Careful with MAc-then-SIGn: A Computational Analysis of the EDHOC Lightweight Authenticated Key Exchange Protocol

Felix Günther
ETH Zurich
Zurich, Switzerland
mail@felixguenther.info

Marc Ilunga Tshibumbu Mukendi
Trail of Bits
New York City, USA
marc.ilunga@trailofbits.com

***Abstract*—EDHOC is a lightweight authenticated key exchange protocol for IoT communication, currently being standardized by the IETF. Its design is a trimmed-down version of similar protocols like TLS 1.3, building on the SIGn-then-MAc (SIGMA) rationale. In its trimming, however, EDHOC notably deviates from the SIGMA design by sending only short, non-unique credential identifiers, and letting recipients perform trial verification to determine the correct communication partner. Done naively, this can lead to identity misbinding attacks when an attacker can control some of the user keys, invalidating the original SIGMA security analysis and contesting the security of EDHOC.**

In this work, we formalize a multi-stage key exchange security model capturing the potential attack vectors introduced by non-unique credential identifiers. We show that EDHOC, in its draft version 17, indeed achieves session key security and user authentication even in a strong model where the adversary can register malicious keys with colliding identifiers, given that the employed signature scheme provides so-called exclusive ownership. Through our security result, we confirm cryptographic improvements integrated by the IETF working group in recent draft versions of EDHOC based on recommendations from our and others' analysis.

1. Introduction

Low-powered devices such as smart appliances, colloquially referred to as "Internet of Things" (IoT), are becoming increasingly ubiquitous in many public spaces and private homes [1]. Besides their computational limitations, IoT devices often operate in environments with stringent network constraints such as LoRaWAN [46].

The proliferation of IoT enables numerous fascinating applications and a certain level of convenience, but also brings its share of challenges. Security considerations are often out of the picture due to said restrictions of IoT applications, leading to the famous saying *"the 'S' in IoT stands for security"*. Examples include applications transferring sensitive data over an insecure channel or firmware updates over insecure channels allowing for injection of arbitrary code. To illustrate a few, [38] demonstrates how a network attacker can defeat smart locks to gain unauthorized access to households, [3] exploits a vulnerability in baby monitoring cameras that gives an attacker access into the privacy of children, [2] demonstrates how a network attacker can exploit insecure smart fridges to get access to their owners' Google accounts, and [18] shows how a malicious entity may interfere with highly intimate details of lovers. If compromised, the ubiquity of IoT devices gives attackers access to sensitive networks, compounding the initial low-cost compromise of the devices.

Designing a secure communication protocol for constrained environments comes with two main challenges: First, low-powered devices support only a limited set of cryptographic primitives. Second, IoT security protocols must incur only minimal bandwidth, round-trip time, and power consumption overhead to fit the network constraints. The Internet Engineering Task Force (IETF) set out to standardize protocols and efficient communication data formats for constrained devices. Of particular relevance are: the Concise Binary Object Representation (CBOR) data format (RFC 8949 [12]) for extremely small message formats; the CBOR Object Signing and Encryption (COSE) protocol (RFC 9052 [54]) defining basic security services like signing, MACing, and encryption for CBOR-serialized data; and the Object Security for Constrained RESTful Environments (OSCORE) protocol (RFC 8613 [55]) for end-to-end application data encryption based on CBOR and COSE. To protect application data, OSCORE requires that protocol participants have established a so-called "security context", effectively a cryptographic session key. An essential missing piece in this technological chain is hence a lightweight and secure key exchange protocol establishing these session keys. To close this gap, the IETF Lightweight Authenticated Key Exchange (LAKE) working group was chartered to standardize such a key exchange protocol. It leads the development of this standard under the name EDHOC ("Ephemeral Diffie–Hellman Over COSE") [56]. The working group has invited (and received) formal and computational security analysis of EDHOC [57]; at the time of this submission, the EDHOC draft standard is in "Working Group Last Call" for final comments.

WHY NOT TLS 1.3? One might ask why a dedicated key exchange protocol is needed for the LAKE setting. After all, the IETF has already standardized several well-established key exchange protocols. One of the most prominent examples is the Transport Layer Security (TLS) protocol; its latest version TLS 1.3 [52] has seen substantial security analysis before and after standardization (e.g., [9], [10], [25], [28], [30], [31]). Additionally, TLS 1.3 is efficient, widely adopted, and supported by several highly optimized and interoperable implementa-

tions [45]. EDHOC and the TLS 1.3 key exchange ("handshake") even share a common design, inspired by the "SIGn-and-MAc" (SIGMA) protocol family proposed by Krawczyk [19], [43], also underlying the Internet Key Exchange (IKE) protocol [40]. Lastly, both protocols draw from an overlapping range of cryptographic primitives for Diffie–Hellman (X25519), signatures (Ed25519, ECDSA), key derivation (HMAC, HKDF), and authenticated encryption (AES-GCM, AES-CCM, Chacha20/Poly1305).

Naturally, one may wonder: why re-invent a new key exchange protocol and not simply use TLS 1.3 in EDHOC? The answer lies again in the constraint environment: comparing the bandwidth overhead of TLS and Datagram TLS (DTLS) with EDHOC, Mattsson et al. [47] show that EDHOC, with a minimum total bandwidth usage as low as 101 bytes [56, Section 1.2], outperforms both TLS 1.3 and DTLS 1.3 by a factor up to 6. Notably, EDHOC specifies four different modes (SIG-SIG, SIG-STAT, STAT-SIG, STAT-STAT) with differing bandwidth characteristics; these modes result from the initiator and responder individually choosing whether they want to authenticate using signature keys (SIG) or static Diffie–Hellman keys (STAT). In this work, we focus on the SIG-SIG mode, which most closely follows the SIGMA design.

EDHOC achieves its low bandwidth usage through aggressive savings. Beyond message format optimizations, these in particular include the following two cryptographically interesting changes:

- MAC-THEN-SIGN. In the classical SIGn-then-MAc approach [43] employed by TLS 1.3, parties send a signature and then a MAC to authenticate. EDHOC instead uses a bandwidth-optimized "MAc-then-SIGn" variant (discussed in [43] and also used in IKE [40]), where the MAC is computed first and not sent explicitly (thereby saving bandwidth), but instead put under the signature.
- ABBREVIATED IDENTIFIERS. Usually, parties identify themselves by sending a certificate (e.g., X.509 certificates in TLS) or similar as a means for the communicating peer to unambiguously determine who they are supposedly talking to and should authenticate. By contrast, EDHOC assumes parties may already hold those certificates locally and sends only short so-called *credential identifiers* instead of the full credentials. Crucially, these credential identifiers *need not be unique*: recipients might associate multiple identities with one identifier and need to check—e.g., by trial signature verification—which is the right one.

It turns out that the combination of these two savings changes, if one is not being especially careful, has the potential to introduce new attack vectors.

ALL GOOD WITH MAC-THEN-SIGN? To understand what could go wrong with the tweaks EDHOC introduces, let us recap the SIGMA design and its MAc-then-SIGn variant in a bit more detail. SIGMA, and EDHOC in SIG-SIG mode, build upon a classic, unauthenticated Diffie–Hellman (DH) protocol and then have peers authenticate through signatures under their long-term signing keys and MACs derived from the shared DH key. In the MAc-then-SIGn variant, the authenticating party P computes the MAC tag τ to cover P, then computes the signature σ over the exchanged DH shares and τ, and finally sends σ *but not* τ. The latter is to save bandwidth, leveraging that τ can be recomputed locally by the receiving party. Within the original SIGMA analysis paper by Canetti and Krawczyk [19], this variant of SIGMA was analyzed as secure in a computational key exchange model building on the classical Bellare–Rogaway model [6].

EDHOC, however, deviates in one noticeable aspect from SIGMA: while user identities are assumed to be unique in the analysis of the latter, EDHOC sends only *non-unique* credential identifiers. This means that a receiver of the signature σ above needs to trial-verify σ under possibly multiple public keys matching the sent credential identifier. Such trial verification can be problematic when adversarially-controlled public keys are among the potential matching ones. Indeed, this is precisely the attack recipe for so-called "duplicate-signature key selection" (DSKS) or "exclusive ownership" attacks [11], [42], [48], [51], where an adversary creates a public key under which an (honestly generated) message-signature pair verifies. In the EDHOC setting, such an attack may translate to the adversary fooling the recipient of a signature to assume it originated from a *different* signer that happens to have a colliding credential identifier, which would violate the correct authentication of entities by the protocol.

To properly capture such attacks, we hence ought to study EDHOC in a model that allows the adversary to (1) register its own (potentially maliciously generated) signature keys—an approach similar to the ASICS model [13] incorporating real-world certification— and (2) control the potentially colliding credential identifiers to capture their ambiguity. Worryingly, such a stronger model invalidates the original MAc-then-SIGn analysis in [19]—indeed, there is a trivial *identity misbinding* attack (the technical details of which we discuss in Appendix C): the attacker can register a degenerate key that makes one side of the communication accept with the wrong peer identifier, but the same session key, violating security. The question hence is: what does this mean for EDHOC?

1.1. Contributions

In this work, we perform a computational cryptographic analysis of EDHOC's SIG-SIG mode with signature-based authentication. We confirm that, with a couple of recently introduced cryptographic improvements prompted by our and others' analysis, the protocol in draft version 17 [56] achieves security even in a strong model where the adversary can register malicious keys with colliding identifiers. In more detail, our contributions are as follows.

CRYPTOGRAPHIC CORE OF EDHOC SIG-SIG. With no prior computational analysis, our first step is to extract the core cryptographic operations of the EDHOC SIG-SIG mode (in draft 17 [56]), which we describe in Section 3. Focusing on the SIG-SIG mode and a defined set of algorithms, we abstract away mode and algorithm negotiation, but consider detailed computation and key derivation steps and carefully capture EDHOC's non-unique credential identifiers and the "trial verification" they require in the protocol execution.

STRONG KEY EXCHANGE SECURITY MODEL. For our computational analysis, we base the security definition on the well-established Bellare–Rogaway model [6], capturing strong person-in-the-middle attacks where adversaries are allowed to compromise parties and established session keys. We incorporate a generalization of this model to capture multi-stage key exchange (MSKE) security [32], [34], enabling a fine-grained analysis of the several keys derived in EDHOC (the final session key as well as intermediate keys for earlier data encryption). To capture the effects of EDHOC's shortened and non-unique credential identifiers discussed above, we ultimately extend the security model to allow adversaries to *register* malicious signing keys (akin to [13]) and specify the (potentially colliding) *credential identifiers* to be used in EDHOC. The result is a strong security model, given in Section 4, which asks for secure keys and entity authentication even in the presence of maliciously controlled keys whose identifiers may collide with keys of honest users. To avoid ambiguity, we fully specify our model in pseudocode.

COMPUTATIONAL SECURITY ANALYSIS OF EDHOC SIG-SIG. Following the reductionist proof methodology, in Section 5 we then analyze EDHOC's SIG-SIG mode in our extended MSKE security model. We show that, in its draft version 17, EDHOC SIG-SIG mode establishes keys that are (forward) secure, with peers explicitly authenticated upon signature verification. Our security proof formally reduces the success probability of an adversary violating the key exchange security guarantees to the security of EDHOC's building blocks (Diffie–Hellman key exchange, signatures, and key derivation). Of particular interest is clearly the handling of EDHOC's non-unique credential identifiers and their effect on explicit authentication: we show that through the way EDHOC includes (full, unique) identities under the signature and—following our suggestion—in the key derivation, secure authentication is indeed guaranteed based on the signature scheme's (strong) unforgeability and an "exclusive ownership" property (cf. Section 2), which are shown for the Ed25519 signature scheme [14]; for ECDSA the picture is more blurry, as we discuss in Section 3.2 and our proof.

In Appendix C, we discuss the technical details why the original MAc-then-SIGn variant [43] would not be secure in our security model, formally underlining the need for a dedicated analysis of EDHOC.

IMPROVING THE DRAFT STANDARD. Our security result makes use of a couple of improvements introduced in recent draft versions of EDHOC, based on recommendations we and authors of other security analyses (cf. Section 1.2 below) communicated to the IETF LAKE working group. Most notably, we recommended (1) establishing a dedicated session key for key separation and composability; (2) changes to the transcript hash computation to bind identities to keys; (3) reducing the dependency on encryption security; and (4) fixing key-reuse issues in the key derivation. All of these changes were incorporated into EDHOC (in draft 14 resp. 17) after fruitful interaction with the LAKE working group; they faciliated our security results. We conclude with a more detailed account of our contributions to the draft standard, as well as a discussion of limitations and open questions in Section 6.

1.2. Related Work

Krawczyk introduced the SIGMA [43] family of authenticated key exchange protocols. Among many others, SIGMA informed the design of the Internet Key Exchange protocol [40], the TLS 1.3 handshake protocol [52], and the EDHOC SIG-SIG protocol [56]. A detailed security analysis of SIGMA and the MAc-then-SIGn variant on which EDHOC is built was given by Canetti and Krawczyk [19]. TLS 1.3 was analyzed in the computational setting by Dowling et al. [30], [31] in a multi-stage security model [32]; our model follows their approach, adapting the code-based version of Davis et al. [26].

Following the successful example of TLS 1.3's standardization process [50], the editors of the EDHOC draft standards co-authored a call for formal and computational security analysis by the research community [57], to which this work is not the first to answer. Bruni et al. [16] performed a formal verification of EDHOC (draft 08) using ProVerif. Norman et al. [49] extended that work to cover newly included mixed authentication modes. Cheval et al. [20] used their formal analysis tool chain, SAPIC+, for an initial analysis of EDHOC in draft 07. More recently, Jacomme et al. [36] substantially broadened this analysis, applying SAPIC+ to analyze all four authentication methods in EDHOC draft 12 (and giving preliminary results on draft 14). Their work extensively covers all modes and various security properties in the symbolic model with enhanced idealizations of cryptographic primitives (including some, like key confirmation [**?**], [33], that we do not cover), whereas our work takes a lower-level, computational cryptography perspective and aims at the subtle effects that arise when modeling non-unique credential identifiers. Concurrent to our computational analysis, Cottier and Pointcheval [23] have worked on a tight computational analysis for the EDHOC STAT-STAT mode basing authentication on long-term, static Diffie–Hellman keys. The latter work is closest to ours as it is also computational; it aims at proof tightness in the STAT-STAT mode, whereas our focus is on the SIG-SIG mode and understanding the security ramifications of EDHOC's non-unique credential identifiers.

