Formal Verification of EDHOC-PSK: A Symbolic Approach with SAPIC+

Elsa Lopez Perez
elsa.lopez-perez@inria.fr
Inria
Paris, France

Thomas Watteyne
twatteyne@gmail.com
Unaffiliated
Paris, France

Cristina Onete
maria-cristina.onete@unilim.fr
University of Limoges, CNRS 7252
Limoges, France

Dhekra Mahmoud
dhekra.mahmoud@cispa.de
CISPA Helmholtz Center for Information Security
Saarbruecken, Germany

Pascal Lafourcade
pascal.lafourcade@uca.fr
Université Clermont Auvergne
Clermont-Ferrand, France

Vaishnavi Sundararajan
vaishnavi@cse.iitd.ac.in
IIT Delhi
Delhi, India

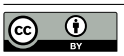
Mališa Vučinić
malisa.vucinic@inria.fr
Inria
Paris, France

Abstract

EDHOC is a lightweight authenticated key exchange protocol designed for constrained IoT devices. It currently supports asymmetric authentication, either using digital signatures or static Diffie-Hellman (DH) keys. Since many IoT deployments rely on Pre-Shared Keys (PSK) for authentication, a new PSK-based authentication method (EDHOC-PSK) is currently under standardization. This paper presents a symbolic analysis EDHOC-PSK (draft version 06) using SAPIC+, which compiles a single formal specification into multiple state-of-the-art verification tools, including Tamarin and ProVerif. Our model extends the typical Dolev-Yao (DY) adversary with additional capabilities, including leakage of ephemeral secrets, leakage of the long-term Pre-Shared Key, leakage of the session key, and a discrete-logarithm oracle. We verify the confidentiality, authentication, and key-agreement properties stated in the draft, and we refine the specification of identity protection by distinguishing anonymity and unlinkability. We show that EDHOC-PSK achieves anonymity for both parties against active attackers, while unlinkability holds only for the Initiator under passive attackers. Finally, we analyze a post-quantum Store-Now-Decrypt-Later (SNDL) adversary and find that all proven properties remain intact except Perfect Forward Secrecy (PFS), which cannot be preserved once DH secrets are recoverable.

CCS Concepts

• **Security and privacy → Formal security models**.



ASIA CCS '26, Bangalore, India

ACM ISBN 979-8-4007-2356-8/2026/06
https://doi.org/10.1145/3779208.3805979

Keywords

Symbolic Analysis, SAPIC+, EDHOC, Formal Verification, DY model

ACM Reference Format:
Elsa Lopez Perez, Thomas Watteyne, Cristina Onete, Dhekra Mahmoud, Pascal Lafourcade, Vaishnavi Sundararajan, and Mališa Vučinić. 2026. Formal Verification of EDHOC-PSK: A Symbolic Approach with SAPIC+. In *ACM Asia Conference on Computer and Communications Security (ASIA CCS '26), June 01–05, 2026, Bangalore, India.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3779208.3805979

1 Introduction

The exponential growth of the Internet of Things (IoT) ecosystem has driven standardization bodies such as the Internet Engineering Task Force (IETF) to develop protocols that are tailored for the particular needs of constrained devices and networks. On the device side, challenges include limited memory, with 10's of kB of RAM and 100's of kB of flash memory, low processing power, with clock frequency in the 10's of MHz, and restricted energy budgets, since many devices rely on non-rechargeable batteries intended to last for multiple years. On the network side, bandwidth is scarce, communication may be intermittent and prone to delays, and maximum transmission units are on the order of 50 bytes.

To address these constraints, the Lightweight Authenticated Key Exchange (LAKE) Working Group (WG) has developed *Ephemeral Diffie-Hellman Over COSE* (EDHOC): a protocol designed to enable an authenticated Diffie-Hellman key exchange between two peers, both possibly operating on resource-constrained devices and utilizing low-power IoT radio communication technologies. Standardized in RFC9528 [38], the solution adopted by the LAKE Working Group requires mutual authentication, and each peer has a choice of using Static Diffie-Hellman credentials, or digital signatures, leading to four protocol versions (see Table 1).

EDHOC has been demonstrated as a competitive alternative for DTLS 1.3 in resource-constrained environments. One impediment

Method	Initiator	Responder
0	Signature	Signature
1	Signature	Static Diffie-Hellman
2	Static Diffie-Hellman	Signature
3	Static Diffie-Hellman	Static Diffie-Hellman

Table 1: Current authentication methods registered by IANA [23] (0-1-2-3).

to its wide-scale adoption, however, is the lack of support for advanced authentication methods in IoT deployments. IoT devices often rely on Pre-Shared Keys (PSKs) for authentication [13, 26, 41]. To address this reality and improve compatibility with existing systems, the IETF LAKE WG has begun extending EDHOC with a PSK-based authentication method. This extension is specified in a recent Internet-Draft describing EDHOC authenticated with PSKs [30], which is currently under consideration for standardization[1]. This addition broadens the protocol applicability and provides a lightweight authentication option aligned with the constraints of many contemporary IoT environments.

In this paper, we set out to analyze the security of the current draft of the PSK-based authenticated EDHOC proposal.

Contributions. Our main contribution is a comprehensive formal analysis of EDHOC authenticated with pre-shared keys (EDHOC-PSK), focusing on draft version 06. We identify precisely which security properties EDHOC-PSK satisfies, including confidentiality, authentication, full agreement, and two flavors of identity protection (anonymity and unlinkability), and we clarify the exact attacker models under which these guarantees hold. Our results highlight the protocol strengths, its limitations (e.g. the lack of perfect forward secrecy under post-quantum "*Store-Now-Decrypt-Later*" attacks), and provide concrete recommendations relevant to the ongoing standardization effort.

For our analysis, we use the SAPIC+ protocol verification platform [11]. SAPIC+ uses the applied π-calculus [1] as a specification language and its code can be exported to state-of-the-art tools, including ProVerif [6] and Tamarin [33].

As in other formal verification tools, our attacker operates in the Dolev-Yao (DY) model [18]: they have full control over the network, and may eavesdrop, intercept, or inject messages at will.

However, to capture the threat models relevant for EDHOC-PSK, including post-quantum attacks and various compromises, we allow the ability to extend the standard symbolic adversary with additional capabilities: *(i)* leakage of ephemeral Diffie–Hellman secrets, *(ii)* leakage of long-term PSKs, *(iii)* leakage of session keys and *(iv)* computation of the Discrete Logarithm (DL). These extensions allow us to analyze classical, post-quantum, and compromise-resilient variants of the protocol within a unified model. We also provide scripts that automate the verification workflow: they execute all relevant properties under different attacker capabilities and record the results in a structured format (CSV). [2]

Outline. Section 2 presents related work on symbolic analysis, including prior analyses of EDHOC and other protocols using PSKs. Section 3 describes EDHOC authenticated with PSK, providing both a protocol overview and a description of the claimed security properties. Section 4 introduces the foundations of symbolic analysis, including adversary model, verification platform, and key modeling principles. Section 5 details the protocol and property specifications, illustrating each security property with concrete examples. Section 6 outlines our analysis methodology. Section 7 presents the obtained results. Section 8 discusses directions for future work. Finally, Section 9 concludes the paper.

2 Related Work

Formal methods play an increasingly critical role in the development and standardization of security protocols. Tools for symbolic and computational verification are now being integrated in protocol design while the protocols are under standardization, as witnessed with TLS 1.3 and EDHOC processes [5, 40]. To this end, several formal verification frameworks can be employed. Some of them are ProVerif [6], Tamarin [33] and AVISPA [3].

ProVerif [6] is an automatic verifier for security protocols in the symbolic DY model. It supports a wide range of cryptographic primitives, such as symmetric and asymmetric encryption, signatures, hash functions, and Diffie–Hellman constructions, through user-defined equational theories. ProVerif can establish *trace* properties such as secrecy and authentication, and *equivalence-based* properties such as anonymity. It works in the setting of an unbounded number of sessions and an unbounded message space, which may lead to non-termination. ProVerif is sound but may reports spurious attacks; its strengths lie in automation and scalability.

Tamarin [33] is a symbolic protocol verifier that combines automated reasoning with interactive proof capabilities. It supports both trace and equivalence properties, and provides advanced algebraic theories (e.g. Diffie–Hellman exponentiation) with support for stateful protocols. Similar to ProVerif, Tamarin handles an unbounded number of sessions, but its interactive mode makes it suitable for debugging non-termination. Unlike ProVerif, Tamarin is both sound and complete in trace mode.

AVISPA [3] is a push-button framework for the automated validation of security-sensitive Internet protocols. It offers a modular and expressive specification language and integrates several back-end analyzers that implement a variety of state-of-the-art symbolic analysis techniques.

An early version of the EDHOC protocol, draft 00, was verified using Tamarin. The analysis, conducted by Norrman *et al.* [35], identified a lack of injective agreement for static Diffie-Hellman, leading to the addition of an optional fourth message in draft 05. As a follow up, Kim *et al.* [25] used BAN logic and the AVISPA tool to scrutinize draft 07, uncovering privacy flaws that lead to the improvements in the authentication of messages 2 and 3. Jacomme *et al.* [24] used SAPIC+ and performed a symbolic analysis on EDHOC draft 12 on all four authentication methods. They found

[1]The LAKE WG has issued a call for formal analysis of EDHOC-PSK draft version 06: https://mailarchive.ietf.org/arch/msg/lake/A4YOMyFDQunzYDVSlsjt_C6yaRE/.

