

Relay Attacks in Intra-handshake Attestation for Confidential Agentic AI Systems

Muhammad Usama Sardar^{1,2}, Viacheslav Dubeyko³,
and Jean-Marie Jacquet⁴

(ACK: Eric Rescorla, Juho Forsén, Markus Rudy, Mariam Moustafa,
Tjaden Hess, Yuning Jiang, Pavel Nikonorov, and Casey Wilson)

¹TU Dresden, Germany

²Co-chair, Trusted Research Environment (TRE) Open Suite,
Global Alliance for Genomics and Health (GA4GH)

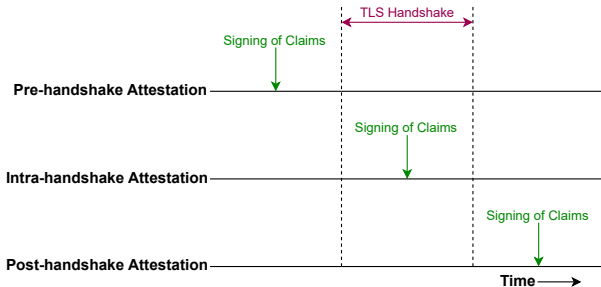
³IBM, San Jose, CA, USA

⁴Nadi Research Institute, University of Namur, Belgium

March 17, 2026

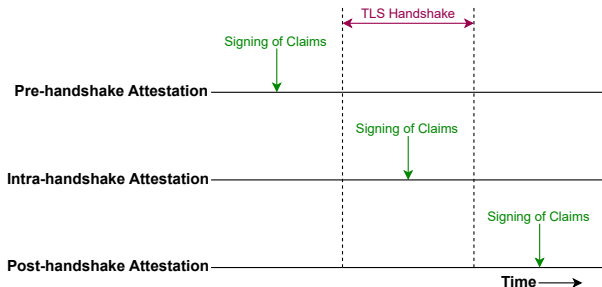
Categories of Attested TLS

- Temporal ordering of RA and TLS at Attester side



Categories of Attested TLS

- Temporal ordering of RA and TLS at Attester side



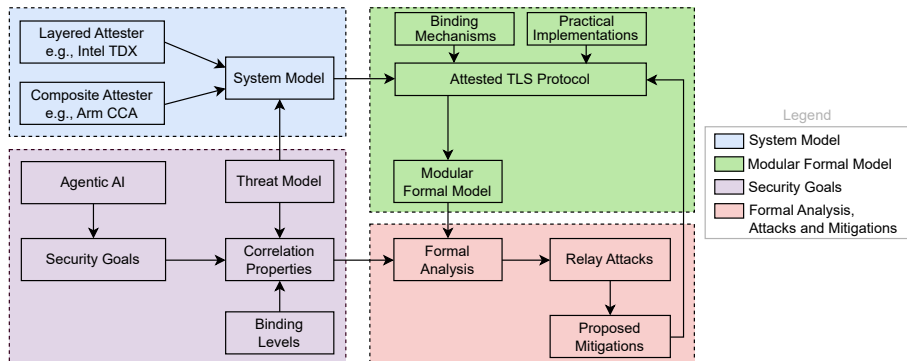
- Typical changes in each category
 - CA/TA = Certification/Trusted Authority
 - ✓ = no change; ✗ = changes required

	Protocol	Deployment	Higher layer
Pre-handshake attestation	✓	✗ (CA/TA)	✗
Intra-handshake attestation	✗ (Invasive)	✓	✗
Post-handshake attestation	✓	✓	✗

AD's instruction to exhaustively explore intra-handshake attestation

Approach¹

- TLS 1.3 as example transport protocol
- Agentic AI as example use case
- Formalization tool: ProVerif



¹https://mailarchive.ietf.org/arch/msg/seat/x3eQxFjQFJLceae614_NgXnmsDY/

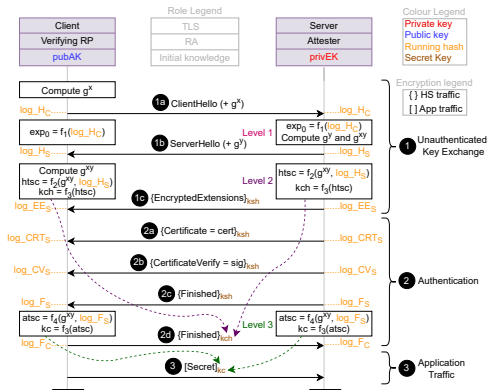
Potential Mechanisms for Binding (TLS Server as Attester)

S. No.	Binding material	Value of rdata field
1	Client's TLS nonce	nc
2	Client's attestation nonce	na
3	Early exporter	exp_0
4	Server's public key	pubEK
5	Client's attestation nonce and early exporter	na exp_0
6	Client's attestation nonce and server's public key	na pubEK
7	Nonce, server's public key, and early exporter	na pubEK exp_0

- **Discussion:** Any other useful binding material/combination?
- Example implementations: **all are vulnerable**
 - #1: Meta's AI (even after **extensive security review** by *Trail of Bits* without formal methods)
 - No Evidence freshness
 - #5: draft-fossati-tls-attestation-06
 - #6: Edgeless Systems Contrast, Cocos AI and CCC PoC²
- **Discussion:** Any other intra-handshake implementation?

