A comprehensive, formal and automated analysis of the EDHOC protocol

Charlie Jacomme[1] Elise Klein[2] Steve Kremer[2] **Maïwenn Racouchot**[2]

August 11, 2023

[1]Inria Paris [2]Inria Nancy

The EDHOC protocol

The EDHOC protocol: Context

- The IETF is currently standardizing a new LAKE (Light-weight Authenticated Key Exchange) protocol [1]
- Light-weight protocol suitable for IoT
- IETF call for formal analysis for draft 12, released in October 2021

[1] `https://github.com/lake-wg/edhoc`

The EDHOC protocol

Features of the protocol:

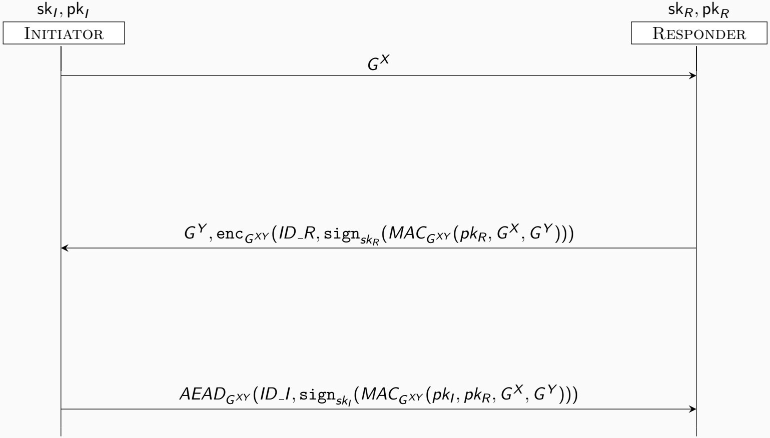
- Variant of MAC-then-Sign Diffie-Hellman for authentication
- 4 methods combining signature key and long-term Diffie-Hellman Key
- 3 messages (and an optional 4th)

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

$+$ KEM-based PQ-secure version

EDHOC modes

The EDHOC protocol: Method 0



The EDHOC protocol: claimed security properties

Authentication

Authentication of some data, implicit key authentication, explicit key confirmation.

Confidentality

Protection of the exchange, even in case of later key compromise.

Identity protection

Confidentiality of identity of the agent (anonymity of Initiator).

Other

KCI (Key Compromise Impersonation), non repudiation, post quantum security.

↪ Many properties (especially regarding authentication and confidentiality), even in the case of key compromise

Protocol model

Symbolic verification

Formal modeling and Analysis of Protocols

- Protocol described by a transition system between protocol states
- Security propreties are stated on execution traces in first order logic
- The adversary is usually modeled with Dolev-Yao model

Symbolic verification gives mathematically sound proofs on the security properties of the protocols.

SAPIC+: protocol verifiers of the world unite [USENIX'22]

The SAPIC+ platform

Protocols modelled in the applied pi-calculus.

Export to different tools that automatically prove the security or find attacks:

- ProVerif: much faster, but looser model of Diffie-Hellman
- Tamarin prover: more precise proofs
- DeepSec: equivalence properties but bounded # of sessions

Translations between tools have been proved: a result proved with one can be reused in the other.

The protocol model

LAKE-EDHOC

- 4 methods executable in parallel;
- includes TOFU (Trust-On-First-Use) paradigm;
- model many key compromise scenarios;
- alternate model with the KEM based variant.

Limitations

- No fine grained modeling of the cipher suite negotiation;
- no modeling of the key update mechanism;
- no modeling of the 4th (optional) message.

Results

Summary of results from automated analysis

Property	Threat model					
	Basic	AEAD$^{\lightning}$	DH$^{\lightning}$	DHShare$^{\lightning}$ + SessKey$^{\lightning}$	Hash$^{\lightning}$ + DH$^{\lightning}$	KEM variant
Confidentiality	✓	✓	✓	✓	✗	✓
Implicit& Explicit Key Auth.	✓	✓	✓	✗	✓	✓
Transcript Auth.	✓	✗	✓	✓	✗	✓
Algo Auth.	✓	✓	✓	✓	✗	✓
Session key uniqueness	✓	✓	✗	✓	✗	✗
Non-repudiation soundness	✓	✓	$\sim$	✓	$\sim$	✓
Identity protection	✗	✗	✗	✗	✗	✗

✓ : property satisfied
✗ : violation of property
$\sim$: unclear security

Weak Sig : weak signatures (malleable, yes keys)
Weak DH : small sub-groups
Weah Hash : Length extensions, chosen-prefix collisions

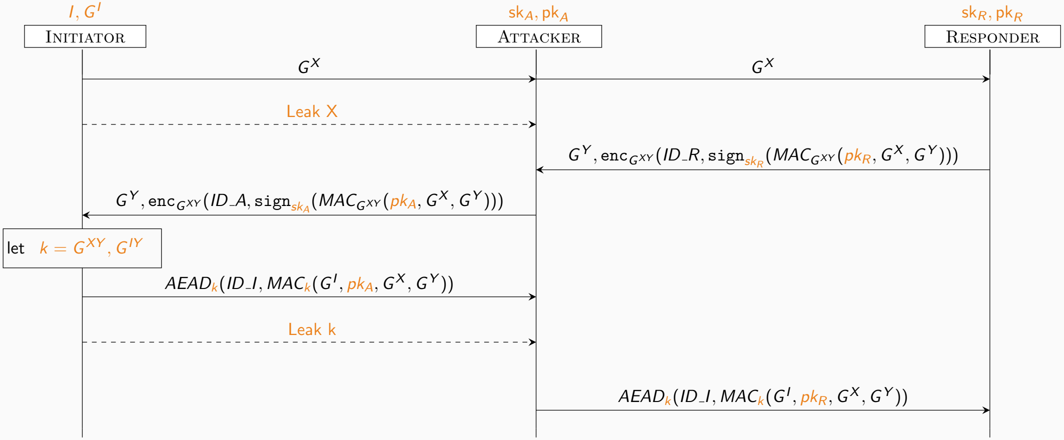
Impersonation attack

Threat model

Authentication operations inside a TEE, but device otherwise compromised.