[2]For reproducibility, we provide a Docker environment and full implementation publicly available at https://github.com/ElsaLopez133/edhoc-psk-formal-analysis

a flaw in the key derivation that led to a weak session key, due to key reuse. They also detected a privacy leak on the Initiator's side, as the list of trusted identities was missing its own identity.

More recent EDHOC drafts have also undergone computational analysis. Günther *et al.* [22], Cottier and Pointcheval [15] and Ferreira [20] used game-based proofs to analyze authentication and key secrecy properties. These efforts uncovered important issues such as salt collisions, and identity mis-binding attacks. We provide a summary of these analyses and their outcomes in Table 2.

Beyond EDHOC, other protocols that use Pre-Shared Keys (PSKs) for authentication have also been extensively analyzed. For instance, TLS 1.3 and its datagram counterpart, DTLS 1.3, both offer PSK-based modes. Studies show [28, 37] that, while PSK authentication is highly efficient for resource-constrained devices, its security critically depends on the entropy and secrecy of the keys. Yong Li *et al.* [29] formally analyze TLS-PSK, establishing strong security guarantees under the assumption of high-entropy keys. Cremers *et al.* [16] prove that Perfect Forward Secrecy (PFS) is achieved in TLS 1.3 only when PSK is combined with an ephemeral Diffie-Hellman key exchange (PSK-DHE), but not in the PSK mode.

EDHOC has been analyzed when authenticated with public-key-based methods. PSK-based authentication has been studied in contexts outside of EDHOC. This creates a gap in the literature, making a dedicated analysis of the security properties of EDHOC-PSK needed. We build on these efforts by applying a symbolic analysis to EDHOC-PSK. We build on the comprehensive symbolic analysis of Jacomme *et al.* [24]. However, their analysis does not cover EDHOC-PSK, as it focuses exclusively on the asymmetric authentication methods of the original RFC9528 [38]. Moreover, several security notions relevant for PSK-based authentication – such as anonymity, unlinkability, and post-quantum "Store-Now-Decrypt-Later" resilience – are absent from their framework. Our work therefore extends and complements that of Jacomme *et al.* by adapting their definitions to the PSK setting, introducing new privacy notions tailored to EDHOC-PSK, and incorporating attacker capabilities specific to post-quantum compromise.

3 EDHOC-PSK

EDHOC is an authenticated key exchange protocol designed for peers running on constrained devices over low-power IoT radio communication technologies. The cryptographic foundation of EDHOC is built on the MAC-then-Sign variant of Krawczyk's SIGMA-I protocol [27]: a family of protocols that combine digital signatures and message authentication code (MAC) functions to guarantee active-attacker-secure Diffie-Hellman (DH) key-exchange.

As standardized in RFC9528, EDHOC supports four authentication methods, 0 to 3, depending on the authentication method that each peer chooses (either signatures or static Diffie-Hellman Keys; see Table 1). The introduction of PSK-based authentication modified the message flow of the original EDHOC protocol. Most notably, the fields ID_CRED_I and ID_CRED_R, which are used in EDHOC in order to identify the Initiator and Responder respectively, are omitted in EDHOC-PSK. Instead, both parties identify themselves using ID_CRED_PSK, an identifier that enables retrieval of the PSK.

Unlike other methods that rely on MACs or digital signatures for authentication, EDHOC-PSK achieves authentication through

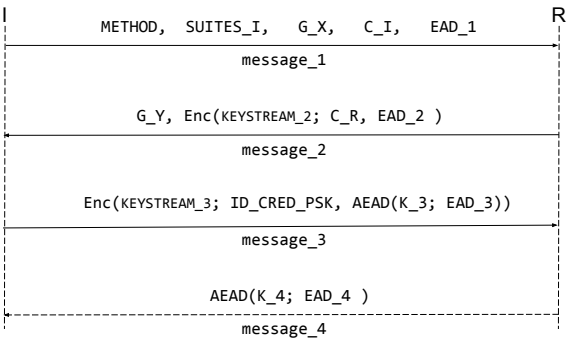


Figure 1: The EDHOC-PSK message flow. G_X **and** G_Y **represent the public ephemeral keys of the Initiator and the Responder, respectively. Field** ID_CRED_PSK **denotes the identifier of the credential containing the Pre-Shared Key.** SUITES_I **is an ordered set of preferred algorithms.**

the use of the PSK. Specifically, the PSK is incorporated in the intermediate pseudo-random keys, which are then used in combination with the Authenticated Encryption with Associated Data (AEAD) function applied in message_3. In EDHOC-PSK, the Initiator authenticates first, unlike in traditional EDHOC (methods 0 to 3), where the Responder is first to authenticate. EDHOC-PSK achieves mutual authentication without depending on an external application layer protocol: this requires a fourth message from the Responder, which ensures that both parties are properly authenticated in the absence of external guarantees.

3.1 Protocol Outline

- **Message 1:** EDHOC-PSK follows the general structure of methods 0 to 3. The Initiator sends the first message containing the setup information. This includes the method identifier (4 in the case of EDHOC-PSK), a ranked list of supported cipher suites (from which the Responder selects the first mutually supported option) [3], the ephemeral Diffie-Hellman public key G_X, a connection identifier C_I, which serves only as a protocol-level handle for correlating messages and does not contribute cryptographically to secrecy or authentication [4], and the additional information called external authorization data EAD_1.
- **Message 2:** Upon receiving message_1, the Responder derives two intermediate pseudo-random keys using the Key Derivation Function EDHOC_KDF: PRK_2e and PRK_3e2m. The Responder also computes a transcript hash TH_2 over the first message and the ephemeral DH public key G_Y, and the Diffie-Hellman shared secret G_XY. It then computes the term CIPHERTEXT_2A by using XOR to encrypt PLAINTEXT_2A = ⟨C_R, EAD_2⟩ with key KEYSTREAM_2A,

[3]We do not model the full negotiation mechanism in our formal analysis; instead, we assume that both parties agree on a single cipher suite.
[4]It nevertheless plays a role in agreement properties, since parties must agree on the same identifiers to achieve injective agreement.

Table 2: Summary of Formal Analyses and Vulnerabilities in EDHOC [36], including our contributions for EDHOC-PSK. *Analysis type* specifies whether a symbolic or a computational analysis was performed, and to which authentication method (as defined in Table 1). *Tool* indicates the used verification platform. *Mitigation* / *Outcome* describes how the vulnerability was fixed in later drafts or what were the results of the analysis.

Draft	Reference	Analysis	Tool	Vulnerability / Focus	Mitigation / Outcome
00	Norrman *et al.* (2020) [35]	Symbolic (all methods)	Tamarin	Lacking injective agreement for static DH; non-repudiation issues	Optional 4th message added for static DH
07	Kim *et al.* (2021) [25]	BAN Logic	AVISPA	Privacy leak in identity credentials	Recommend authenticating message 1, validating message 2
12	Jacomme *et al.* (2023) [24]	Symbolic (all methods)	SAPIC+	Weak session key due to key reuse; privacy leak in trusted list; nonce reuse in AEAD	New key derivation (adopted in 14); include own identity in trusted list; prohibit message resending (in 14)
14	Günther *et al.* (2023) [22]	Computational (method 0)	-	Strengthen against identity misbinding	Unique credentials in transcript hashes
15	Cottier & Pointcheval (2022) [15]	Computational (method 3)	-	Salt collision in key derivation; KCI attack for static DH; 64-bit security with short MACs	Use TH_2 as salt (in 16); recommend longer MACs; 4th message with AEAD gives 128-bit security
23	Ferreira (2024) [20]	Computational (all methods)	-	Need for 4th message for static DH; padding for identity privacy; AEAD choice affects security	Confirms 4th message for static DH; recommends padding; A256GCM preferred for high security
PSK (06)	This work	Symbolic (PSK method)	SAPIC+	Verified confidentiality, authentication, and key agreement; anonymity holds; unlinkability holds only for Initiator under passive attacker; SNDL breaks PFS due to the reliance on DH.	Confirms draft's security claims for PSK mode; clarifies precise identity-protection guarantees.

derived from PRK_2e. Message_2 is then composed by concatenating the ephemeral public key of the Responder, G_Y, and CIPHERTEXT_2A. EDHOC-PSK does not require the use of a MAC function in message_2, since there is no authentication at this point in the protocol.

- **Message 3:** The Initiator computes the shared secret G_XY, the pseudo-random keys PRK_2e and PRK_3e2m, and the decryption key KEYSTREAM_2A derived from PRK_2e. It then computes CIPHERTEXT_3B, output of the AEAD algorithm using EAD_3 as plaintext and TH_3 and the encryption of ID_CRED_PSK as externally supplied data, external_aad. It then generates CIPHERTEXT_3A by encrypting (again, using XOR) PLAINTEXT_3A = ⟨ID_CRED_PSK, CIPHERTEXT_3B⟩ with KEYSTREAM_3A, which also it derives in this step.
- **Message 4.** Message_4 is mandatory in EDHOC-PSK to achieve mutual authentication, specifically to authenticate the Responder. This guarantees explicit key confirmation. The key used for encrypting message_4 is PRK_{4e3m}, which is derived from CRED_PSK.

3.2 Claimed Properties

EDHOC-PSK sets out to guarantee the following properties, which are defined in draft 06 [30].