2. Preliminaries

2.1. Notation

GROUPS AND DIFFIE–HELLMAN. Let $(\mathbb{G}, +)$ be a cyclic group of prime order q generated by G, i.e., $\mathbb{G} = \langle G \rangle = \{xG : x \in \mathbb{Z}_q\}$. 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 $S = xY$.

GAMES, ADVERSARIES, AND ADVANTAGES. We carry out our analysis of EDHOC in the code-based game-playing framework for provable security [7]. The security of a cryptographic scheme or protocol Π is captured by a game $\mathbf{G}(\Pi)$ played by an adversary $\mathcal{A}$ that interacts with the game through several named oracles. Each game provides (sometimes implicitly) two oracles INITIALIZE (executed once at the outset of the game) and FINALIZE (through which the adversary's success is evaluated, called

once at the end); a winning condition is defined and the oracle FINALIZE outputs 1 if that condition is satisfied and 0 otherwise. The advantage of an adversary $\mathcal{A}$ in winning the game $\mathbf{G}(\Pi)$, denoted by $\mathsf{Adv}^{\mathbf{G}}_{\Pi}(\mathcal{A})$, captures the performance of $\mathcal{A}$ which is the probability that FINALIZE outputs 1, i.e.:

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

When the context is unambiguous, we simply write $\mathsf{Adv}^{\mathbf{G}}(\mathcal{A})$ instead of $\mathsf{Adv}^{\mathbf{G}}_{\Pi}(\mathcal{A})$.

2.2. Cryptographic Primitives

The main cryptographic primitives in EDHOC are a cryptographic hash function denoted H, key derivation functions Extract and Expand following the extract-then-expand paradigm of HKDF [44], a digital signature scheme Sig, and an authenticated encryption scheme AEAD. In the following, we recap some of the less established security properties of these schemes we will rely on for our security analysis. For further formal definitions and standard security notions (like collision resistance, signature unforgeability, etc.), see Appendix A.

2.2.1. Key derivation. EDHOC uses a key derivation function to derive session keys, but also MAC tags or IV values. The key derivation in EDHOC closely follows the design of HKDF [44], which is realized via two modules:

1) $\underline{\mathsf{Extract}(s, \mathsf{ikm})}$ extracts a pseudo-random key prk from some (high-entropy, but not necessarily uniform) input material ikm using a salt s.
2) $\underline{\mathsf{Expand}(\mathsf{prk}, \mathit{info}, \mathit{len})}$ generates from a pseudo-random key prk a pseudo-random output string of length len bits, taking as further input a context string info.

We refer to [44] for the detailed rationale behind HKDF. As we will see in more detail in Section 3, EDHOC uses Extract to derive from a Diffie–Hellman secret a uniformly random intermediate key and Expand to from that intermediate key derive the actual session keys and more. Analogous to prior computational analysis of real-world protocols like TLS 1.3 [31], we will employ the PRF-ODH assumption [15], [37] on Extract and assume Expand to be a pseudorandom function (with variable output length). See Appendices A.3, A.4 for more details.

2.2.2. Exclusive ownership of signatures. Classical unforgeability of signature schemes—existential (EUF-CMA) or strong (SUF-CMA)—ask for signatures to be unforgeable for any message of an adversary's choice, wrt. some *fixed* and *honestly generated* public key pk. In general, unforgeability does not imply that it is difficult to find a *different, adversarially-chosen* public key pk' under which some honest message-signature pair (m, σ) *also* verifies. Indeed, such attacks are sometimes easy and in the literature referred to as "duplicate-signature key selection" (DSKS) or "exclusive ownership" attacks [11], [42], [48], [51]. Such an attack vector would be problematic for EDHOC, as it interferes with its approach to deduce the peer's identity through "trial verification" against multiple, non-uniquely identified credentials.

$\underline{\text{INITIALIZE}()}$
$(sk, pk) \leftarrow \mathfrak{S}.\mathsf{KGen}$
$\mathcal{M} \leftarrow \emptyset$
return pk

$\underline{\text{SIGN}(m)}$
$\sigma \xleftarrow{\$} \mathfrak{S}.\mathsf{Sign}(sk, m)$
$\mathcal{M} \leftarrow \mathcal{M} \cup \{(m, \sigma)\}$
return σ

$\underline{\text{FINALIZE}(m, m', \sigma, pk')}$
return $\begin{pmatrix} (m,\sigma) \in \mathcal{M} \\ \wedge\ pk \neq pk' \\ \wedge\ \mathfrak{S}.\mathsf{Vf}(pk', m', \sigma) = 1 \end{pmatrix}$

Figure 1. The strong universal exclusive ownership (S-UEO) game $\mathbf{G}^{\text{S-UEO}}$ for a signature scheme $\mathfrak{S}$.

In our analysis, we therefore rely on the following additional security property of signature schemes introduced as *strong universal exclusive ownership* by [14] when establishing this property for Ed25519.

Definition 2.1 (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 the following advantage of $\mathcal{A}$ in the game $\mathbf{G}^{\text{S-UEO}}$ defined in Figure 1 is small:*

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

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) [11], [48]. 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 [24] for further details.

3. EDHOC and Its SIG-SIG Mode

The EDHOC protocol is a lightweight authenticated key exchange that enables constrained devices to establish a shared session key that is secret and mutually authenticated. Its lightweight operations and very compact messages target, for instance, Internet of Things (IoT) devices operating in low-bandwidth environments such as LoRaWAN [46]. The primary goal of EDHOC is to establish a security context for the OSCORE protocol [55], i.e., key material for constrained application-layer end-to-end encryption, but also allows deriving keys for other applications.

EDHOC specifies four authentication modes, depending on whether the initiator resp. responder authenticates itself through signatures (SIG) or static Diffie–Hellman keys (STAT). In this work, we focus on the SIG-SIG mode of EDHOC which is inspired by the SIGMA ("SIGn-and-MAc") family of key exchange protocols of Krawczyk [43]. The SIGMA design involves an unauthenticated, ephemeral Diffie–Hellman key exchange that is authenticated through signatures and MACs sent by both peers. It is the basis of widely deployed protocols like the Internet Key Exchange (IKE) protocol [35], [40] and the Transport Layer Security (TLS) protocol [52].

To save on bandwidth, EDHOC's SIG-SIG mode follows the *"MAc-then-SIGn"* version of SIGMA [43, Sec-

tion 5.4].[1] Here, instead of sending first a signature and then the MAC (covering the signature), the MAC tag is put "under" the signature and recomputed locally by the receiver, thereby avoiding the need to send the MAC. In addition, EDHOC assumes that devices usually store the credentials of peers (e.g., X.509 certificates) locally. Further bandwidth savings (compared to, e.g., TLS 1.3) can then be achieved by sending only a short *credential identifier* kid rather than the credential itself. Notably, these credential identifiers *need not be unique*:

> applications MUST NOT assume that 'kid' values are unique and several keys associated with a 'kid' may need to be checked [by the recipient] before the correct one is found. [56, Section 3.5.3]

EDHOC is a self-negotiating protocol, meaning participants agree on the authentication mode and the further cryptographic components (the so-called "cipher suite") within the first two protocol messages. We do not capture negotiation here, but focus on the SIG-SIG mode of authentication and assume participants agree on a cipher suite (defining the to-be-used algorithms for authenticated encryption, hashing, DH key exchange, signatures, etc.), omitting the corresponding values from the protocol description.

3.1. Protocol Details

The EDHOC protocol consists of three mandatory messages[2] (msg_1, msg_2, msg_3) exchanged between the initiator I and the responder R. The EDHOC SIG-SIG protocol flow is illustrated in Figure 2 and goes like this:

EDHOC message 1. The initiator begins by sampling an ephemeral Diffie–Hellman secret x and forms its DH share $G_x = xG$. It sends G_x together with a connection identifier C_I and optional external authorization data ead_1.[3]

EDHOC message 2. Upon receiving msg_1, the responder also generates an ephemeral Diffie–Hellman secret y. It computes its DH share $G_y = yG$ and the shared DH secret $G_{xy} = yG_x$. From the shared DH secret, it derives EDHOC's core secret value $\mathsf{PRK}_{2e} = \mathsf{Extract}(\mathsf{th}_2, G_{xy})$ from which all further keys will be derived, using the HKDF Extract function [44].

To authenticate itself, the responder first computes a MAC tag τ_2 via HKDF Expand, keyed width PRK_{2e}, covering in particular its credential identifier kid_R, the hashed transcript so far th_2, and its credential cred_R. It then signs the same values together with τ_2, obtaining a signature σ_2. The signature together with kid_R and further optional external authorization data ead_2 form a plaintext ptxt_2, which is XOR-encrypted into ctxt_2 with a keystream K_2 derived from PRK_{2e} as $\mathsf{Expand}(\mathsf{PRK}_{2e}, (0,$

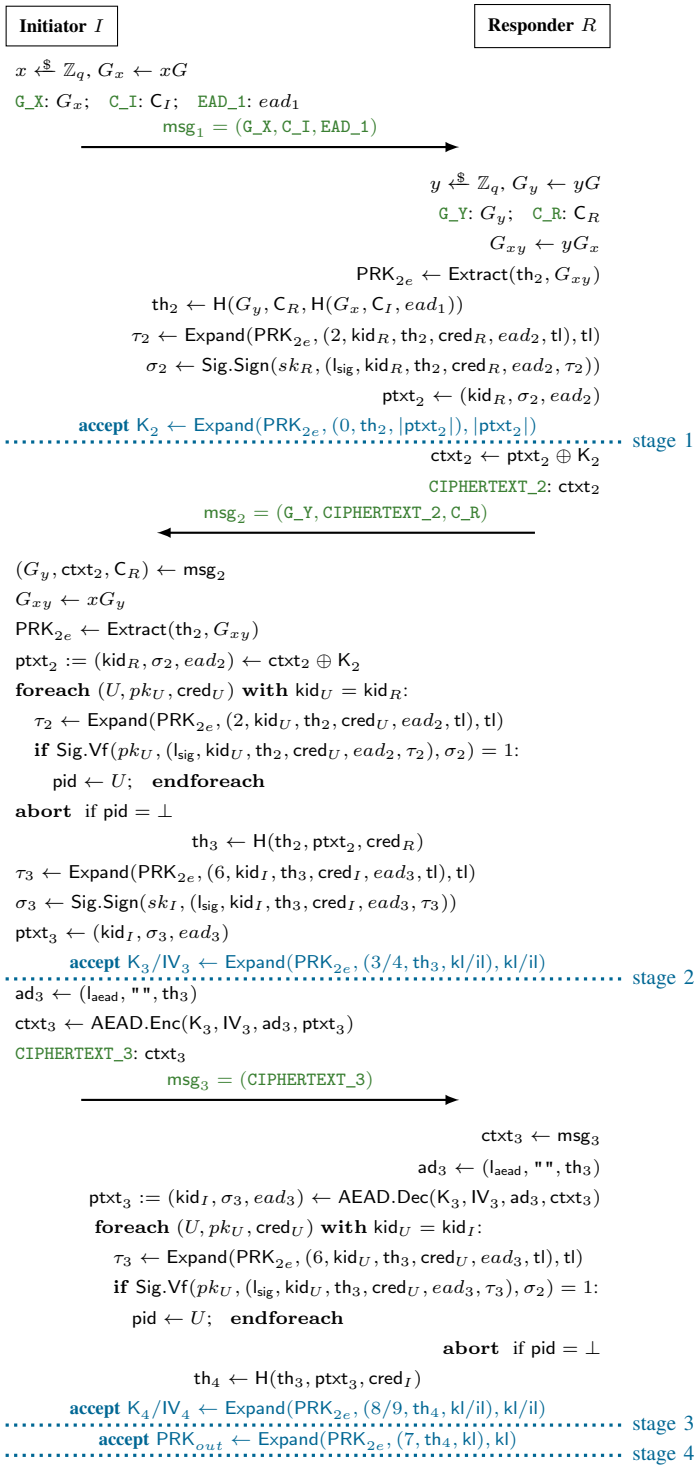

Figure 2. The EDHOC SIG-SIG protocol with three messages. MSG terms highlight message components in the terminology of the EDHOC specification [56] for better reference.

1. This is similar to the IKE design [35], [40], but differs from the more common "SIGn-then-MAc" approach used, e.g., in TLS 1.3 [52].

2. The responder sends a fourth message for key confirmation if EDHOC is used for authentication only and no application data is exchanged. We focus on the three-message case.

3. Applications may send external authorization data (EAD) to "reduce round trips and the number of messages" [56, Section 3.8] by transporting "authorization related data." EAD is opaque to EDHOC (and we treat it as such), but can benefit from the security of keys it is encrypted under—see ead_2 and ead_3 in message 2 and 3 below.

$\mathsf{th}_2, |\mathsf{ptxt}_2|), |\mathsf{ptxt}_2|)$. Finally, the responder sends as second message its DH share G_y, connection identifier C_R, and the ciphertext ctxt_2.

EDHOC message 3. Upon receiving msg_2, the initiator computes $G_{xy} = xG_y$, then derives PRK_{2e} and K_2 to decrypt ctxt_2. It now needs to determine the responder's identity/credential cred_R from the *not necessarily unique* credential identifier kid_R. The EDHOC draft [56] is currently underspecified in how ambiguous identifiers should be handled; we assume a "trial verification" loop is performed: For every identity U with matching $\mathsf{kid}_U = \mathsf{kid}_R$, the initiator computes a trial MAC τ_2 and assumes kid_R identifies U if the received signature σ_2 verifies for the corresponding credential cred_U and τ_2. Note that there might potentially be multiple identities U for which the signature verifies and hence the protocol participants may assume a wrong peer; in our security analysis in Section 5 we will implicitly give the adversary control over the order of trials to capture this.

