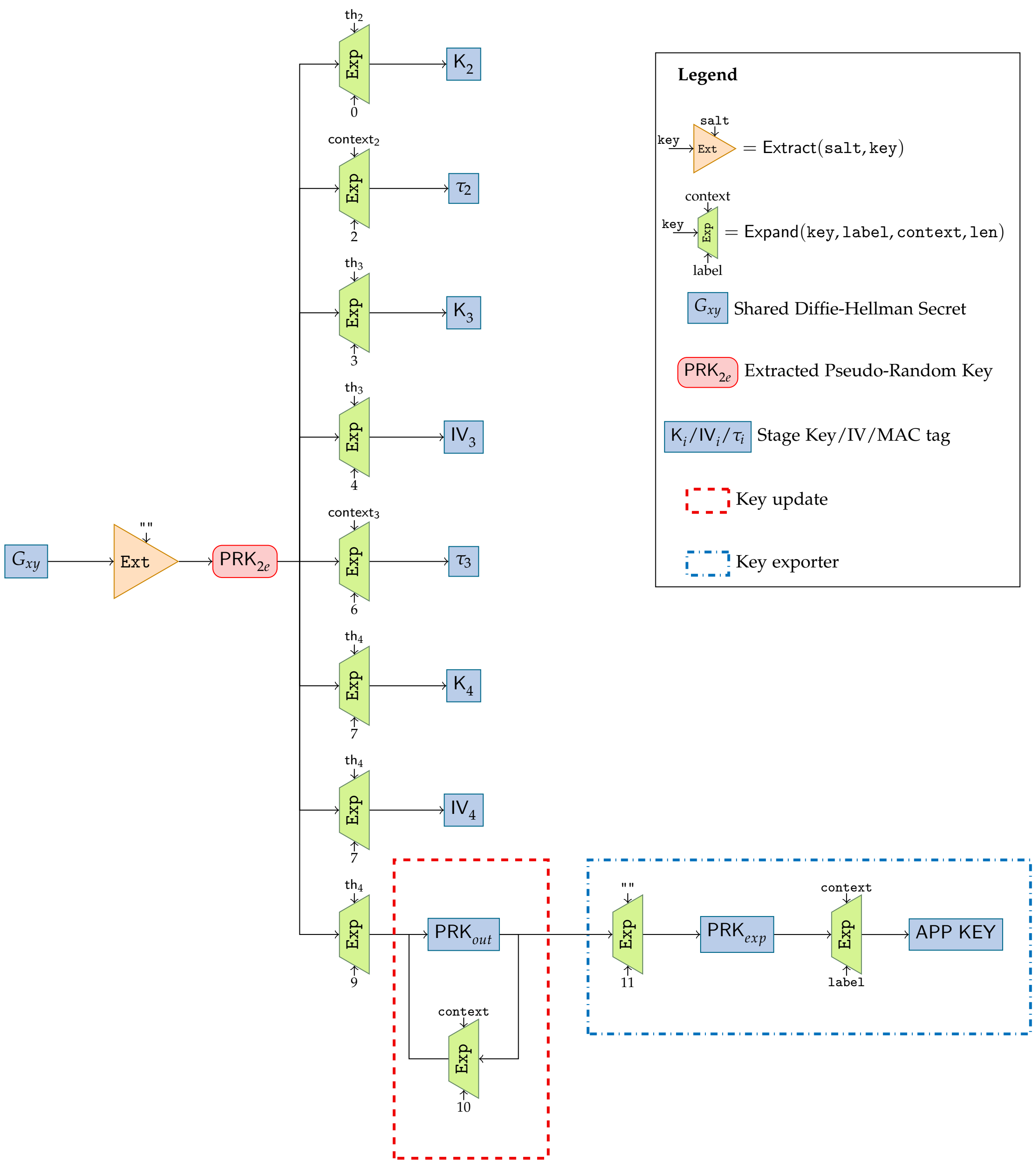


**Figure 3.2:** The EDHOC key schedule, including the key update and exporter mechanism. The computation of the transcript hashes $(th_2, th_3, th_4)$ is shown in Table 3.1. The values of $context_2$ and $context_3$ are shown in Table 3.2.

Chapter 4

# Code-based Multi-stage Key Exchange Model

This chapter introduces the security model we use to analyze the EDHOC protocol, namely the Multi-Stage Key Exchange model. We first describe the MSKE model and then specify our MSKE model tailored for EDHOC.

## 4.1 The Multi-stage Key Exchange Model

We analyze EDHOC in the Multi-Stage Key Exchange Model [FG14, Gü18]. It is the state-of-the-art model for analyzing key exchange protocols wherein not only one but multiple keys are derived at various stages throughout the protocol. It is an extension of the seminal work by Bellare and Rogaway [BR94]. Their work models the use of a key exchange protocol by several parties; each party may run multiple concurrent protocol instances called *sessions*. In BR-like models, the security of the session key derived through a key exchange protocol is captured by the notion of indistinguishability of the session key. Namely, a key exchange protocol is shown secure via a game played by a strong adversary with complete control over the network; the adversary can additionally reveal session keys and long-term secrets. Key indistinguishability states that the adversary should not be able to differentiate a "fresh" session key from a randomly generated key.

The MSKE model is formally specified as a code-based game, also capturing a strong adversary who can arbitrarily modify the communication between sessions. In the MSKE model, the adversary's control over the network is captured by providing an Send oracle. Additionally, the adversary can create new users with the oracle NewUser, it can instantiate new sessions by querying the NewSession oracle; it can reveal stage keys with the RevSessionKey oracle and can reveal the long-term secret signing key of a protocol participant with the RevLongTermKey oracle. The MSKE model extends the notion of indistinguishability of keys to all stage keys. Namely, an adversary with the capabilities described by the oracles listed above is given access to an Test oracle, where it can ask to reveal a stage key of a chosen session and stage. The oracle returns either the real stage key or a random one. Security then demands that the adversary cannot distinguish which key is received, assuming that the said adversary did not perform any action that allows winning trivially. Overall, we are interested in the following security guarantees for each stage key:

**Key indistinguishability.** This notion states that stage keys should "look" random from the adversary's standpoint. More formally, we let the MSKE challenger choose a random challenge bit $b$; then, the adversary may issue Test queries, and at the end, the adversary outputs a guess $b'$. Key indistinguishability is captured by the adversary's advantage in guessing the challenge bit $b$. To exclude trivial winning conditions, our model specifies a freshness condition for tested stage keys. Therefore, the guessing advantage is only valid for adversaries who respect freshness.

**Forward Secrecy.** Even if the long-term secret of the session owner is compromised at a given time, the already established stage keys should remain indistinguishable from random once established.

**Explicit Authentication.** This notion states that for a given stage, only the intended peer has knowledge of the stage key *and* the peer session is guaranteed to exist.

Before we define our model syntax, we briefly discuss some specificities of the EDHOC protocol that require careful treatment and justify our modifications to the MSKE model.

### 4.1.1 MSKE model for EDHOC

We recall the usage of credential identifiers in EDHOC during authentication. The credential identifier of user `X` denote `ID_CRED_X`. The responder sends its credential identifier (`ID_CRED_R`) in the second protocol message; the initiator sends its identifier (`ID_CRED_I`) in the third protocol message. While these credential identifiers may contain the actual credential itself, the primary operating mode is that each protocol participant will have local access to the credentials of most peers. In this case, a credential identifier is simply a small value that allows a party to retrieve the credential of the peer. For instance, the `kid` parameter of a `COSE_key` can be used as a credential identifier. However, per the COSE standard, users should not assume that all types of credential identifiers identify a unique identity. Quoting the EDHOC draft 14 Section 3.5.3:

> As stated in Section 3.1 of [I-D.ietf-cose-rfc8152bis-struct], applications MUST NOT assume that 'kid' values are unique, and several keys associated with a 'kid' may need to be checked before the correct one is found. Applications might use additional information such as 'kid context' or lower layers to determine which key to try first. Applications should strive to make ID_CRED_X as unique as possible since the recipient may otherwise have to try many keys.

Based on this observation, we take a conservative approach and assume that all types of credential identifiers may not necessarily identify a unique credential. Formally, we translate this fact in our MSKE model by augmenting our game with global credential sets denoted $\mathit{peerpk}_{\mathsf{kid}}$ that contain all credentials identified by the credential identifier kid. Furthermore, we give the adversary full control over the choice of the credential identifiers for both honest and compromised users. More precisely, the oracle NewUser is modified to allow the adversary to provide a credential identifier and an optional adversarial key pair, in which case the newly created user is considered comprised upon creation. Our approach follows that of Boyd *et al.* [BCF+13]. One can consider EDHOC SIG-SIG as an ASICS protocol, where adversarial registration of long-term keys is limited to pkregister queries.

## 4.2 Model Syntax

We now specify the syntax we use in our model. First, we discuss the specificities of EDHOC to justify our extension to the MSKE model, then we describe the protocol-specific syntax, including our abstraction of key exchange protocols and protocol specific-variables. Finally, we describe the session-specific syntax used in our security game.

### 4.2.1 Protocol Syntax

In our model, a key exchange protocol KE is abstracted as a triple of algorithms $(\mathsf{KGen}, \mathsf{Activate}, \mathsf{Run})$ that we describe below.

**KGen.** The KGen algorithm generates long-term public and secret key pairs for each user. For every user $U$, the associated key pair is denoted by $(sk_U, pk_U) \xleftarrow{\$} \mathsf{KGen}()$. In the context of EDHOC, these key pairs are signing and verification keys.

**Activate.** The algorithm $\mathsf{Activate}(U, sk_U, \{\mathsf{pid}\}_U, \mathit{peerpk}, \mathsf{role}) \xrightarrow{\$} (\pi_U^i, m)$ start a new session $\pi_U^i$ owned by the user $U$, with a list $\{\mathsf{pid}\}_U$ of peers that the user $U$ is willing to engage with in the key exchange protocol. If $\mathsf{role} = \mathit{initiator}$, Activate returns the first protocol message $m$ and $\perp$ otherwise.

**Run.** The algorithm $\mathsf{Run}(U, sk_U, \pi_U^i, peerpk, m) \xrightarrow{\$} (\pi_U^i, m')$ delivers the protocol message $m$ to the session $\pi_U^i$. The message $m$ is processed according to the protocol specification, and $\pi_U^i$ is updated accordingly. Finally, Run outputs a response message $m'$ or the symbol $\perp$ in case of an error.

.

### 4.2.2 Protocol variables

For our security game, we need to extend the exchange protocol KE with the following variables:

- KE.$\mathbf{S} \in \mathbb{N}$: the number of stages in the protocol. The set of all stages is denoted by $\mathcal{S} = [1, \mathbf{S}]$.
- KE.$\mathbf{use}[s] \in \{internal, external\}$: indicates whether the $s$'th stage key is used internally within the protocol to encrypt protocol messages, for instance, or externally to protect application data.
- KE.$\mathbf{eauth}[r, s] \in \mathcal{S} \cup \{\infty\}$: indicates the stage at which the session in role $r$ receives explicit authentication for the stage $s$. In other words, the session in role $r$ considers the other peer explicitly authenticated in stage $s$ upon acceptance of stage $\mathbf{auth}[r, s]$.
- KE.$\mathbf{fs}[s] \in \{\perp, fs\}$: indicates whether stage $s$ is forward-secret, predicated on acceptance of stage $s$.

### 4.2.3 Session variables

A session owned by the user $U$ and uniquely[1] identified by the label, $i$ is denoted by $\pi_U^i$. Each session holds the following variables:

- $\pi_U^i.\mathsf{id} \in \mathcal{U}$: the identity of the session owner.
- $\pi_U^i.\mathsf{pid} \in \mathcal{U} \cup \{*\}$: the identity of the intended peer. $\pi_U^i.\mathsf{pid}$ is initialized with the value $*$, which denotes an a-priory unknown identity that may be eventually specified later.
- $\pi_U^i.\mathsf{role} \in \{init, resp\}$: the role of the session owner.
- $\pi_U^i.\mathsf{stage} \in \mathcal{S}$: the current execution stage. $\pi_U^i.\mathsf{stage}$ is initialized with the value 1. It is incremented by one after acceptance of the stage.
- $\pi_U^i.\mathsf{status}[s] \in \{\perp, running, accepted, rejected\}$: the state of execution of stage $s$. $\pi_U^i.\mathsf{status}[s]$ is initialized with $\perp$; it is set to *running* once $\pi_U^i.\mathsf{stage}$ is set to $s$. Upon acceptance of stage $s$, $\pi_U^i.\mathsf{status}[s]$ is set to *accepted*; otherwise, it is set to *rejected* if the stage is rejected.

[1]Session labels only need to be unique for each user.

- $\pi_U^i.\mathsf{key}[s] \in \mathcal{K}_s$: the session key derived in stage $s$. The stage $s$ key has key space $\mathcal{K}_s$. $\pi_U^i.\mathsf{key}[s]$ is initialized with the value $\perp$.
- $\pi_U^i.\mathsf{revealed}[s] \in \mathbb{N} \cup \{\infty\}$: the time at which the $s$'th stage key was revealed. It is initialized with the value $\infty$, which means that the key has not been revealed. When used as a predicate, $\mathsf{revealed}[s]$ means $\mathsf{revealed}[s] \neq \infty$.
- $\pi_U^i.\mathsf{accepted}[s] \in \mathbb{N} \cup \{\infty\}$ : the time at which the $s$'th stage was accepted. $\mathsf{accepted}[s]$ is initially set to $\infty$, which means that the stage has not been accepted. When used as a predicate, $\mathsf{accepted}[s]$ means $\mathsf{accepted}[s] \neq \infty$.
- $\pi_U^i.\mathsf{tested}[s] \in \mathbb{N} \cup \{\infty\}$: time at which the $s$'th stage key was tested. $sessTested[s]$ is initially set to $\infty$. When used as a predicate, $\mathsf{tested}[s]$ means $\mathsf{tested}[s] \neq \infty$.

**Session identifiers.** To be meaningful, our security notion must exclude trivial winning conditions, in particular by disallowing the adversary to test and reveal two partnered sessions. We use session identifiers to define when two sessions are partnered, namely if they hold the same session identifier at a given stage. Therefore, we augment our session objects with the following variables:

- $\pi_U^i.\mathsf{sid}[s] \in \{0,1\}^* \cup \{\perp\}$: the session identifier of stage $s$. $\pi_U^i.\mathsf{sid}[s]$ is initialized with the value $\perp$.

**Contributive identifiers.** To exclude other trivial winning conditions, we also keep track of contributive identifiers. These specify the values a session must have honestly received before allowing the adversary to test a stage where no authentication guarantees are provided. Therefore, session objects also keep the following variables:

- $\pi_U^i.\mathsf{cid}[r,s] \in \{0,1\}^* \cup \{\perp\}$: the contributive identifier for the session in role $r$ in stage $s$. Let $\bar{r}$ denote the role opposite to $r$, then $\mathsf{cid}[\bar{r},s]$ contains the values that the session in role $r$ has honestly received. $\mathsf{cid}[r,s]$ is initially set to $\perp$.

## 4.3 MSKE Security Game

Our MSKE security game is formally defined in Figure 4.1.

### 4.3.1 MSKE security of Key Exchange Protocols

**Security properties** The security properties are formally captured in the game shown in Figure 4.2 and encoded in the predicates Sound, ExplicitAuth.

Furthermore, the Fresh predicate encodes all trivial winning conditions that must be excluded when proving key indistinguishability.