3.2.1 Mutual Authentication. In EDHOC-PSK, both peers must authenticate to each other in each handshake. Authentication must be guaranteed even in the presence of an active Man-in-the-Middle (MitM) attacker. This includes protection against Selfie or Reflection attacks, in which the attacker might fool a peer into believing it is communicating with the attacker, while it is talking to itself. On the other hand, as opposed to authentication methods 0 to 3, EDHOC-PSK is not required to (and cannot) guarantee Key-Compromise Impersonation (KCI), as attackers learning the PSK may impersonate either the Initiator or the Responder to the other peer.

3.2.2 Explicit Key Confirmation. EDHOC-PSK must ensure that, before the two peers accept the handshake as valid, they must be sure that they have both computed the same (secure) session-key.

3.2.3 Confidentiality. An active Man-in-the-Middle adversary must be unable to distinguish the session key from a random value of the same length. Moreover, EDHOC-PSK must guarantee Perfect-Forward-Secrecy (PFS): the corruption of a long-term PSK should

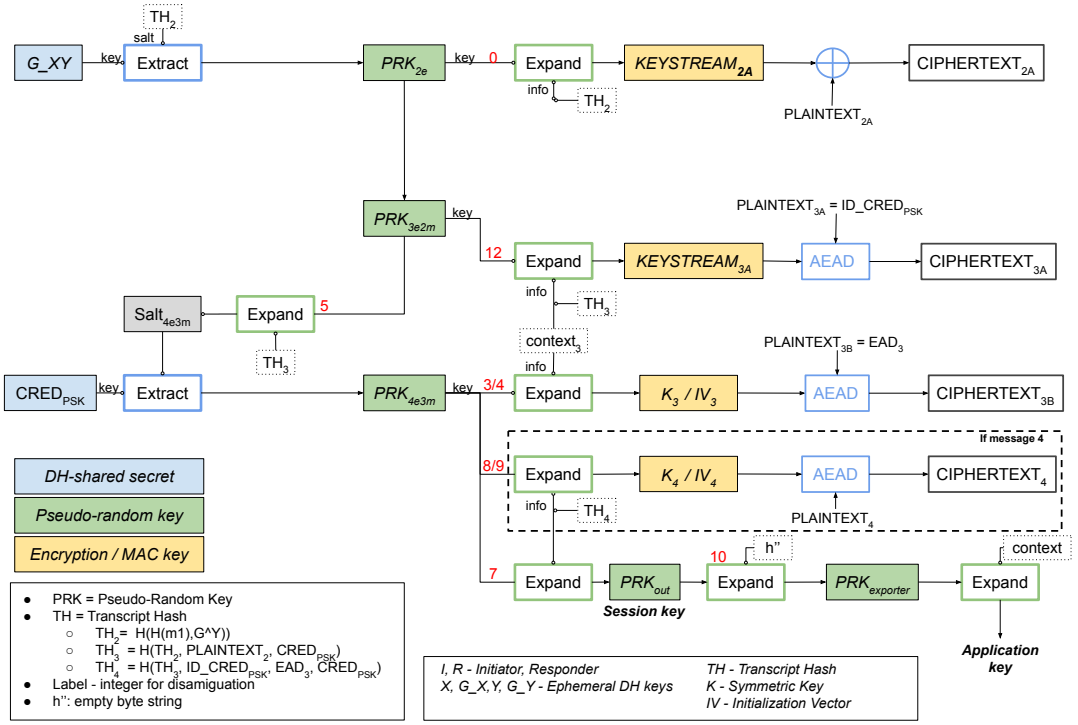


Figure 2: Key derivation scheme for PSK authentication method [31].

not allow the attacker to distinguish session keys of past sessions from random.

3.2.4 Identity Protection. EDHOC-PSK should protect the identities of both peers against passive Man-in-the-Middle attackers. Note that this is a slight weakening of the guarantees of traditional EDHOC, for which identity protection can be achieved against an active adversary for the initiator.

3.2.5 Post-Quantum Security. EDHOC-PSK aims to guarantee resistance to "*Store-Now-Decrypt-Later*" attacks by quantum computers. In particular, the protocol must guarantee identity-protection, and confidentiality for all sessions, as long as the PSK does not leak.

4 Background on Symbolic Analysis

This section provides some background on the symbolic verification of cryptographic protocols.

4.1 Adversary Model

Our analysis operates under the well-established Dolev-Yao (DY) adversary model [18]. This model assumes an active network adversary who has full control over the communication channel. The DY model treats messages as abstract terms within a term algebra, rather than assuming some bit-level representation, and cryptographic operations as functions over these terms. Mathematical properties of these functions are specified using equations, which can be exploited to generate new terms from old ones. This approach exposes the internal algebraic structure of messages. The DY attacker possesses the following capabilities:

- *Interception and manipulation:* The attacker can intercept, block, redirect, eavesdrop on, and alter messages over the network, as well as replay previously observed messages exchanged between honest parties.
- *Active communication:* The attacker can actively communicate with honest parties in protocol sessions, impersonating legitimate parties or initiating sessions.
- *Cryptographic operations:* The attacker can use cryptographic operations which are part of the protocol specification, including encryption, signing, and decryption.

A fundamental assumption of the DY model is the *perfect cryptography assumption*: the attacker cannot break cryptographic primitives themselves (e.g. by brute-forcing keys or exploiting weaknesses in the cryptographic algorithms). Therefore, our analysis of EDHOC with PSK focuses on logical flaws in the protocol design and message exchange, rather than vulnerabilities in the underlying cryptography.[5]

4.2 Verification Platform

For our formal security analysis of EDHOC, we use the SAPIC+ protocol verification platform [11]. SAPIC+ enables the use of multiple verification tools as backends. It currently supports Tamarin

[5]EDHOC relies on widely deployed and well-studied primitives, including curves P-256/Ed25519 for ECDH, SHA-256 for hashing, or AEAD schemes such as AES-CCM. These primitives are considered secure, lending credence to the perfect cryptography assumption.

and ProVerif, which we are going to use for our formal analysis, but it also supports GSVerif [9] and DeepSec [12].[6]

In SAPIC+, protocols are described using a dialect of the applied π-calculus [1], similarly to ProVerif. The property specification language, however, is that of Tamarin, and therefore uses a temporal first-order-logic. Its direct integration within Tamarin allows for the straightforward verification of input files, while also providing command-line options to produce valid input files for ProVerif. For more details on this framework, we recommend the documentation in [11].

4.3 Symbolic Modeling Principles

4.3.1 Messages as Terms. In symbolic models, messages are not treated as raw sequences of bits, rather as abstract terms. This approach highlights the structure of the message itself, ignoring the details of how it is actually represented or transmitted. For instance, an EDHOC message m_1 containing a connection identifier, an ephemeral public key, the cipher suite, and the method would be modeled as a composite term, usually a tuple containing four elements, one for each of the above objects, rather than dealing with the actual bytes and metadata transmitted.

4.3.2 Atomic Terms and Variables. Atomic terms are indivisible symbolic elements that represent basic values in the protocol model, i.e. nothing inside them can be extracted or examined. They may represent freshly generated data (e.g. an EDHOC ephemeral private key), but also fixed constants, agent identities, or any other uninterpreted values.

Variables, on the other hand, serve as placeholders for unknowns such as the inputs of a protocol, and are replaced with actual values during protocol execution. For example, if an agent expects to receive a two-field message $< X, D >$, whose contents are not yet known, the symbolic model may represent these variables using the notation m_2 =< X, D >. As execution proceeds, these variables are instantiated with concrete terms, such as atomic terms, nonces, keys, or ciphertexts, depending on what the agent actually receives.

4.3.3 Cryptographic Primitives as Function Symbols. Cryptographic operations, such as authenticated encryption and key derivation, are modeled using function symbols. The behavior of these functions can be specified by a set of equations i.e. an *equational theory*.

For AEAD, a fundamental building block of EDHOC, the symbol enc of arity 4 can be used, with the encryption operation being represented as enc(k, n, ad, m), where:

- k is the secret key,
- n is a unique value (nonce) for each encryption,
- ad is the associated data,
- m is the plaintext to encrypt.

The decryption function is represented as dec(k, n, ad, c), where the first three symbols have the same meanings as above, and c stands for the ciphertext being decrypted. This decryption function returns the original message if the ciphertext (c) and associated data (ad) are valid, or an error otherwise.

[6]GSVerif is used for stateful protocols, so we do not use it since our model of EDHOC-PSK is stateless. DeepSec is not used for our formal analysis as it does not support DH modeling.

Thus, for AEAD encryption, the following equation specifies the link between the encryption and decryption functions:

$$\mathsf{dec}(\mathsf{k}, \mathsf{n}, \mathsf{ad}, \mathsf{enc}(\mathsf{k}, \mathsf{n}, \mathsf{ad}, \mathsf{m})) = \mathsf{m}$$

This equation expresses the fact that decrypting a ciphertext produced by encrypting a message m (with key k, nonce n, and associated data ad) returns the original message m only upon using the same parameters. The only equalities between symbolic terms are those explicitly stated in the equational theory. In particular, equations may specify when function symbols can "cancel each other out"; otherwise, applying any function symbol to any term produces a "different" term.