If the signature does not verify against any matching user, the initiator aborts. Otherwise, the initiator authenticates by producing a MAC tag τ_3 and signature σ_3 similarly to the responder's in msg_2, but for its own credentials and the extended transcript hash $\mathsf{th}_3 = \mathsf{H}(\mathsf{th}_2, \mathsf{ptxt}_2, \mathsf{cred}_R)$. It sends the plaintext $\mathsf{ptxt}_3 = (\mathsf{kid}_I, \sigma_3, ead_3)$, AEAD-encrypted into ctxt_3 using a key $\mathsf{K}_3 = \mathsf{Expand}(\mathsf{PRK}_{2e}, (3, \mathsf{th}_3, \mathsf{kl}), \mathsf{kl})$ and corresponding initialization vector/nonce IV_3 derived from PRK_{2e}. (Here, ead_3 is again optional external authorization data, to be protected under K_3.)

Upon receiving msg_3, the receiver derives $\mathsf{K}_3/\mathsf{IV}_3$ and decrypts ctxt_3. Like the initiator, it performs a trial verification loop to determine the initiator's identity from the possibly ambiguous credential identifier kid_I. After successfully sending/processing msg_3, both parties compute two final keys from PRK_{2e}: key and IV $\mathsf{K}_4/\mathsf{IV}_4$ for optionally sending a fourth EDHOC message for key confirmation in authentication-only mode (which we omit in our analysis), and key PRK_{out} as the final "session key" that is used to derive application-level keys.

Key exporter, the OSCORE context, and key updates. From the established session key PRK_{out}, initiator and responder can derive different application keys as needed. To this end, EDHOC derives an exporter key PRK_{exp} from PRK_{out} with Expand. Any application-specific key is then derived from PRK_{exp} using distinct labels. In particular, for the OSCORE security context, the exporter mechanism is used to derive a master key and master salt.

EDHOC further allows to update PRK_{out} to extend the lifetime of an EDHOC connection while providing forward security. For this, a new session key PRK_{out} is derived by invoking Expand on the old PRK_{out} and a designated key-update label.

The full key schedule of EDHOC in SIG-SIG mode including keys, IVs, the key export and (optional) key update mechanisms are shown in Figure 3.

3.2. Cryptographic Algorithms in EDHOC

Our analysis of EDHOC treats its cryptographic building blocks generically; see Section 2 and Appendix A for their syntax and security definitions. Nevertheless,

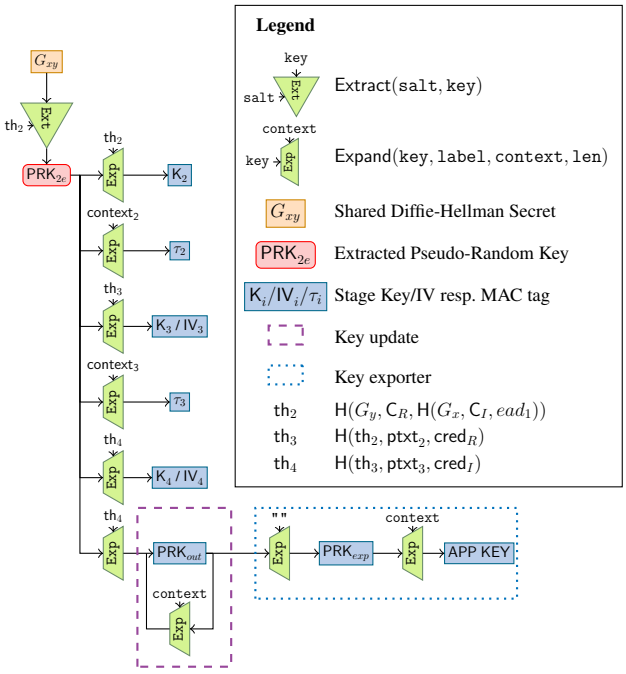

Figure 3. The EDHOC SIG-SIG key schedule, including the key update and exporter mechanism. The transcript hashes (th_2, th_3, th_4) are summarized in the legend; context values ($\mathsf{context}_2$, $\mathsf{context}_3$) as well as labels and output lengths (`label`, `len`) for Expand are given in Figure 2.

in the following we briefly summarize the cryptographic algorithms specified for EDHOC in its cipher suites [56, Section 3.6];

- The hash function H is instantiated with one of SHA2, Shake128, or Shake256. Shake128 and Shake256 are sponge-based extendable output functions (XOF [41]). In our analysis, we require H to be collision resistant.
- The KDF extraction function Extract is instantiated with HKDF.Extract = HMAC [4] when the hash function is SHA2. For hash algorithm Shake128 or Shake256, it is instantiated with KMAC [41] as $\mathsf{Extract}(s, \mathsf{ikm}) = \mathsf{KMAC}(s, \mathsf{ikm}, len, \texttt{""})$ with the desired output length len. In our analysis, we employ the PRF-ODH assumption [15], [37] on Extract.
- The KDF expansion function Expand is instantiated with HKDF.Expand, an iterated application of HMAC [4]. For hash algorithm Shake128 or Shake256, it is instantiated with KMAC as $\mathsf{KMAC}(\mathsf{prk}, info, len, \texttt{""})$. In our analysis, we assume Expand to be a pseudorandom function (with variable output length).
- The signature scheme Sig is instantiated with either Ed25519 [8] or ECDSA [39]. In our analysis, we require strong unforgeability (SUF-CMA) and strong universal exclusive ownership (S-UEO) [14] from the signature.
  We note here that Ed25519 was studied by Brendel et al. [14] and was shown to be SUF-CMA and S-UEO secure, making our results directly applicable to it. In contrast, plain ECDSA is only

EUF-CMA secure and fails to meet SUF-CMA security[4] as well as S-DEO (and hence S-UEO) security [51]. The former can be fixed by making signatures unique, the latter can be mitigated by including the verification key under the signature [51], as done in EDHOC. Formally establishing these properties for ECDSA as used in EDHOC is however beyond the scope of this work.

- The AEAD scheme AEAD is instantiated with one of AES-GCM, AES-CCM, or ChaCha20/Poly1305. For the goals of our analysis, we do not need to make any assumptions on the AEAD scheme beyond correctness.

4. Security Model

We analyze EDHOC in a computational security model in the style of the classical Bellare–Rogaway key exchange model [6], adapted to the multi-stage (MSKE) setting [32], [34], and with our own extensions to capture the specifics of EDHOC. Through the Bellare–Rogaway basis of our model, it captures strong adversaries with full control over the network, able to passively observe and actively modify messages arbitrarily: The adversary can create protocol participants via a NEWUSER oracle and orchestrate protocol sessions (i.e., the execution of the protocol by one party) between these participants via a SEND oracle. The adversary is further allowed to reveal established session keys (through a REVSESSIONKEY oracle) and compromise long-term signing keys (REVLONGTERMKEY).

On a high level, the targeted security guarantees are:

1) Key indistinguishability. An adversary cannot distinguish an established session key from random (via a TEST oracle), as long as it is not trivially compromised ("fresh").
2) Forward security. Keys are indistinguishable from random even if the long-term secrets of involved parties are later compromised.
3) Explicit authentication. When a session accepts with an authenticated peer, there is indeed a corresponding session of that peer.

These guarantees apply to *all* keys established in the protocol and must hold even if other keys in the same sessions are compromised. For example, in EDHOC, an attacker might leverage leakage of the intermediate key K_2 to attack the indistinguishability or authentication of K_3. This is the *multi-stage* aspect of our model, a state-of-the-art concept that has been applied to other modern real-world protocols like TLS 1.3 [31] or Signal [21]. To minimize ambiguity, we give both a high-level description as well as a fully code-based description of our model in the following; the latter is based on the model for TLS 1.3 by Davis et al. [26].

4.1. Capturing EDHOC's Specifics

Recall that, in EDHOC, participant's credentials are identified through short *credential identifier* values kid (cf. Figure 2). As per the underlying COSE standard [54], "applications MUST NOT assume that 'kid' values are unique." In contrast, key exchange models generally assume parties (and their key material) are uniquely identifiable by protocol participants.

To properly capture the non-uniqueness of credential identifiers in EDHOC, in our extension of the MSKE model we grant the adversary additional power when it comes to creating participants in the model: it can *register* users with long-term keys of its choice (as an option in the NEWUSER oracle, drawing inspiration from Boyd et al. [13]) and, most importantly, *specify* their (potentially colliding) credential identifiers. Protocol sessions can then address the true (unique) identities and public keys of other participants through lists $peerpk_{\mathsf{kid}}$ indexed by (non-unique) credential identifiers kid. This mimics EDHOC's process of potentially having to check several candidate credentials matching some identifier kid and allows us to capture the security requirements emerging from it.

4.2. Model Syntax

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

- $\mathsf{KGen}()$ generates long-term signing and verification key pairs for a protocol participant.
- $\mathsf{Activate}(U, i, sk_U, \{\mathsf{pid}\}_U, peerpk, \mathsf{role}) \xrightarrow{\$} (\pi^i_U, m)$ starts a new session π^i_U 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 $\mathsf{role} = initiator$, Activate returns the first protocol message m and $\perp$ otherwise.
- $\mathsf{Run}(\pi^i_U, sk_U, peerpk, m) \xrightarrow{\$} (\pi^i_U, m')$ delivers the protocol message m to the session π^i_U. The message m is processed according to the protocol specification, and π^i_U is updated accordingly. Finally, Run outputs a response message m' or the symbol $\perp$ in case of an error.

4.2.1. Protocol properties. The key exchange protocol KE is augmented with the following variables which will determine its aimed-at security properties:

- $\mathsf{KE}.\mathbf{S}$: the number of stages in the protocol (i.e., first-order keys to be derived).
- $\mathsf{KE}.\mathbf{use}[s]$: whether the s-th stage key is used within the protocol ($internal$) or not.
- $\mathsf{KE}.\mathbf{eauth}[r, s]$: the stage upon whose acceptance a session in role r considers the peer explicitly authenticated in stage s.[5]
- $\mathsf{KE}.\mathbf{fs}[s]$: whether stage s is forward secure.

4.2.2. Session variables. The i-th session owned by user U is denoted by π^i_U. Each session holds, among others, the following variables:

- $\pi^i_U.\mathsf{pid}$: the identity of the intended peer.
- $\pi^i_U.\mathsf{role}$: the role of the session owner.
- $\pi^i_U.\mathsf{stage}$: the current execution stage.
- $\pi^i_U.\mathsf{status}[s]$: the state of execution of stage s.

4. For an ECDSA signature $\sigma = (r, s) \in \mathbb{F}_q^2$, on m, $(r, -s)$ is also a valid signature on m.

5. This in particular captures "retroactive" authentication [31]: E.g., $\mathbf{eauth}[resp, 1] = 3$ encodes that the stage-1 key accepted by a responder will be explicitly authenticated once stage 3 is reached.

$G_{\mathcal{A}}^{\text{MSKE}}(\mathsf{KE})$

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

NEWUSER(sk, pk, 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** (sk, pk) is valid key pair :
7 : / only valid key pairs allowed, i.e., sk must match pk (but beyond that is adversarially generated)
8 : $(sk_U, pk_U) \leftarrow (sk, pk)$
9 : $revltk_U \leftarrow time$
10 : / adversarially-registered keys 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$(U, i, sk_U, \{\mathsf{pid}\}_U, peerpk, \mathsf{role})$
1 : $time \leftarrow time + 1$
2 : **if** $\pi_U^i \neq \perp$: **return** $\perp$
3 : $(\pi_U^i, m) \xleftarrow{\$} \mathsf{Activate}(U, i, sk_U, \{\mathsf{pid}\}_U, peerpk, \mathsf{role})$
4 : $\pi_U^i.\mathsf{id} \leftarrow U$
5 : $\pi_U^i.\mathsf{role} \leftarrow role$
6 : **return** m

SEND(U, i, m)
1 : $time \leftarrow time + 1$
2 : **if** $\pi_U^i = \perp$: **return** $\perp$
3 : $(\pi_U^i, m') \xleftarrow{\$} \mathsf{Run}(\pi_U^i, sk_U, peerpk, m)$
4 : $s \leftarrow \pi_U^i.\mathsf{stage}$
5 : **if** $\pi_U^i.\mathsf{status}[s] = accepted$:
6 : $\pi_u^i.accepted[s] \leftarrow time$
7 : **if** $b = 0$ **and** $\mathsf{KE.use}[s] = internal$: / Random world: if key is used internally...
8 : $\exists \pi_V^j : (\pi_U^i, \pi_V^j) \in \mathcal{P}_s$ **and** $\pi_V^j.\mathsf{tested} = \mathsf{true}$: / and partnered session was tested
9 : $\pi_U^i.\mathsf{key}[s] \leftarrow \pi_V^j.\mathsf{key}[s]$ / copy the key from the partner for consistency
10 : **return** $(\pi_U^i.\mathsf{status}[s], m')$

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

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

TEST(U, i, s)
1 : $time \leftarrow time + 1$
2 : **if** $\pi_U^i = \perp$ **or**
3 : $\pi_U^i.\mathsf{status}[s] \neq accepted$ **or**
4 : $\pi_U^i.\mathsf{tested}[s] = \mathsf{true}$:
5 : **return** $\perp$
6 : **if** $\exists \pi_V^j : (\pi_U^i, \pi_V^j) \in \mathcal{P}_s$ **and** / π_V^j is partnered to π_U^i and...
7 : $\mathsf{KE.use}[s] = internal$ **and** / the key is used internally and...
8 : $\pi_V^j.\mathsf{status}[s+1] \neq \perp$: / the partnered already proceeded to the next stage
9 : **return** $\perp$ / reject the request (since the stage-s key may have been used already)
10 : $\pi_U^i.\mathsf{tested}[s] \leftarrow \mathsf{true}$
11 : $\mathcal{T}_s \leftarrow \mathcal{T}_s \cup \{\pi_U^i\}$
12 : $k_0 \xleftarrow{\$} \mathcal{K}_i$
13 : $k_1 \leftarrow \pi_U^i.\mathsf{key}[s]$
14 : **if** $b = 0$ **and** $\mathsf{KE.use}[s] = internal$: / Random world: if key is used internally
15 : $\pi_U^i.\mathsf{key}[s] \leftarrow k_b$ / 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. The multi-stage key exchange security game for a key exchange protocol KE. The predictates Sound, ExplicitAuth and Fresh are given in Figure 5.