- leak the initiator ephemeral key at the beginning, and the session key at the end;
- but no access to authentication keys.

Impersonation attack



Impersonation attack

Main concern

- In method 1,2,3, the session key is actually the MAC key, and is sufficient for impersonation.
- Safety of all authentication operations is insufficient to ensure authentication.
- Storing G_I inside a TEE does not increase the security level.

Mitigation

Additional "Master Secret" derivation solves this issue.

High-level feedback

Security proofs

In basic model, the protocol provides almost all expected security properties.

Suggestions for improvements

Simple changes and clarifications, identified through the automated analysis:

1. avoid potential misuse of the existing design;
2. strengthen the TEE implementation;
3. improve the future resilience of the protocol.

Discussions made with IETF working group for improvements.

Conclusion

Conclusion

- In-depth case study of LAKE-EDHOC using state-of-the-art tools and models
  $\rightarrow$ detected a number of weaknesses (not all mentioned in the talk)
- Discussion with IETF LAKE working group :
  - Weaknesses acknowledged + mitigations wip
  - 8 issues reported; 4 Pull Requests

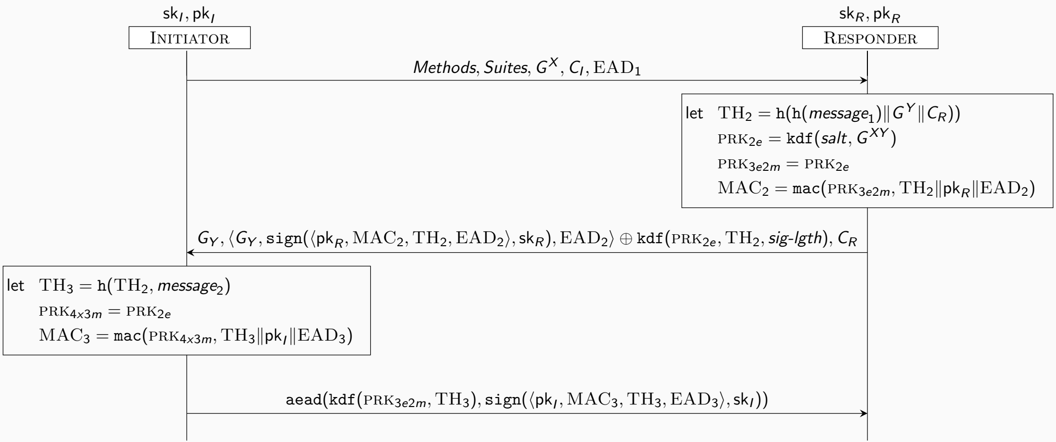
  $\rightarrow$ **draft 14 released after our discussion, in May 2022**
- Improve and deepen the analysis (key update, fourth message . . .)
- Keep the models up to date with the drafts and up to the final RFC (current version : draft 20, in July 2023)

Summary of our attacks and action takens

Attack type	Requirements	Found by	Action
Initiator Impersonation	Ephemeral share and Session key leaks	PROVERIF (846 s)	✓(draft 14)
Secrecy & Auth. breach & Downgrade attack	Hash Chosen-prefix collisions and no neutral DH check	TAMARIN (16 h)	✓(draft 14)
Final transcript mismatch	Leak session key or Non deterministic encoding or Leak share and Malleable Sig.	PROVERIF (56 s)	✓(draft 14)
Party Controlled Session key	No neutral DH check or KEM variant	PROVERIF (49 s)	✓(draft 14)
Identity leak	Initiator refuses to exchange with its identity	DEEPSEC (1 s)	To be clarified
Duplicated non-repudiation	Malleable Sig.	PROVERIF (81 s)	Judged irrelevant
AEAD Key/IV reuse	Message recomputation from stored state	Manual	✓(draft 14)

Questions?

The EDHOC protocol: Method 0



Transcript collisions

Threat model

- The attacker can compute chosen prefix collisions.

  Given p_1, p_2, it can compute c_1, c_2 such that $h(p_1|c_1) = h(p_2|c_2)$
- Agents accept as DH share the identity element (or low-order points).

  The identity element e is such that $e^x = e$.

Consequences

Breaks secrecy, and may allow for downgrade attacks. (EDHOC allows SHA-2 and SHA-256)

```
Trans_E := method |  suitesI  | G_X |  C_I  |  EAD_1  | G_Y | C_R

Trans_I :=  zero  | "suitesI" | g^x | "C_I" | "EAD_1" |  e  | c2 | g^y | "C_R"
Trans_R :=  zero  | "suitesI" |  e  | "C_I" |   c1    | g^y | "C_R"
```