4.3.4 Protocols as Processes. To formally specify the EDHOC protocol's behavior and interactions, we utilize a variant of the applied π-calculus [1] used by SAPIC+. The applied π-calculus is a language for modeling security protocols that allows to detail the actions of the participants in a protocol, emphasizing their communication. In contrast with the *pure* π-calculus, which lacks built-in functions, the *applied* π-calculus allows to form values from names (atomic terms) via the application of built-in functions (e.g. cryptographic operations), subject to equations, and their subsequent transmission as messages. The values known to the attacker are also defined through a built-in keyword K(t): K(t) is true whenever the attacker can deduce the term t from its initial knowledge, publicly available information, or any messages it has received or intercepted.

Rather than presenting a fully-formal description of the protocol, we illustrate the main features of the modeling language through representative examples. The following code snippet models the first message sent by the Initiator in an EDHOC handshake authenticated with PSK:

```
let I(~cid, method, ~credI, ~psk, ~cred) =
  in(<suitesI,C_I,EAD_1>);
  event MethodOk(method);
  let CRED_I = ~credI in
  let CRED_R = ~cred in
  let ID_CRED_PSK=make_id(CRED_I, CRED_R,~psk) in
  new ~X;
  if CRED_I = CRED_R then(
    0
  ) else
  ( CompromiseShare(~X) |
    event StartI(~cid,method,CRED_I,CRED_R,~psk);
    let G_X = 'g'^~X in
    let m1 = <method,suitesI,G_X,C_I,EAD_1> in
    out(m1);
```

The Initiator role takes as parameters a freshly generated connection identifier (~ cid, where ~ indicates that it is a fresh value), the used method (method four in this case), a freshly-generated pre-shared key ~ psk, and the identities of both itself and the intended Responder (~ cred_I and ~ cred_R respectively). The 'let ... in' construct is the standard term-evaluation and variable-binding mechanism of the applied π-calculus. in(< suitesI, C_I, EAD_1 >) indicates that the Initiator receives (from the network) a tuple consisting of three parts, namely the cipher suite to be used (indicated by the variable suitesI), a connection identifier (indicated by C_I), and the associated data EAD_1. Because inputs come from the public

network, the attacker may arbitrarily choose values to instantiate these variables with.

The instruction new allows us to generate fresh values (where freshness is indicated to the compiler by the ~ symbol).[7] In the above snippet of code, ~ X denotes the ephemeral DH value. The event construct records the occurrence of protocol-relevant actions without altering the control flow of the process. To prevent reflection and selfie attacks, the model explicitly checks that the Initiator and Responder identities are distinct. By G_X we refer to a Diffie-Hellman public value, modeled by raising the generator g to the freshly generated exponent ~ X, and by m1 the first protocol message, constructed as a tuple of protocol parameters. The exponentiation operation is defined as part of the built-in Diffie-Hellman equational theory, which is specified at the beginning of the file. In fact, we include three built-in equational theories, those for Diffie-Hellman, hashing, and xor.

```
theory edhoc_psk_sapic
begin
builtins: diffie-hellman, hashing, xor
```

4.3.5 Properties as First-Order Logic Formulas. While protocol modeling is specified using the applied-π-calculus, security properties can be modeled as first-order logic formulas over events and timepoints, following Tamarin's syntax. To specify a property to be verified, we use the keyword **lemma** followed by a name for the property and the corresponding formula. The following snippet shows an example of an executability lemma, used to verify that the protocol is indeed running as a sanity check:

```
lemma executableI_simple:
  exists-trace
  "Ex cid method psk subj_i #i.
  Start_I(cid, method, psk, subj_i) @i"
```

This states that there exists an execution where the event Start_I is triggered at a timepoint i, indicating that the Initiator has started a session with those values.

Additionally, security properties can be expressed as equivalence relations. In general terms, two processes P and Q, are said to be equivalent or indistinguishable (P ≡ Q) if an attacker cannot tell them apart. This type of relation is particularly suitable for modeling anonymity and unlinkability properties. For example, a protocol is said to guarantee the anonymity of participants if running the protocol with a participant of identity id_1 is indistinguishable to the attacker (or equivalent) from running it with a participant of identity id_2. We note that while Tamarin can verify equivalence properties, SAPIC+ does not support a Tamarin model when operating in equivalence mode. ProVerif is therefore the only tool we used to verify equivalence properties. The equivalence relation verified by ProVerif is *observational equivalence* [8].

5 Modeling EDHOC-PSK and its Properties

5.1 Modeling the EDHOC-PSK Protocol

The main process (reproduced below) defines the protocol's operational model:

[7]The instruction new is sufficient to generate fresh values in SAPIC+, and the ~ symbol is optional.

```
process:
  !(new ~psk; event Honest(~psk);
    !(new ~cid; new ~id; event HonestSubject(~id);
      out(~id);
      ((!in(cred);I(~cid,method_four(),~id,~psk,
          cred))
        | (!R(~cid, ~id, ~psk))
        | (compromise(~psk))
      )
    )
  )
```

The top-level replication indicated by the (!) symbol, following the applied-π syntax, creates an unbounded number of protocol instances, each containing a freshly generated PSK. This key, denoted by ~ psk, is immediately marked as honest (through the event annotation Honest(psk)) and remains secret unless explicitly compromised. For each ~ psk, the model then generates an unbounded number of peers, each equipped with their own fresh identity ~ id and connection identifier ~ cid. The identity is marked as honest using the event HonestSubject(id). Identities are output on the public channel to reflect that they may be known a priori to, and later supplied by the adversary. Each peer can act as an Initiator or as a Responder in any number of sessions, captured by the parallel composition indicated by |. This includes sessions where an agent takes on both the role of Initiator and Responder, hence reflection and selfie attacks can be captured within our model. Initiators obtain the identity of their peer from the attacker-controlled network, allowing the attacker to engage with honest Initiators. The attacker can also act as an Initiator choosing its own credentials. This abstraction, therefore, represents a system with multiple parallel protocol runs, where several peers share the same PSK, and the adversary can affect role instantiation and peer selection.

5.2 Extension to the Threat Model

We extend the DY threat model in a modular way, allowing the attacker to have additional capabilities. This is achieved by making compromises available as processes executed in parallel with the main process. We consider four compromise processes:

- compromise(psk) : compromise of the long-term PSK psk.
- LeakShare(x) : compromise of the ephemeral DH key x.
- LeakSKey(prk) : compromise of the session key prk.
- PQDL : a post-quantum discrete-logarithm oracle allowing the attacker to recover ephemeral DH secrets.

These extensions are fully modular: each leakage or adversary capability can be enabled independently or in combination with others. This modularity allows us to systematically explore how different attacker models affect the protocol security across a variety of threat scenarios.

5.3 Modeling Properties

Our formal analysis of the EDHOC protocol with PSK aims at verifying the security properties defined in Section 3.2, which are formalized using first-order logic formulas or equivalence relations.

5.3.1 Full Agreement. We formalize authentication and key agreement according to Lowe's hierarchy of authentication properties [32].

We focus on full (injective) agreement, which is the symbolic analogue of AKE correctness [17]. We therefore verify that *(1)* if an honest party completes a session with an honest peer, then there exists a unique matching session run by the intended peer and *(2)* both peers agree on all session data, including identities, session identifiers, and the derived session key. This definition concerns the internal correctness of the protocol execution between honest parties (i.e. in the absence of an active adversary), and it additionally implies that both sides derive the same session key.

To prove that there is full injective agreement from the Responder's point of view, we state a lemma called **full_agreement_RI**, which asserts that if a Responder concludes a session (indicated by the event annotation AcceptR), then an Initiator with the corresponding identity should also have the necessary values to derive matching session parameters (event DerivedI). Moreover, for a given Initiator-Responder pair and a given session key, a Responder can participate in such a session only once. This ensures injectivity: each session completed by a Responder with a given session key and identifiers corresponds to exactly one matching Initiator session. This can be modeled as:

```
lemma full_agreement_RI:
  All cid1 cid2 m credI1 credI2 credR1 credR2 prk
      gx1 gx2 y1 y2 psk #i #j #k #l #c.
     AcceptR(cid1, m, credI1, credR1, prk, gx1, y1
         , psk)@i & AcceptR(cid2, m, credI2,
         credR2, prk, gx2, y2, psk)@j & Honest(psk
         )@k & HonestSubject(credI1)@l &
         HonestSubject(credI2)@c
   ==> (Ex cid credI credR x gy prk #t. DerivedI(
       cid, m, credI, credR, prk, x, gy, psk)@t &
       t < i & t < j & cid1 = cid2 & gx1 = gx2 &
       y1 = y2 & credI1 = credI2 & credR1 = credR2
        & credI = credI1 & credR = credR1 & #i = #
       j)
    | (Ex #t. Compromise(psk)@t & t < j)
```

The lemma includes an alternative scenario where the long-term PSK has been compromised before the session completes. In this case, injective agreement cannot be guaranteed, which is why the "Compromise(psk)" branch is included.

One can write a symmetric lemma **full_agreement_IR** to represent full injective agreement from the point of view of the Initiator.

5.3.2 Confidentiality. The lemmas in this section verify whether honest parties establish a shared, secret session key, and whether this key remains confidential from the adversary.

Confidentiality of the Session Key. We verify whether the negotiated session key (prk) remains confidential to the legitimate protocol participants and cannot be learned by the DY adversary [8].