- $\pi_U^i.\mathsf{key}[s]$: the session key of stage s.
- $\pi_U^i.\mathsf{revealed}[s]$, $\pi_U^i.\mathsf{accepted}[s]$, $\pi_U^i.\mathsf{tested}[s]$: the time at which the s-th stage key was revealed, accepted, resp. tested in the game.
- $\pi_U^i.\mathsf{sid}[s]$, $\pi_U^i.\mathsf{cid}[r, s]$: the session and contributive identifiers of stage s (and role r), explained next.

4.2.3. Session and contributive identifiers. We use *session identifiers* [5] to define when two sessions are considered *partnered*, namely if they hold the same session identifier at a given stage. Partnering in turn is used to exclude trivial winning conditions in our model, for instance, an adversary testing and revealing two partnered sessions. A session records its session identifier for stage s in $\pi_U^i.\mathsf{sid}[s]$.

Furthermore, we use *contributive identifiers* [30] to specify the values a session must have honestly received before allowing the adversary to test a stage without authenticated peer. Contributive identifers hence let the key exchange model capture the (passive) security of unauthenticated keys. The session variable $\pi_U^i.\mathsf{cid}[r, s]$ holds the contributive identifier for the role r session in the protocol run, for stage s. Let $\overline{r}$ denote the role opposite to r, then $\pi_U^i.\mathsf{cid}[\overline{r}, s]$ contains the values that the session π_U^i in role r should have honestly received to allow testing it if stage s is not authenticated.

4.2.4. Game variables. In addition to the protocol properties and session variables, the security game tracks the following game-specific variables:

- $\mathcal{T}_s$: the set of all sessions that $\mathcal{A}$ tested in stage s.
- $\mathcal{R}_s$: the set of all sessions for which $\mathcal{A}$ revealed the s-th stage key.
- $\mathcal{P}_s$: the set of sessions partnered in stage s, eval-

Fresh

1 : / The same session was tested and revealed in stage s
2 : **if** $\exists s : \mathcal{T}_s \cap \mathcal{R}_s \neq \emptyset$ **then**
3 : **return** false
4 : / Partnered sessions...
5 : **if** $\exists s : \mathcal{P}_s \cap \mathcal{R}_s \times \mathcal{T}_s$ **then**
6 : / one tested, one revealed in stage s
7 : **return** false
8 : / Forward-secure stages are allowed to be tested unless...
9 : **if** $\exists s, \pi_U^i \in \mathcal{T}_s : \mathsf{KE.fs}[s] = fs$
10 : / they accepted after peer compromise and...
11 : $\wedge\ (revltk_{\pi_U^i.\mathsf{pid}} < \pi_U^i.\mathsf{accepted}[s])$
12 : / they do not have a contributive partner
13 : $\wedge\ \forall \pi_V^j : \pi_U^i.\mathsf{cid}[\overline{\pi_U^i.\mathsf{role}}, s] \neq \pi_V^j.\mathsf{cid}[\overline{\pi_U^i.\mathsf{role}}, s]$
14 : **return** false
15 : / Unauthenticated stages are allowed to be tested unless...
16 : **if** $\exists s, \pi_U^i \in \mathcal{T}_s : \mathbf{eauth}[\overline{\pi_U^i.\mathsf{role}}, s] = \bot$
17 : / they do not have a contributive partner
18 : $\wedge\ \forall \pi_V^j : \pi_U^i.\mathsf{cid}[\overline{\pi_U^i.\mathsf{role}}, s] \neq \pi_V^j.\mathsf{cid}[\overline{\pi_U^i.\mathsf{role}}, s]$
19 : **return** false
20 : **return** true

ExplicitAuth

1 : / Explicit authentication requires that, for all sessions and stages s...
2 : $\forall(\pi_U^i, s, s') : \pi_U^i.\mathsf{accepted}[s] \wedge$
3 : $\mathbf{eauth}[\overline{\pi_U^i.\mathsf{role}}, s] = s' < \infty$ / that should achieve expl. auth. at stage s'...
4 : $\wedge\ \pi_U^i.\mathsf{accepted}[s'] < revltk_{\pi_U^i.\mathsf{pid}}$ / and accepted stage s' before peer compromise, ...
5 : $\implies \exists \pi_V^j$: / there exists a session...
6 : $\pi_U^i.\mathsf{pid} = V$ / owned by V, the peer that π_U^i considers communicating with, and...
7 : $\wedge\ \pi_U^i.\mathsf{sid}[s'] = \pi_V^j.\mathsf{sid}[s']$ / partnered with π_U^i in stage s', and...
8 : / if π_V^j accepts stage s before U is comprised, partnered with π_U^i also in stage s.
9 : $\wedge\ \pi_V^j.\mathsf{accepted}[s] < revltk_{\pi_U^i.\mathsf{id}} \implies \pi_U^i.\mathsf{sid}[s] = \pi_V^j.\mathsf{sid}[s]$

Sound

1 : / More than two sessions are partnered in statge s
2 : **if** $\exists s, \pi_U^i, \pi_V^j, \pi_W^k : (\pi_U^i, \pi_V^j) \in \mathcal{P}_s \wedge$
3 : $(\pi_U^i, \pi_W^k) \in \mathcal{P}_s \wedge (\pi_V^j, \pi_W^k) \in \mathcal{P}_s$ **then**
4 : **return** false
5 : / Partnered sessions...
6 : **if** $\exists s, (\pi_U^i, \pi_V^j) \in \mathcal{P}_s$:
7 : $\wedge\ (\pi_U^i.\mathsf{accepted}[s] \wedge \pi_V^j.\mathsf{accepted}[s])$
8 : $\wedge\ (\pi_U^i.\mathsf{key}[s] \neq \pi_V^j.\mathsf{key}[s])$ **then**
9 : / have different keys
10 : **return** false
11 : / Partnered sessions...
12 : **if** $\exists s, (\pi_U^i, \pi_V^j) \in \mathcal{P}_s$:
13 : $(\pi_U^i.\mathsf{role} = \pi_V^j.\mathsf{role})$ **then**
14 : / in the same role
15 : **return** false
16 : / Partnered sessions...
17 : **if** $\exists s, (\pi_U^i, \pi_V^j) \in \mathcal{P}_s, r \in \{init, resp\}$:
18 : $\pi_U^i.\mathsf{cid}[r, s] \neq \pi_V^j.\mathsf{cid}[r, s]$ **then**
19 : / do not agree on contributive identifiers
20 : **return** false
21 : / Partnered sessions...
22 : **if** $\exists s, (\pi_U^i, \pi_V^j) \in \mathcal{P}_s$:
23 : $\pi_U^i.\mathsf{pid} \neq \bot \neq \pi_V^j.\mathsf{pid} \wedge$ / upon authentication/setting pid...
24 : $(\pi_U^i.\mathsf{pid} \neq V \vee \pi_V^j.\mathsf{pid} \neq V)$ **then**
25 : / set the wrong peer identity
26 : **return** false
27 : / Session identifiers...
28 : **if** $\exists s \neq t, \pi_U^i, \pi_V^j$:
29 : $(\pi_U^i.\mathsf{sid}[s] = \pi_V^j.\mathsf{sid}[t])$ **then**
30 : / collide across different stages
31 : **return** false
32 : **return** true

Figure 5. The predicates Fresh, ExplicitAuth, and Sound used in the MSKE game (Figure 4).

uated dynamically as

$$\mathcal{P}_s = \left\{(\pi_U^i, \pi_V^j) : \pi_U^i.\mathsf{sid}[s] = \pi_V^j.\mathsf{sid}[s]\right\}.$$

- $users$: the current number of users in the game.
- $time$: a discrete value used to order queries/events in the game.
- $revltk_U$: the time at which the long-term secret of U was compromised; set to ∞ by default.
- $peerpk_{\mathsf{kid}}$: the set of all credentials identified by some credential identifier kid.

4.3. Adversary Model and Goal

Adversary $\mathcal{A}$ interacts with the protocol KE through a security game $\mathbf{G}^{\mathrm{MSKE}}(\mathsf{KE})$ with the following oracles. We summarize the oracles' main functionality here and give their detailed, code-based definition in Figure 4.

- NEWUSER(sk, pk, kid). Register a new user U (with honestly generated keys if $pk = \bot$, else adversarially-controlled keys) and credential identifier kid; add $\{(U, pk_U)\}$ to $peerpk_{\mathsf{kid}}$.
- NEWSESSION$(U, i, sk_U, \{\mathsf{pid}\}_U, peerpk, \mathsf{role})$. Create and activate a new session π_U^i.
- SEND(U, i, m). Let π_U^i process message m and return the response to $\mathcal{A}$.
- REVSESSIONKEY(U, i, s). Reveal the session key $\pi_U^i.\mathsf{key}[s]$ to $\mathcal{A}$ and mark it as revealed.
- REVLONGTERMKEY(U). Reveal the long-term signing key sk_U of U to $\mathcal{A}$ and mark U as compromised.
- TEST(U, i, s). Depending on the game's challenge bit b, return either the real session key $\pi_U^i.\mathsf{key}[s]$ or a randomly sampled key. (The oracle ensures consistency across multiple TEST queries and of $internal$-use keys.)

4.4. Security

The adversary's goal in the security game $\mathbf{G}^{\mathrm{MSKE}}$ is to violate the protocol's

1) soundness, negating a predicate Sound which checks that the protocol-specified session identifiers correctly capture partnering (e.g., that partnered session derive the same keys, that at most two sessions are partnered, etc.),

   or

2) explicit authentication, negating a predicate ExplicitAuth which, in essence, checks that any session that accepts a stage s and is promised

explicit authentication as per $\mathbf{eauth}$ indeed has an honest partner in stage s (unless the involved parties were compromised prior to accepting),

or

3) key indistinguishability, by correctly guessing the challenge bit b after testing only *fresh* sessions; freshness is encoded via a predicate Fresh which checks that tested sessions are not trivially revealed, their forward-security conditions are met, and when unauthenticated, they have an honest contributive-identifier partner.

See Figure 5 for the full, code-based definition of the predicates Sound, $\mathrm{ExplicitAuth}$, and Fresh.

Definition 4.1 (Multi-stage key exchange security)**.** *Let* KE *be a key exchange protocol. Let* $\mathbf{G}^{\mathrm{MSKE}}(\mathsf{KE})$ *be the MSKE game defined above and formalized in Figure 4. We define the advantage of an MSKE adversary* $\mathcal{A}$ *against* KE *as:*

$$\mathsf{Adv}^{\mathrm{MSKE}}_{\mathcal{A}}(\mathsf{KE}) = 2 \cdot \Pr\big[\mathbf{G}^{\mathrm{MSKE}}(\mathsf{KE}) \to 1\big] - 1.$$

5. Security Analysis

We are now ready to analyze the EDHOC SIG-SIG protocol in the security model of Section 4.

5.1. Protocol Properties

For our analysis, we first need to specify the protocol's targeted properties: its stages, how keys are used, when explicit authentication is expected, and whether stages are forward secure.

Stages. EDHOC consists of $\mathbf{S} = 4$ stages. These correspond to establishing the keys (and potentially associated IVs) K_2, $\mathsf{K}_3/\mathsf{IV}_3$, $\mathsf{K}_4/\mathsf{IV}_4$, resp. PRK_{out}.

Key usage. The first three stage keys (and IVs) K_2, $\mathsf{K}_3/\mathsf{IV}_3$, and $\mathsf{K}_4/\mathsf{IV}_4$ are used internally within the protocol to encrypt EDHOC messages. In contrast, we will show that PRK_{out} is fit for external use, e.g., to protect application data, as intended. I.e., $\mathbf{use} = [\mathit{internal}, \mathit{internal}, \mathit{internal}, \mathit{external}]$.

Explicit authentication. For initiator sessions, stages 2, 3, and 4 are explicitly authenticated upon acceptance of stage 2; stage 1 then receives explicit authentication retroactively. For responder sessions, the peer is explicitly authenticated upon acceptance of stage 3; hence, stages 3 and 4 are explicitly authenticated upon acceptance of stage 1, while stages 1 and 2 receive explicit authentication retroactively.

Formally, for a given role r and stage s, we define $\mathbf{eauth}[r, s]$ as:

$$\forall s \in [1,4]: \quad \begin{aligned} &\mathbf{eauth}[\mathit{init}, s] = 2, \\ &\mathbf{eauth}[\mathit{resp}, s] = 3. \end{aligned}$$

Forward security. Through the ephemeral Diffie–Hellman shares freshly sampled by both participants in each run of the protocol run and keys derived from them, all four stages are forward secure: $\mathbf{fs} = [\mathit{fs}, \mathit{fs}, \mathit{fs}, \mathit{fs}]$.

Session identifiers. The session identifier for stage s is a tuple $(\text{“}s\text{”}, \mathbf{tx}_s, \mathbf{auth}_s)$, where “$s$” serves as unique label, $\mathbf{tx}_s$ is the plaintext message transcript containing elements that enter the key schedule, and $\mathbf{auth}_s$ is the (potentially empty) list of 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 for the responder's identity.