²<https://github.com/CCC-Attestation/attested-tls-poc>

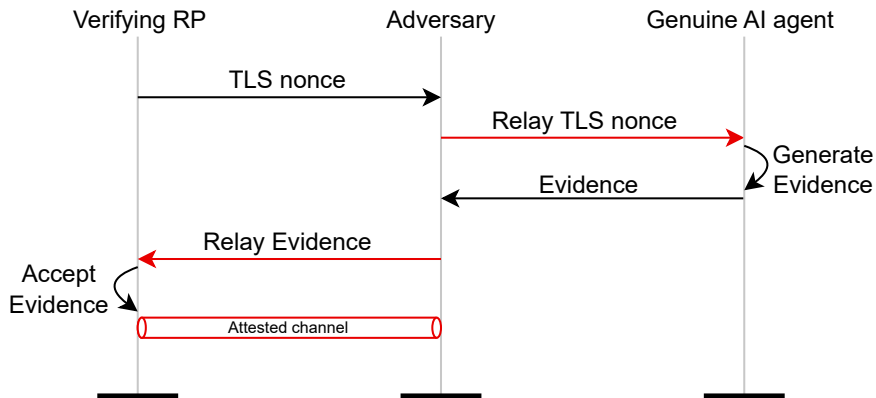
Devil is in the Details!



- $htsc$: used for encryption of clientFinished message (2d).
 - Irrelevant for security goals
 - Server **not yet authenticated** at this point
- $atsc$: used for encryption of application data (client's secret, e.g., decryption key)
 - Relevant for security goals

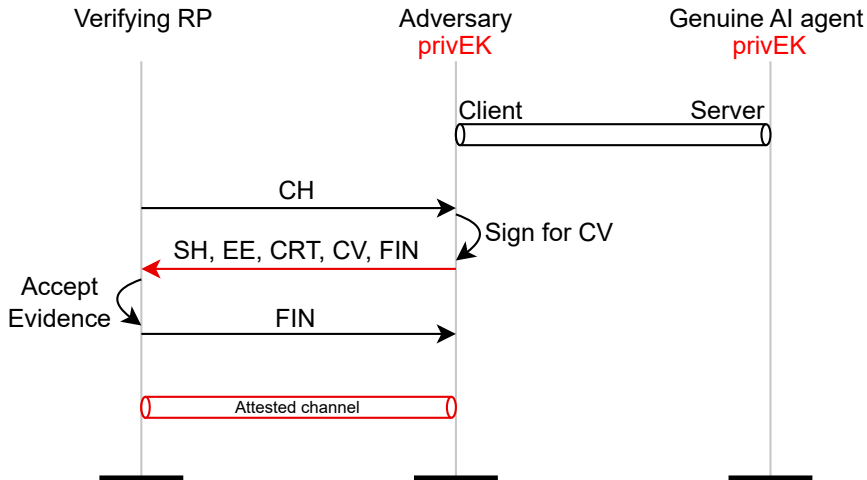
Relay Attack 1 (Abstracted): Mechanism # 1 (nc)

- Adversary can relay nonce to a genuine Attester



Relay Attack 2 (Abstracted): Mechanism # 4 (pubEK)

- No protection if **privEK** is accessible to some entity other than an AI agent (e.g., provisioning at runtime) or leaked via vulnerability in code
- Adversary gets signed Evidence from genuine AI agent



Binding Evidence to unauthenticated peer (Server)

Need for Implementation Guidance

- Meta's AI: TLS nonce (hence, no Evidence freshness)
- Conveyance of attestation nonce
 - Edgeless Systems Contrast (previously) and Cocos AI abuse SNI (intended for hostname)
 - Edgeless Systems Contrast (now) abuse ALPN (intended for protocol names)
- CCC Attested TLS proof-of-concept³: No update for 3 years
- **Discussion:** Any other intra-handshake attestation implementation within SEAT WG charter?

³<https://github.com/CCC-Attestation/attested-tls-poc>

Summary of Security Analysis So Far

- Pre-handshake attestation: replay and diversion attacks
 - Intra-handshake attestation: diversion and relay attacks
 - Post-handshake attestation: no known attacks
-
- Unless one re-establishes connection each time attestation is required, post-handshake attestation is required, and hence, intra-handshake only adds unnecessary complexity.