This is modeled as follows through the **secret_psk** lemma

```
lemma secret_psk:
  All cid1 cid2 m subj_i subj_r prk x gy y gx psk
      #i #j #k #p.
     AcceptI(cid1, m, subj_i, subj_r, prk, x, gy,
         psk)@i & AcceptR(cid2, m, subj_i, subj_r,
          prk, gx, y, psk)@j & K(prk)@k & Honest(
         psk)@p
   ==>
   (Ex #t #u. Compromise(psk)@t & t < k &
       LeakShare(x)@u & u < k)
   | (Ex #t #u. Compromise(psk)@t & t < k &
       LeakShare(y)@u & u < k)
   | (Ex #t. LeakSKey(prk)@t)
```

In essence, if both parties accept with the same parameters and the attacker learns the session key prk, then one of the following must have occurred: *(1)* the session key *prk* was compromised (witnessed by the event annotation LeakSKey(·)), or *(2) psk* was compromised before the attacker learned prk (event Compromise(·)) *(3)* one of the private ephemeral values was compromised (event LeakShare(·)).

Perfect Forward Secrecy (PFS). This property captures that any compromise of long-term secrets (in this case, the PSK) does not compromise the confidentiality of previously-established session keys. This is expressed by the lemma pfs as follows:

```
lemma pfs:
    All cid1 cid2 m credI credR prk psk gy x gx y
        #i #j #c #k #l #p #u. AcceptI(cid1, m,
        credI, credR, prk, x, gy, psk)@i & AcceptR
        (cid2, m, credI, credR, prk, gx, y, psk)@j
         & Compromise(psk)@c  & i < c & Honest(psk
        )@k & HonestSubject(credI)@p &
        HonestSubject(credR)@u & K(prk)@l & j < i
     ==>
     (Ex #t. LeakSKey(prk)@t & t < l)
     | (Ex #t. LeakShare(x)@t & t < l )
     | (Ex #t. LeakShare(y)@t & t < l )
```

The lemma states that, if the PSK is compromised after the session is complete (both Accept events have occurred), the attacker can get to know the session key only if one of the ephemeral DH secrets was leaked (event LeakSKey(·)), or the session key itself was leaked (event LeakShare(·)).

5.3.3 Identity Protection. The EDHOC-PSK draft claims that the protocol protects the identities of both peers against passive attackers. However, the term identity protection is not used uniformly in the security-protocols literature: different works refer to different, though related, notions. We adopt the standard symbolic notions of anonymity and unlinkability introduced by Chothia and Paquet [2].

Both are expressed as diff-equivalence properties in ProVerif, following Blanchet *et al.* [7]. Formally, if $P(x)$ denotes an EDHOC protocol execution where the peer whose identity we are interested in protecting is the parameter x, then anonymity corresponds to establishing:

$$P(id_1) \equiv P(id_2), \text{ for } id_1 \neq id_2.$$

meaning that no adversarial context can distinguish a run with identity id_1 from one with id_2. ProVerif provides explicit support for

[8] Although confidentiality is stated as an indistinguishability property (see Section 3.2, our lemma establishes it as a non-derivability property. In the perfect-cryptography model, non-derivability of a term by the adversary can be equivalent to its indistinguishability from any other fresh value; hence, proving secrecy may imply key indistinguishability. A discussion of the formal relationship between non-derivability and indistinguishability can be found in [14].

this reasoning by introducing the constructor diff[M_1, M_2], which denotes a symbolic value that evaluates to M_1 on the left-hand side of the equivalence and to M_2 on the right-hand side.

Anonymity. Informally, it means that an attacker cannot determine which party executed a given protocol instance. In this sense, anonymity is not intended to protect the user's identity, but rather the link between a use of a service and the identity of the user.

We encode this using ProVerif's diff-equivalence and model the Initiator's (resp. Responder's) identity using a symbolic choice:

```
process
    [...]
    let idI:bitstring=choice[id1, id2] in
    let cidI:bitstring=choice[cid1, cid2] in
    (( !R(cidR, idR, psk) )
      |
      ( !I(cidI, method_four, idI, psk, idR) ))
```

This corresponds to single-session indistinguishability, in the sense of [2]: the attacker must not distinguish whether the single symbolic session uses *id*_1 or *id*_2.

Unlinkability. Unlinkability requires that an attacker cannot determine whether two (or more) sessions were executed by the same party. In our analysis, we adopt the standard symbolic interpretation: one session uses a symbolic identity, while a second session uses a fixed identity. The attacker attempts to determine whether these two sessions belong to the same principal.

We encode this by adding a second Initiator (resp. Responder) that uses a known identity. Concretely, we model two sessions: one that always uses identity id_1, and a second that uses either id_1 or id_2. This yields two worlds: (i) both sessions use id_1, and (ii) the sessions use different identities. We then prove that these two worlds are indistinguishable to the attacker, implying that the attacker cannot link sessions executed by the same identity.

```
set attacker = passive.
process
    [...]
    let idI:bitstring=choice[id1, id2] in
    let cidI:bitstring=choice[cid1, cid2] in
    (( !R(cidR, idR, psk) )
      |
      ( !I(cidI, method_four, idI, psk, idR) )
      |
      ( !I(cid1, method_four, id1, psk, idR ) ))
```

5.3.4 Post-Quantum Security. To capture post-quantum security in our formal analysis, we model the capability of an adversary to record all exchanged messages and later break the ephemeral Diffie–Hellman values (Store-Now-Decrypt-Later attack (SNDL) [34]). This corresponds to a quantum attacker that passively observes protocol flows today and then, in a later phase, uses a quantum computer to compute discrete logarithms.

ProVerif supports such reasoning through the `phase` construct, which allows the protocol to be divided into sequential stages. The attacker is assumed to have full access to all messages from earlier phases, but cannot access capabilities of later phases prematurely. This lets us model that the ephemeral DH keys are secure in phase 0 (classical world) but fully breakable in phase 1 (quantum world).

Formally, we introduce a phase 1 process in which the attacker receives DH public values and applies a hypothetical quantum-enabled Discrete Logarithm (DL) function to recover the ephemeral secrets:

```
process
[...]
  (
   (!(in(att,cred:bitstring);I(cid,
       method_four, id, psk, cred)))
   | (!R(cid, id, psk))
   | (compromise(psk))
   | phase 1; in(att, G:bitstring); out(att,
       DL(G,g))
  )
```

Here, the phase 1 block ensures that the DL oracle is only available after all protocol interactions have taken place. We then verify whether session secrecy still holds when ephemeral DH secrets are revealed.

6 Methodology

This section outlines the methodology we use to analyze the models described in Section 5. Our analysis relies on a single SAPIC+ specification of EDHOC-PSK, from which we automatically generate the input files for both ProVerif and Tamarin. The purpose of this section is not to provide a tutorial on SAPIC+, rather to explain how we combine both verification engines to obtain trustworthy and complementary results.

ProVerif and Tamarin differ in expressiveness and proof techniques. ProVerif offers fully automated verification and is particularly effective for reachability and diff-equivalence reasoning, making it well-suited for identity-protection properties and sanity checks. Tamarin combines automatic and interactive proofs and includes a built-in equational theory for Diffie–Hellman exponentiation, allowing precise reasoning about algebraic properties and subtle attacker behaviors.

These capabilities are complementary. In rare cases, ProVerif may report a property as holding that Tamarin later refutes. In such cases, we consider Tamarin's counterexamples as valid, as they rely on a more precise algebraic model and interactive proof exploration. For equivalence properties such as anonymity and unlinkability, ProVerif's diff-equivalence engine is the current state of the art.

Step 1: Sanity checks in ProVerif. We begin with ProVerif to ensure that the specification is well-formed and that honest participants can complete executions. This provides rapid model feedback before engaging in more expensive analysis.

Step 2: Checking for partial deconstructions in Tamarin. We then examine the Tamarin translation, in particular the automatically generated multiset-rewrite rules, to ensure the absence of *partial deconstructions*: destructor applications that syntactically fire but cannot reduce. This can be checked automatically using the `--precompute-only` flag in the command line. For example, if we have a function `pair(x, y)` and a destructor `fst(pair(x, y)) = x`, then `fst(enc(k, m))` is a partial deconstruction. Indeed, it looks like an

application of fst but it cannot be reduced since it is not applying the correct destructor (pair). By eliminating partial deconstructions, we ensure that the translation from SAPIC+ to Tamarin accurately reflects the intended cryptographic model.

Step 3: Authentication, confidentiality and key-agreement proofs in Tamarin. Once we have confidence in the correctness of the protocol specification and the absence of partial deconstructions, we conduct a more detailed analysis in Tamarin, taking into account the algebraic structure of Diffie-Hellman exponentiation and other cryptographic operations. The interactive mode also allows the inspection of potential attack traces that might not be visible in ProVerif.

Step 4: Anonymity and Unlinkability Properties in ProVerif. Identity-protection properties are expressed as observational equivalence in ProVerif, following the standard methodology of Blanchet *et al.* [7].

We distinguish two flavors of identity protection:

- Anonymity: We model unbounded sessions, but the identity of the party under test (Initiator or Responder) is represented by a symbolic choice between two identities. ProVerif proves observational equivalence between the two protocol executions with different identities.
- Unlinkability: To model unlinkability, we run an additional Initiator (resp. Responder) instance using a fixed identity while the other session uses a symbolic choice. ProVerif then checks that the attacker cannot link the chosen identity to the concrete session.

If ProVerif proves diff-equivalence, then the attacker observes indistinguishable traces and cannot infer the identities of the communicating peers.