Concretely, the session identifiers sid for $s \in [1,4]$ are defined as follows:

$$\begin{aligned}
\mathsf{sid}[1] &= (\text{“1”}, G_x, \mathsf{C}_I, \mathit{ead}_1, G_y, \mathsf{C}_R), \\
\mathsf{sid}[2] &= (\text{“2”}, G_x, \mathsf{C}_I, \mathit{ead}_1, G_y, \mathsf{C}_R, \mathsf{kid}_R, \sigma_2, \mathit{ead}_2, R), \\
\mathsf{sid}[3] &= (\text{“3”}, G_x, \mathsf{C}_I, \mathit{ead}_1, G_y, \mathsf{C}_R, \mathsf{kid}_R, \sigma_2, \mathit{ead}_2, \\
&\qquad \mathsf{kid}_I, \sigma_3, \mathit{ead}_3, R, I), \\
\mathsf{sid}[4] &= (\text{“4”}, G_x, \mathsf{C}_I, \mathit{ead}_1, G_y, \mathsf{C}_R, \mathsf{kid}_R, \sigma_2, \mathit{ead}_2, \\
&\qquad \mathsf{kid}_I, \sigma_3, \mathit{ead}_3, R, I).
\end{aligned}$$

Contributive identifiers. The contributive identifier for a stage s corresponds to the values that a session π must have *honestly* received (i.e., untampered) from a peer session to allow testing π in the *unauthenticated* stage s. Such testing is then allowed, even when other message parts are not or only partially delivered to either party involved in that protocol run. To allow the adversary to test as many sessions as possible, we shall choose the entries in the contributive identifiers to be minimal.

For a session π in the role $\mathsf{role} \in \{\mathit{init}, \mathit{resp}\}$, let $\overline{\mathsf{role}}$ denote the opposite role. The contributive identifier $\pi.\mathsf{cid}[\overline{\mathsf{role}}, s]$ captures the messages that π must have received honestly from its peer as a prerequisite to allow testing π in stage s, if s is unauthenticated.

For EDHOC, we have the initiator (resp. responder) set $\mathsf{cid}[\mathit{init}, 1]$ to $(\text{“1”}, G_x)$ upon sending (resp. receiving) message 1, which captures that an initiator must have contributed a 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 $\mathsf{cid}[\mathit{resp}, 1]$ to $(\text{“1”}, G_x, G_y)$ upon receiving (resp. sending) message 2. This captures that the responder must have contributed its G_y share before a legitimate test query against an initiator session is allowed (without authentication). For all other stages $s \in \{2,3,4\}$, $\mathsf{cid}[\mathit{init}, s] = \mathsf{cid}[\mathit{resp}, s] = (\text{“}s\text{”}, G_x, G_y)$.

In summary:

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

5.2. Security Result

For EDHOC SIG-SIG, we establish the following security theorem, which bases the protocol's MSKE security on the used hash function's collision resistance (CR), the signature's unforgeability (SUF-CMA) and strong universal exclusive ownership (S-UEO), the PRF-ODH [15] security of $\mathsf{Extract}$, and the PRF security of Expand. We give the main proof steps here, focusing on the technically challenging bits when establishing explicit authentication despite EDHOC's use of non-unique credential identifiers; the full proof can be found in Appendix B.

Theorem 5.1 (MSKE security of EDHOC SIG-SIG)**.** *Let* EDHOC-Sig-Sig *be the EDHOC SIG-SIG protocol as defined in Section 3, using a cyclic group* $\mathbb{G}$ *of order* q*. Let* $\mathcal{A}$ *be an* MSKE *adversary against* EDHOC-Sig-Sig*, interacting with at most* n_U *users and* n_S *sessions. Then we can construct 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*

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

Proof (main steps). The proof proceeds via a series of games starting with the MSKE game as defined in Section 4 and ending with games where the adversary has zero advantange. Each game hop introduces a slight variation; bounding the advantage difference introduced by those variations yields the stated theorem bound. (See the full proof in Appendix B for detailed reductions for all advantage bounds.) The high-level strategy is to first ensure that soundness holds ($\mathrm{Sound} = \mathsf{true}$), then to split into two disjoint cases: Branch I treats the case that the adversary breaks explicit authentication ($\mathrm{ExplicitAuth}$), Branch II shows that the adversary cannot guess the challenge bit when satisfying the Fresh condition. In the explicit authentication branch of the proof, we carefully analyze the effect of non-unique credential identifiers, which is the most critical part of the proof. We show that the specific usage of signatures in EDHOC provides S-UEO security, thereby preventing attacks on explicit authentication that would exploit the ambiguity of credential identifiers.

GAME $\mathbf{G}_0$. The unmodified MSKE game.

GAMES $\mathbf{G}_1$/$\mathbf{G}_2$. We introduce a "bad event", aborting the game whenever two honest sessions sample the same DH key shares. By the birthday bound, this bad event happens with probability at most ${n_S}^2/q$. We argue (in Appendix B.1) that if $\mathbf{G}_2$ does not abort (i.e., if no DH shares collide), then the adversary $\mathcal{A}$ cannot cause the Sound predicate to become false by definition of the session identifers.

GAMES $\mathbf{G}_3$/$\mathbf{G}_4$. We log all hash function computations done by honest sessions and abort the game if a hash collision occurs. The probability of this happening translates into a reduction $\mathcal{B}_4$ breaking H's collision resistance.

At this point, we split the proof into two disjoint branches, I and II, each starting from Game $\mathbf{G}_4$. In Branch I, the adversary attempts to break explicit authentication for at least one session. In Branch II, the adversary attempts to violate key indistinguishability by guessing the challenge bit, assuming explicit authentication is not violated. The adversary's advantage in $\mathbf{G}_4$ is then bounded by the sum of its advantage in the two branches.

Branch I. Ensuring explicit authentication. We first treat explicit authentication.

GAME $\mathbf{G}_I$. Continuing from $\mathbf{G}_4$, we first guess at random a session $\pi^i_U \in [1..n_S]$ and stage $s \in [1..4]$ for which $\mathcal{A}$ breaks explicit authentication, introducing a factor $4n_S$. In the following, we refer to π^i_U as the *target* session.

GAMES $\mathbf{G}_{I.1}$/$\mathbf{G}_{I.2}$. We abort the game if the target session π^i_U receives a message-signature pair which is valid under the public key of a non-corrupted user but was not produced by an honest session of that user. This would constitute a SUF-CMA forgery and hence can be bound by $n_U \cdot \mathsf{Adv}^{\text{SUF-CMA}}_{\mathsf{Sig}}(\mathcal{B}_{I.2})$ via a reduction $\mathcal{B}_{I.2}$ that first guesses the peer user of π^i_U and outputs the forgery when it occurs.

ON SUF-CMA SECURITY. EDHOC including signatures in the key derivation means any modification to signatures implies different keys computed by initiator and responder. As soundness requires same session identifiers implying same keys, we consequently include the signatures in the session identifiers. Explicit authentication then requires agreement on these signatures (as a modification would lead to non-partnered sessions, violating explicit authentication), which brings up SUF-CMA here.[6]

We remark that since the initiator-to-responder signature is AEAD-protected within msg_3, one might be able to leverage AEAD integrity in this direction. Our analysis however intentionally does not rely on AEAD security but allows revealing of internal keys like K_2, K_3, capturing key independence [32]. These keys enable protection of identities and EAD; modeling them as stage keys facilitates a composable treatment [17], [26], [30].

In the following games, we carefully analyze the consequences on authentication due to *non-unique* credential identifiers. More precisely, we show that an honest session will not set a wrong peer identifier.

GAME $\mathbf{G}_{I.3}$. In this game, we set a flag $sig_{ambigous}$ if a session accepts a message-signature pair (m', σ) verifying under a public key $pk_{U'}$ such that: (1) there exists an uncompromised user U and both pk_U and $pk_{U'}$ are identified by the same credential identifier kid, and (2) an honest session π^i_U of U produced the signature σ on some message m. Note that $pk_{U'}$ is a key that is potentially chosen and registered maliciously by $\mathcal{A}$.

GAME $\mathbf{G}_{I.4}$. We now abort the game whenever $sig_{ambigous}$ is set. We first observe that each session signs a message that includes the user's credentials cred_U, which uniquely identifies the user identity U. Therefore, it must be that $m' \neq m$, as the accepted message m' must have included U', but the honest session π^i_U would only sign a message including U. Additionally, we can restrict our analysis to $pk_U \neq pk_{U'}$: if $pk_U = pk_{U'}$ uncompromised, m would have been a forgery caught in the prior games.

By definition of $sig_{ambigous}$, we can now directly relate $\Pr[sig_{ambigous} \leftarrow \mathsf{true}]$ to the advantage of an S-UEO adversary $\mathcal{B}_{I.4}$. More precisely, $\mathcal{B}_{I.4}$ associates pk_U with the challenge public key pk^* received in the S-UEO game, i.e., set $pk_U = pk^*$. The reduction uses the S-UEO signing oracle whenever it needs to sign a message on behalf of U; for all other users it, picks the key itself and answers oracles in the usual manner. If

6. Note that, in contrast to SIGMA which only requires EUF-CMA security, EDHOC has the MAC "under" the signature, meaning the former cannot guarantee integrity of the latter.

$sig_{ambigous}$ is set for a public key $pk_{U'}$ and message-signature pair (m', σ), $\mathcal{B}_{I.4}$ outputs $(pk_{U'}, m')$ to win the S-UEO game.

Conclusion of Branch I. At this point, we argue that if $\mathbf{G}_{I.4}$ does not abort, then the adversary cannot win by causing the predicate $\mathrm{ExplicitAuth}$ to evaluate to false. Let us recall what it means for explicit authentication to be violated for a session π_U^i and stage s which should be explicitly authentication once stage $s' = \mathbf{eauth}[\overline{\pi_U^i.\mathsf{role}}, s]$ is reached. For this, π_U^i must have accepted stage s, and accepted stage s' while its peer $V = \pi_U^i.\mathsf{pid}$ was not compromised, and one of the following must hold:

(I.a) No (honest) session π_V^j is partnered with π_U^i in stage s'.
(I.e., π_U^i has no stage-s' partner, despite s' giving the explicit authentication.)

(I.b) There exists π_V^j partnered with π_U^i in stage s'; however, the two sessions are not partnered in stage s although π_U^i accepts stage s while U is uncompromised.
(I.e., π_U^i reaches stage s uncompromised (note that possibly $s \geq s'$), but does not have a stage-s partner as promised by explicit authentication.)

Recall from Section 5.1 that session identifiers, determining partnering, include the exchange message transcript as well as the so-far authenticated peers. Non-partnered sessions must hence disagree on one or the other.

The case (I.a) corresponds to one of two attacks: Either, there is no session π_V^j agreeing on the message transcript part of the session identifier; that means the adversary must have forged the signature π_U^i received, but this is excluded through Game $\mathbf{G}_{I.2}$. Or, π_V^j agrees on the transcript, but not on the authenticated identities, i.e., π_U^i accepts with an "erroneous" peer identity. However, this corresponds to the ambiguous signatures ruled out in Game $\mathbf{G}_{I.4}$. Hence, case (I.a) cannot occur anymore at this point.

For case (I.b), we first note that if $s \leq s'$, agreement on $\mathsf{sid}[s']$ implies agreement on $\mathsf{sid}[s]$ as the latter contains a subset of elements of the former. For the later stages $s \in \{3, 4\}$, the disagreement in the session identifier can be traced to either a forged or an ambiguous signature, which are ruled out in Game $\mathbf{G}_{I.2}$ resp. $\mathbf{G}_{I.4}$. So also case (I.b) cannot occur anymore at this point, and hence the explicit authentication properties are guaranteed.

Branch II. Ensuring key indistinguishability. We now turn to the proof branch handling challenge bit guesses of the adversary on fresh, tested sessions.

GAME $\mathbf{G}_{II}$. Continuing from Game $\mathbf{G}_4$, we begin by restricting the adversary $\mathcal{A}$ to a single TEST query only. In the following, π_U^i refers to *the* tested session. Following Dowling et al. [31], the advantage loss can be bounded by a factor at most $n_S \cdot \mathbf{S}$ (the maximum number of TEST queries possible), via a hybrid argument. Here, n_S is the number of sessions and $\mathbf{S} = 4$ is the number of stages. Therefore, we get the following bound:

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

We proceed with our analysis of Branch II by considering two disjoint cases, predicated on whether the tested session has a contributive partner in the first stage or not. We start by analyzing the case with no contributive partner.

GAME $\mathbf{G}_{II.A1}$/$\mathbf{G}_{II.A2}$. We abort the game whenever the tested session π_U^i accepts a signature that verifies under an honest user's public key for some message that was never signed by a session of that user. Similarly to Games $\mathbf{G}_{I.1}$/$\mathbf{G}_{I.2}$ in Branch I, this reduces to the existential[7] unforgeability of the signature scheme, times a factor for guessing the involved peer, i.e., $n_U \cdot \mathsf{Adv}^{\text{EUF-CMA}}_{\mathsf{Sig}}(\mathcal{B}_{II.A2})$.

With forgeries rules out, a tested session only accepts authenticated stages $s \geq 2$ (as initiator) resp. $s \geq 3$ (as responder) if it received a valid signature from an honest session; since this honest session however then agrees on the contributive identifier, the test session has a contributive partner, contradicting the assumption. For unauthenticated stage 1 (and stage 2 for responders), a session without contributive identifier cannot be tested (as otherwise freshness is violated). Hence, at this point, the adversary cannot issue a valid TEST query, leaving it with guessing the challenge bit and $\mathsf{Adv}^{\mathbf{G}_{\mathrm{II.A2}}}(\mathcal{A}) = 0$.

We now turn to π_U^i having a contributive partner.

GAME $\mathbf{G}_{II.B1}$. We start by guessing the contributively partnered session π_V^j, introducing a guessing factor of n_S. From this point on, we know both the tested session and its contributive partner at the outset of the game.

GAME $\mathbf{G}_{II.B2}$. We now replace PRK_{2e} computed by the tested session with a uniform random value $\widetilde{\mathsf{PRK}_{2e}}$. Likewise, we replace PRK_{2e} with $\widetilde{\mathsf{PRK}_{2e}}$ at the contributive partner π_V^j *if* the π_V^j holds the same two DH shares as the tested session. We rely on the $\mathrm{snPRF\text{-}ODH}$ security of $\mathsf{Extract}$ to justify this step, where the challenge DH shares are embedded as the shares of the tested session and its contributive partner, PRK_{2e} is the challenge PRF value, and we use the DH oracle to compute a deviating PRK_{2e} value in case a contributive-partner initiator session receives an adversarially-modified DH share.