**Game variables.** Throughout the lifetime of the game $\mathbf{G}_{\text{MSKE}}(\mathsf{KE})$, we track the following game-specific variables:

- $\mathcal{T}_s = \{\pi : \pi.\mathsf{tested}[s]\}$ : the set of all sessions that $\mathcal{A}$ tested at stage $s$. It is initialized with the value $\emptyset$. Whenever $\mathcal{A}$ makes a Test query for a session $\pi_U^i$ in stage $s$, $\mathcal{T}_s$ is updated with the value $\mathcal{T}_s = \mathcal{T}_s \cup \{\pi_U^i\}$. The set of all tested sessions is denoted by $\mathcal{T} = \bigcup\limits_{s \leq \mathbf{S}} \mathcal{T}_s$.
- $\mathcal{R}_s = \{\pi : \pi.\mathsf{revealed}[s]\}$: the set of all sessions for which $\mathcal{A}$ revealed at $s'$t stage key. The set of all revealed sessions is denoted by $\mathcal{R} = \bigcup\limits_{s \leq \mathbf{S}} \mathcal{R}_s$.
- $\mathcal{P}_s = \{(x, y) : x \neq y \wedge x.\mathsf{sid}[s] = y.\mathsf{sid}[s] \neq \bot\}$: the set of sessions partnered at stage $s$. The $\mathcal{P}_s$ are not explicitly constructed (nor explicitly updated with insertions) but are defined via the predicate on the session identifiers. As a consequence of the definition $(x, y) \in \mathcal{P}_s \iff (y, x) \in \mathcal{P}_s$. The multiset of all partnered sessions is $\mathcal{P} = \bigcup\limits_{s \leq \mathbf{S}} \mathcal{P}_s$.
- $\mathit{users} \in \mathbb{N}$: the number of users in the current game.
- $\mathit{time} \in \mathbb{N}$: a discrete value used to order events in the game.
- $\mathit{revltk}_U$: the time at which the long-term secret of $U$ was compromised.
- $\mathit{peerpk}_{\mathsf{kid}} = \{(U, pk_U)\}$: the set of all credentials (identity and public key pairs) identified by the credential identifier $\mathsf{kid}$. $\mathit{peerpk}_{\mathsf{kid}}$ is initially set to $\emptyset$. We demand that $\mathit{peerpk}_{\mathsf{kid}} \cap \mathit{peerpk}_{\mathsf{kid}_I} = \emptyset$ for all $\mathsf{kid} \neq \mathsf{kid}'$. That is, one $\mathsf{kid}$ may refer to multiple key pairs, but a key pair is associated to a single $\mathsf{kid}$ The global list of credentials is $\mathit{peerpk} = \bigcup\limits_{\mathsf{kid}} \mathit{peerpk}_{\mathsf{kid}}$.

**MSKE security definition.** We now state our main definition of multi-stage key exchange security for a protocol $\mathsf{KE}$ against an MSKE adversary $\mathcal{A}$. At a high level, the adversary wins the game by violating the soundness of the session identifiers, breaking explicit authentication, or distinguishing the challenge bit; the latter condition is available only if $\mathcal{A}$ is a freshness-respecting adversary.

**Definition 4.1 (Multi-stage key exchange security)** *Let* $\mathsf{KE}$ *be a key exchange protocol. Let* $\mathbf{G}^{\text{MSKE}}(\mathsf{KE})$ *be the MSKE game defined in Figure 4.1. We define the advantage of an MSKE adversary* $\mathcal{A}$ *against* $\mathsf{KE}$ *is defined by:*

$$\mathsf{Adv}_{\mathcal{A}}^{\text{MSKE}}(\mathsf{KE}) = \Pr\Big[\mathbf{G}^{\text{MSKE}}(\mathsf{KE}) \to 1\Big].$$

```
$G^{\mathsf{MSKE}}_{\mathcal{A},\mathsf{KE}}$

INITIALIZE
 1 : $time \leftarrow 0$
 2 : $b \xleftarrow{\$} \{0,1\}$
 3 : $peerpk \leftarrow \emptyset$

NEWUSER($sk, pk, \mathsf{kid}$)
 1 : $time \leftarrow time + 1$
 2 : $users \leftarrow users + 1$
 3 : $U \leftarrow users$
 4 : $(pk_U, sk_U) \xleftarrow{\$} \mathsf{KGen}()$
 5 : $revltk_U \leftarrow \infty$
 6 : if $pk \neq \perp$ and $pk$ is valid then :
 7 : // Only valid verification keys are allowed.
 8 :    $(sk_U, pk_U) \leftarrow (sk, pk)$
 9 :    $revltk_U \leftarrow time$ // adversarially registerd keys
10 :          // are considered compromised.
11 : // Add $(U, pk_U)$ to $peerpk_{\mathsf{kid}}$
12 : $peerpk_{\mathsf{kid}} \leftarrow peerpk_{\mathsf{kid}} \cup \{(U, pk_U)\}$
13 : return $pk_U$

NEWSESSION($U, sk_U, \{\mathsf{pid}\}_U, peerpk, \mathsf{role}$)
 1 : $time \leftarrow time + 1$
 2 : if $\pi^i_U \neq \perp$ :    return $\perp$
 3 : $(\pi^i_U, m) \xleftarrow{\$} \mathsf{Activate}(U, sk_U, \{\mathsf{pid}\}_U, peerpk, \mathsf{role})$
 4 : return $m$

SEND($U, i, m$)
 1 : $time \leftarrow time + 1$
 2 : if $\pi^i_U = \perp$ :    return $\perp$
 3 : $(\pi^i_U, m') \xleftarrow{\$} \text{RUN}(\pi^i_U, sk_U, peerpk, m)$
 4 : $i \leftarrow \pi^i_U.\mathsf{stage}$
 5 : if $\pi^i_U.\mathsf{status}[s] = accepted$ :
 6 :    $\pi^i_u.accepted[s] \leftarrow time$
 7 : return $(\pi^i_U.\mathsf{status}[s], m')$

ACTIVATE($U, sk_U, \{\mathsf{pid}\}_U, peerpk, \mathsf{role}$)
 1 : $\pi^i_U.\mathsf{id} \leftarrow U$
 2 : $\pi^i_U.\mathsf{role} \leftarrow role$
 3 : $m \leftarrow \text{RUN}(\pi^i_U, peerpk, sk_U, \{\mathsf{pid}\}_U, m)$
 4 : return $(m, \pi^i_U)$

REVSESSIONKEY($U, s, i$)
 1 : $time \leftarrow time + 1$
 2 : if $\pi^i_U = \perp$ or $\pi^i_U.\mathsf{status}[s] \neq accepted$ :
 3 :       return $\perp$
 4 : $\pi^i_U.\mathsf{revealed}[s] \leftarrow \mathsf{true}$
 5 : $\mathcal{R}_s \leftarrow \mathcal{R}_s \cup \{\pi^i_U\}$
 6 : return $\pi^i_U.\mathsf{key}[s]$

REVLONGTERMKEY($U$)
 1 : $time \leftarrow time + 1$
 2 : $revltk_U \leftarrow time$
 3 : return $sk_U$

TEST($U, s, i$)
 1 : $time \leftarrow time + 1$
 2 : if $\pi^i_U = \perp$ or
 3 :    $\pi^i_U.\mathsf{status}[s] \neq accepted$ or
 4 :    $\pi^i_U.\mathsf{tested}[s] = \mathsf{true}$ :
 5 :       return $\perp$
 6 : if $\exists \pi^j_V : (\pi^i_U, \pi^j_V) \in \mathcal{P}_s$ and // $\pi^j_V$ is partnered to $\pi^i_U$ and...
 7 :    $\mathsf{KE}.\mathbf{use}[s] = internal$ and // the key is used internally and...
 8 :    $\pi^j_V.\mathsf{status}[s+1] \neq \perp$ : // the partnered hasn't accepted
 9 :       return $\perp$ // the stage $s$, return $\perp$
10 : $\pi^i_U.\mathsf{tested}[s] \leftarrow \mathsf{true}$
11 : $\mathcal{T}_s \leftarrow \mathcal{T}_s \cup \left\{\pi^i_U\right\}$
12 : $k_0 \xleftarrow{\$} \mathcal{K}_i$
13 : $k_1 \leftarrow \pi^i_U \mathsf{key}[s]$
14 : if $\mathsf{KE}.\mathbf{use}[s] = internal$ : // If key is used internally
15 :    $\pi^i_U \mathsf{key}[s] \leftarrow k_b$ // copy the key in the session for consistency
16 : return $k_b$

FINALIZE($b'$)
 1 : // The adversary wins by...
 2 : if $\neg$Sound : return 1 // breaking soundeness or...
 3 : if $\neg$ExplicitAuth : return 1 // explicit authentication or..
 4 : if $\neg$Fresh : $b' \leftarrow 0$ // (if it respected freshness)...
 5 : return $b = b'$ // ...by guessing the challenge bit
```

**Figure 4.1:** The multi-stage key exchange security game for a key exchange protocol KE. The predictates Sound, ExplicitAuth and Fresh are show in Figure 4.2

Fresh

```
// A session was tested and revealed
if $\exists s : \mathcal{T}_s \cap \mathcal{R}_s \neq \emptyset$ then // in stage s.
  return false
// Parterned sessions...
if $\exists s : \mathcal{P}_s \cap \mathcal{R}_s \times \mathcal{T}_s$ then
  // one tested, one revealed in stage $s$
    return false
// Parterned sessions...
if $\exists i, \pi^i_U, \pi^j_V : (\pi^i_U, \pi^j_V) \in \mathcal{P}_s$
// one is revealed and the other tested in stage $s$
  $\wedge (\pi^i_U \in \mathcal{T}_s) \wedge (\pi^j_V \in \mathcal{R}_s)$ then
    return false
// Forward secret stages are fresh...
if $\exists \pi^i_U, s : \mathsf{KE}.\mathbf{fs}[s] = fs \wedge (\pi^i_U \in \mathcal{T}_s) \wedge$
// accepted before peer compromise or...
  $(revltk \pi^i_U.\mathsf{pid} < \pi^i_U.\mathsf{accepted}[t])$
// there is a cid partnered
  $\wedge (\forall \pi^j_V \neq \pi^i_U : \pi^i_U.\mathsf{cid}[\pi^j_V.\mathsf{role}, i] \neq \pi^j_V.\mathsf{cid}[\pi^i_U.\mathsf{role}, i])$
    return false
// Unauthenticated stages are fresh ony if..
if $\exists \pi^i_U \in \mathcal{T}_s : \mathbf{eauth}[\overline{\pi^i_U.\mathsf{role}}, s] = \perp \wedge$
  $\forall \pi^j_V : \pi^i_U.\mathsf{cid}[\overline{\pi^i_U.\mathsf{role}}, s] \neq \pi^j_V.\mathsf{cid}[\overline{\pi^i_U.\mathsf{role}}, s]$
// there's a cid partner
  return false
return true
```

ExplicitAuth

```
// Explicit auth is true all session and stages if...
// All session having accepted stage $s$..
$\forall \pi^i_U, s : \pi^i_U.\mathsf{accepted}[s] \wedge$
  $\mathbf{eauth}[\overline{\pi^i_U.\mathsf{role}}, s] = s' < \infty$ // (achieving exp auth at stage $s'$)...
  $\wedge\, \pi^i_U.\mathsf{accepted}[s'] < revltk_{\pi^i_U.\mathsf{pid}}$ // and accepted stage $s'$
// before compromise of peers's secret,
  $\implies \exists \pi^j_V :$ // The exists a session...
  $\pi^i_U.\mathsf{sid}[s'] = \pi^j_V.\mathsf{sid}[s']$ // partnered with $\pi^i_U$ in stage $s'$ and..
// if $\pi^j_V$ accepts stage $s$ before $U$ is comprised...
  $\wedge\, \pi^j_V.\mathsf{accepted}[s] < revltk_{\pi^i_U.\mathsf{id}} \implies \pi^i_U.\mathsf{sid}[s] = \pi^j_V.\mathsf{sid}[s]$
// $\pi^j_V$ accepts with peer identity $U$
```

Sound

```
// More than two sessions are partnered in statge $s$
if $\exists s, \pi^i_U, \pi^j_V, \pi^k_W : (\pi^i_U, \pi^j_V) \in \mathcal{P}_s \wedge$
  $(\pi^i_U, \pi^k_W) \in \mathcal{P}_s \wedge (\pi^j_V, \pi^k_W) \in \mathcal{P}_s$ then
    return false
// Parterned sessions..
if $\exists s, \pi^i_U, \pi^j_V : (\pi^i_U, \pi^j_V) \in \mathcal{P}_s$
  $\wedge (\pi^i_U.\mathsf{accepted}[s] \wedge \pi^j_V.\mathsf{accepted}[s])$
  $\wedge (\pi^i_U.\mathsf{key}[s] \neq \pi^j_V.\mathsf{key}[s])$
// have different keys
  then return false
// Condition 2
if $\exists s, \pi^i_U, \pi^j_V : (\pi^i_U, \pi^j_V) \in \mathcal{P}_s \wedge$
  $(\pi^i_U.\mathsf{role} = \pi^j_V.\mathsf{role})$ // in the same role
    then return false
// Parterned sessions...
if $\exists s, \pi^i_U, \pi^j_V : (\pi^i_U, \pi^j_V) \in \mathcal{P}_s \wedge$
  $\exists r \in \{init, resp\}$ : // don't agree
  $\pi^i_U.\mathsf{KE}.\mathbf{auth}[r, s] \neq \pi^j_V.\mathsf{KE}.\mathbf{auth}[r, s]$
// on authentication level for $r$
    then return false
// Parterned sessions...
if $\exists s, (\pi^i_U, \pi^j_V) \in \mathcal{P}_s : \exists r \in \{init, resp\} \wedge$
  $\pi^i_U.\mathsf{cid}[r, s] \neq \pi^j_V.\mathsf{cid}[r, s]$
// don't agree on contributive identifiers
  then return false
// Parterned sessions...
if $\exists s, \pi^i_U, \pi^j_V : (\pi^i_U, \pi^j_V) \in \mathcal{P}_s \wedge$
  $\pi^i_U.\mathsf{pid} \neq V \vee \pi^j_V.\mathsf{pid} \neq V$
// don't agree on peer identities
  return false
// The session identifiers...
if $\exists s \neq t, \pi^i_U : (\pi^i_U, \pi^j_V) \in \mathcal{P}_s$
  $\wedge (\pi^i_U.\mathsf{sid}[t] = \pi^i_U.\mathsf{sid}[s])$ then
// are not different across stages
    return falsereturn true
```

**Figure 4.2:** The MSKE predicates The predictates Sound, ExplicitAuth and Fresh

Chapter 5

# Security Analysis of EDHOC SIG-SIG Mode

In this chapter, we employ the MSKE model introduced in Chapter 4 to analyze the security of the SIG-SIG mode of the EDHOC protocol. To do so, we cast EDHOC in SIG-SIG mode as a multi-stage key exchange protocol, formally modeled in pseudocode (Figure 5.1). We specify the protocol properties of EDHOC in SIG-SIG mode, then prove its security in the MSKE model.

## 5.1 EDHOC SIG-SIG as an MSKE protocol

In the following, we formalize the EDHOC protocol in SIG-SIG mode as an MSKE protocol. More precisely, we define the stages in the protocol with the associated stage keys; we specify whether a stage is internal or not; we specify the stage-specific session identifiers and contributive identifiers. We specify when and which stages are explicitly authenticated, and finally, when and which stages are forward secret.

## 5.2 Protocol properties

**Stages.** The EDHOC protocol in SIG-SIG mode, as described in draft 14, has four stages, hence $\mathbf{S} = 4$. These correspond to the keys (and potentially associated IVs) $\mathsf{K}_2$, $\mathsf{K}_3/\mathsf{IV}_3$, $\mathsf{K}_4/\mathsf{IV}_4$ and $\mathsf{PRK}_{out}$.