7 Results

Our experiments are performed on a 13th Gen Intel(R) Core(TM) i7-1370P with 16GB of RAM. To automate the procedure, we use a Python script that proves each lemma independently under various attacker capabilities.

Table 3 summarizes the results of our automated analysis of draft 06 of EDHOC-PSK. The columns correspond to different symbolic attacker capabilities, defined in Section 5:

- *Basic.* This is the standard DY model, extended with the attacker's ability to compromise the PSK. All other attacker models build on top of this one.
- *DHShare.* In this model the adversary can learn the ephemeral DH exponent of one party (either X or Y). It is relevant for testing properties that should remain secure even if DH secrets leak, such as mutual authentication or key agreement.
- *SesKeyShare.* Here the attacker gains access to the session key. This is similarly useful for testing whether authentication and agreement guarantees continue to hold despite session-key compromise.
- *DHShare & SesKeyShare.* This combines the previous two capabilities: the attacker can simultaneously obtain a DH exponent and session key.
- *Post-Quantum (SNDL).* In the Store-Now-Decrypt-Later model, the attacker can eventually recover all DH exponents, modeling the impact of a future quantum computer.

For each security property, we list the supporting lemmas/equivalence relations that were proven and report the verification outcome together with the corresponding proof time. When we report a positive verification result (✓) under a given attacker model, this should be interpreted as follows: the property holds in all traces except those explicitly excluded by the compromise conditions appearing in the lemma statement. Thus, a ✓result indicates correctness *conditional* on the negation of the compromise events explicitly listed in the lemma's right hand side, not correctness in the presence of compromise.

Mutual Authentication. Contrary to EDHOC methods 0 to 3, a fourth message is mandatory in EDHOC-PSK in order for the Responder to authenticate, hence allowing the possibility for mutual authentication. Indeed, an important consideration in EDHOC-PSK is that the PSK information is only included for constructing the pseudorandom key PRK_4e3m. This design renders the ephemeral DH shared secret G_XY vulnerable to active attacks such as Man-in-the-Middle (MitM), as early protocol messages do not cryptographically bind the participants until the PSK is mixed in. As a result, no message or key authentication against active adversaries is guaranteed before this point. The keys used for AEAD encryption in messages 3 and 4 are computed using PRK_4e3m and are therefore authenticated. For instance, when the Responder receives message 3 it verifies the AEAD. If AEAD verification fails, this indicates a processing problem or that the message was tampered with. If it succeeds, the Responder concludes that the Initiator possesses the PSK, correctly derived TH_3, and is actively participating in the protocol. Similarly for the Initiator upon receiving message 4.

Formulating authentication-related lemmas relies on the fact that only messages and session keys computed after incorporation of the PSK can be cryptographically authenticated. This is why events such as AcceptI, AcceptR, or DerivedI are defined after the PSK has been used in the protocol. From Table 2, we see that the protocol offers full agreement, meaning injective authentication guarantees.

Explicit Key Confirmation. In order to provide explicit key confirmation, a fourth message is needed. Without it, only implicit key confirmation would be guaranteed. This property is verified by means of the lemma full_agreement_IR. It ensures that the Initiator's acceptance of the session key (event AcceptI) is explicitly confirmed by the Responder's prior acceptance of the same key (event AcceptR).

Confidentiality. Lemma secretR_psk establishes confidentiality of the session key from the Responder's point of view. It states that whenever a Responder accepts a session with key prk, an attacker can only learn this key if either: *(i)* the PSK was compromised before the key is revealed, or *(ii)* the session key itself was leaked. Importantly, no leakage of DH shares is required in this lemma: if both the PSK and identities are compromised, the attacker can simply impersonate one party from the start. Similarly, lemma secretI_psk proves confidentiality from the Initiator's perspective.

A stronger result is given by lemma secret_psk, which captures secrecy when both peers have successfully completed the protocol with matching parameters. In this case, the PSK and one of the peers' ephemeral DH secrets must have been compromised prior to key

Property	Lemma	Basic		LeakShare		LeakSKey		LeakShare & LeakSKey		PQ (SNDL)	
		Result	Time (s)	Result	Time (s)	Result	Time (s)	Result	Time (s)	Result	Time (s)
Authentication and Key Agreement	full_agreement_IR	✓	46.13	✓	47.30	✓	44.28	✓	42.93	✓	5.85
	full_agreement_RI	✓	45.11	✓	50.77	✓	45.97	✓	46.57	✓	7.581
Confidentiality	secretR_psk	✓	34.71	✓	48.15	✓	40.46	✓	44.07	✓	15.41
	secretI_psk	✓	41.37	✓	45.62	✓	43.24	✓	45.47	✓	10.66
	pfs	✓	107.58	✓	118.58	✓	116.05	✓	120.19	×	-
	secret_psk	✓	73.81	✓	65.94	✓	67.37	✓	71.77	✓	18.11
Identity Protection	anonymity_I_active	✓	22.30	✓	81.45	✓	19.19	✓	84.61	✓	337.55
	anonymity_R_active	✓	24.63	✓	126.95	✓	21.94	✓	122.23	✓	569.38
	unlinkability_I_active	×	-	×	-	×	-	×	-	×	-
	unlinkability_I_passive	✓	5.18	✓	7.90	✓	5.84	✓	8.25	×	-
	unlinkability_R_active	×	-	×	-	×	-	×	-	×	-
	unlinkability_R_passive	×	-	×	-	×	-	×	-	×	-

Table 3: Summary of automated verification results of EDHOC-PSK across adversary capabilities. The classic results are computed using Tamarin. The Post-Quantum (PQ) results are computed using ProVerif.

derivation. This reflects the fact that once both parties contribute DH shares to the transcript, the session key can only be computed if the adversary learns at least one of the ephemeral exponents in addition to the PSK.

Lemma pfs formalizes that the protocol provides perfect forward secrecy: even if the long-term PSK is compromised after a session completes, all previously established keys remain confidential unless an ephemeral exponent was leaked during the execution of that session.

When the PSK and the key-derivation steps (e.g. the computation of PRK_4e3m) are confined to a Trusted Execution Environment (TEE), the confidentiality guarantees above become significantly stronger. A TEE isolates the PSK and all intermediate keying material from the rest of the device, preventing an attacker with control over the main operating system from reading memory, extracting secrets, or observing intermediate values [19].

This has two major consequences:

- The PSK cannot be obtained through standard device compromise. Since the PSK never leaves the TEE, attackers who compromise the normal world (e.g. via malware, physical access to RAM, or debugging interfaces) cannot obtain it.
- The attacker cannot satisfy the secrecy-breaking preconditions of the lemmas. Indeed, all attacks that violate confidentiality require prior PSK compromise. If the PSK is protected by a TEE, these conditions become unattainable unless the TEE itself is breached.

Identity Protection. Our ProVerif analysis considers two levels of identity protection as defined in Section 3.2: anonymity and unlinkability. We analyze both properties for the Initiator and the Responder, under both passive and active attackers. Our anonymity and unlinkability results correspond to a symbolic model with two identities, under the assumption that `ID_CRED_PSK` is not identity-unique and is role-independent.

We prove that EDHOC-PSK satisfies anonymity for both the Initiator and the Responder in the sense of Chothia and Paquet [2], even against active attackers, and in both the classical and post-quantum setting (equivalence relations anonymity_I_active and anonymity_R_active). This means that an active adversary cannot discover whether a session was executed under identity *id*_1 or *id*_2.

We prove that EDHOC-PSK provides unlinkability (as defined in [2]) for the Initiator under a passive attacker in a classical setting (witnessed by the equivalence relation unlinkability_I_passive). In this setting, the attacker cannot determine whether two protocol sessions originate from the same Initiator. Unlinkability does not hold for the Responder even under passive attackers, nor for either role under active attackers. The root cause is structural: the Initiator chooses a concrete Responder identity, and the value `ID_CRED_PSK` sent in message_3 is bound to that choice. Thus, in an unlinkability biprocess where the two Responder identities differ only in the diff-branch, an attacker can replay or reroute an Initiator-to-Responder flow to the "wrong" Responder. The resulting decryption failure (or acceptance) allows the attacker to link the session to a specific Responder identity. Hence, EDHOC-PSK cannot achieve unlinkability for the Responder in the presence of an active attacker.

Because unlinkability fails against active attackers, EDHOC-PSK does not satisfy the identity protection notion described by Cottier and Pointcheval [15]. This property requires security even against an active man-in-the-middle (MitM) adversary who may modify encrypted identity information (e.g. a forged `ID_CRED_PSK`), and observe whether the peer continues or aborts. Indeed, in EDHOC-PSK, such an active adversary can mount a chosen-ciphertext attack on the encrypted identity `ID_CRED_PSK` contained in message_3. Because message_3 is protected by an AEAD, any modification of the ciphertext causes the authentication to fail. However, the attacker can observe the Responder's behavior when processing the malformed message. In particular, differences in observable failure behavior – such as distinct error messages (e.g., unknown credential" versus authentication failure") or timing differences – may leak information about whether the manipulated identity corresponds to a valid credential. By exploiting these observable differences, a MitM attacker can distinguish the Initiator's actual identity from other candidate identities, violating strong anonymity, even though the AEAD tag prevents successful impersonation. A more detailed explanation of the attack can be found in Appendix A.