GAME $\mathbf{G}_{II.B3}$. Finally, we replace the function Expand keyed with $\widetilde{\mathsf{PRK}_{2e}}$ with a (variable output length) random function F at the tested session (as well as the contributive partner if it shares $\widetilde{\mathsf{PRK}_{2e}}$). We justify this step by the PRF security of Expand. The result of this step is that all keys derived in the tested session are replaced with uniformly random values (and likewise for the partnered session).

At this point, the tested session key is random and independent of the challenge bit. It remains to argue that REVSESSIONKEY queries on non-partnered sessions do not help the adversary. Interestingly, this is not straightforward in EDHOC, again due to the ambiguity of credential identifiers: sessions might agree on the entire message transcript (only including the non-unique credential identifiers), but disagree on the obtained identities. Fortunately,

7. Here we only care about the *messages* (containing the contributive identifier) being signed, not the signatures themselves; hence existential unforgeability suffices in this case.

(since draft version 16) EDHOC in addition to the transcript also hashes the identities into th_3 and th_4, which in turn enter the key derivation and, by Game $\mathbf{G}_4$ do not collide. Hence, non-partnered sessions derive different keys, making RevSessionKey useless and leaving the adversary with no chance to win.

Through the initial games and Branches I and II, we now guaranteed Sound, ExplicitAuth and key indistinguishability. This completes the proof; collecting the bounds gives the result in Theorem 5.1. □

6. Conclusion and Discussion

In this work, we analyzed EDHOC in SIG-SIG mode for authentication and proved its security in a strong, multi-stage model for authenticated key exchange. We gave a security proof EDHOC SIG-SIG, carefully analyzing its authentication guarantees when an attacker is allowed to leverage EDHOC's non-unique credential identifiers. Our analysis also reveals that the "MAc-the-SIGn" variant of the SIGMA protocol [43] can be a bit brittle in terms of security when giving the adversary more control over how signatures are verified.

6.1. Contributions to EDHOC

During our analysis, we provided several recommendations to the IETF LAKE working group that led to constructive and fruitful discussions and have by now been integrated into draft version 17. We provide some further detail on the most notable proposed cryptographic improvements and their consequences for our analysis:

- Dedicated session key. We recommended establishing a dedicated session key instead of reusing the last key-exchange–internal key (K_4) for clear key separation and composable security [17], [34]. Such key was added (PRK_{out}) in draft 14, in agreement with a similar proposal by Jacomme et al. [36].[8] Our analysis confirms that PRK_{out} is a secure "external" key, i.e., can be used securely and independently of the other keys established.
- Transcript hashes. To strengthen EDHOC against potential attacks taking advantage of non-unique credential identifiers, we suggested that the transcript hashes (th_3, th_4) should include the full/unique credentials of the party just authenticated (responder, resp. initiator). This prevents parties from deriving the same keys without agreeing on peer identities, which we leverage in Branch II of our security proof. The proposed change was incorporated in draft 17:[9]

  > By including the authentication credentials in the transcript hash, EDHOC protects against Duplicate Signature Key Selection (DSKS)-like identity misbinding attack that the MAC-then-Sign variant of SIGMA-I is otherwise vulnerable to. [56, Section 8.1]

  We further suggested to build transcript hashes based on the plaintext, not the ciphertext, version of messages (similar to TLS 1.3); a change integrated in draft 14.[10] Our analysis confirms that this avoids depending on integrity properties of the message encryption for key exchange security.
- Key separation in key derivation. We suggested to not reuse keys across HKDF calls of Extract and Expand for key separation; a change executed in draft 14.[11] This enables cleanly applying PRF security of both functions independently.

6.2. Limitations and Open Research Questions

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 affect other authentication methods. In particular, the concern that non-unique credential identifiers can lead to ambiguous signatures is also valid for the STAT-SIG and SIG-STAT modes and should be analyzed in those contexts as well. We focus on the generic security properties of EDHOC's building blocks when proving our results, but not on tightness of our bounds. Indeed, due to various guessing steps in our proof (cf. Section 5), our security bound is rather loose. Tighter bounds are desirable as they meaningfully inform the choice of concrete parameters to instantiate the protocol both securely and efficiently, We anticipate that recent advances in proving tight security for real-world protocols like TLS 1.3 [22], [26], [27], [29] can be applied to EDHOC as well and that to this end the tight analysis of EDHOC's STAT-STAT mode by Cottier and Pointcheval [23] could potentially be combined with our SIG-SIG analysis.

We restricted our analysis to the cryptographic core of EDHOC, striving for an appropriate balance between abstraction and completeness. Hence, we do not capture all aspects of this complex protocol like negotiation of authentication mechanisms and cipher suites or properties like key confirmation, neither do we consider other attack surfaces in low-powered devices (e.g., due to insecure implementations) that may undermine the security guarantee of EDHOC. Tool-based analyses like those of EDHOC by Norrman et al. [49] or Jacomme et al. [36] can paint a more complete picture of protocol interactions across different modes, at a complexity level that is potentially out-of-reach for classical pen-and-paper computational analyses. We view those approaches as complementary, with computational analyses providing insights into the security of lower-level wiring cryptographic building blocks into protocols.

To the best of our knowledge, the OSCORE protocol has not yet received a formal security analysis. Transcribing compositional results by Brzuska et al. [17] for Bellare–Rogaway key exchange and follow-up ones for MSKE [32], our analysis suggests that the final session key PRK_{out} in EDHOC can be securely composed with symmetric-key primitives. This enables a modular security analysis limited to the OSCORE protocol itself which, together with our results, then would give further confidence in the overall protocol to be deployed.

8. https://github.com/lake-wg/edhoc/pull/276
9. https://github.com/lake-wg/edhoc/pull/318
10. https://github.com/lake-wg/edhoc/pull/277
11. https://github.com/lake-wg/edhoc/pull/286

References

[1] Cisco Annual Internet Report (2018–2023) White Paper. https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.

[2] Samsung connected home fridge becomes weapon in MITM attacks. https://www.zdnet.com/article/samsung-connected-home-fridge-becomes-weapon-in-mitm-attacks, August 2015.

[3] National Vulnerability Database CVE-2021-28372. https://nvd.nist.gov/vuln/detail/CVE-2021-28372, August 2021.

[4] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In Neal Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 1–15. Springer, Heidelberg, August 1996.

[5] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 139–155. Springer, Heidelberg, May 2000.

[6] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 232–249. Springer, Heidelberg, August 1994.

[7] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Heidelberg, May / June 2006.

[8] 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.

[9] Karthikeyan Bhargavan, Bruno Blanchet, and Nadim Kobeissi. Verified models and reference implementations for the TLS 1.3 standard candidate. In *2017 IEEE Symposium on Security and Privacy*, pages 483–502. IEEE Computer Society Press, May 2017.

[10] Karthikeyan Bhargavan, Christina Brzuska, Cédric Fournet, Matthew Green, Markulf Kohlweiss, and Santiago Zanella-Béguelin. Downgrade resilience in key-exchange protocols. In *2016 IEEE Symposium on Security and Privacy*, pages 506–525. IEEE Computer Society Press, May 2016.

[11] Simon Blake-Wilson and Alfred Menezes. Unknown key-share attacks on the station-to-station (STS) protocol. In Hideki Imai and Yuliang Zheng, editors, *PKC'99*, volume 1560 of *LNCS*, pages 154–170. Springer, Heidelberg, March 1999.

[12] C. Bormann and P. Hoffman. Concise Binary Object Representation (CBOR). RFC 8949 (Internet Standard), December 2020.

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

[14] Jacqueline Brendel, Cas Cremers, Dennis Jackson, and Mang Zhao. The provable security of Ed25519: Theory and practice. In *2021 IEEE Symposium on Security and Privacy*, pages 1659–1676. IEEE Computer Society Press, May 2021.

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

[16] 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 (SSR 2018)*, volume 11322, pages 21–36. Springer International Publishing, Cham, 2018. Series Title: Lecture Notes in Computer Science.

[17] Christina Brzuska, Marc Fischlin, Bogdan Warinschi, and Stephen C. Williams. Composability of Bellare-Rogaway key exchange protocols. In Yan Chen, George Danezis, and Vitaly Shmatikov, editors, *ACM CCS 2011*, pages 51–62. ACM Press, October 2011.

[18] Matt Burgess. Smart dildos and vibrators keep getting hacked – but Tor could be the answer to safer connected sex. *WIRED UK*, February 2018.

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

[20] Vincent Cheval, Charlie Jacomme, Steve Kremer, and Robert Künnemann. SAPIC+: protocol verifiers of the world, unite! In Kevin R. B. Butler and Kurt Thomas, editors, *USENIX Security 2022*, pages 3935–3952. USENIX Association, August 2022.

[21] Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. A formal security analysis of the signal messaging protocol. *Journal of Cryptology*, 33(4):1914–1983, October 2020.

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

[23] Baptiste Cottier and David Pointcheval. Security Analysis of the EDHOC protocol. https://arxiv.org/abs/2209.03599, September 2022.

[24] 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*, pages 1696–1714. IEEE Computer Society Press, May 2021.

[25] Cas Cremers, Marko Horvat, Jonathan Hoyland, Sam Scott, and Thyla van der Merwe. A comprehensive symbolic analysis of TLS 1.3. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1773–1788. ACM Press, October / November 2017.

[26] Hannah Davis, Denis Diemert, Felix Günther, and Tibor Jager. On the concrete security of TLS 1.3 PSK mode. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 876–906. Springer, Heidelberg, May / June 2022.

[27] Hannah Davis and Felix Günther. Tighter proofs for the SIGMA and TLS 1.3 key exchange protocols. In Kazue Sako and Nils Ole Tippenhauer, editors, *ACNS 21, Part II*, volume 12727 of *LNCS*, pages 448–479. Springer, Heidelberg, June 2021.

[28] Antoine Delignat-Lavaud, Cédric Fournet, Markulf Kohlweiss, Jonathan Protzenko, Aseem Rastogi, Nikhil Swamy, Santiago Zanella-Béguelin, Karthikeyan Bhargavan, Jianyang Pan, and Jean Karim Zinzindohoue. Implementing and proving the TLS 1.3 record layer. In *2017 IEEE Symposium on Security and Privacy*, pages 463–482. IEEE Computer Society Press, May 2017.

[29] 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, July 2021.

[30] Benjamin Dowling, Marc Fischlin, Felix Günther, and Douglas Stebila. A cryptographic analysis of the TLS 1.3 handshake protocol candidates. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015*, pages 1197–1210. ACM Press, October 2015.

[31] 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, October 2021.

[32] Marc Fischlin and Felix Günther. Multi-stage key exchange and the case of Google's QUIC protocol. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014*, pages 1193–1204. ACM Press, November 2014.

[33] Marc Fischlin, Felix Günther, Benedikt Schmidt, and Bogdan Warinschi. Key confirmation in key exchange: A formal treatment and implications for TLS 1.3. In *2016 IEEE Symposium on Security and Privacy*, pages 452–469. IEEE Computer Society Press, May 2016.

[34] Felix Günther. *Modeling Advanced Security Aspects of Key Exchange and Secure Channel Protocols*. Ph.D. Thesis, Technische Universität Darmstadt, 2018.

[35] Dan Harkins and Dave Carrel. The Internet Key Exchange (IKE). IETF RFC 2409 (Proposed Standard), 1998.

[36] Charlie Jacomme, Elise Klein, Steve Kremer, and Maïwenn Racouchot. A comprehensive, formal and automated analysis of the EDHOC protocol. In *32nd USENIX Security Symposium, USENIX Security 2023*, Anaheim, CA, United States, August 2023.

[37] Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk. On the security of TLS-DHE in the standard model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 273–293. Springer, Heidelberg, August 2012.

[38] Jmaxxz. DEF CON 24 - Backdooring the Frontdoor. https://archive.org/details/youtube-MMB1CkZi6t4, October 2022.

[39] 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.

[40] C. Kaufman, P. Hoffman, Y. Nir, P. Eronen, and T. Kivinen. Internet Key Exchange Protocol Version 2 (IKEv2). RFC 7296 (Internet Standard), October 2014. Updated by RFCs 7427, 7670, 8247, 8983.

[41] 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.

[42] Neal Koblitz and Alfred Menezes. Another look at security definitions. *Advances in Mathematics of Communications*, 7(1):1–38, February 2013.

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

[44] Hugo Krawczyk. Cryptographic extraction and key derivation: The HKDF scheme. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 631–648. Springer, Heidelberg, August 2010.

[45] 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.

[46] LoRa Alliance. Long range wide area network (LoRaWAN) specification. https://lora-alliance.org/about-lorawan/.

[47] 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.

[48] Alfred Menezes and Nigel P. Smart. Security of signature schemes in a multi-user setting. *Designs, Codes and Cryptography*, 33(3):261–274, November 2004.

[49] K. Norrman, V. Sundararajan, and A. Bruni. Formal Analysis of EDHOC Key Establishment for Constrained IoT Devices. In *SECRYPT*, 2021.

[50] Kenneth G. Paterson and Thyla van der Merwe. Reactive and proactive standardisation of TLS. In Lidong Chen, David A. McGrew, and Chris J. Mitchell, editors, *Security Standardisation Research (SSR 2016)*, volume 10074 of *Lecture Notes in Computer Science*, pages 160–186, Gaithersburg, MD, USA, December 2016. Springer.

[51] Thomas Pornin and Julien P. Stern. Digital signatures do not guarantee exclusive ownership. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *ACNS 05*, volume 3531 of *LNCS*, pages 138–150. Springer, Heidelberg, June 2005.

[52] E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446 (Proposed Standard), August 2018.

[53] Phillip Rogaway. Authenticated-encryption with associated-data. In Vijayalakshmi Atluri, editor, *ACM CCS 2002*, pages 98–107. ACM Press, November 2002.

[54] J. Schaad. CBOR Object Signing and Encryption (COSE): Structures and Process. RFC 9052 (Internet Standard), August 2022.

[55] G. Selander, J. Mattsson, F. Palombini, and L. Seitz. Object Security for Constrained RESTful Environments (OSCORE). RFC 8613 (Proposed Standard), July 2019.