**Key usage.** The stage keys and potentially associated IVs $\mathsf{K}_2$, $\mathsf{K}_3/\mathsf{IV}_3$, and $\mathsf{K}_4/\mathsf{IV}_4$ are used internally within the protocol. In contrast, $\mathsf{PRK}_{out}$ is used to protect application data; hence, its usage is external. Therefore,

$$\mathbf{use} = [internal, internal, internal, external].$$

**Explicit authentication.** In our model for EDHOC, an initiator session considers the peer responder session as explicitly authenticated once the initiator accepts stage 2. More precisely, stages 2, 3, and 4 are explicitly authenticated upon acceptance of stage 2, while stage 1 receives explicit authentication retroactively. For responder sessions, the peer initiator session is explicitly authenticated upon acceptance of stage 3; stages 3 and 4 are explicitly authenticated upon acceptance of stage 1. Stages 2 and 1 receive explicit authentication retroactively. Therefore, for a given role $r$ and stage $s$, we define $\mathbf{eauth}[r, s]$ by the following matrix:

$$\begin{aligned} \forall s \in [0,4]: \quad & \mathbf{eauth}[init, s] = 2 \\ & \mathbf{eauth}[resp, s] = 3. \end{aligned}$$

**Forward secrecy.** Each protocol participant samples ephemeral DH shares for each run of the protocol, and all keys derived throughout the execution of the protocol depend on these DH shares. Therefore, all stages are forward secret, and we have:

$$\mathbf{fs} = [fs, fs, fs, fs].$$

**Session identifiers.** The session identifier of stage $s$ is a tuple $(\text{“}s\text{''}, \mathbf{tx}_s, \mathbf{auth}_s)$, where $\mathbf{tx}_s$ denotes the conversation transcript containing elements that enter the key schedule, $\mathbf{auth}_s$ is a potentially empty list containing the identities of the peers that are explicitly authenticated at stage $s$. Within $\mathbf{auth}_s$, $I$ is a placeholder for the identity of the initiator session, and $R$ is for the responder's identity. For $s \in [1,4]$, the $s$'th session identifier is defined as follows:

$$\begin{aligned} \mathsf{sid}[1] &= (\text{“1''}, \mathsf{M}, \mathsf{S}, G_x, \mathsf{C}_I, ead_1, G_y, \mathsf{C}_R) \\ \mathsf{sid}[2] &= (\text{“2''}, \mathsf{M}, \mathsf{S}, G_x, \mathsf{C}_I, ead_1, G_y, \mathsf{C}_R, \mathsf{kid}_R, \sigma_2, ead_2, R) \\ \mathsf{sid}[3] &= (\text{“3''}, \mathsf{M}, \mathsf{S}, G_x, \mathsf{C}_I, ead_1, G_y, \mathsf{C}_R, \mathsf{kid}_R, \sigma_2, ead_2, \mathsf{kid}_I, \sigma_3, ead_3, R, I) \\ \mathsf{sid}[4] &= (\text{“4''}, \mathsf{M}, \mathsf{S}, G_x, \mathsf{C}_I, ead_1, G_y, \mathsf{C}_R, \mathsf{kid}_R, \sigma_2, ead_2, \mathsf{kid}_I, \sigma_3, ead_3, R, I) \end{aligned}$$

**Contributive Identifiers.** The contributive identifier for a stage $s$ corresponds to the messages that a session $\pi$ must *honestly* receive (untampered) from the peer session to allow testing $\pi$ in the *unauthenticated* stage $s$, even though any other message is not delivered or only partially to either party involved in a run of the protocol. For a session $\pi$, in the role $\mathsf{role} \in \{init, resp\}$, let $\overline{\mathsf{role}}$ denote the opposite role; $\pi.\mathsf{cid}[\overline{\mathsf{role}}, s]$ captures the messages that $\pi$ must have received honestly from its peer as a prerequisite to allow testing $\pi$ in stage $s$, if $s$ is unauthenticated. Specifically, the initiator (resp. responder) sets $\mathsf{cid}[init, 1]$ to $(\text{“1''}, G_x)$ upon sending (resp. receiving) message 1, which captures that an initiator must have contributed a

```
$\textsc{Run}(\pi^i_U, peerpk, sk_U, \{\mathsf{pid}\}_U, m)$
$\mathsf{stage} \leftarrow \pi^i_U.\mathsf{stage}$
$\textbf{if } \pi^i_U.\mathsf{role} = init \textbf{ and } \textbf{ if } \mathsf{stage} = 1:$
    $m' \leftarrow \textsc{RunInit1}(\pi^i_U, peerpk, sk_U, \{\mathsf{pid}\}_U, m)$
$\textbf{if } \pi^i_U.\mathsf{role} = init \textbf{ and } \mathsf{stage} = 2:$
    $m' \leftarrow \textsc{RunInit2}(\pi^i_U, peerpk, sk_U, \{\mathsf{pid}\}_U, m)$
$\textbf{if } \pi^i_U.\mathsf{role} = resp \textbf{ and } \mathsf{stage} = 1:$
    $m' \leftarrow \textsc{RunResp1}(\pi^i_U, peerpk, sk_U, \{\mathsf{pid}\}_U, m)$
$\textbf{if } \pi^i_U.\mathsf{role} = resp \textbf{ and } \mathsf{stage} = 2:$
        $m' \leftarrow \textsc{RunResp2}(\pi^i_U, peerpk, sk_U, \{\mathsf{pid}\}_U, m)$
$\textbf{return } (\pi^i_U, m')$
```

```
$\textsc{RunResp1}(\pi^i_U, peerpk, sk_U, \{\mathsf{pid}\}_U, m)$
$(M, \mathsf{S}, G_x, \mathsf{C}_I, ead_1) \leftarrow m$
$y \xleftarrow{\$} \mathbb{Z}_q$
$G_y \leftarrow y * G_y$
$\mathsf{PRK}_{2e} \leftarrow \mathsf{Extract}("", G_{xy})$
$\mathsf{th}_2 \leftarrow \mathsf{H}(G_y, \mathsf{C}_R, \mathsf{H}((M, \mathsf{S}, G_x, \mathsf{C}_I, ead_1)))$
$\tau_2 \leftarrow \mathsf{Expand}(\mathsf{PRK}_{2e}, (2, \mathsf{kid}_R \| \mathsf{th}_2 \| \mathsf{cred}_R \| ead_2, \mathsf{taglen}_2), \mathsf{taglen}_2)$
$\sigma_2 \leftarrow \mathsf{Sig.Sign}(sk_R, (\mathsf{l}_{\mathsf{sig}}, \mathsf{kid}_R, \mathsf{th}_2 \| \mathsf{cred}_R \| ead_2, \tau_2))$
$\mathsf{ptxt}_2 \leftarrow (\mathsf{kid}_R, \sigma_2, ead_2)$
$\mathsf{K}_2 \leftarrow \mathsf{Expand}(\mathsf{PRK}_{2e}, (0, \mathsf{th}_2, |\mathsf{ptxt}_2|), |\mathsf{ptxt}_2|)$
$\mathsf{ctxt}_2 \leftarrow \mathsf{ptxt}_2 \oplus \mathsf{K}_2$
$\mathsf{msg}_2 \leftarrow (G_y, \mathsf{ctxt}_2, \mathsf{C}_R)$
$\textbf{return } \mathsf{msg}_2$
```

```
$\textsc{RunResp2}(\pi^i_U, peerpk, sk_U, \{\mathsf{pid}\}_U, m)$
$\mathsf{ctxt}_3 \leftarrow \mathsf{msg}_3$
$\mathsf{ad}_3 \leftarrow (\mathsf{l}_{\mathsf{aead}}, "", \mathsf{th}_3)$
$(\mathsf{kid}_I, \sigma_3, ead_3) \leftarrow \mathsf{AEAD.Dec}(\mathsf{K}_3, \mathsf{ctxt}_3, \mathsf{ad}, \mathsf{IV}_3)$
$\mathsf{th}_3 \leftarrow \mathsf{H}(\mathsf{th}_2, \mathsf{ptxt}_2, \mathsf{cred}_R)$
$\mathsf{pid} \leftarrow \textbf{null}$
$\textbf{for } (pk, U) \textbf{ in } credentials[\mathsf{kid}_I]:$
    $\tau_3 \leftarrow \mathsf{Expand}(\mathsf{PRK}_{3e2m}, (6, \mathsf{kid}_I \| \mathsf{th}_3 \| \mathsf{cred}_I \| ead_3, \mathsf{taglen}_3), \mathsf{taglen}_3)$
    $\textbf{if } \mathsf{Sig.Vf}(pk_U, (\mathsf{l}_{\mathsf{sig}}, \mathsf{kid}_I, \mathsf{th}_3 \| \mathsf{cred}_I \| ead_3, \tau_3), \sigma_3) = 1:$
        $\mathsf{pid} \leftarrow U$
        $\textbf{endfor}$
$\textbf{abort } \text{ if pid is } \textbf{null}$
$\mathsf{th}_4 \leftarrow \mathsf{H}(\mathsf{th}_3, \mathsf{ptxt}_3, \mathsf{cred}_I)$
$\mathsf{K}_4 \leftarrow \mathsf{Expand}(\mathsf{PRK}_{2e}, (8, \mathsf{th}_4, \mathsf{klen}_4), \mathsf{klen}_4)$
$\mathsf{IV}_4 \leftarrow \mathsf{Expand}(\mathsf{PRK}_{2e}, (7, \mathsf{th}_4, \mathsf{ivlen}_4), \mathsf{ivlen}_4)$
$\mathsf{PRK}_{out} \leftarrow \mathsf{Expand}(\mathsf{PRK}_{2e}, (9, \mathsf{th}_4, \mathsf{klen}_{out}), \mathsf{klen}_{out})$
$\textbf{return } \perp$
```

```
$\textsc{RunInit1}(\pi^i_U, peerpk, sk_U, \{\mathsf{pid}\}_U, m)$
$x \xleftarrow{\$} \mathbb{Z}_q$
$G_x \leftarrow x * G$
$msg_1 \leftarrow (M, \mathsf{S}, G_x, \mathsf{C}_I, ead_1)$
$\textbf{return } msg_1$
```

```
$\textsc{RunInit2}(\pi^i_U, peerpk, sk_U, \{\mathsf{pid}\}_U, m)$
$(G_y, \mathsf{ctxt}_2, \mathsf{C}_R) \leftarrow m$
$G_{xy} \leftarrow x * G_y$
$\mathsf{PRK}_{2e} \leftarrow \mathsf{Extract}("", G_{xy})$
$\mathsf{th}_2 \leftarrow \mathsf{H}(G_y, \mathsf{C}_R, \mathsf{H}((M, \mathsf{S}, G_x, \mathsf{C}_I, ead_1)))$
$\mathsf{K}_2 \leftarrow \mathsf{Expand}(\mathsf{PRK}_{2e}, (0, \mathsf{th}_2, |\mathsf{ptxt}_2|), |\mathsf{ptxt}_2|)$
$(\mathsf{kid}_R, \sigma_2, ead_2) \leftarrow \mathsf{ctxt}_2 \oplus \mathsf{K}_2$
$\mathsf{pid} \leftarrow \textbf{null}$
$\textbf{for } (pk, U) \textbf{ in } peerpk_{\mathsf{kid}_R}:$
    $\tau_2 \leftarrow \mathsf{Expand}(\mathsf{PRK}_{2e}, (2, \mathsf{kid}_U \| \mathsf{th}_2 \| \mathsf{cred}_U \| ead_2, \mathsf{taglen}_2), \mathsf{taglen}_2)$
    $\textbf{if } \mathsf{Sig.Vf}(pk_U, (\mathsf{l}_{\mathsf{sig}}, \mathsf{kid}_R, \mathsf{th}_2 \| \mathsf{cred}_U \| ead_2, \tau_2), \sigma_2) = 1:$
        $\mathsf{pid} \leftarrow U$
        $\textbf{endfor}$
$\textbf{abort } \text{ if pid is } \textbf{null}$
$\mathsf{th}_3 \leftarrow \mathsf{H}(\mathsf{th}_2, \mathsf{ptxt}_2, \mathsf{cred}_R)$
$\tau_3 \leftarrow \mathsf{Expand}(\mathsf{PRK}_{3e2m}, (6, \mathsf{kid}_I \| \mathsf{th}_3 \| \mathsf{cred}_I \| ead_3, \mathsf{taglen}_3), \mathsf{taglen}_3)$
$\sigma_3 \leftarrow \mathsf{Sig.Sign}(sk_I, (\mathsf{l}_{\mathsf{sig}}, \mathsf{kid}_I, \mathsf{th}_3 \| \mathsf{cred}_I \| ead_3, \tau_3))$
$\mathsf{ptxt}_3 \leftarrow (\mathsf{kid}_I, \sigma_3, ead_3)$
$\mathsf{ad}_3 \leftarrow (\mathsf{l}_{\mathsf{aead}}, "", \mathsf{th}_3)$
$\mathsf{K}_3 \leftarrow \mathsf{Expand}(\mathsf{PRK}_{2e}, (3, \mathsf{th}_3, \mathsf{klen}_3), \mathsf{klen}_3)$
$\mathsf{IV}_3 \leftarrow \mathsf{Expand}(\mathsf{PRK}_{2e}, (4, \mathsf{th}_3, \mathsf{ivlen}_3), \mathsf{ivlen}_3)$
$\mathsf{ctxt}_3 \leftarrow \mathsf{AEAD.Enc}(\mathsf{K}_3, \mathsf{ptxt}_3, \mathsf{ad}, \mathsf{IV}_3)$
$\mathsf{th}_4 \leftarrow \mathsf{H}(\mathsf{th}_3, \mathsf{ptxt}_3, \mathsf{cred}_I)$
$\mathsf{K}_4 \leftarrow \mathsf{Expand}(\mathsf{PRK}_{2e}, (8, \mathsf{th}_4, \mathsf{klen}_4), \mathsf{klen}_4)$
$\mathsf{IV}_4 \leftarrow \mathsf{Expand}(\mathsf{PRK}_{2e}, (7, \mathsf{th}_4, \mathsf{ivlen}_4), \mathsf{ivlen}_4)$
$\mathsf{PRK}_{out} \leftarrow \mathsf{Expand}(\mathsf{PRK}_{2e}, (9, \mathsf{th}_4, \mathsf{klen}_{out}), \mathsf{klen}_{out})$
$\mathsf{msg}_3 \leftarrow \mathsf{ctxt}_3$
$\textbf{return } \mathsf{msg}_3$
```

**Figure 5.1:** Pseudocode description of EDHOC in Sig-Sig mode as an MSKE protocol. The RUN dispatches the message to the relevant session in the MSKE game. The Initator's action are specified in the oracles RUNINIT1 and RUNINIT2. The reponnder's action are specified in the oracles RUNRESP1 and RUNRESP2.

DH share $G_x$ as a prerequisite to allow testing of the responder session in stage 1. At a later point, the initiator (resp. responder) sets $\mathsf{cid}[resp, 1]$ to $(\text{"1"}, G_x, G_y)$ upon receiving (resp. sending) message 2. In this case, we capture the fact that the responder also contributed a $G_y$ before a legitimate test query against the initiator session is allowed. For all other stages $2 \leq s \leq 4$, $\mathsf{cid}[init, s] = \mathsf{cid}[resp, s] = (\text{"s"}, G_x, G_y)$. We summarize the contributive identifiers for the initiator and responder here-below.

$$\begin{aligned} \mathsf{cid}[init, 1] &= (\text{"1"}, G_x) \\ \mathsf{cid}[resp, 1] &= (\text{"1"}, G_x, G_y) \\ \mathsf{cid}[init, s] &= \mathsf{cid}[resp, s] = (\text{"s"}, G_x, G_y), \forall s \in \{2, 3, 4\} \end{aligned}$$

## 5.3 MSKE security of EDHOC SIG-SIG

We now state our MSKE security result for the EDHOC protocol in SIG-SIG mode.