Post-Quantum Security. EDHOC-PSK derives authentication and session keys primarily from a pre-shared key (PSK). Symmetric-key primitives remain secure against quantum adversaries, except for a quadratic speedup due to Grover's algorithm. That is, a PSK

of sufficient entropy (e.g. ≥128 bits) provides post-quantum security [21, 34]. Consequently, even if ephemeral ECDHE keys are broken by a quantum adversary (i.e. SNDL attack), the PSK still ensures secrecy of the session key and authentication.

To model this, we analyze EDHOC-PSK in a Post-Quantum (PQ) adversarial model, where the attacker can record all DH public values G_X and G_Y during protocol execution and later recover the corresponding exponents using a Discrete Logarithm (DL) oracle. This models the SNDL attack, in which a future quantum computer breaks the hardness of ECDH. We build a post-quantum variant of the Basic verification model by extending it with a PQDL oracle. The reason is that the *DHShare* and *DHShare & SesKeyShare* models already grant the attacker direct access to ephemeral DH secrets; hence, adding a PQ DL oracle would not increase the attacker's power. Our PQ analysis therefore focuses on the meaningful case in which DH secrecy is normally preserved but becomes breakable only after the protocol run. From Table 3 we see that confidentiality of the session key is preserved, and so is authentication and key agreement. However, in the presence of the *DL oracle*, EDHOC-PSK does not provide perfect forward secrecy (PFS), nor does it ensure unlinkability for either the Initiator or the Responder. Once the ephemeral Diffie–Hellman secrets are recoverable, the attacker can derive the shared secrets used to protect identity-related information in the protocol messages. Regarding forward secrecy, EDHOC-PSK uses the long-term pre-shared key (PSK) in the key derivation, so the final session key depends on both the ephemeral Diffie–Hellman secret and the PSK. When the DL oracle leaks the ephemeral DH values after the session has completed, an attacker who subsequently compromises the PSK can recompute the PRK and derive all session keys of past sessions.

Disclosure procedure. Following the IETF's call for formal analysis of EDHOC-PSK [9], we have notified the LAKE WG and authors of relevant analyses about the results. Additionally, we aim to present our findings during the IETF 125 meeting (March 2026), allowing the community to review and discuss any potential issues.

8 Future Work

In this work, we adopt the perfect cryptography assumption, i.e. cryptographic primitives are modeled as ideal black-box operations that cannot be broken. This approach simplifies the analysis while still allowing us to detect a broad range of protocol-level attacks and logical flaws. However, symbolic models may fail to capture weaknesses that arise from the concrete instantiation of primitives.

For example, Cheval *et al.* [10] show that although many cryptographic protocols assume hash functions to be perfect (as in the Random Oracle Model (ROM)), this assumption does not accurately reflect reality. The ROM, introduced by Bellare and Rogaway [4], models a hash function as a perfect random function: for every new input, the output is chosen uniformly at random, and the same input always produces the same output. Under this abstraction, hash functions satisfy collision resistance (given H it is hard to find $y \neq x$ such that $H(y) = H(x)$), preimage resistance (given $H(x)$, it is infeasible to find x), and second-preimage resistance (given x, it is infeasible to find $y \neq x$ such that $H(y) = H(x)$). In practice, these ideal properties do not always hold. Collisions are unavoidable for any fixed-length hash function and have already been demonstrated for weaker algorithms such as SHA-1. Also, the ROM abstraction does not capture structural weaknesses such as *length extension attacks* that affect Merkle–Damgard constructions [39]. In such cases, given $H(x)$ and the length of x, an attacker can compute $H(x\|y)$ for arbitrary y without knowing x.

A similar limitation arises with Diffie–Hellman exponentiation. For instance, the existence of an identity element e such that $e^x = e$ for all x is ignored in symbolic models. Furthermore, for some elliptic curves, there exist small subgroups $h_1, \ldots, h_n$ that introduce additional algebraic properties, such as: *(1)* collisions of the form $h_i^x = h_j^y$, and *(2)* given g^x, with non-negligible probability, $(h_i g^x)^z = g^{xz}$. Such subgroup-related issues are invisible in a perfect model but can lead to real-world attacks if not taken into account. An interesting direction for future work is therefore to relax this assumption. Recent research has shown that it is possible to extend tools such as Tamarin and ProVerif with such models, thereby providing a more faithful representation of real-world cryptographic algorithms. Integrating such models into SAPIC+ would allow a more flexible and realistic analysis of EDHOC and similar IoT protocols.

9 Conclusions

In this paper, we perform an automated analysis of EDHOC authenticated with PSK (draft version 6). Using a unified SAPIC+ model, we verify EDHOC-PSK security properties across two complementary tools: ProVerif and Tamarin. ProVerif allows us to efficiently prove equivalence-based privacy properties such as anonymity and unlinkability, and it supports the same lemmas and reasoning principles we use for confidentiality and authentication. Tamarin provides more detailed control over proofs and relies on a more precise built-in equational theory for Diffie–Hellman exponentiation. Although both tools implement the same symbolic cryptographic assumptions, they differ in expressiveness, automation, and proof paradigms, and using them in combination increases confidence in the robustness of our results.

We confirm the authentication, confidentiality, and key agreement properties claimed in the specification. We also analyze two flavors of identity protection: anonymity and unlinkability. Our analysis shows that anonymity holds for both the Initiator and the Responder even in the presence of active attackers. In contrast, unlinkability holds only for the Initiator, and only against passive attackers in the classical setting. We further investigate post-quantum security implications. In particular, we show that under the SNDL attack, EDHOC-PSK does not provide perfect forward secrecy or Initiator unlinkability against a passive attacker. However, it still guarantees session key secrecy, anonymity, and full agreement, provided that the PSK has sufficient entropy.

Acknowledgments

This work was funded by the French government under the France 2030 ANR program "PEPR Networks of the Future" (ref. ANR-22-PEFT-0009), and ANR Project PRIVA-SIQ (ref. ANR-23-CE39- 0008), and from the European Union's Horizon Europe Framework Programme under Grant Agreement No. 101093046.

[9] https://mailarchive.ietf.org/arch/msg/lake/A4YOMyFDQunzYDVSlsjt_C6yaRE/

References

[1] Martín Abadi, Bruno Blanchet, and Cédric Fournet. 2017. The Applied Pi Calculus: Mobile Values, New Names, and Secure Communication. *J. ACM* 65, 1 (2017), 41 pages.

[2] Myrto Arapinis, Tom Chothia, Eike Ritter, and Mark Ryan. 2010. Analysing Unlinkability and Anonymity Using the Applied Pi Calculus. In *IEEE Computer Security Foundations Symposium*. 107–121.

[3] Alessandro Armando, David Basin, Yohan Boichut, Yannick Chevalier, Luca Compagna, Jorge Cuellar, Paul Hankes Drielsma, Pierre-Cyrille Héam, Olga Kouchnarenko, Jacopo Mantovani, Sebastian Mödersheim, David von Oheimb, Michaël Rusinowitch, Judson Santos Santiago, Luca Vigano, Mathieu Turuani, and Laurent Vigneron. 2005. The AVISPA Tool for the automated validation of internet security protocols and applications. In *Lecture Notes in Computer Science (Lecture Notes in Computer Science, Vol. 3576)*. Springer, Edinburgh, Scotland/UK, France, 281–285.

[4] Mihir Bellare and Phillip Rogaway. 1993. Random Oracles are Practical: a Paradigm for Designing Efficient Protocols. In *ACM Conference on Computer and Communications Security (CCS)*. Association for Computing Machinery, New York, NY, USA, 62–73.

[5] Karthikeyan Bhargavan, Vincent Cheval, and Christopher Wood. 2022. A Symbolic Analysis of Privacy for TLS 1.3 with Encrypted Client Hello. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22)*. Association for Computing Machinery, New York, NY, USA, 365–379.

[6] Bruno Blanchet. 2016. Modeling and Verifying Security Protocols with the Applied Pi Calculus and ProVerif. In *Modeling and Verifying Security Protocols with the Applied Pi Calculus and ProVerif*. Number 1-2 in Foundations and Trends® in Privacy and Security. 1 – 135. doi:10.1561/3300000004

[7] Bruno Blanchet, Martín Abadi, and Cédric av. 2008. Automated verification of selected equivalences for security protocols. *The Journal of Logic and Algebraic Programming* 75, 1 (2008), 3–51.

[8] Bruno Blanchet and Ben Smyth. 2016. Automated Reasoning for Equivalences in the Applied Pi Calculus with Barriers. In *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*. 310–324. doi:10.1109/CSF.2016.29

[9] Vincent Cheval, Véronique Cortier, and Mathieu Turuani. 2018. A Little More Conversation, a Little Less Action, a Lot More Satisfaction: Global States in ProVerif. *2018 IEEE 31st Computer Security Foundations Symposium (CSF)* (2018), 344–358. https://api.semanticscholar.org/CorpusID:49213331

[10] Vincent Cheval, Cas Cremers, Alexander Dax, Lucca Hirschi, Charlie Jacomme, and Steve Kremer. 2023. Hash Gone Bad: Automated discovery of protocol attacks that exploit hash function weaknesses. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association, Anaheim, CA, 5899–5916.

[11] Vincent Cheval, Charlie Jacomme, Steve Kremer, and Robert Künnemann. 2022. SAPIC+: protocol verifiers of the world, unite!. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, 3935–3952.