[56] Göran Selander, John Preuß Mattsson, and Francesca Palombini. Ephemeral Diffie-Hellman Over COSE (EDHOC) – draft-ietf-lake-edhoc-17. https://tools.ietf.org/html/draft-ietf-lake-edhoc-17, October 2022.

[57] Mališa Vučinić, Göran Selander, John Preuss Mattsson, and Thomas Watteyne. Lightweight authenticated key exchange with EDHOC. *Computer*, 55(4):94–100, 2022.

A. Cryptographic Primitives

A.1. Hash Functions

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.

Definition A.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 $\mathbf{G}^{\mathrm{CR}}_{\mathcal{A}}(H)$. The advantage of $\mathcal{A}$ is defined as:*

$$\mathsf{Adv}^{\mathrm{CR}}_{H}(\mathcal{A}) = \Pr\left[(m, m') \xleftarrow{\$} \mathcal{A} : \begin{matrix} H(m) = H(m') \\ \wedge\ m \neq m' \end{matrix}\right].$$

A.2. Pseudo-Random Functions

EDHOC generates multiple keys, IVs, and MAC tags from an already established session key using pseudo-random functions.

Definition A.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 $\mathcal{Y}$.*

Security of PRFs. We use the term "PRF" to refer to a function defined in this section and the associated security notion that we formalize next. Let $\mathcal{A}$ be any efficient distinguisher for F in the sense of the game $\mathbf{G}^{\mathrm{PRF}}$ (Figure 6). We define $\mathcal{A}$'s advantage against the PRF security of F as follows:

$$\mathsf{Adv}^{\mathrm{PRF}}_{F}(\mathcal{A}) = 2 \cdot \Pr\left[\mathbf{G}^{\mathrm{PRF}}(F) \to 1\right] - 1.$$

Key expansion as a PRF in EDHOC. The Expand module in EDHOC defined as $\mathsf{Expand}(k, (label, context, len), len)$ (cf. Section 2.2.1 and [56, Section 4.1.2]) is a *variable output length* PRF that outputs bit strings of length len when queried on the key k and input $label$.

A.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

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

PRF(x)
$1: y_0 = F(k, x)$
$2: y_1 = f(x)$
$3:$ **return** y_b

FINALIZE(b')
$1:$ **return** $b' = b$

Figure 6. The PRF security game $\mathbf{G}^{\mathrm{PRF}}(F)$ for a function F.

the design of HKDF [44], it is realized via two modules. The first $\mathsf{Extract}(s, \mathsf{ikm})$ that extracts a pseudo-random key from ikm. The second module is the function $\mathsf{Expand}(\mathsf{prk}, \mathit{info}, \mathit{len})$ that generates from the pseudo-random key prk another length-len pseudo-random key.

A.3.1. 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 [44]. In our analysis of EDHOC, we will rely on the PRF-ODH assumption described in Appendix A.4.

A.3.2. The Expand module. Once a pseudo-random key prk is obtained from the extractor, one wishes to use it to generate other keys. The $\mathsf{Expand}(\mathsf{prk}, (\mathit{label}, \mathit{context}, \mathit{len}), \mathit{len})$ module is used for this purpose. We assume that Expand is a PRF as defined in Appendix A.2.

A.4. Pseudo-Random Function Diffie–Hellman Oracle Assumption

The PRF-ODH [15], [37] assumption has been introduced and used to analyze real-world Diffie–Hellman based key exchange protocols (including TLS 1.2, TLS 1.3, Signal, Wireguard). In DH-based protocols, participants exchange DH shares xG, yG and compute the shared secret $ss = \mathsf{DH}(x, yG)$, 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 $ss' = \mathsf{DH}(v, xG)$ 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 A.3 (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 Appendix A.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 $k = u(vG)$ for $(u, v) \xleftarrow{\$} \mathbb{Z}_q^2$. Similiarly to the usual PRF security notion, an adversary is given access to an oracle* CHALL *(once) that returns either the output of F or a uniform random value. In addition, the adversary is given uG, vG, and also access to an oracle* U *that can be called once on input $(T, x) \neq (vG, x^*)$ to compute $F(T^u, x)$. The security notion is made more formal in the game $\mathbf{G}^{\mathrm{snPRF\text{-}ODH}}$ (Figure 7). The advantage of an adversary $\mathcal{A}$ is defined as:*

$$\mathsf{Adv}_F^{\mathrm{snPRF\text{-}ODH}}(\mathcal{A}) = 2 \cdot \Pr\left[\mathbf{G}^{\mathrm{snPRF\text{-}ODH}}(F) \to 1\right] - 1.$$

INITIALIZE()
$1: b \xleftarrow{\$} \{0,1\}$
$2: (u, v) \xleftarrow{\$} \mathbb{Z}_q^2$
$3:$ **return** (uG, vG)

CHALL(x^*)
$1: y_0^* \leftarrow F(u(vG), x^*)$
$2: y_1^* \xleftarrow{\$} \mathcal{Y}$
$3:$ **return** y_b^*

FINALIZE(b')
$1:$ **return** $b = b'$

U(T, x)
$1:$ **if** $T \notin \mathbb{G}:$
$2:$ **return** $\perp$
$3:$ **if** $(T, x) = (vG, x^*):$
$4:$ **return** $\perp$
$5: y \leftarrow F(uT, x)$
$6:$ **return** y

Figure 7. The snPRF-ODH game $\mathbf{G}^{\mathrm{snPRF\text{-}ODH}}(F)$. The adversary may call each of the oracles CHALL and U only once, and must call CHALL first.

The PRF-ODH assumption was studied in [15], and the authors showed that in the random oracle model, the strongest PRF-ODH variant is achievable under the strong Diffie–Hellman assumption.

PRF-ODH security of Extract. Brendel et al. [15] showed that HMAC is snPRF-ODH-secure in the random oracle model, 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 directly uses KMAC for MAC-ing and Shake128 or Shake256 for hashing and as 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.

A.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, signatures are used to authenticate the peers.

Definition A.4 (Digital signature scheme)**.** *A digital signature scheme $\mathfrak{S}$ is a triple of efficiently computable algorithms* $(\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ *where:*

- 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 σ and outputs a bit $b = \mathsf{Vf}(pk, m, \sigma)$. The output is 1 when the signature is valid and 0 otherwise.*

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

$\mathbf{G}^{\text{EUF-CMA}}$

INITIALIZE()
1 : $(sk, pk) \leftarrow \mathfrak{S}.\mathsf{KGen}$
2 : $\mathcal{M} \leftarrow \emptyset$
3 : **return** pk

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

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

$\mathbf{G}^{\text{SUF-CMA}}$

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

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

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

Figure 8. The EUF-CMA game (top) and SUF-CMA game (bottom) for a signature scheme $\mathfrak{S}$.

A.5.1. Security of Digital Signatures Schemes.

Definition A.5 (Existential unforgeability under chosen-message attacks)**.** *For a signature scheme $\mathfrak{S}$ and an efficient adversary $\mathcal{A}$,* existential unforgeability under chosen-message attacks *(*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 8). The advantage of $\mathcal{A}$ is defined by:*

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

Definition A.6 (Strong unforgeability under chosen-message attacks)**.** *For a signature scheme $\mathfrak{S}$ and an efficient adversary $\mathcal{A}$,* strong unforgeability under chosen-message attacks *(*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 8). We define $\mathcal{A}$'s advantage as follows:*

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

A.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 confidentiality for m and integrity for both m and ad. In EDHOC, AEAD is used to encrypt part of the handshake. We use the nonce-based syntax for AEAD [53].

Definition A.7 (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:*

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

Correctness. We demand that for all $k \in \mathcal{K}$, $m \in \mathcal{M}$, $ad \in \mathcal{AD}$, $n \in \mathcal{N}$:

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

B. Full Proof of Theorem 5.1

We provide here a full proof for Theorem 5.1.

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

B.1. Phase 1: Ensuring Soundness

GAME $\mathbf{G}_0$. We start with the normal MSKE game defined in Figure 4 and played by $\mathcal{A}$. By definition,

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

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 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:

$$\mathsf{Adv}^{\mathbf{G}_1}(\mathcal{A}) = \mathsf{Adv}^{\mathbf{G}_0}(\mathcal{A}).$$

GAME $\mathbf{G}_2$. This game aborts 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 [7], the advantage difference of $\mathcal{A}$ can be bounded as follows:

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

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:

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

Conclusion of Phase 1. At this point, we argue that if $\mathbf{G}_2$ does not abort, 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 5), there are six 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 abort, then none of the six events occurred.

Proposition B.1. *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[12]*. 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 B.2. *Matching session identifiers for a given stage implies matching stage session keys.*

Proof. *We show that matching session identifiers implies 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) starts by computing the key $\mathsf{PRK}_{2e} = \mathsf{Extract}(\mathsf{th}_2, G_{xy})$, where G_{xy} is the shared Diffie-Hellman secret. Hence, the equality of the session identifiers implies the equality of the derived PRK_{2e}. The key schedule proceeds to derive further stage keys (and potentially associated IVs) using the* Expand *function keyed with PRK_{2e}. For each key/IV,* Expand *is evaluated on an input composed of the (partial) transcript hash and a stage-specific label. 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 B.3. *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 B.1) 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 kG in the session identifier at a fixed position. Having Two sessions with the same session identifier and the same role at a given stage implies 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 abort.*

Proposition B.4. *Matching session identifiers for a given stage implies agreed-upon contributive identifiers.*

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: $\mathsf{cid}[init, 1] = (\text{“1”}, G_x)$ and for all roles r and stages 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 B.5. *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 B.3), 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.*

12. Every session identifier includes two DH shares.

Proposition B.6. *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*

B.2. Phase 2: Ensuring Explicit Authentication and Key Indistinguishability

We proceed with the second phase 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 2, 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 lookups. 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

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

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

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

We bound $\Pr[hash_{coll} \leftarrow \mathsf{true}]$ using a reduction $\mathcal{B}_4$ to the collision resistance of 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, $\Pr[hash_{coll} \leftarrow \mathsf{true}] \leq \mathsf{Adv}^{\mathrm{CR}}_{\mathsf{H}}(\mathcal{B}_4)$ and as consequence:

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

At this point, we split the proof into two branches **I** and **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:

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

We start with branch **Branch I**.

B.2.1. Branch I: The adversary cannot break explicit authentication. In this branch, 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 (π_U^i, s) such that ExplicitAuth evaluates to false since the adversary broke explicit authentication of π_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:

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

From now on, π_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, σ, pk_U, U). In concrete terms, an initiator session with identity I will sign a message of the form $m_3 = (\mathsf{I_{sig}}, \mathsf{kid}_I, \mathsf{th}_3, \mathsf{cred}_I, ead_3, \tau_3)$. Whereas, the responder R will sign a message of the form $m_2 = (\mathsf{I_{sig}}, \mathsf{kid}_R, \mathsf{th}_2, \mathsf{cred}_R, ead_2, \tau_2)$. 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 msg_2 that includes the credential identifier kid_U, initiator sessions will attempt to validate the received signature σ_2 against each public key pk_U referenced by kid_U, adapting the a priori signed message to the messages $m_U = (\mathsf{I_{sig}}, \mathsf{kid}_U, \mathsf{th}_2, \mathsf{cred}_U, ead_2, \tau_2)$. 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 σ_3 received within msg_3 and all possible messages $m_U = (\mathsf{I_{sig}}, \mathsf{kid}_U, \mathsf{th}_3, \mathsf{cred}_U, ead_3, \tau_3)$ against each public key pk_U.

In addition to logging messages, we set the flag sig_{forged} if the targeted session receives and validates a message signature pair (m, σ) under the public key pk_V of an honest[13] 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:

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

GAME $\mathbf{G}_{I.2}$. In this game, we abort whenever sig_{forged} is set. 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, σ) that caused sig_{forged} to be set and aborts $\mathbf{G}_{I.2}$. See Section 5.2 for further discussion on the need for SUF-CMA security.

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.

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, σ) under a pk_V such that $(m, \sigma, pk_V, V) \notin T_{sig}$. This implies that (m, σ) 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:

$$\Pr[sig_{forged} \leftarrow \mathsf{true}] = \mathsf{Adv}_{\mathsf{Sig}}^{\text{SUF-CMA}}(\mathcal{A}).$$

And as a consequence:

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

GAME $\mathbf{G}_{I.3}$. In this game, we set the flag $sig_{ambigous}$ if there exists an honest session π that receives and accepts a message-signature pair (m', σ) under a public key $pk_{U'}$, where a session of some user U produced σ on some other message m, and there exists a value kid such that $(pk_U, U) \in peerpk_{\mathsf{kid}}$ and $(pk_{U'}, U') \in peerpk_{\mathsf{kid}}$. In other words, 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 NEWUSER($sk_{U'}, pk_{U'}, \mathsf{kid}$). From the standpoint of π, there is an ambiguity about the identity of the peer that (presumably) authenticated themselves via the received message signature pair (m', σ). These changes are unobservable to the adversary, therefore:

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

GAME $\mathbf{G}_{I.4}$. The game aborts 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 $m' \neq m$. 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 $(\mathsf{I_{sig}}, \mathsf{kid}_{U'}, \mathsf{th}, \mathsf{cred}_{U'}, ead, \tau)$. 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 relate $\Pr[sig_{ambigous} \leftarrow \mathsf{true}]$ to the advantage of an S-UEO adversary $\mathcal{B}_{I.4}$ against Sig. 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 aborts the game when $sig_{ambigous}$ is set.

Simulation soundness: We only need to consider the REVLONGTERMKEY(U) queries, as the reduction can answer all other queries consistently. On the one hand, we

13. More precisely, we only expect that V is honest at the time the message-signature pair is received.

do not need to consider what happens after $sig_{ambigous}$ is set. The simulation aborts and need not answer REVLONGTERMKEY(U) queries. On the other hand, the predicate ExplicitAuth requires pk_U is not compromised before the session π 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, π accepted and verified a message signature pair (m', σ) 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', σ, pk, pk') is a valid S-UEO forgery. Therefore, we get that

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

Finally, we get:

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

Note. The signature schemes in EDHOC are Ed25519 and ECDSA. The former is known to be S-UEO-secure [14]. Moreover, 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: $\widehat{\mathsf{Sig}}.\mathsf{Sign}(sk, m) = \mathsf{Sig}.\mathsf{Sign}(sk, (m, pk))$. Pornin and Stern [51] 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.