**Theorem 5.1 (MSKE security of EDHOC SIG-SIG)** *Let* EDHOC-Sig-Sig *denote the EDHOC protocol in SIG-SIG mode for authentication as defined in Chapter 3. Moreover, let* $\mathbb{G}$ *be a cyclic group of order* $q$*, and* H *be a hash function,* Sig *be a digital signature scheme,* Extract *be a* PRF*,* Expand *be a variable-length* PRF*,* $n_U$ *be the total number of users and* $n_S$ *be the total number of sessions. Finally, let* $\mathcal{A}$ *be an* MSKE *adversary against* EDHOC-Sig-Sig*. Then there exist adversaries* $\mathcal{B}_4, \mathcal{B}_{I.2}, \mathcal{B}_{I.4}, \mathcal{B}_{II.A2}, \mathcal{B}_{II.B2}, \mathcal{B}_{II.B3}$ *such that:*

$$\begin{aligned} &\mathsf{Adv}_{\mathcal{A}}^{\mathrm{MSKE}}(\text{EDHOC-Sig-Sig}) \leq \frac{{n_S}^2}{q} + \mathsf{Adv}_{\mathcal{B}_4}^{\mathrm{CR}}(\mathsf{H}) \\ &+ 4n_S \left( n_U \cdot \mathsf{Adv}_{\mathcal{B}_{I.2}}^{\mathsf{SUF\text{-}CMA}}(\mathsf{Sig}) + \mathsf{Adv}_{\mathcal{B}_{I.4}}^{\mathrm{S\text{-}UEO}}(\mathsf{Sig}) \right) \\ &+ 4n_S \begin{pmatrix} n_U \cdot \mathsf{Adv}_{\mathcal{B}_{II.A2}}^{\mathsf{SUF\text{-}CMA}}(\mathsf{Sig}) + \mathsf{Adv}_{\mathcal{B}_{II.B2}}^{\mathrm{snPRF\text{-}ODH}}(\mathsf{Extract}) \\ + \mathsf{Adv}_{\mathcal{B}_{II.B3}}^{\mathsf{PRF}}(\mathsf{Expand}) \end{pmatrix} \end{aligned}$$

### 5.3.1 Proof details

**Proof overview.** We proceed with our proof in phases. In the first phase, we ensure that the adversary cannot win the game by causing the predicate Sound to evaluate to false. Assuming soundness is unconditionally guaranteed in the second phase, we split the proof into two disjoint branches, depending on whether the adversary attempts to cause the ExplicitAuth predicate to evaluate to false. In the first branch, we show that the adversary cannot win the game by forcing the ExplicitAuth predicate to evaluate to false. In the second branch, we show that the adversary that respects the freshness conditions defined in Figure 4.2 cannot win the game by guessing the challenge bit with a non-negligible probability. Finally, we collect the bounds to provide an upper bound on the adversary's advantage against EDHOC-Sig-Sig in the MSKE game.

**Proof** Let $\mathcal{A}$ be an MSKE-adversary against EDHOC-Sig-Sig, we bound $\mathcal{A}$'s advantage, denoted by $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{MSKE}}(\text{EDHOC-Sig-Sig})$, with the following sequence of games.

## Phase 1: Ensuring that Sound is true

**Game $\mathbf{G}_0$.** We start with the normal MSKE game, defined in Figure 4.2, played by $\mathcal{A}$. By definition,

$$\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_0}(\text{EDHOC-Sig-Sig}) = \mathsf{Adv}_{\mathcal{A}}^{\mathrm{MSKE}}(\text{EDHOC-Sig-Sig}).$$

**Game $\mathbf{G}_1$.** In this game, we log all Diffie-Hellman shares chosen by honest sessions in a table $T_{dh}$. Additionally, we set the flag $dh_{coll}$ to $\mathsf{true}$ whenever a collision occurs in $T_{dh}$, i.e., when two honest sessions sample the same DH key shares. These changes are not noticeable to the adversary, therefore:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_1}(\text{EDHOC-Sig-Sig}) = \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_0}(\text{EDHOC-Sig-Sig}).$$

**Game $\mathbf{G}_2$.** This game terminates whenever $dh_{coll}$ is set. Before $dh_{coll}$ is set, $\mathbf{G}_2$ is equivalent to $\mathbf{G}_1$. By the *identical-until-bad* lemma of [BR06], the advantage difference of $\mathcal{A}$ can be bounded as follows:

$$|\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_2}(\text{EDHOC-Sig-Sig}) - \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_1}(\text{EDHOC-Sig-Sig})| \leq \Pr[dh_{coll} \leftarrow \mathsf{true}].$$

We use the birthday paradox to bound $\Pr[dh_{coll} \leftarrow \mathsf{true}]$. Let $q = |\mathbb{G}|$ be the order of the prime-order group used in EDHOC-Sig-Sig and assuming that DH shares are chosen uniformly at random, we directly obtain the bound $\Pr[dh_{coll} \leftarrow \mathsf{true}] \leq \frac{{n_S}^2}{q}$, where $n_S$ is the total number of sessions. As a consequence:

$$|\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_2}(\text{EDHOC-Sig-Sig}) - \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_1}(\text{EDHOC-Sig-Sig})| \leq \frac{{n_S}^2}{q}.$$

**Conclusion of phase 1.** At this point, we argue that if $\mathbf{G}_2$ does not terminate, then the adversary $\mathcal{A}$ cannot cause the Sound predicate to become false. Recalling the definition of the predicate Sound in our MSKE model (see Figure 4.2), there are seven events, at least one of which must occur for Sound to be false. In the following, we argue that if $\mathbf{G}_2$ did not terminate, then none of the seven events occurred.

**Proposition 5.2** *At any given stage, no more than two sessions share the same session identifier.*

***Proof.*** We show that there is no “triple-partnering”. Assume that $\exists s, x, y, z :$ $(x, y) \in \mathcal{P}_s, (x, z) \in \mathcal{P}_s, (y, z) \in \mathcal{P}_s$, that is, sessions $x, y, z$ are pair-wise partnered in stage $s$. We have three pairs of partnered sessions, but at most two DH shares[1]. We recall that in $\mathbf{G}_2$, the challenger aborts the game if such a situation occurs, which contradicts the assumption of triple partnering. Therefore, from now on, we assume that at most two sessions are partnered. ■

**Proposition 5.3** *Matching session identifiers for a given stage implies matching stage session keys.*

***Proof.*** We show that matching session identifiers imply that partnered sessions derive the same shared DH secret and transcript hashes, which is sufficient to compute the stage keys deterministically. We recall that the key schedule of EDHOC-Sig-Sig (Figure 3.2) starts by computing the key $\mathsf{PRK}_{2e} = \mathsf{Extract}(\texttt{""}, G_{xy})$, where $G_{xy}$ is the shared Diffie-Hellman secret. Hence, the equality of the session identifiers implies the equality of the derived $\mathsf{PRK}_{2e}$. The key schedule proceeds to derive further stage keys (and potentially associated IVs) using the Expand function keyed with $\mathsf{PRK}_{2e}$. For each key/IV, Expand is evaluated on an input composed of the (partial) transcript hash and a stage-specific label value (see Section 3.4). By the definition of transcript hashes, the equality of session identifiers implies the equality of the transcript hash. Therefore, two partnered sessions at any stage $s$ will always derive the same stage key and IV if the latter is required. ■

**Proposition 5.4** *Matching stage session identifiers implies opposite roles.*

***Proof.*** Assume that not more than two sessions have the same session identifier for a given stage (Proposition 5.2) i.e., $\forall s \in \mathcal{S} : \neg\exists x, y, z : (x, y) \in \mathcal{P}_s \wedge (x, z) \in \mathcal{P}_s \wedge (y, z) \in \mathcal{P}_s$. Each session includes its DH share $k * G$ in the session identifier at a fixed position. Two sessions with the same session identifier and the same role at a given stage imply that the sessions sampled the same DH key shares, which contradicts the uniqueness of the DH key shares guaranteed at this point since $\mathbf{G}_2$ did not terminate. ■

**Proposition 5.5** *Matching session identifiers for a given stage implies an agreed-upon authentication level.*

***Proof.*** This property holds trivially, since EDHOC fixes the authentication level for each stage. ■

**Proposition 5.6** *Matching session identifiers for a given stage implies agreed-upon contributive identifiers.*

[1]Every session identifier includes two DH shares.

***Proof.*** For any stage $s \in [1, 4]$, the session identifier for that stage includes the DH shares of both parties. We recall that the contributive identifiers for EDHOC-Sig-Sig are defined as follows: $\mathsf{cid}[init, 1] = (\text{“}1\text{”}, G_x)$ and for all roles $r$ and stage $s$, $\mathsf{cid}[r, s] = (\text{“}s\text{”}, G_x, G_y)$. Therefore, matching session identifiers means agreement on the DH shares, which in turn means agreement on the contributive identifiers. ■

**Proposition 5.7** *Matching session identifiers in authenticated stages implies that the partner session is intended.*

***Proof.*** Assuming that agreement on the session identifier ($\mathsf{sid}[s]$) for an authenticated stage $s$ implies different roles (see Proposition 5.4), honest initiators and responders write their identity in the $I/R$ placeholder in the session identifier. If these values are both honestly set, agreement on the session identifier implies agreement on the peer's identity and respective roles. ■

**Proposition 5.8** *Session identifiers are different across stages.*

***Proof.*** For any stage $s \in [1, 4]$, the session identifier $\mathsf{sid}[s] = (\text{“}s\text{”}, \ldots)$ is a sequence whose first element is “$s$”. For any $t \neq s$, $\mathsf{sid}[t] = (\text{“}t\text{”}, \ldots) \neq (\text{“}s\text{”}, \ldots)$. Therefore, session identifiers are distinct across stages ■

## Phase 2: Ensuring Explicit Authentication and Key Indistinguishability

We proceed with phase two of our proof, assuming that soundness is unconditionally guaranteed from now on. In this phase, we show that the adversary cannot win by breaking explicit authentication or distinguishing the challenge bit.

Preparing for our analysis of phase II, we introduce the following two games to exclude collisions in the partial transcript hashes. Moreover, from now on, we drop EDHOC-Sig-Sig from advantage expressions for the sake of readability.

**Game $\mathbf{G}_3$.** In this game, we log the hash values computed by honest sessions in a table $T_{hash}$ that provides efficient look-ups. Given an arbitrary value $m$, $T_{hash}$ maps $\mathsf{H}(m)$ to $m$, that is, $T_{dh}[h] \leftarrow m$. Additionally, we set the flag $hash_{coll}$ if an honest session computes a hash $h$ on a value $m$ such that $h \in T_{hash}$ and $T_{hash}[h] \neq m$. These changes are unobservable to the adversary, therefore

$$\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_3} = \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_2}.$$

**Game $\mathbf{G}_4$.** The $\mathbf{G}_4$ terminates whenever $hash_{coll}$ is set. Using the *identical-until-bad* lemma, we have that

$$|\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_4} - \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_3}| \leq \Pr[hash_{coll} \leftarrow \mathsf{true}].$$

We bound $\Pr[hash_{coll} \leftarrow \mathsf{true}]$ using a reduction $\mathcal{B}_4$ to the collision resistance of $\mathsf{H}$. $\mathcal{B}_4$ honestly, simulates $\mathbf{G}_4$ towards $\mathcal{A}$; whenever $hash_{coll}$ is set, $\mathcal{B}_4$ wins the collision resistance game by outputting the strings $m \neq m'$ that caused the collisions. Therefore, $\Pr[hash_{coll} \leftarrow \mathsf{true}] \leq \mathsf{Adv}_{\mathcal{B}_4}^{\mathrm{CR}}(\mathsf{H})$ and as consequence:

$$|\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_4} - \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_3}| \leq \mathsf{Adv}_{\mathcal{B}_4}^{\mathrm{CR}}(\mathsf{H}).$$

**Note.** At this point, we split the proof into two branches **2.I** and **2.II**, each starting from $\mathbf{G}_4$ and proceeding with the games $\mathbf{G}_I$ and, $\mathbf{G}_{II}$ respectively. In the first branch, the adversary attempts to break explicit authentication for at least one session; in the second branch, explicit authentication is unconditionally guaranteed, and the adversary attempts to guess the challenge bit. Since the two cases are disjoint, we have the following bound:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_4} \leq \max(\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_\mathrm{I}}, \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_\mathrm{II}}) \leq \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_\mathrm{I}} + \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_\mathrm{II}}.$$

We start with branch **Branch 2.I**.

**Branch 2.I: The adversary Cannot Break Explicit Authentication.** In this phase, we use a hybrid argument to analyze explicit authentication. Namely, we will zoom in on a single session for which $\mathcal{A}$ attempts to break explicit authentication. We will use the term *targeted* session to refer to the session for which the adversary attempts to break explicit authentication.

**Game $\mathbf{G}_I$:** Continuing from $\mathbf{G}_4$, in this game, we guess a session-stage pair $(\pi_U^i, s)$ such that ExplicitAuth evaluates to false since the adversary broke explicit authentication of $\pi_U^i$ in stage $s$. This restriction to a single targeted session and stage reduces the advantage by a factor of $n_S \times \mathbf{S}$, where $n_S$ is the total number of sessions and $\mathbf{S}$ is the number of stages. We therefore get the following:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_4} \leq 4n_S \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_\mathrm{I}}.$$

From now on, $\pi_U^i$ refers to the targeted session.

**Game $\mathbf{G}_{I.1}$.** In this game, we log all messages signed by honest users in a table $T_{sig}$ with efficient lookups, along with the corresponding public key and the signature produced. More precisely, for a (honest) user $U$ that owns the long-term key pair $(sk_U, pk_U)$, let $\sigma = \mathsf{Sign}(sk_U, m)$ be the signature computed on a message $m$, then $T_{sig}$ is a list of tuples $(m, \sigma, pk_U, U)$. In concrete terms, an initiator session with identity $I$ will sign a message of the form $m_3 = (\mathsf{l}_{\mathsf{sig}}, \mathsf{kid}_I, \mathsf{th}_3, \mathsf{cred}_I, ead_3, \tau_3)$. Whereas, the responder $R$ will sign a message of the form $m_2 = (\mathsf{l}_{\mathsf{sig}}, \mathsf{kid}_R, \mathsf{th}_2, \mathsf{cred}_R, ead_2, \tau_2)$. Due to credential identifiers potentially referencing multiple credentials, protocol participants may have to verify the received signatures against multiple public keys. Upon receiving the protocol message $\mathsf{msg}_2$ that includes the credential identifier $\mathsf{kid}_U$, initiator sessions will attempt to validate the received signature $\sigma_2$ against each public key $pk_U$ referenced by $\mathsf{kid}_U$, adapting the a priori signed message to the messages $m_U = (\mathsf{l}_{\mathsf{sig}}, \mathsf{kid}_U, \mathsf{th}_2, \mathsf{cred}_U, ead_2, \tau_2)$. These validation attempts are performed until for one public key $\mathsf{Vf}(pk_U, \sigma_2, m_U) = 1$; otherwise, the protocol is aborted. Similarly, responder sessions verify the signature $\sigma_3$ received within $\mathsf{msg}_3$ and all possible messages $m_U = (\mathsf{l}_{\mathsf{sig}}, \mathsf{kid}_U, \mathsf{th}_{3U}, \mathsf{cred}_U, ead_3, \tau_3)$ against each public key $pk_U$. Where $\mathsf{th}_{3U}$ must also be re-computed for each $m_U$ and takes the value(s) $\mathsf{H}(\mathsf{th}_2, \mathsf{ptxt}_2, pk_U, U)$.

In addition to logging messages, we set the flag $sig_{forged}$ if the targeted session receives and validates a message signature pair $(m, \sigma)$ under the public key $pk_V$ of an honest[2] user $V$ such that $(m, \sigma, pk_V, V) \notin T_{sig}$. These changes are only administrative and are not observable by the adversary. Therefore:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_{I.1}} = \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_{I}}.$$

**Game $\mathbf{G}_{I.2}$.** $\mathbf{G}_{I.2}$ terminates whenever $sig_{forged}$ is set. We analyze the probability of the event $sig_{forged} \leftarrow \mathsf{true}$. We bound $\Pr[sig_{forged} \leftarrow \mathsf{true}]$ by a reduction $\mathcal{B}_{I.2}$ to the SUF-CMA security of Sig. Namely, $\mathcal{B}_{I.2}$ first guesses the identity $(V)$ of the peer session which reduces the advantage by a factor $n_U$ and associates $pk_V$ with the public key $pk^*$ from the SUF-CMA challenge i.e. $pk^* = pk_V$. The reduction answers all game queries and calls its signing oracle whenever a query needs $V$ to produce a signature. Upon $sig_{forged}$ being set, $\mathcal{B}_{I.2}$ outputs the message signature pair $(m, \sigma)$ that caused $sig_{forged}$ to be set and terminates $\mathbf{G}_{I.2}$.

