# When to Attest? Intra- and Post-Handshake Attestation for IoT Swarms

Yuxuan Song[1], Muhammad Usama Sardar[2], Geovane Fedrecheski[1], Mališa Vučinić[1], Thomas Watteyne[1]
[1]Inria, France first.last@inria.fr
[2]Technical University of Dresden, Germany
muhammad_usama.sardar@tu-dresden.de

***Abstract*—Remote attestation is a security mechanism that allows a device to prove its integrity and trustworthiness by generating fresh verifiable evidence to be assessed by a verifier. It is gaining increasing attention in the context of IoT security for both IoT devices and services. Within the ongoing standardization efforts at the IETF, two distinct approaches have emerged and are actively discussed by different working groups and protocol designers: *(1)* intra-handshake attestation, where attestation is performed during the handshake process; *(2)* post-handshake attestation, where it occurs after the handshake is complete. This position paper analyzes the respective security properties and discusses their applicability across different IoT deployment scenarios. We highlight the key trade-off: intra-handshake attestation enables early trust establishment prior to session setup, making it suitable for onboarding scenarios, while post-handshake attestation provides continuous assurance and supports runtime integrity validation.**

## I. Introduction

The evolution of the Internet-of-Things (IoT) is leading to a scenario where swarms of IoT devices such as sensors and drones are numerously applied in industry, smart cities and agriculture [1]. In many deployments, embedded devices operate collaboratively in physically accessible environments, where a single compromised device may jeopardize the entire swarm. In swarm settings, attestation can be coordinated so that devices verify each other or perform group-wide trust validation before collaborative tasks.

These challenges highlight the need for remote attestation, enabling a remote Verifier to assess the trustworthiness across use cases, including: *(1) Secure Onboarding:* Attesting devices before network admission. Similarly, a network service attestation may also be required before sharing sensitive data. *(2) Update Verification:* Confirming integrity after software or firmware updates. *(3) Software Version Synchronization:* Ensuring consistent versions across collaborating devices. *(4) Runtime or Periodic Integrity Checks:* Detecting unauthorized modifications in long-lived deployments. *(5) Device Mobility and Gateway Handoff:* Re-establishing trust during gateway transitions.

Several standardization efforts on attestation have taken place within the Internet Engineering Task Force (IETF). The IETF is one of the main standards development organizations for the Internet, responsible for defining protocols and architectures. The attestation architecture has been standardized in the IETF [2], and ongoing work continues to explore attestation support across various protocols, including Transport Layer Security (TLS) [3], [4] and Ephemeral Diffie-Hellman over COSE (EDHOC) [5].

However, debate remains active regarding the most appropriate attestation type between intra-handshake attestation [3], [5], [6] and post-handshake attestation [4]. Intra-handshake attestation over TLS [3] requires invasive protocol changes, prompting active efforts to standardize a post-handshake alternative [4]. In contrast, intra-handshake attestation over EDHOC [5], which targets constrained IoT devices, operates without changes to the protocol itself. Still, the IETF community continues to emphasize the need for continuous attestation in the context of IoT devices and swarms. This position paper discusses the different security properties and features of intra- and post-handshake attestation, with a focus on their applicability to IoT use cases.

The remainder of the paper is organized as follows: Section II provides background on remote attestation and relevant IoT security protocols. Section III and Section IV describe how intra- and post-handshake attestation are performed in IoT contexts. Section V gives core arguments between intra- and post-handshake attestation. Section VII concludes the paper.

## II. Background

### A. Remote Attestation

Remote attestation is a security mechanism that enables a Verifier to assess the integrity and trustworthiness of a remote device. The attestation roles and architecture have been standardized by the IETF [2]. It typically involves three roles: *(1)* Attester, which generates reliable evidence of its state; *(2)* Verifier, which evaluates the evidence and generates attestation result; *(3)* Relying Party, which uses the attestation result to make trust decisions. In some scenarios, the Verifier and Relying Party can reside on the same device.