[12] Vincent Cheval, Steve Kremer, and Itsaka Rakotonirina. 2024. DeepSec: Deciding Equivalence Properties for Security Protocols – Improved theory and practice. *TheoretiCS* Volume 3 (March 2024). doi:10.46298/theoretics.24.4

[13] Cisco Systems. 2020. *8.5 Identity PSK Feature Deployment Guide*. https://www.cisco.com/c/en/us/td/docs/wireless/controller/technotes/8-5/b_Identity_PSK_Feature_Deployment_Guide.html Describes the configuration and deployment of Identity PSK (iPSK) for device-specific or group-specific pre-shared keys in enterprise and IoT environments.

[14] Véronique Cortier, Michaël Rusinowitch, and Eugen Zalinescu. 2007. Relating two standard notions of secrecy. *CoRR* abs/0706.0502 (2007).

[15] Baptiste Cottier and David Pointcheval. 2022. Security Analysis of Improved EDHOC Protocol. In *Foundations and Practice of Security: 15th International Symposium, FPS 2022, Ottawa, ON, Canada, December 12–14, 2022, Revised Selected Papers*. Springer-Verlag, Berlin, Heidelberg, 3–18.

[16] Cas Cremers, Marko Horvat, Jonathan Hoyland, Sam Scott, and Thyla van der Merwe. 2017. A Comprehensive Symbolic Analysis of TLS 1.3. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17)*. Association for Computing Machinery, New York, NY, USA, 1773–1788.

[17] Cyprien Delpech de Saint Guilhem, Marc Fischlin, and Bogdan Warinschi. 2020. Authentication in Key-Exchange: Definitions, Relations and Composition. In *2020 IEEE 33rd Computer Security Foundations Symposium (CSF)*. 288–303.

[18] D. Dolev and A. Yao. 1983. On the security of public key protocols. *IEEE Transactions on Information Theory* 29, 2 (1983), 198–208.

[19] Jan-Erik Ekberg, Alexandra Afanasyeva, and N. Asokan. 2012. Authenticated Encryption Primitives for Size-Constrained Trusted Computing. In *Trust and Trustworthy Computing*, Stefan Katzenbeisser, Edgar Weippl, L. Jean Camp, Melanie Volkamer, Mike Reiter, and Xinwen Zhang (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–18.

[20] Loïc Ferreira. 2024. Computational Security Analysis of the Full EDHOC Protocol. In *Cryptographers' Track at the RSA Conference (CT-RSA)*. Springer Nature Switzerland, Cham, 25–48.

[21] Lov K. Grover. 1998. A framework for fast quantum mechanical algorithms. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing (STOC '98)*. Association for Computing Machinery, New York, NY, USA, 53–62.

[22] Felix Günther and Marc Ilunga Tshibumbu Mukendi. 2023. Careful with MAc-then-SIGn: A Computational Analysis of the EDHOC Lightweight Authenticated Key Exchange Protocol. In *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*. IEEE, 773–796.

[23] Internet Assigned Numbers Authority (IANA). [n. d.]. *IANA EDHOC Parameters Registry*. Accessed: 2025-02-10.

[24] Charlie Jacomme, Elise Klein, Steve Kremer, and Maïwenn Racouchot. 2023. A comprehensive, formal and automated analysis of the EDHOC protocol. In *Proceedings of the 32nd USENIX Conference on Security Symposium (SEC '23)*. USENIX Association, USA, Article 329, 18 pages.

[25] Jiyoon Kim, Daniel Gerbi Duguma, Sangmin Lee, Bonam Kim, JaeDeok Lim, and Ilsun You. 2021. Scrutinizing the Vulnerability of Ephemeral Diffie–Hellman over COSE (EDHOC) for IoT Environment Using Formal Approaches. *Mobile Information Systems* (2021), 1–18.

[26] J. Koo. 2021. *Internet of Things Security Case Studies and Solutions*. Ph. D. Dissertation. California State University, San Bernardino. https://scholarworks.lib.csusb.edu/cgi/viewcontent.cgi?article=2455&context=etd Contains case studies of IoT products (e.g., smart meters, EV chargers, IP cameras) using PSK or password authentication, and discusses their vulnerabilities.

[27] Hugo Krawczyk. 2003. SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and Its Use in the IKE Protocols. In *Advances in Cryptology - CRYPTO 2003*. Springer Berlin Heidelberg, Berlin, Heidelberg, 400–425.

[28] Fang-Chun Kuo, Hannes Tschofenig, Fabian Meyer, and Xiaoming Fu. 2006. Comparison Studies between Pre-Shared and Public Key Exchange Mechanisms for Transport Layer Security. In *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*. IEEE, Barcelona, Spain, 1–6.

[29] Yong Li, Sven Schäge, Zheng Yang, Florian Kohlar, and Jörg Schwenk. 2014. On the Security of the Pre-shared Key Ciphersuites of TLS. In *Proceedings of the 17th International Conference on Public-Key Cryptography — PKC 2014 - Volume 8383*. Springer-Verlag, Berlin, Heidelberg, 669–684.

[30] Elsa Lopez-Perez, Göran Selander, John Preuß Mattsson, Rafael Marin-Lopez, and Francisco Lopez-Gomez. 2025. EDHOC Authenticated with Pre-Shared Keys (PSK). Work in Progress.

[31] Elsa Lopez Perez, Thomas Watteyne, Rafael Marin Lopez, Cristina Onete, Clement Papon, and Mališa Vučinić. 2025. Pre-Shared Key Authentication with EDHOC: the Security-Performance Tradeoff. . *IEEE Access* (2025).

[32] G. Lowe. 1997. A hierarchy of authentication specifications. In *Proceedings 10th Computer Security Foundations Workshop*. 31–43.

[33] Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. 2013. The TAMARIN Prover for the Symbolic Analysis of Security Protocols. In *Computer Aided Verification, 25th International Conference, CAV 2013, Princeton, USA, Proc. (Lecture Notes in Computer Science, Vol. 8044)*, Natasha Sharygina and Helmut Veith (Eds.). Springer, 696–701. doi:10.1007/978-3-642-39799-8_48

[34] NIST. 2016. *Report on Post-Quantum Cryptography*. Technical Report NISTIR 8105. National Institute of Standards and Technology.

[35] Karl Norrman, Vaishnavi Sundararajan, and Alessandro Bruni. 2020. Formal Analysis of EDHOC Key Establishment for Constrained IoT Devices. *CoRR* abs/2007.11427 (2020).

[36] Elsa López Pérez, Göran Selander, John Preuß Mattsson, Thomas Watteyne, Mališa Vučinić, and James Bret Michael. 2024. EDHOC Is a New Security Handshake Standard: An Overview of Security Analysis. *IEEE Computer Magazine* 57, 9 (2024), 101–110.

[37] Gabriele Restuccia, Hannes Tschofenig, and Emmanuel Baccelli. 2020. Low-Power IoT Communication Security: On the Performance of DTLS and TLS 1.3. In *2020 9th IFIP International Conference on Performance Evaluation and Modeling in Wireless Networks (PEMWN)*. 1–6.

[38] Göran Selander, John Preuß Mattsson, and Francesca Palombini. 2024. Ephemeral Diffie-Hellman Over COSE (EDHOC).

[39] M.M.J. Stevens. 2021. A Survey of Chosen-Prefix Collision Attacks. *Computational Cryptography* (2021). https://api.semanticscholar.org/CorpusID:236467675

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

[41] Thomas Watteyne, Lance Doherty, Jonathan Simon, and Kris Pister. 2013. Technical Overview of SmartMesh IP. In *2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*. 547–551.

A Unlinkability Attack

We provide a more detailed explanation of the attack that an active adversary can mount, showing that EDHOC-PSK does not provide identity protection against active attackers under the unlinkability notion of Cottier and Pointcheval [15].

In message 3, the Initiator sends an AEAD-protected ciphertext:

$$\texttt{CIPHERTEXT\_3A} = \texttt{PLAINTEXT\_3A} \oplus \texttt{KEYSTREAM\_3A}.$$

where PLAINTEXT_3A contains ID_CRED_PSK. An active attacker can intercept this ciphertext and introduce a controlled modification δ, producing

$$\texttt{CIPHERTEXT\_3A'} = \texttt{CIPHERTEXT\_3A} \oplus \delta.$$

Due to the malleability of XOR-based stream encryption, this induces a corresponding change in the decrypted plaintext:

$$\begin{aligned} \texttt{PLAINTEXT\_3A'} &= \texttt{PLAINTEXT\_3A} \oplus \delta \\ &= \texttt{ID\_CRED\_PSK} \oplus \delta \\ &= \texttt{ID\_CRED\_PSK}'. \end{aligned} \tag{1}$$

Because message 3 is protected by an AEAD scheme, any modification of the ciphertext will cause the authentication tag verification to fail. Consequently, the protocol run will not complete successfully when processing CIPHERTEXT_3A'.

The potential privacy leakage arises only if the Responder's behavior on failure depends on the decrypted ID_CRED_PSK′ in a way that is externally observable. For instance, if the Responder returns distinguishable error messages (e.g., "unknown credential" versus "authentication/tag failure"), or exhibits measurable timing differences, then an active adversary can use this as an oracle. By crafting different values of δ and observing the Responder's failure behavior, the attacker can test whether candidate credential references are recognized by the Responder.

Received 12 December 2025; revised 10 March 2026; accepted 13 March 2026