*Simulation soundness.* Besides RevLongTermKey queries, $\mathcal{B}_{I.2}$ can consistently answer all queries. Next, we argue that RevLongTermKey queries are of no concern. Indeed, after $sig_{forged}$ is set, we do not need to answer this query. Before the flag is set, such a query does not help the adversary either. The ExplicitAuth predicate requires the value of $revltk_V$ designates a time after acceptance of the stage $s'$, where the stage $s$ receives explicit authentication, perhaps retroactively. Since the targeted session must have accepted stage $s'$ which requires receiving and accepting a message-signature pair under $pk_V$; therefore, a RevLongTermKey$(V)$ query before $sig_{forged}$ is unhelpful for the adversary in its quest to break explicit authentication. Hence, the reduction need not answer RevLongTermKey queries.

[2]More precisely, we only expect that $V$ is honest at the time the message-signature pair is received.

*Validity of the Forgery.* By definition of $T_{sig}$, the flag $sig_{forged}$ is set only when the targeted session receives and accepts a message signature pair $(m, \sigma)$ under a $pk_V$ such that $(m, \sigma, pk_V, V) \notin T_{sig}$. This implies that $(m, \sigma)$ is a new message-signature pair that $V$ did not previously produce. Therefore, $\mathcal{B}_{I.2}$ produces a legitimate SUF-CMA forgery, and we have the following:

$$\Pr\left[sig_{forged} \to 1\right] = \mathsf{Adv}^{\mathsf{SUF\text{-}CMA}}_{\mathcal{B}_{I.2}}(\mathsf{Sig}).$$

And as a consequence:

$$|\mathsf{Adv}^{\mathbf{G}_{I.2}}_{\mathcal{A}} - \mathsf{Adv}^{\mathbf{G}_{I.1}}_{\mathcal{A}}| \leq n_U \cdot \mathsf{Adv}^{\mathsf{SUF\text{-}CMA}}_{\mathcal{B}_{I.2}}(\mathsf{Sig}).$$

**Game $\mathbf{G}_{I.3}$.** In this game, we set the flag $sig_{ambigous}$ if there exists an honest session $\pi$ that receives and accepts a message-signature pair $(m, \sigma)$ under a public key $pk_{U'}$ and, there exists a value $\mathsf{kid}$ such that: $(pk_U, U) \in peerpk_{\mathsf{kid}}$ and $(pk_{U'}, U') \in peerpk_{\mathsf{kid}}$. In other words, $\mathsf{kid}$ identifies both $pk_U$ and $pk_{U'}$; and for some $m'$ it holds that $(m', \sigma, pk_U, U) \in T_{sig}$. We view $pk_{U'}$ as a key chosen by the adversary $\mathcal{A}$ and registered using the query $\textsc{NewUser}(sk_{U'}, pk_{U'}, \mathsf{kid})$. From the standpoint of $\pi$, there is an ambiguity about the identity of the peer that (presumably) authenticated themselves via the received message signature pair $(m, \sigma)$. These changes are unobservable to the adversary, therefore:

$$\mathsf{Adv}^{\mathbf{G}_{I.3}}_{\mathcal{A}} = \mathsf{Adv}^{\mathbf{G}_{I.4}}_{\mathcal{A}}.$$

**Game $\mathbf{G}_{I.4}$.** In this game, we terminate whenever $sig_{ambigous}$ is set. We first observe that for $m$ and, $m'$ as described in the previous game, it is always the case that the following hold: $m' \neq m$. This is because each session signs a message that includes the user's credentials, i.e., each user $U'$ signs a message of the form $(\mathsf{l_{sig}}, \mathsf{kid}_{U'}, \mathsf{th} \| \mathsf{cred}_{U'} \| ead, \tau)$. Furthermore, the credentials are unique to each identity, and the CBOR encoding is unambiguous. Consequently, we can restrict our analysis to $pk_U \neq pk_{U'}$. If $pk_U = pk_{U'}$, the attacker knows the secret key, or they have to devise a forgery since $m$ is never signed by $U$. By definition of $sig_{ambigous}$, we can straightforwardly relate $\Pr\left[sig_{ambigous} \leftarrow \mathsf{true}\right]$ to the advantage of an S-UEO adversary $\mathcal{B}_{I.4}$. More precisely, $\mathcal{B}_{I.4}$ associates $pk_U$ with the public key from the $pk^*$ received from the challenger S-UEO, i.e. $pk_U = pk^*$. The reduction uses the signing oracle of its challenger whenever a query needs $U$ to sign a message; else, it responds to the other oracle queries in the usual manner. It terminates the game when $sig_{ambigous}$ is set.

*Simulation soundness*: We only need to consider the $\textsc{RevLongTermKey}(U)$ queries, as the reduction can answer all other queries consistently. On the one hand, we do not need to consider what happens after $sig_{ambigous}$ is set. The simulation terminates and need not answer $\textsc{RevLongTermKey}(U)$

queries. On the other hand, the predicate ExplicitAuth requires $pk_U$ is not compromised before the session $\pi$ accepts stage $s$. Therefore, for our reduction, the RevLongTermKey($U$) queries are not a concern, and the simulation is sound.

*Validity of the attack.* If the flag $sig_{ambigous}$ is set, $\pi$ accepted and verified a message signature pair $(m, \sigma)$ under a public key $pk_{U'}$ and $\exists t \in T_{sig} : t = (m', \sigma, pk_U, U)$. As observed above $m \neq m'$; therefore, no honest session sought to sign $m$. As a consequence, the tuple $(m, m', \sigma, pk, pk')$ is a valid S-UEO forgery. Therefore,

$$\Pr\left[sig_{ambigous} \leftarrow \mathsf{true}\right] \leq \mathsf{Adv}^{\text{S-UEO}}_{\mathcal{B}_{I.4}}(\mathsf{Sig}).$$

**Remark 5.9** *Again, we note that $\mathcal{A}$ does not see the messages signed by honest sessions as they contain a (random) MAC tag unknown to the adversary; however, seeing messages would only increase the advantage. Therefore, the upper bound above is justified.*

Finally, we get:

$$|\mathsf{Adv}^{\mathbf{G}_{I.4}}_{\mathcal{A}} - \mathsf{Adv}^{\mathbf{G}_{I.3}}_{\mathcal{A}}| \leq n_S \cdot \mathsf{Adv}^{\text{S-UEO}}_{\mathcal{B}_7}(\mathsf{Sig}).$$

**Note 1.** The signature schemes in EDHOC are Ed25519 and ECDSA. The former is known to be S-UEO-secure [BCJZ20] while the latter is only secure under certain conditions. However, in EDHOC, the signing algorithm unambiguously places the public key of the message together with the actual message via the credential. Therefore, we could view the signature scheme (Sig) in EDHOC as another scheme $\widehat{\mathsf{Sig}}$ that takes a message and signs the message along with the corresponding verification key. That is, for a key pair $(sk, pk)$, the signing algorithm is modified and behaves as follows: $\widehat{\mathsf{Sig}}.\mathsf{Sign}(sk, m) = \mathsf{Sig}.sign(sk, (m, pk))$. Pornin and Stern [PS05] showed that unambiguous inclusion of the verification key is enough to thwart S-UEO attacks, provided there are no weak keys. This property holds for ECDSA assuming that the concrete implementation of ECDSA performs all the necessary checks to prevent "weak keys." Finally, we note that *Destructive Ownership* would be sufficient.

**Note 2.** The MAc-then-SIGn version of the SIGMA protocol [Kra03], that informed the design of EDHOC and that we denote by $\mathsf{SIGMA}_{\frac{\sigma}{\tau}}$[3], can be vulnerable to attacks similar to those targeting the ambiguity of the message signer. In Appendix A.1, we show that standard unforgeability alone is not sufficient; one requires that there are no *weak keys*. Fortunately, the signature schemes in EDHOC provide S-UEO security.

[3] $\mathsf{SIGMA}_{\frac{\sigma}{\tau}}$ stands for "MAC under the Signature".

**Conclusion of phase 2.I** At this point, we argue that if $\mathbf{G}_{I.4}$ does not abort, then the adversary cannot win by causing the predicate ExplicitAuth to evaluate to false.

We recall that the predicate ExplicitAuth is defined as follows:

$$\forall(\pi_U^i, s) : \left\{ \begin{array}{l} \left\{ \begin{array}{ll} \pi_U^i.\mathsf{accepted}[s] & \wedge \\ \mathbf{eauth}[\overline{\pi_U^i.\mathsf{role}}, s] = s' < \infty & \wedge \\ \pi_U^i.\mathsf{accepted}[s'] < \mathit{revltk}_{\pi_U^i.\mathsf{pid}} & \end{array} \right\} \implies \\ \\ \exists \pi_V^j : \left\{ \begin{array}{ll} \pi_U^i.\mathsf{sid}[s'] = \pi_V^j.\mathsf{sid}[s'] & \wedge \\ \pi_V^j.\mathsf{accepted}[s] < \mathit{revltk}_{\pi_U^i.\mathsf{id}} & \implies \pi_U^i.\mathsf{sid}[s] = \pi_V^j.\mathsf{sid}[s] \end{array} \right\} \end{array} \right\} .$$

Breaking explicit authentication is equivalent to a state of the game where the following holds:

$$\exists(\pi_U^i, s) : \left\{ \begin{array}{l} \left\{ \begin{array}{ll} \pi_U^i.\mathsf{accepted}[s] & \wedge \\ \mathbf{eauth}[\overline{\pi_U^i.\mathsf{role}}, s] = s' < \infty & \wedge \\ \pi_U^i.\mathsf{accepted}[s'] < \mathit{revltk}_{\pi_U^i.\mathsf{pid}} & \end{array} \right\} \bigwedge \\ \\ \forall \pi_V^j : \left\{ \begin{array}{ll} \pi_U^i.\mathsf{sid}[s'] \neq \pi_V^j.\mathsf{sid}[s'] & \vee \\ \pi_V^j.\mathsf{accepted}[s] < \mathit{revltk}_{\pi_U^i.\mathsf{id}} & \wedge \;\; \pi_U^i.\mathsf{sid}[s] \neq \pi_V^j.\mathsf{sid}[s] \end{array} \right\} \end{array} \right\} .$$

In other words, the following holds:

1. (1) $\pi_U^i$ accepted stage $s$ (resp. $s'$) at time $t$ (resp. $t'$). The session accepts with a peer identifier $V$ (one must be set).
2. $V$'s long-term secret was not compromised at time $t'$.
3. (I.a) Either, no (honest) session $\pi_V^j$ is partnered with $\pi_U^i$ in stage $s'$.
4. (I.b) *Or*, There exists an honest session $\pi_V^j$ that is partnered with $\pi_U^i$ in stage $s'$; however, the two sessions are not partnered in stage $s$.

For an initiator session, stages 2, 3, and 4 are explicitly authenticated once stage 2 is accepted; stage 1 receives authentication retroactively. For a responder session, stages 3 and 4 are explicitly authenticated once stage 3 is accepted; previous stages receive authentication retroactively. Regardless of the role, each session must have received a valid signature $\sigma$ on a MAC tag before accepting the relevant $s$'th stage. Concretely, an initiator session with identity $I$, must have received from its responder peer with identity $R$ a valid signature $\sigma_2$ within the message $\mathsf{msg}_2$; where $\sigma_2$ is computed over a message of the form $m_2 = (\mathsf{l}_{\mathsf{sig}}, \mathsf{kid}_R, \mathsf{th}_2, \mathsf{cred}_R, ead_2, \tau_2)$. The responder session must have received a signature $\sigma_3$ over the message $m_3 = (\mathsf{l}_{\mathsf{sig}}, \mathsf{kid}_I, \mathsf{th}_3, \mathsf{cred}_I, ead_3, \tau_3)$ within the message $\mathsf{msg}_3$. The attacker breaks explicit authentication if either case (I.a) or (I.b) occurs. We address the possibility that either event occurs.

**Case (I.a).** The targeted session, $\pi_U^i$, accepted a message signature pair $(m, \sigma)$ under the public key of $V$ i.e. $\mathsf{Vf}(pk_V, m, \sigma) = 1$, but no session $\pi_V^j$ is partnered with $\pi_U^i$ in stage 2 (resp. stage 3) if $\pi_U^i$ is the initiator (resp. responder). We consider two cases that we call (i) and (ii), based on whether the message and signature received by $\pi_U^i$ verifies the following: $(m, \sigma, pk_V, V) \notin T_{sig}$. By the definition of case (i), no honest session produced the pair of message signatures $(m, \sigma)$. Therefore, the adversary must have forged a signature. At this point, if $\mathbf{G}_{I.2}$ did not abort, then the adversary could not have forged a signature. If case (ii) occurs, an honest session $\pi_V^j$ produced the message signature pair received and accepted by $\pi_U^i$. In particular, if $\pi_U^i$ is in the initiator role, the message $(\mathsf{l}_{\mathsf{sig}}, \mathsf{kid}_R, \mathsf{th}_2, \mathsf{cred}_R, ead_2, \tau_2)$ was signed (resp. verified) by $\pi_V^j$ (resp. $\pi_U^i$). Hence, $\pi_U^i$ and $\pi_V^j$ agree on the values of $\sigma_2$, $\mathsf{kid}_R$, $ead_2$ in $\mathsf{sid}[2]$. Additionally, they also agree on the values of $\mathsf{th}_2 = \mathsf{H}(G_y, \mathsf{C}_R, \mathsf{H}(\mathsf{msg}_1))$. Thanks to $\mathbf{G}_4$, partial collisions in transcript hashes are excluded. Therefore, $\pi_U^i$ must also agree on the values of $G_y$, $\mathsf{C}_R$ and  and therefore agree on their respective stage-2 session identifiers and are partnered in stage 2, contradicting the assumption that $\pi_U^i$ does not have a partner session in stage 2. Analogously, if $\pi_U^i$ is a responder session ($\pi_V^j$ is an initiator), the message signed is of the form $m_3 = (\mathsf{l}_{\mathsf{sig}}, \mathsf{kid}_I, \mathsf{th}_3, \mathsf{cred}_I, ead_3, \tau_3)$. Therefore, there is agreement on the values of $\sigma_3$, $\mathsf{kid}_I$ and $ead_3$. Furthermore, agreement on $\mathsf{th}_3 = \mathsf{H}(\mathsf{th}_2, \mathsf{ptxt}_2, \mathsf{cred}_R)$ implies agreement on the remaining values of the stage 3 session identifiers, thanks to $\mathbf{G}_4$.

Finally, suppose that the targeted session $\pi_U^i$ is in the responder role. The attacker can cause case (ii) to occur by mounting an attack against the intended initiator session $\pi_V^j$ such that $\pi_V^j$ would accept with a malicious peer identity $U'$ while not modifying the conversation transcript. The subtlety of this attack is that although $\pi_V^j$ has been "tricked" into accepting with an unintended peer, the adversary does not, in fact, break explicit authentication for $\pi_V^j$; the adversary broke explicit authentication for the responder session $\pi_U^i$. At the end of the protocol run, $\pi_U^i$ ends up without a partner in stage 2 and above; hence $\pi_U^i$ is indeed the targeted session. We expand a bit more on the details of this attack that exploit ambiguity about the identity of the responder $\pi_U^i$. Upon receiving $\mathsf{msg}_2$ from $\pi_U^i$, $\mathcal{A}$ registers a new key pair by calling $\textsc{NewUser}(sk'_U, pk_{U'}, \mathsf{kid}_U)$. The malicious key pair is selected such that $\pi_V^j$ would accept $\sigma_2$ under $pk_{U'}$ when delivered via the relevant $\textsc{Send}$ query. Careful observation of the protocol specification reveals that such an attack would not disturb the protocol run. However, the result is an identity mis-binding attack.

Thanks to $\mathbf{G}_{I.4}$, ambiguity about the responder of the initiator is excluded. Furthermore, if the initiator session $\pi_V^j$ accepts another peer identity $U'$, the value of $\mathsf{th}_3$ computed by $\pi_U^i$ (resp. $\pi_V^j$) are $\mathsf{H}(\mathsf{th}_2, \mathsf{ptxt}_2, \mathsf{cred})$ (resp. $\mathsf{H}(\mathsf{th}_2, \mathsf{ptxt}_2, \mathsf{cred}_{U'})$). These are different values, and since honest sessions only sign transcript hashes corresponding to their session identifiers, the adversary must come up with a new forgery for $\pi_U^i$ to later accept $\mathsf{msg}_3$.

We have shown that case (I.a) does not occur. Next, we analyze the case (I.b).