### B. Ephemeral Diffie-Hellman over COSE (EDHOC) protocol

Ephemeral Diffie-Hellman over COSE (EDHOC) is a lightweight authenticated key exchange protocol standardized by the IETF [7]. EDHOC is designed for constrained IoT devices, which operates with an Initiator and a Responder to establish authenticated session keys after a three-way handshake. It consists of three messages. The Initiator sends `message_1` including a cipher suite, ephemeral public key, connection identifier, and an optional External Authorization Data field (`EAD_1`). In `message_2`, the Responder replies

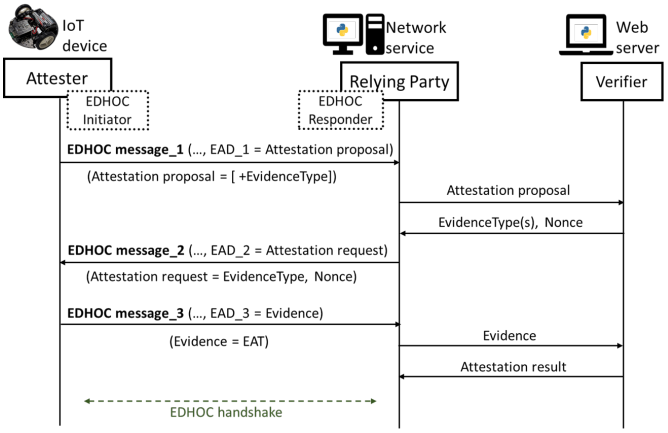


Fig. 1: Intra-handshake attestation with EDHOC.

with its ephemeral public key, encrypted credential identifier, authentication item, connection identifier and optional `EAD_2`. Initiator authenticates itself in `message_3`, sending a ciphertext that the corresponding plaintext contains its credential identifier, authentication item, and optional `EAD_3`. This exchange allows both parties to derive a shared secret to secure subsequent communication. EDHOC supports compact authentication by allowing credentials to be referenced via identifiers to reduce message size.

### *C. Object Security for Constrained RESTful Environments (OSCORE) protocol*

Object Security for Constrained RESTful Environments (OSCORE) [8] is an application-layer protocol that secures Constrained Application Protocol (CoAP) messages in constrained IoT environments. It provides end-to-end confidentiality, integrity, and replay protection. OSCORE operates on a shared secret established via a key exchange protocol such as EDHOC, and derives keys and nonces using a lightweight context based on the shared secret and unique identifiers (Sender/Recipient IDs). Once established, OSCORE transforms CoAP messages into protected messages using Authenticated Encryption with Associated Data (AEAD).

An optimization standardized in [9] enables combining EDHOC with the first OSCORE transaction, reducing round-trips required to establish and use the OSCORE security context.

## III. Intra-Handshake: Remote Attestation with EDHOC

An ongoing work at the IETF for intra-handshake attestation in IoT environments is the Internet-Draft named *Remote attestation over EDHOC* [5]. A detailed message flow is illustrated in Fig. 1. Remote attestation is tightly integrated into the EDHOC key exchange phase. The Attester (constrained IoT device) initiates an EDHOC session with the Relying Party, which is responsible for granting network access based on the attestation result. A Verifier operates in the background to assess the device's integrity.

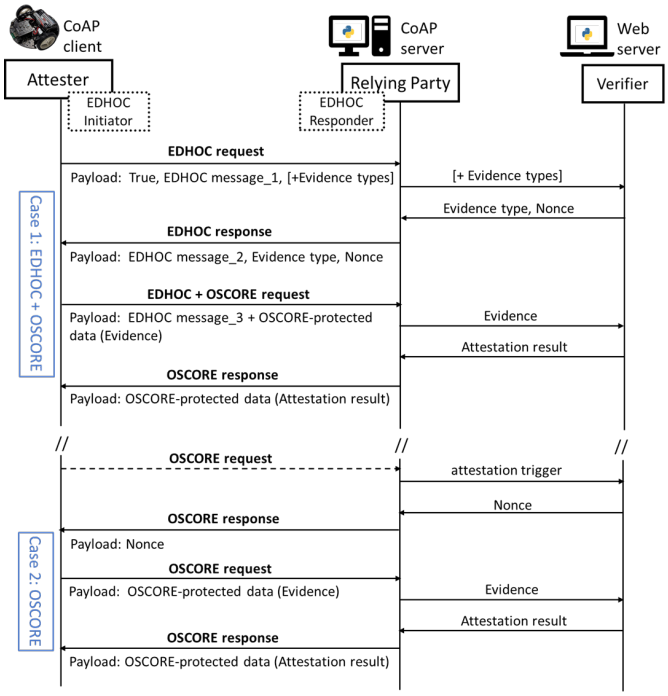


Fig. 2: Post-handshake attestation with OSCORE.