Establishing ExplicitAuth $=$ true**.** 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.

The predicate ExplicitAuth and its negation are shown in Figure 9 for illustration. When explicit authentication is violated (¬ExplicitAuth in Figure 9), the following holds:

1) π^i_U accepted stage s (resp. s') at time t (resp. t'). The session accepts with a peer identity $\pi^i_U.\mathsf{pid} = V$ (one must be set).[14]
2) V's long-term secret was not compromised at time t'.
3) (I.a) Either, no (honest) session π^j_V is partnered with π^i_U in stage s'.
4) (I.b) *Or*, There exists an honest session π^j_V that is partnered with π^i_U 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 σ 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 σ_2 within the message msg_2, where σ_2 is computed over a message of the form $m_2 = (\mathsf{I}_{\mathsf{sig}}, \mathsf{kid}_R, \mathsf{th}_2, \mathsf{cred}_R, ead_2, \tau_2)$. The responder session must have received a signature σ_3 over the message $m_3 = (\mathsf{I}_{\mathsf{sig}}, \mathsf{kid}_I, \mathsf{th}_3, \mathsf{cred}_I, ead_3, \tau_3)$ within the message 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, π^i_U, accepted a message signature pair (m, σ) under the public key of V, i.e., $\mathsf{Vf}(pk_V, m, \sigma) = 1$, but no session π^j_V is partnered with π^i_U in stage 2 (resp. stage 3) if π^i_U is the initiator (resp. responder). We consider two cases that we call (i) and (ii), based on whether the message and signature received by π^i_U 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, σ). 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 π^j_V produced the message signature pair received and accepted by π^i_U. In particular, if π^i_U is in the initiator role, the message $(\mathsf{I}_{\mathsf{sig}}, \mathsf{kid}_R, \mathsf{th}_2, \mathsf{cred}_R, ead_2, \tau_2)$ was signed (resp. verified) by π^j_V (resp. π^i_U). Hence, π^i_U and π^j_V agree on the values of σ_2, 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, π^i_U must also agree on the values of G_y, C_R, and and therefore agree on their respective stage-2 session identifiers and are partnered in stage 2, contradicting the assumption that π^i_U does not have a partner session in stage 2. Analogously, if π^i_U is a responder session (π^j_V is an initiator), the message signed is of the form $m_3 = (\mathsf{I}_{\mathsf{sig}}, \mathsf{kid}_I, \mathsf{th}_3, \mathsf{cred}_I, ead_3, \tau_3)$. Therefore, there is agreement on the values of σ_3, 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 π^i_U is in the responder role. The attacker can cause case (ii) to occur by mounting an attack against the intended initiator session π^j_V such that π^j_V would accept with a malicious peer identity U' while not modifying the conversation transcript. The subtlety of this attack is that although π^j_V has been "tricked" into accepting with an unintended peer, the adversary does not, in fact, break explicit authentication for π^j_V; the adversary broke explicit authentication for the responder session π^i_U. At the end of the protocol run, π^i_U ends up without a partner in stage 2 and above; hence π^i_U 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 π^i_U. Upon receiving msg_2 from π^i_U, $\mathcal{A}$ registers a new key pair by calling NEWUSER($sk_{U'}, pk_{U'}, \mathsf{kid}_U$). The malicious key pair is selected such that π^j_V would accept σ_2 under $pk_{U'}$ when delivered via the relevant SEND query. Careful observation

14. Note that we can focus on potential partner sessions owned by $\pi^i_U.\mathsf{pid} = V$, as sessions $\pi^k_{V'}$ for $V' \neq V$ trivially satisfy the $\pi^i_U.\mathsf{pid} \neq V$ of the $\forall$ clause of ¬ExplicitAuth.

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

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

Figure 9. The top predicate corresponds to explicit authentication (predicate ExplicitAuth) being satisfied, the bottom predicate is its negation and corresponds to explicit authentication being violated; used when establishing $\text{ExplicitAuth} = \mathsf{true}$ in the proof in Game $\mathbf{G}_{I.4}$.

of the protocol specification reveals that such an attack would not disturb the protocol run. However, the result is an identity misbinding attack.

Thanks to $\mathbf{G}_{I.4}$, ambiguity about the responder of the initiator is excluded. Furthermore, if the initiator session π_V^j accepts another peer identity U', the value of th_3 computed by π_U^i (resp. π_V^j) are $\mathsf{H}(\mathsf{th}_2, \mathsf{ptxt}_2, \mathsf{cred}_U)$ (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 π_U^i to later accept 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 in which π_U^i receives explicit authentication. Recall that $s' = 2$ if π_U^i is in the initiator role and $s' = 3$ if π_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 π_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 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 π_U^i's long-term secret, sk_U, is not comprised before π_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 π_V^j accepts the peer identity U' after receiving 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,

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

Conclusion of Branch I. 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 true. We do so by showing that the challenge bit is random and independent of the adversary's guess.

B.2.2. Branch 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, π_U^i. We follow the approach of Dowling

et al. [31] 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 $n_S \cdot \mathbf{S}$. Here, n_S is the number of sessions, and $\mathbf{S} = 4$ is the number of stages. Therefore, we get the following bound:

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

We proceed with our analysis of Branch II by considering two disjoint cases. Namely,

- Case $\mathbf{A}$: the tested session does not have a (honest) contributive partner in the first stage, i.e.,

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

- Case $\mathbf{B}$: The tested session has a (honest) contributive partner in the first stage, that is,

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

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

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

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 5). In particular, $\mathcal{A}$ may not test stage 1 nor stage 2 in the 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 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 π^i_U's peer is compromised, that is, TEST must be issued *before* REVLONGTERMKEY$(\pi^i_U.\mathsf{pid})$. 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 π^i_U 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:

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

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

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

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 π^i_U's peer, and associates the challenge public key pk^* to V's long-term verification key, i.e. $pk_V = pk^*$. Consequently, $\mathcal{A}$' 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, σ) as its forgery towards its EUF-CMA challenger. Here, σ 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 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 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 abort 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, σ) is a valid forgery, that is, when $\mathcal{B}_{II.A2}$ outputs (m, σ), the EUF-CMA challenger also outputs 1. In EDHOC, signatures are computed on (amongst other things) the MAC tag (τ) and the transcript hashes (th). More precisely, the messages to be signed is $m = (\mathsf{I}_{\mathsf{sig}}, \mathsf{kid}_U, \mathsf{th}, \mathsf{cred}_U, ead, \tau)$. 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 kid_U refers to multiple credentials. In this case, for each cred_X associated with 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 abort. This implies that without a contributive partner, no honest session signed the message that the tested session received and accepted after successfully validating the signature. As a result, given the challenge (EUF-CMA) public key pk^*, $\mathcal{B}_{II.A2}$ can verify that the pair (m, σ) is a valid forgery; allowing $\mathcal{B}_{II.A2}$ to abort the game and present (m, σ) to the challenge EUF-CMA. Therefore, we have:

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

It follows that:

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

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 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 π_V^j that is contributive partner of the tested session π_U^i. This step reduces the advantage of $\mathcal{A}$ by a factor n_S and we get:

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

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 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 PRK_{2e}. We note here that the cid partner is not guaranteed to have received the honest DH shares from the tested session; namely, 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 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 PRK_{2e} at the tested session π_U^i and eventually at its partnered session π_V^j. $\mathcal{B}_{II.B2}$ consistently answers all other queries. We observe that if the tested session is in the initiator role, then π_U^i and π_V^j have the same $\widetilde{\mathsf{PRK}_{2e}}$. If, however, π_U^i is a responder session, π_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 π_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 uG and vG from its snPRF-ODH challenger and simulates $\mathbf{G}_{II.B2}$ towards $\mathcal{A}$ answering all queries unrelated to PRK_{2e} as needed. $\mathcal{B}_{II.B2}$ encodes the received DH shares (uG, vG) into the Diffie–Hellman shares (G_x, G_y) of the tested session and its partner, respectively. To derive PRK_{2e}, $\mathcal{B}_{II.B2}$ makes a PRF query on input th_2 (recall that $\mathsf{PRK}_{2e} = \mathsf{Extract}(\mathsf{th}_2, G_{xy})$) and copies the result into the state of the tested session. $\mathcal{B}_{II.B2}$ copies 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 $v = \mathsf{H}(G'_y, \mathsf{C}_R, \mathsf{H}(G_x, \mathsf{C}_I, ead_1))$.

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:

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

GAME $\mathbf{G}_{II.B3}$. In this game, we replace the function Expand keyed with $\widetilde{\mathsf{PRK}_{2e}}$ with a random function F at the tested session. The contributive partner also replaces 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 Expand. To answer queries that require deriving any key, IV, or MAC tag derived from PRK_{2e}, $\mathcal{B}_{II.B3}$ queries its PRF oracle with the appropriate input. By the Game $\mathbf{G}_{II.B2}$, we have replaced 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:

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

Having replaced Expand with a random function F, we can readily replace all values derived by the tested session using a call to 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 (τ_2, τ_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. Fortunately, EDHOC includes the identities in the transcript hashes. Therefore, disagreement on the peer's identity leads to divergent keys.

Conclusion of Branch II. 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:

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

To summarize the proof:

1) Sound. As discussed in the conclusion of Phase 1, the predicate Sound remains true.

2) $\mathrm{ExplicitAuth}$. As discussed in the conclusion of Phase 2, Branch I, the predicate $\mathrm{ExplicitAuth}$ remains true.
3) Key secrecy. As discussed in the conclusion of Phase 2, Branch 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 $\mathsf{EDHOC\text{-}Sig\text{-}Sig}$ hold, which concludes the proof of Theorem 5.1. □

C. Identity Misbinding in MAc-then-SIGn Under Strong Adversaries

In the following, we discuss an identity misbinding attack in our model on the MAc-then-SIGn protocol [43], hereafter denoted by $\mathsf{SIGMA}_{\frac{\sigma}{\tau}}$ for "MAC under the signature".[15] We stress that this attack does not contradict the original security analysis of SIGMA's MAc-then-SIGn variant by Canetti and Krawczyk [19]; it arises through giving an adversary the ability to register its own malicious keys in the key exchange model, as our model does to capture the potential ambiguous interpretation of credential identifiers in EDHOC (cf. Section 4).

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. The original analysis [19] requires that a secure instantiation must use an unforgeable signature scheme. We show that for the security of $\mathsf{SIGMA}_{\frac{\sigma}{\tau}}$ against adversaries that can register malicious keys, it is not enough for the signature scheme to be $\mathrm{EUF\text{-}CMA}$ secure; it must indeed be unforgeable for all keys. In particular, there cannot be any weak key, for instance, one that accepts any message-signature pair. We note that this observation is in line with a remark by Pornin and Stern [51] that weak keys can lead to DSKS/exclusive ownership attacks. (An alternative approach, which is beyond the scope of this work, would be to establish security for $\mathsf{SIGMA}_{\frac{\sigma}{\tau}}$ under an exclusive ownership property of the signature scheme, akin to our EDHOC analysis.)

THE SETUP. Recall that in $\mathsf{SIGMA}_{\frac{\sigma}{\tau}}$, the initiator sends the first message (G_x). The responder picks its own DH share G_y and computes a shared session key K and a MAC key K_m from G_{xy} using a key derivation function. It responds with the second protocol message $(R, G_y, \sigma_R = \mathsf{Sign}(sk_I, G_x, G_y, \mathsf{MAC}(\mathsf{K}_m, R)))$.[16] The initiator completes the key exchange by sending the third message, $(I, \sigma_I = \mathsf{Sign}(sk_I, G_y, G_x, \mathsf{MAC}(\mathsf{K}_m, I)))$.

Assume now that the attacker intercepts the second message and modifies the responder's identity to some R'; i.e., sends to the initiator the message (R', G_y, σ). For the identity R', the adversary has registered a weak verification key pk such that for all messages m and signatures σ, $\mathsf{Vf}(pk, m, \sigma) = 1$. As a consequence, the attacker need not know the MAC key, yet the initiator will accept the signature with peer identity R'. Furthermore, the attack modifies the second message only; in particular, it does not modify the keys computed by the initiator and the responder. Hence, the responder successfully accepts the initiator's final message—which the adversary relays unmodified. The result is an identity misbinding attack: initiator I and responder R share the same session key, but the initiator (incorrectly) thinks it is talking to R', while the responder correctly deems I as its peer.

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 $\mathrm{EUF\text{-}CMA}$ unforgeability notion is that it only captures "average-case" unforgeability, i.e., for an honestly, randomly generated key. In contrast, the attack described here requires unforgeability essentially for all keys, as the adversary is able to register them with the key exchange game.

Formally, $\mathrm{EUF\text{-}CMA}$ security is insufficient by the following separating example. Let Sig be a $\mathrm{EUF\text{-}CMA}$ secure signature scheme. Define $\widehat{\mathsf{Sig}}$ such that the key space of $\widehat{\mathsf{Sig}}$ is the key space of Sig augmented with the special key pair (sk^*, pk^*). Signing in $\widehat{\mathsf{Sig}}$ is as before unless the signing key is sk^*, in which case a random signature value is returned. The verification algorithm is also the same, except that when the verification key is pk^*, verification always returns 1. Observe that $\widehat{\mathsf{Sig}}$ is still $\mathrm{EUF\text{-}CMA}$ as the advantage of any $\mathrm{EUF\text{-}CMA}$ adversary is only increased by the probability that the challenge key pair happens to be (sk^*, pk^*), which for practical key spaces of Sig is small. However, the $\mathsf{SIGMA}_{\frac{\sigma}{\tau}}$ protocol is clearly vulnerable to the attack described above if instantiated with $\widehat{\mathsf{Sig}}$.

15. For simplicity, we exclude identity protection from our treatment.
16. We omit further distinguishing label inputs for clarity.