**Case (I.b)** Let $s'$ be the stage a which $\pi_U^i$ receives explicit authentication. Recall that $s' = 2$ if $\pi_U^i$ is in the initiator role and $s' = 3$ if $\pi_U^i$ is in the responder role. For a given stage $s \in [1,4]$, we use $\underline{\mathsf{sid}}[s]$ to denote the subsequence of $\mathsf{sid}[s]$ that does not contain the stage label. We proceed with this analysis stage by stage, assuming that $\pi_U^i.\mathsf{sid}[s'] = \pi_V^j.\mathsf{sid}[s']$.

- Stage 1. This stage receives retroactively explicit authentication upon acceptance of stage 2 for initiator sessions and upon acceptance of stage 3 for responder sessions. Assuming that $\pi_U^i.\mathsf{sid}[s'] = \pi_V^j.\mathsf{sid}[s']$, we also know that $\underline{\mathsf{sid}}[1] \prec \underline{\mathsf{sid}}[s]$ for $s \in \{2,3\}$. Therefore, case (I.b) cannot occur for $s = 1$.
- Stage 2. For initiation sessions, case (I.b) is trivially impossible since $s = s' = 2$. For responder session, $\underline{\mathsf{sid}}[2] \prec \underline{\mathsf{sid}}[3]$ and thus case (I.b) cannot occur.
- Stage 3. In case $\pi_U^i$ is in the responder role, then case (I.b) is trivially excluded since $s = s' = 3$. For an initiator session, the only possible divergences in $\pi_U^i.\mathsf{sid}[3]$ and $\pi_V^j.\mathsf{sid}[3]$ are (i) different values in the field corresponding to $\mathsf{msg}_3$ or (ii) different values in the initiator placeholder position ($I$). Case (i) comprises modifications that would require the adversary to forge a signature, since honest sessions only sign messages in transcript hashes that correspond to their session identifiers, and the predicate ExplicitAuth requires that $\pi_U^i$'s long-term secret, $sk_U$, is not comprised before $\pi_V^j$ accepts stage 3. Therefore, case (i) is prevented thanks to $\mathbf{G}_{I.2}$ where forgeries are excluded. Case (ii) requires that the attacker can create ambiguity about the initiator's identity. Namely, the attacker would have to mount an attack such

that $\pi_V^j$ accepts the peer identity $U'$ after receiving $\mathsf{msg}_3$. Thanks to $\mathbf{G}_{I.4}$, this cannot occur.

- Stage 4. We observe that $\underline{\mathsf{sid}}[4] = \underline{\mathsf{sid}}[3]$. Therefore, the analysis of stag 4 is identical to the analysis of stage 3.

Since adversaries $\mathcal{A}$ cannot break explicit authentication, they can no longer win in this branch of the proof. Thus,

$$\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_{\mathrm{I.4}}} \leq 0.$$

**Conclusion of Branch II.1** We have shown that the adversary cannot break explicit authentication. We now analyze the key secrecy properties of stage keys in EDHOC, assuming that the predicate ExplicitAuth is always $\mathsf{true}$. We do so by showing that the challenge bit is random and independent of the adversary's guess.

### Branch 2.II: Ensuring that the challenge bit is random and independent of the adversary's guess

**Game $\mathbf{G}_{II}$.** Continuing from $\mathbf{G}_4$, in this game, we restrict the adversary $\mathcal{A}$ by allowing a single Test query. From this point on, we assume that the tested session is known in the subsequent games, and we will talk of *the* tested session, $\pi_U^i$. We follow the approach of Dowling *et al.* [DFGS21] who presented a careful hybrid argument for their analysis of TLS 1.3 and argued that this restriction reduces the advantage of $\mathcal{A}$ by a factor at most $n_S \times \mathbf{S}$. Here, $n_S$ is the number of sessions and $\mathbf{S} = 4$ is the number of stages. Therefore, we get the following bound:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_4} \leq 4n_S \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_{\mathrm{II}}}.$$

**Note.** We proceed with our analysis of phase II by considering two disjoint cases. Namely,

- Case **A**: In this case, the tested session does not have a (honest) contributive partner in the first stage, i.e.,

$$\forall \pi \neq \pi_U^i : \pi_U^i.\mathsf{cid}\left[\overline{\pi_U^i.\mathsf{role}}, 1\right] \neq \pi.\mathsf{cid}\left[\overline{\pi_U^i.\mathsf{role}}, 1\right].$$

- Case **B**: The tested session has a (honest) contributive partner in the first stage, that is,

$$\exists \pi \neq \pi_U^i : \pi_U^i.\mathsf{cid}\left[\overline{\pi_U^i.\mathsf{role}}, 1\right] = \pi.\mathsf{cid}\left[\overline{\pi_U^i.\mathsf{role}}, 1\right].$$

Since the two cases above are disjoint, we can bound $\mathcal{A}$'s advantage as follows:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_{\text{II}}} \leq \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_{\text{II}}\text{ case }\mathbf{A}} + \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_{\text{II}}\text{ case }\mathbf{B}}.$$

**Case A: The tested session has no contributive partner.**

As a first observation, the adversary cannot test for unauthenticated stages, such a test query is considered non-fresh in the model (see Figure 4.2). In particular, $\mathcal{A}$ may not test stage 1 nor stage 2 in case of a responder session. Test queries are only allowed from stage 2 onward for an initiator session and from stage 3 onwards for a responder. Having established the appropriate restrictions on Test queries, we now analyze the conditions under which a session accepts a stage that can be legally tested. For an initiator session, acceptance of stage 2 is predicated on the reception of a valid tuple $(\sigma_2, \mathsf{kid}_R, ead_2)$ containing of a signature and a key identifier. Analogously, a responder session accepts stage 3 only if it received a valid triple $(\sigma_3, \mathsf{kid}_I, ead_3)$. Finally, we also observe that a Test query is allowed only *before* the long-term key of $\pi_U^i$'s peer is compromised, that is, Test must be issued *before* RevLongTermKey$(\pi_U^i.\mathsf{pid})$. Based on the three observations previously made, one sees that a prerequisite for $\mathcal{A}$ to have a chance of winning the game is to be able to send valid messages and signatures to the tested session on behalf of honest users. In the next game hops, we analyze $\mathcal{A}$'s likelihood of causing such an event.

**Game $\mathbf{G}_{II.A1}$.** In this game, we set a flag $sig_{forged}$ whenever the tested session $\pi_U^i$ in the role of initiator (resp. Responder) receives a tuple $(\mathsf{kid}_R, \sigma_2, ead_2)$ (resp. $(\mathsf{kid}_I, \sigma_3, ead_3)$) such that the signature verifies under an honest public key $pk_V \in peerpk_{\mathsf{kid}_R}$ (resp. $peerpk_{\mathsf{kid}_I}$). These changes are only administrative and unobservable to $\mathcal{A}$, therefore:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_{\text{II.A1}}} = \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_4}.$$

**Game $\mathbf{G}_{II.A2}$.** The game $\mathbf{G}_{II.A2}$ terminates whenever $sig_{forged}$ is set. By the identical-until-bad lemma, we have that

$$|\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_{\text{II.A2}}} - \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_{\text{II.A1}}}| \leq \Pr[sig_{forged} \leftarrow \mathsf{true}].$$

We bound $\Pr[sig_{forged} \leftarrow \mathsf{true}]$ by a reduction $\mathcal{B}_{II.A2}$, to the EUF-CMA security of the signature scheme. $\mathcal{B}_{II.A2}$, an EUF-CMA adversary, emulates $\mathbf{G}_{II.A2}$ towards $\mathcal{A}$. To this end, $\mathcal{B}_{II.A2}$ first guesses the identity $V$ of $\pi_U^i$'s peer, and associates the challenge public key $pk^*$ to $V$'s long-term verification key, i.e. $pk_V = pk^*$. As a consequence, $\mathcal{A}'$ s advantage is reduced by a factor $n_U$ where $n_U$ is the total number of users. For each Send query that requires $V$ to produce a signature, $\mathcal{B}_{II.A2}$ queries its signing oracle with the message to be signed. Otherwise, $\mathcal{B}_{II.A2}$ answers the remaining queries from $\mathbf{G}_{II.A2}$ as appropriate. Finally, if $sig_{forged}$ is set, $\mathcal{B}_{II.A2}$ outputs the relevant message-signature pair $(m, \sigma)$ as its forgery towards its EUF-CMA challenger. Here, $\sigma$ is the signature value received by the tested session and $m$ is the message for which the tested session verified the signature.

*Simulation soundness.* We argue that $\mathcal{B}II.A2$'s simulation of $\mathbf{G}_{II.A2}$ is sound. First, we observe that $\mathcal{B}_{II.A2}$ can perfectly answer all queries in $\mathbf{G}_{II.A2}$ but $\textsc{RevLongTermKey}(V)$ given that the secret key corresponding to $pk_V = pk^*$ is unknown. However, $\mathcal{B}_{II.A2}$ does not need to be able to answer such queries. Namely, if such a query is issued *before* acceptance of the tested stage, the test query is now non-fresh, and the attacker loses the game. On the other hand, $\mathcal{B}_{II.A2}$ cannot be bothered by $\textsc{RevLongTermKey}(V)$ queries issued *after* $sig_{forged}$ is set. By then, $\mathcal{B}_{II.A2}$ has a valid forgery for the EUF-CMA game and can terminate the game. This shows that until $sig_{forged}$ is set, the $\mathbf{G}_{II.A2}$ and $\mathbf{G}_{II.A1}$ are equivalent, and the simulation is sound.

*Validity of the forgery.* Having shown simulation soundness, it remains to show that $(m, \sigma)$ is a valid forgery, that is, when $\mathcal{B}_{II.A2}$ outputs $(m, \sigma)$, the EUF-CMA challenger also outputs 1. In EDHOC, signatures are computed on (amongst other things) the MAC tag ($\tau$) and the transcript hashes ($\mathsf{th}$). More precisely, the messages to be signed is $m = (\mathsf{l_{sig}}, \mathsf{kid}_U, \mathsf{th}, \mathsf{cred}_U, ead, \tau)$. $\mathsf{cred}_U$ is the credential of $U$ that contains $pk_U$ and $U$'s unique identity. Recall that for signature verification, when an initiator (resp. responder) session receives message 2 (resp. message 3), the session may verify the signature against multiple public keys if the received $\mathsf{kid}_U$ refers to multiple credentials. In this case, for each $\mathsf{cred}_X$ associated with $\mathsf{kid}_U$, the message(s) to be verified is $m_X = (l_{sig}, \mathsf{kid}_U, th, \mathsf{cred}_X, ead, \tau)$ until one verification is successful. Due to the game $\mathbf{G}_4$, collisions in the transcript hashes are excluded if $\mathbf{G}_4$ did not terminate. This implies that without a contributive partner, no honest session signed the message that the tested session received and accepted after a successful validation of the signature. As a result, given the challenge (EUF-CMA) public key $pk^*$, $\mathcal{B}_{II.A2}$ can verify that the pair $(m, \sigma)$ is a valid forgery; allowing $\mathcal{B}_{II.A2}$ to terminate the game and present $(m, \sigma)$ to the challenge EUF-CMA. Therefore, we have:

$$\Pr[sig_{forged} \leftarrow \mathsf{true}] \leq n_U \cdot \mathsf{Adv}^{\text{EUF-CMA}}_{\mathcal{B}_{II.A2}}(\mathsf{Sig}).$$

It follows that:

$$|\mathsf{Adv}^{\mathbf{G}_{\text{II.A2}}}_{\mathcal{A}} - \mathsf{Adv}^{\mathbf{G}_{\text{II.A1}}}_{\mathcal{A}}| \leq n_U \cdot \mathsf{Adv}^{\text{EUF-CMA}}_{\mathcal{B}_{II.A2}}(\mathsf{Sig}).$$

At this point, we remark that if $sig_{forged}$ is never set, then the tested session without a contributive partner never accepts either stage 2 or stage 3. Consequently, an attacker cannot make a valid $\textsc{Test}$ query, and their guess bit $b'$ is truly independent of the challenge bit $b$.

**Case B: The tested session has a contributive partner.**

**Game $\mathbf{G}_{II.B1}$** In this game, we guess the session $\pi_V^j$ that is contributive partner of the tested session $\pi_U^i$. This step reduces the advantage of $\mathcal{A}$ by a factor $n_S$ and we get:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_4}(\text{EDHOC-Sig-Sig}) \leq n_S \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_{\text{II.B1}}}(\text{EDHOC-Sig-Sig}).$$

From this point on, we consider the games to have a specified tested session and its partner at the outset.

**Game $\mathbf{G}_{II.B2}$.** In this game, we replace $\mathsf{PRK}_{2e}$ computed by the tested session with a uniform random value $\widetilde{\mathsf{PRK}_{2e}} \xleftarrow{\$} \mathcal{K}_{\mathsf{PRK}_{2e}}$. where $\mathcal{K}_{\mathsf{PRK}_{2e}}$ is the key space of $\mathsf{PRK}_{2e}$. We note here that the cid partner is not guaranteed to have received the honest DH shares from the tested session; that is, if the cid partner is in the initiator role, the adversary $\mathcal{A}$ could have delivered a malicious share, for which $\mathcal{A}$ could even know the corresponding secret scalar. Therefore, we also replace $\mathsf{PRK}_{2e}$ at the contributive partner with the same $\widetilde{\mathsf{PRK}_{2e}}$ *only if* the contributive partner holds the same DH shares as the tested session. To justify this step, we exhibit a reduction to an snPRF-ODH adversary $\mathcal{B}_{II.B2}$ which, at a high level, receives Diffie-Hellman shares from its challenger and encodes them in the shares $G_x$ and $G_y$ used by the partnered sessions.

*Simulation soundness.* $\mathcal{B}_{II.B2}$ simulates $\mathbf{G}_{II.B2}$ towards $\mathcal{A}$ and must answer all queries consistently. The queries of interest here are Send queries that induce the computation of the $\mathsf{PRK}_{2e}$ at the tested session $\pi_U^i$ and eventually at its partnered session $\pi_V^j$. $\mathcal{B}_{II.B2}$ consistently answers all other queries. We observe that if the tested session is in the initiator role, then $\pi_U^i$ and $\pi_V^j$ have the same $\widetilde{\mathsf{PRK}_{2e}}$. If, however, $\pi_U^i$ is a responder session, $\pi_V^j$ may have received a modified $G_y'$ for which the attacker knows the private scalar $z$ such that $G_y' = zG$. $\mathcal{B}_{II.B2}$ must be able to compute $xzG$, which is achievable given access to the "left" PRF-ODH oracle $\mathcal{O}_x(S, v)$. Finally, we observe that in EDHOC, $\mathcal{A}$ is only allowed to deliver a potentially modified $G_y'$ once to $\pi_V^j$; this implies that the reduction only needs access to a single $\mathcal{O}_x(S, v)$ query.

*Details of the reduction.* $\mathcal{B}_{II.B2}$ receives DH shares $u * G$ and $v * G$ from its snPRF-ODH challenger and simulates $\mathbf{G}_{II.B2}$ towards $\mathcal{A}$ answering all queries unrelated to $\mathsf{PRK}_{2e}$ as needed. $\mathcal{B}_{II.B2}$ encodes the received DH shares $(u * G, v * G)$ into the Diffie-Hellman shares $(G_x, G_y)$ of the tested session and its partner, respectively. To derive $\mathsf{PRK}_{2e}$, $\mathcal{B}_{II.B2}$ makes a PRF query on input the empty string `""` (recall that $\mathsf{PRK}_{2e} = \mathsf{Extract}(\texttt{""}, xy * G)$) and copies the result into the state of the tested session. $\mathcal{B}_{II.B2}$ copies $\mathsf{PRK}_{2e}$ into the state of the contributive partner if it received the DH share $G_y$. If the partner session receives a modified $G'_y$, $\mathcal{B}_{II.B2}$ calls the left oracle $\mathcal{O}_x(S, v)$ on the inputs $S = G'_y$ and $v =$ "".

As a consequence, the advantage difference between $\mathbf{G}_{II.B1}$ and $\mathbf{G}_{II.B2}$ can be bounded by the advantage of the snPRF-ODH adversary $\mathcal{B}_{II.B2}$, and we get:

$$|\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_{\text{II.B2}}} - \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_{\text{II.B1}}}| \leq \mathsf{Adv}_{\mathcal{B}_{II.B2}}^{\text{snPRF-ODH}}(\mathsf{Extract}).$$

**Game** $\mathbf{G}_{II.B3}$**.** In this game, we replace the function $\mathsf{Expand}$ keyed with $\widetilde{\mathsf{PRK}_{2e}}$ with a random function $F$ at the tested session. The contributive partner also replaces $\mathsf{Expand}$ with $F$ *only if* it received honest DH shares. We justify this step by relating and bounding the advantage difference of $\mathcal{A}$ to the advantage of an PRF adversary $\mathcal{B}_{II.B3}$. $\mathcal{B}_{II.B3}$ simulates $\mathbf{G}_{II.B3}$ towards $\mathcal{A}$ answering all queries that do not trigger a call to $\mathsf{Expand}$. To answer queries that require deriving any key, IV or MAC tag derived from $\mathsf{PRK}_{2e}$, $\mathcal{B}_{II.B3}$ queries its PRF oracle with the appropriate input. We summarized the label used to derive each key, IV, and MAC tags in Table 3.2. By the game $\mathbf{G}_{II.B2}$, we have replaced $\mathsf{PRK}_{2e}$ by a random value, and each key, IV or MAc tag is computed with a unique and distinct label. Therefore, the simulation is sound, and we get the following:

$$|\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_{\text{II.B3}}} - \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_{\text{II.B3}}}| \leq \mathsf{Adv}_{\mathcal{B}_{II.B3}}^{\mathsf{PRF}}(\mathsf{Expand}).$$

Having replaced $\mathsf{Expand}$ with a random function $F$, we can readily replace all values derived by the tested session using a call to $\mathsf{Expand}$ with uniform random values. Again, we replace these values in the partner session only if it received an honest DH share. Concretely, we replace in the tested session (and possibly in the contributing partner) the keys $(\mathsf{K}_2, \mathsf{K}_3, \mathsf{K}_4, \mathsf{PRK}_{out})$, the initialization vectors $(\mathsf{IV}_3, \mathsf{IV}_4)$, and the mac tags $(\tau_2, \tau_3)$ with values drawn at random from the corresponding domains. We call the newly sampled values $\widetilde{\mathsf{K}_2}, \widetilde{\mathsf{K}_3}, \widetilde{\mathsf{K}_4}, \widetilde{\mathsf{PRK}_{out}}, \widetilde{\mathsf{IV}_3}, \widetilde{\mathsf{IV}_4}, \widetilde{\tau_2}, \widetilde{\tau_3}$. Since all values are derived in EDHOC by evaluating the random function $F$ on a unique input per value, $F$ produces independent random values. Here, one may object that the key spaces $\mathcal{K}_k, k \in \{\mathsf{K}_2, \mathsf{K}_3, \ldots, \tau_3\}$ may be different; therefore, it is not clear that $F$ can produce random values from each key space. We note that $\mathcal{K}_k = \{0,1\}^{klen}$, where *klen* is the length of $k$. Furthermore, we assume that $F$ is a variable-length random function with output space $\{0,1\}^*$. Therefore, all keys can be computed accordingly.

*Remark.* At this stage, all keys are random values independent of the challenge bit $b$, and it remains to argue that RevSessionKey queries do not help the adversary. Interestingly, keys may not necessarily be independent when sessions are not partnered. The following problem may arise in EDHOC: With credential identifiers (kid) that can refer to multiple identities and associated public keys, a session may believe they are talking to a session with a different identity than the one involved in the protocol. By the definition of the session identifiers, the two sessions will no longer be partnered. However, the two sessions will derive the same stage keys. Therefore, the adversary may use RevSessionKey queries to guess the challenge bit with high probability. This observation suggests that we must analyze the eventuality of *ambiguous signatures*.

**Conclusion of phase II.** Fortunately, the case of *ambiguous signatures* corresponds to the adversary breaking explicit authentication. Given that the adversary does not break explicit authentication in this branch of the proof, we conclude that the challenge bit is random and independent of the adversary's guess. Therefore:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_{\text{II.B3}}} \leq 0.$$

To summarize the proof:

1. Sound. As discussed in the conclusion of phase 1, the predicate Sound remains true.
2. ExplicitAuth. As discussed in the conclusion of phase II, the predicate ExplicitAuth remains true.
3. Key secrecy. As discussed in the conclusion of phase II, the adversary $\mathcal{A}$ cannot guess the challenge bit with non-negligible probability, which proves that the stage keys are indistinguishable from uniform random values. Therefore, we have shown that all the security properties defined in our MSKE model for EDHOC-Sig-Sig hold, which concludes the proof of Theorem 5.1. □

Chapter 6

# Discussion and Conclusion

In this final chapter, we offer closing remarks on the EDHOC protocol and its development process. We report on our involvement with the LAKE working group and our contributions to the protocol specification. We discuss the limitations of our work, including aspects of the protocol that we excluded from the scope of our analysis. Finally, we conclude by offering some direction for future work.

## 6.1 Comments on the design of EDHOC

In the following, we would like to offer some thoughts on the design of the EDHOC protocol; we discuss some of its peculiarities and contrast them with the existing TLS 1.3 handshake protocol.

### 6.1.1 Chasing a moving target

In the course of this thesis project, the EDHOC draft underwent numerous modifications. Starting with draft 12, our analysis evolved along the versions and finally focused on draft 14. The evolution of the draft versions was driven by various contributions from engineering teams and teams performing security analysis, such as ours. This relatively dynamic environment required us to stay attentive to the inputs of multiple stakeholders and identify relevant inputs from other security analyses.

### 6.1.2 Non-unique credential identifiers

A striking difference between EDHOC and TLS 1.3 (besides being different instantiations of SIGMA) is the use of credential identifiers in EDHOC, sent during authentication, in lieu of certificates commonly used in the handshake messages of TLS 1.3. Small-sized credential identifiers have the benefit of keeping the bandwidth footprint of protocol messages relatively low, hence their attractiveness for constrained networks. However, some implementations may not enforce uniqueness of the credential related to a credential identifier. Non-unique credential identifiers can potentially create ambiguity about the identity of peers engaging in the protocol and open the door for identity mis-binding attacks. This observation influenced the design of our security model to pay special attention to explicit authentication guarantees in EDHOC, as discussed in Section 4.1.1.

### 6.1.3 Identity protection

In specific scenarios, it is desirable to protect the identity of the protocol participants. The SIGMA protocol elegantly offers this feature through encryption of the key exchange messages where peers mutually authenticate. Secrecy of the responder's identity is guaranteed against passive adversaries, while an active attacker should be unable to learn the initiator's identity. Interestingly, the protocol message 2 is encrypted by XORing with a key stream of the appropriate length derived from the shared Diffie-Hellman secret. The reason for avoiding using an AEAD scheme is to mitigate the overhead caused by the ciphertext expansion due to the additional MAC tag. Additionally, one could argue that secrecy of the responder's identity is guaranteed only against passive adversaries, i.e., an adversary that faithfully relays the first two protocol messages. From our perspective, the use of an AEAD scheme would be preferable. Considering the noticeable bandwidth economy achieved by EDHOC in comparison to TLS 1.3, an additional 32 bytes for the MAC tag does not seem to be unreasonable and leaves a sizeable bandwidth margin.

## 6.2 Interactions with the IETF

**The standardization process.** The Internet Engineering Task Force has established the LAKE working group to guide the development of the EDHOC protocol. The work on EDHOC started in early July 2020. The draft has gone through several versions, and the current version is draft 15[1]. The working group is an open environment, allowing various entities to contribute ideas and improvement suggestions freely. On the one hand, engineering teams have reported their performance evaluation of the EDHOC protocol; multiple open-source implementations of EDHOC are available for experimentation. On the other hand, many other entities have and are still working on analyzing the security of the EDHOC protocol. In particular, Jacomme *et al.* have been actively working on analyzing the security of the EDHOC protocol in the symbolic model. They use the formal analysis tool chain SAPIC+ [CJKK22] to analyze all authentication modes in EDHOC. They

[1]EDHOC draft 15: https://datatracker.ietf.org/doc/draft-ietf-lake-edhoc/15/.

also suggest numerous changes to the draft. On the other hand, Cottier and Pointcheval investigated the security of EDHOC STAT-STAT[CP22], focusing on tightness and the security of ciphersuite with 64 bits MACs. They study the possibility of "augmenting" the security level of such ciphersuites and recommend a few improvements to insure tight security bounds. Our work is a humble contribution to this effort.

**Our involvement with the working group.** Throughout this thesis project, we have had a handful of interactions with the working group. In the early months of this project, we attended the IETF 113 meeting to get a feel for the progress of the working group, to gain visibility into concurrent security analyses, and to announce our ongoing work on EDHOC. Later on, on the 31st of March 2022, we provided some early comments on the protocol design; we suggested several improvements to the protocol, mainly targeting the key schedule and the computation of the transcript hashes [IGb]. These changes were positively received by the working group that integrated them into the current draft version. We discuss these suggestions in detail below. On the 8th of July 2022, we provided additional suggestions for improving explicit authentication based on our security analysis[IGa]. Finally, we also presented our final analysis and main results to the IETF 114[2] meeting on the 27th of July 2022.

Through it all, we have found the interaction with the working group very cordial. We thank the LAKE chairs and all members for their kind reception of our work.

## 6.3 Contributions to the Design of EDHOC

We describe our main contributions, starting with the early comments and concluding with our final suggestions extracted from the insight of our final analysis. Our suggestions on the early drafts have been integrated into the current protocol specification. The latest contribution is expected to be added to the next draft.

### 6.3.1 Early comments on draft 12

**Lack of a final session key.** The draft 12 [SMP] was ambiguous regarding what should be considered the final session key. Based on the key schedule, the key $\mathsf{PRK}_{4e3m}$ appears in the final session key. It is used to derive the OSCORE master secret and the OSCORE master salt; the exporter and the key update mechanism were also keyed with $\mathsf{PRK}_{4e3m}$. In EDHOC SIG-SIG $\mathsf{PRK}_{4e3m} = \mathsf{PRK}_{2e}$. We observed that $\mathsf{PRK}_{4e3m}$ is also used within the

[2] https://datatracker.ietf.org/meeting/114/materials/agenda-114-lake-02.

key exchange to derive MAC tags, encryption keys for protocol messages, and so on. Therefore, it is impossible to prove key indistinguishability for this session key. Though this impossibility may not have any practical impact, we suggested that a dedicated session key be defined to cleanly prove its security. This suggestion was also brought up by Jacomme *et al.* who recommended the addition of the final session key[3] $\mathsf{PRK}_{out}$.

**Reuse of keying material in the key schedule.** The key schedule specified in draft 12 uses the pseudo-random keys $\mathsf{PRK}_{2e}$, $\mathsf{PRK}_{3e2m}$ and $\mathsf{PRK}_{4e3m}$ as keying material in both Extract and Expand. The precise usage depends on the authentication method. We pointed out that such reuse of keying material is generally not secure unless done with careful attention to domain separation. We referenced the usage of *derived secrets*[4] in the TLS 1.3 specification to separate domains of Extract and Expand. In response, the working group redesigned the key schedule [5]; which now includes *stage salts* used to separate the calls to Expand and Extract, similarly to TLS 1.3.

**Transcript hashes.** In draft 12, the transcript hashes are computed on the encrypted protocol messages. More precisely, the partial transcript hashes three and four are computed as follows: $\mathsf{th}_3 = \mathsf{H}(\mathsf{th}_2, \mathsf{ctxt}_2)$ and $\mathsf{th}_4 = \mathsf{H}(\mathsf{th}_3, \mathsf{ctxt}_3)$. From an analysis perspective, the inclusion of ciphertext introduces a dependency on the encryption method. We included this in our previous remarks on the design of EDHOC. In response, the working group redesigned the partial transcript hash computation; the underlying plaintext messages are now used in computing the transcript hashes. We additionally remark here that the partial hash computation diverges from how it is performed in TLS 1.3 using a *running hash*. The recursive nature of the transcript hash computation in EDHOC is due to the lack of appropriate cryptographic APIs in low-end devices.

### 6.3.2 Further comments on draft 14

Towards the end of the thesis project, we offered further improvement suggestions based on a complete analysis of the EDHOC protocol. Our study of the authentication guarantee of the EDHOC protocol provided us with insights into the MAc-then-SIGn protocol and the particular requirement for the digital signature schemes used in EDHOC. To strengthen EDHOC against potential attacks taking advantage of credential identifiers that identify multiple credentials, we suggested that the transcript hashes $\mathsf{th}_3$ (resp. $\mathsf{th}_4$) should include the credentials of the responder (resp. the initiator).

[3] https://github.com/lake-wg/edhoc/pull/276.

[4] https://github.com/tlswg/tls13-spec/pull/875.

[5] https://openwsn.atlassian.net/wiki/spaces/LAKE/pages/1932427302/Key+Schedule.

An official pull request was opened to integrate our suggestions into the draft [Mat22]. The modification we propose has minimal impact on the performance and was positively received by the working group. We expect that the next draft version will include our suggestions.

## 6.4 Limitations

### 6.4.1 Scope limited to the SIG-SIG mode

Our analysis of the EDHOC protocol is limited to the SIG-SIG mode for authentication, although some of our comments to the working group also affected other authentication methods. The SIG-SIG mode and its intricacies proved challenging enough, hence, the scope limitation. We note, however, that the risks of ambiguous signatures are also valid for the STAT-SIG and SIG-STAT modes.

### 6.4.2 Loose security bounds

In this work, we show EDHOC to be secure and provide a security proof to this effect. Our proof follows the strategies used by Dowling *et al.* to analyze the TLS 1.3 handshake protocol [DFGS21]. Due to various guessing strategies used in our proof (see Chapter 5), our security bound is rather loose. In particular, the security bound depends on the number of users and sessions. Loose security bounds are problematic since they cannot meaningfully inform the choice of concrete parameters to instantiate the protocol both securely and efficiently with existing cryptographic primitives. Davis and Günther [DG21] point out that a concrete evaluation of the bounds in [DFGS21] results in an uncomfortably high probability that a state adversary may break the TLS 1.3 key exchange if instantiated with commonly used cryptographic primitives. We fully expect the same criticism to be relevant when concretely evaluating our security bound.

### 6.4.3 Out-of-scope considerations

We restricted our analysis to the cryptographic core of the EDHOC protocol. To carry out a security analysis, one needs to find the right abstraction level while still capturing the essence of the protocol. Although we did our best to find the appropriate balance between abstraction and completeness, such an analysis cannot capture all aspects of a complex protocol. In particular, we do not model negotiation of authentication mechanism. We assume in our analysis that the SIG-SIG mode is used. Moreover, we also leave the negotiation of the ciphersuite out of scope.

Furthermore, the actual real-world security of the protocol depends on far more factors than can be captured in this thesis. Insecure programming practices and various other attack surfaces in low-powered devices may undermine the security guarantee of the EDHOC protocol, despite our analysis. For instance, a real-world attacker may use so-called software exploits to bypass the security guarantees offered by the EDHOC protocol.

## 6.5 Future Work

**SIG-STAT and STAT-SIG modes.** As discussed in the previous sections, our analysis solely focused on the SIG-SIG mode. From our interaction with the working group, we were made aware of the work of Baptiste Cottier [Cot22] on the STAT-STAT mode. The security of *mixed* modes such as SIG-STAT and STAT-SIG are currently open questions worth investigating.

**Tighter analysis** A significant improvement to the work presented in this thesis would be a tighter security analysis. Davis and Günther [DG21] presented a tight analysis of SIGMA and the TLS 1.3 handshake. The authors concluded with optimism about the applicability of their results to multi-stage settings. To carry out a concrete and tight analysis on PSK modes of the TLS 1.3 handshake [DDGJ22], Davis *et al.* employed techniques from [DG21, DJ21]. We believe that the techniques used in [DG21] can serve as a foundation towards a tighter analysis of the EDHOC protocol.

**Security analysis of OSCORE.** Establishing a security context is the main use case for EDHOC. The security of the overall deployment must normally be analyzed, given that a protocol proven secure in a standalone manner may not remain secure when combined arbitrarily. Brzuska *et al.* [BFWW11] have shown that Bellare-Rogaway like key exchange protocols security with other security symmetric primitives. To the best of our knowledge, the OSCORE protocol has not yet received a formal security analysis. Nevertheless, a formal analysis limited to the security of the OSCORE protocol can augment our work and give further confidence in the overall protocol without the need to analyze the OSCORE and EDHOC in conjunction.

Appendix A

# Appendix

## A.1 Identity Mis-Binding in MAc-then-SIGn

We present an identity mis-binding on the MAc-then-SIGn protocol, hereafter denoted by $\mathsf{SIGMA}_{\frac{\sigma}{\tau}}$ for "MAC under the signature". For simplicity, we exclude identity protection from our treatment. The core issue is caused by a lack of explicit verification of the MAC tag; instead, the tag is implicitly verified through the verification of the signature. Existing analysis of SIGMA (SIGn-and-MAc instantiation) show that a secure instantiation must use an unforgeable signature scheme, including the original analyze of C. We show that for the security of $\mathsf{SIGMA}_{\frac{\sigma}{\tau}}$, it is not enough for the signature scheme to be EUF-CMA secure; it must be unforgeable for all keys. In particular, there cannot be any weak key, for instance one that accepts any message signature pair.

Consider a simplified run of the MAc-then-SIGn, where we additionally omit encryption. The initiator sends the first message $(G_x, I)$. The responder then computes a shared session key $\mathsf{K}$ and a MAC key $\mathsf{K}_m$ using a key derivation function. It responds with the second protocol message $\big(R, G_y, \sigma = \mathsf{Sign}(sk_I, \mathsf{MAC}(\mathsf{K}_m, G_y, G_x, R))\big)$. Assume now that the attacker intercepts this message and modifies the message with the value $\big(R', G_y, \sigma\big)$. The identity $R'$ has a weak verification key $pk$ such that for all message $m$ and signature $\sigma$, $\mathsf{Vf}(pk, m, \sigma) = 1$. As a consequence, the attacker need not know the MAC key and the initiator will accept the signature with peer identity $R'$. Furthermore, the attack does not modify the transcript, hence the initiator's third message is successfully accepted by the responder.

**The issue.** At a high level, in MAc-then-SIGn, peers do not explicitly prove knowledge of the key via the MAC. Instead, the MAC is only implicitly verified once the signature is accepted. The insufficiency of the standard EUF-CMA notion is not enough because EUF-CMA it captures unforgeability “on average”. While the attack described here requires unforgeability almost everywhere. In other words, let Sig be a EUF-CMA secure signature scheme. Now create $\widehat{\mathsf{Sig}}$ such that the key space of $\widehat{\mathsf{Sig}}$ is the key space of Sig augmented with the special key pair $(sk^*, pk^*)$. Furthermore, modify signing is the same unless the signing key is $sk^*$ in which case a random signature value is return. The verification algorithm is also similar to the original signature scheme unless the verification key is $pk^*$ in which case the verification algorithm always returns 1. Observe that $\widetilde{sig}$ is still EUF-CMA as the advantage of any EUF-CMA adversary is only increased by a negligible factor, corresponding to the choice of $(sk^*, pk^*)$ as challenge key pair. However, the $\mathsf{SIGMA}_{\frac{\sigma}{\tau}}$ protocol is vulnerable to the attack described above if instanciated with $\widetilde{sig}$.

# Bibliography

[BCF+13] Colin Boyd, Cas Cremers, Michèle Feltz, Kenneth G. Paterson, Bertram Poettering, and Douglas Stebila. ASICS: Authenticated Key Exchange Security Incorporating Certification Systems. In Jason Crampton, Sushil Jajodia, and Keith Mayes, editors, *Computer Security – ESORICS 2013*, Lecture Notes in Computer Science, pages 381–399, Berlin, Heidelberg, 2013. Springer. doi:10.1007/978-3-642-40203-6_22.

[BCJZ20] Jacqueline Brendel, Cas Cremers, Dennis Jackson, and Mang Zhao. The Provable Security of Ed25519: Theory and Practice. Technical Report 823, 2020. URL: https://eprint.iacr.org/2020/823.

[BCK96] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying Hash Functions for Message Authentication. In Neal Koblitz, editor, *Advances in Cryptology — CRYPTO '96*, Lecture Notes in Computer Science, pages 1–15, Berlin, Heidelberg, 1996. Springer. doi:10.1007/3-540-68697-5_1.

[BDL+12] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. *Journal of Cryptographic Engineering*, 2(2):77–89, September 2012. doi:10.1007/s13389-012-0027-1.

[BFGJ17] Jacqueline Brendel, Marc Fischlin, Felix Günther, and Christian Janson. PRF-ODH: Relations, Instantiations, and Impossibility Results. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017*, Lecture Notes in Computer Science, pages 651–681, Cham, 2017. Springer International Publishing. doi:10.1007/978-3-319-63697-9_22.

[BFWW11] Christina Brzuska, Marc Fischlin, Bogdan Warinschi, and Stephen C. Williams. Composability of bellare-rogaway key exchange protocols. In *Proceedings of the 18th ACM conference on Computer and communications security*, CCS '11, pages 51–62, New York, NY, USA, October 2011. Association for Computing Machinery. doi:10.1145/2046707.2046716.

[BH20] Carsten Bormann and Paul E. Hoffman. Concise Binary Object Representation (CBOR). Request for Comments RFC 8949, Internet Engineering Task Force, December 2020. Num Pages: 66. URL: https://datatracker.ietf.org/doc/rfc8949, doi:10.17487/RFC8949.

[BR94] Mihir Bellare and Phillip Rogaway. Entity Authentication and Key Distribution. In Douglas R. Stinson, editor, *Advances in Cryptology — CRYPTO' 93*, Lecture Notes in Computer Science, pages 232–249, Berlin, Heidelberg, 1994. Springer. doi:10.1007/3-540-48329-2_21.

[BR06] Mihir Bellare and Phillip Rogaway. The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006*, Lecture Notes in Computer Science, pages 409–426, Berlin, Heidelberg, 2006. Springer. doi:10.1007/11761679_25.

[BS] Dan Boneh and Victor Shoup. *A Graduate Course in Applied Cryptography*. URL: https://toc.cryptobook.us/.

[BSJGPS18] Alessandro Bruni, Thorvald Sahl Jørgensen, Theis Grønbech Petersen, and Carsten Schürmann. Formal Verification of Ephemeral Diffie-Hellman Over COSE (EDHOC). In Cas Cremers and Anja Lehmann, editors, *Security Standardisation Research*, volume 11322, pages 21–36. Springer International Publishing, Cham, 2018. Series Title: Lecture Notes in Computer Science. URL: http://link.springer.com/10.1007/978-3-030-04762-7_2, doi:10.1007/978-3-030-04762-7_2.

[CDF$^+$21] Cas Cremers, Samed Düzlü, Rune Fiedler, Marc Fischlin, and Christian Janson. BUFFing signature schemes beyond unforgeability and the case of post-quantum signatures. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1696–1714, May 2021. ISSN: 2375-1207. doi:10.1109/SP40001.2021.00093.

[CH98] David Carrel and Dan Harkins. The Internet Key Exchange (IKE). Request for Comments RFC 2409, Internet Engineering

Task Force, November 1998. Num Pages: 41. URL: https://datatracker.ietf.org/doc/rfc2409, doi:10.17487/RFC2409.

[CHH$^+$17] Cas Cremers, Marko Horvat, Jonathan Hoyland, Sam Scott, and Thyla van der Merwe. A Comprehensive Symbolic Analysis of TLS 1.3. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 1773–1788, New York, NY, USA, October 2017. Association for Computing Machinery. doi:10.1145/3133956.3134063.

[CJKK22] Vincent Cheval, Charlie Jacomme, Steve Kremer, and Robert Künnemann. {SAPIC+}: protocol verifiers of the world, unite! pages 3935–3952, 2022. URL: https://www.usenix.org/conference/usenixsecurity22/presentation/cheval.

[Cot22] Baptiste Cottier. slides-113-lake-computational-model-analysis-02.pdf, 2022. URL: https://datatracker.ietf.org/meeting/113/materials/slides-113-lake-computational-model-analysis-02.pdf.

[CP22] Baptiste Cottier and David Pointcheval. Security Analysis of the EDHOC protocol. September 2022. URL: https://arxiv-export1.library.cornell.edu/abs/2209.03599.

[DDGJ22] Hannah Davis, Denis Diemert, Felix Günther, and Tibor Jager. On the Concrete Security of TLS 1.3 PSK Mode. Technical Report 246, 2022. URL: https://eprint.iacr.org/2022/246.

[DFGS21] Benjamin Dowling, Marc Fischlin, Felix Günther, and Douglas Stebila. A Cryptographic Analysis of the TLS 1.3 Handshake Protocol. *Journal of Cryptology*, 34(4):37, July 2021. doi:10.1007/s00145-021-09384-1.

[DG21] Hannah Davis and Felix Günther. Tighter Proofs for the SIGMA and TLS 1.3 Key Exchange Protocols. In *Applied Cryptography and Network Security: 19th International Conference, ACNS 2021, Kamakura, Japan, June 21–24, 2021, Proceedings, Part II*, pages 448–479, Berlin, Heidelberg, June 2021. Springer-Verlag. doi:10.1007/978-3-030-78375-4_18.

[DJ21] Denis Diemert and Tibor Jager. On the Tight Security of TLS 1.3: Theoretically Sound Cryptographic Parameters for Real-World Deployments. *Journal of Cryptology*, 34(3):30, June 2021. doi:10.1007/s00145-021-09388-x.

[FG14] Marc Fischlin and Felix Günther. Multi-Stage Key Exchange and the Case of Google's QUIC Protocol. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 1193–1204, New York, NY, USA, November 2014. Association for Computing Machinery. doi:10.1145/2660267.2660308.

[Gü18] Felix Günther. *Modeling Advanced Security Aspects of Key Exchange and Secure Channel Protocols*. Ph.D. Thesis, Technische Universität, Darmstadt, 2018. URL: https://tuprints.ulb.tu-darmstadt.de/7162/.

[IGa] Marc Ilunga and Felix Günther. [Lake] Computational analysis of EDHOC Sig-Sig - improvement suggestions for transcript hashes. URL: https://mailarchive.ietf.org/arch/msg/lake/BTxPdJl0Z8ylSe1P7i0BFx1T_Oo/.

[IGb] Marc Ilunga and Felix Günther. [Lake] Computational EDHOC analysis - Some early comments and questions. URL: https://mailarchive.ietf.org/arch/msg/lake/i2NYxn3vchQ1jUv91frlcqQ4sP0/.

[JMV01] Don Johnson, Alfred Menezes, and Scott Vanstone. The Elliptic Curve Digital Signature Algorithm (ECDSA). *International Journal of Information Security*, 1(1):36–63, August 2001. doi:10.1007/s102070100002.

[KCP16] John Kelsey, Shu-jen Chang, and Ray Perlner. SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash, and ParallelHash. Technical Report NIST Special Publication (SP) 800-185, National Institute of Standards and Technology, December 2016. URL: https://csrc.nist.gov/publications/detail/sp/800-185/final, doi:10.6028/NIST.SP.800-185.

[KMO+14] Markulf Kohlweiss, Ueli Maurer, Cristina Onete, Bjoern Tackmann, and Daniele Venturi. (De-)Constructing TLS. *Cryptology ePrint Archive*, 2014. URL: https://eprint.iacr.org/2014/020.

[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*, pages 400–425. Springer, Berlin, Heidelberg, 2003. URL: https://link.springer.com/chapter/10.1007/978-3-540-45146-4_24, doi:10.1007/978-3-540-45146-4_24.

[Kra10] Hugo Krawczyk. Cryptographic Extraction and Key Derivation: The HKDF Scheme. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, Lecture Notes in Computer Science, pages 631–648, Berlin, Heidelberg, 2010. Springer. doi:10.1007/978-3-642-14623-7_34.

[LKK21] Hyunwoo Lee, Doowon Kim, and Yonghwi Kwon. TLS 1.3 in Practice:How TLS 1.3 Contributes to the Internet. In *Proceedings of the Web Conference 2021*, WWW '21, pages 70–79, New York, NY, USA, April 2021. Association for Computing Machinery. doi:10.1145/3442381.3450057.

[Mat22] John Preuß Mattsson. Change TH_3 and TH_4 by emanjon · Pull Request #318 · lake-wg/edhoc, 2022. URL: https://github.com/lake-wg/edhoc/pull/318.

[MPV20] John Preuß Mattsson, Francesca Palombini, and Mališa Vučinić. Comparison of CoAP Security Protocols. Internet Draft draft-ietf-lwig-security-protocol-comparison-05, Internet Engineering Task Force, November 2020. Num Pages: 39. URL: https://datatracker.ietf.org/doc/draft-ietf-lwig-security-protocol-comparison-05.

[MS04] Alfred Menezes and Nigel Smart. Security of Signature Schemes in a Multi-User Setting. *Designs, Codes and Cryptography*, 33(3):261–274, November 2004. doi:10.1023/B:DESI.0000036250.18062.3f.

[NSB21] K. Norrman, V. Sundararajan, and A. Bruni. Formal Analysis of EDHOC Key Establishment for Constrained IoT Devices. In *SECRYPT*, 2021. doi:10.5220/0010554002100221.

[PS05] Thomas Pornin and Julien P. Stern. Digital Signatures Do Not Guarantee Exclusive Ownership. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *Applied Cryptography and Network Security*, pages 138–150, Berlin, Heidelberg, 2005. Springer. doi:10.1007/11496137_10.

[Res18] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. Request for Comments RFC 8446, Internet Engineering Task Force, August 2018. Num Pages: 160. URL: https://datatracker.ietf.org/doc/rfc8446, doi:10.17487/RFC8446.

[Rog02] Phillip Rogaway. Authenticated-encryption with associated-data. In *Proceedings of the 9th ACM conference on Computer and* *communications security*, CCS '02, pages 98–107, New York, NY, USA, November 2002. Association for Computing Machinery. doi:10.1145/586110.586125.

[Sch17] Jim Schaad. CBOR Object Signing and Encryption (COSE). Request for Comments RFC 8152, Internet Engineering Task Force, July 2017. Num Pages: 121. URL: https://datatracker.ietf.org/doc/rfc8152, doi:10.17487/RFC8152.

[Shr04] Tom Shrimpton. A Characterization of Authenticated-Encryption as a Form of Chosen-Ciphertext Security. *Cryptology ePrint Archive*, 2004. URL: https://eprint.iacr.org/2004/272.

[SMP] Göran Selander, John Preuß Mattsson, and Francesca Palombini. Ephemeral Diffie-Hellman Over COSE (EDHOC). Internet Draft draft-ietf-lake-edhoc-12, Internet Engineering Task Force. Num Pages: 80. URL: https://datatracker.ietf.org/doc/draft-ietf-lake-edhoc-12.

[SMP22a] Göran Selander, John Preuß Mattsson, and Francesca Palombini. Ephemeral Diffie-Hellman Over COSE (EDHOC). Internet Draft draft-ietf-lake-edhoc-15, Internet Engineering Task Force, July 2022. Num Pages: 92. URL: https://datatracker.ietf.org/doc/draft-ietf-lake-edhoc.

[SMP22b] Göran Selander, John Preuß Mattsson, and Francesca Palombini. Ephemeral Diffie-Hellman Over COSE (EDHOC). Internet Draft draft-ietf-lake-edhoc-14, Internet Engineering Task Force, May 2022. Num Pages: 90. URL: https://datatracker.ietf.org/doc/draft-ietf-lake-edhoc-14.

[SMPS19] Göran Selander, John Preuß Mattsson, Francesca Palombini, and Ludwig Seitz. Object Security for Constrained RESTful Environments (OSCORE). Request for Comments RFC 8613, Internet Engineering Task Force, July 2019. Num Pages: 94. URL: https://datatracker.ietf.org/doc/rfc8613, doi:10.17487/RFC8613.