EDHOC supports remote attestation through its External Authorization Data (EAD) fields. During the intra-handshake attestation, three EAD fields are exchanged, without any additional changes to the EDHOC protocol. In `EAD_1`, the Attester lists supported evidence types as integers indicating the encoding and content formats (e.g., 60 for `application/cbor`). The Relying Party forwards this proposal to the Verifier, which selects an evidence type and generates a fresh nonce to ensure session integrity. The selected parameters are returned to the Attester in `EAD_2` as an attestation request. The Attester responds with `EAD_3`, containing the attestation evidence.

## IV. Post-Handshake: Remote Attestation with OSCORE

A general way to perform post-handshake attestation for IoT devices is to carry the attestation-related materials within an application-layer message secured by OSCORE. This assumes the EDHOC three-way handshake has performed beforehand, and the OSCORE Security Context has also been derived. An example process is illustrated in Fig. 2, case 2. The Relying Party can trigger remote attestation at any moment, and request a nonce from the Verifier. The nonce is sent in an OSCORE response payload, and followed by direct transmission of the evidence in the subsequent OSCORE request. Then the attestation is finished by an OSCORE response containing the attestation result to the CoAP client.

An efficient method for combining authentication and post-handshake attestation is OSCORE piggybacking on EDHOC [9]. The case 1 in Fig 2 presents an example of this approach, where a CoAP client acts as the EDHOC Initiator and a CoAP server acts as the EDHOC Responder. The EDHOC

| Property | Intra-handshake | Post-handshake |
|---|---|---|
| Evidence Freshness and integrity | ✓ | ✓ |
| Replay Protection | ✓ | ✓ |
| Authentication Binding | ✓ | ✗ |
| Runtime Integrity | ✗ | ✓ |
| Pre-session Trust Establishment | ✓ | ✗ |
| Continuous Assurance | ✗ | ✓ |
| On-demand Availability | ✗ | ✓ |

TABLE I: Comparison of security properties between intra- and post-handshake attestation. Symbols: ✓= yes, ✗= no.

messages serve for authentication, while all attestation-related materials are transmitted in the CoAP message payload. In the first message payload, the client (EDHOC Initiator) sends the `true` value together with EDHOC message_1 to signal that the EDHOC exchange will be combined with an OSCORE request. The attestation-related evidence type negotiation is also included in the payload. In the second message, the selected evidence type and nonce are sent in the payload alongside EDHOC message_2. Based on EDHOC's features, after the client receives EDHOC message_2, it already has all the information to derive the OSCORE Security Context. The client can use OSCORE to protect communication in the third message, enabling the attestation evidence to be sent within an OSCORE-protected payload. The server then responds with the attestation result in the payload.

## V. Discussion

Remote attestation provides security properties including evidence integrity, freshness guarantees, and replay protection. Additional properties vary by attestation type (Table. I), notably: *(1)* Authentication Binding: whether attestation is cryptographically bound to the authentication process; *(2)* Runtime Integrity: whether dynamic measurements can be obtained during the device's execution; *(3)* Pre-session Trust Establishment: whether successful attestation is a necessary condition for session establishment; *(4)* Continuous Assurance: whether attestation can be repeatedly performed throughout the session; *(5)* On-demand Availability: whether attestation can be triggered at any point during operation in response to specific requests or events. These properties influence how each method handles edge cases, such as compromised devices attempting late-session reentry or swarms requiring collective re-attestation after partial failures.

Intra-handshake attestation embeds evidence exchange into the authentication handshake, cryptographically tying identity to device state and blocking unauthenticated and untrusted devices from completing the handshake. However, intra-handshake attestation may require modifications to existing handshake protocols (e.g., TLS protocol), and transmits larger data during the handshake process, which increases handshake completion time. Besides, the attestation evidence is only as fresh as the handshake, and it cannot re-verify device state during the session.

Post-handshake attestation separates attestation from the authentication handshake, providing modularity that allows attestation mechanisms to be integrated independently of the handshake protocol. This separation enables transmission of larger evidence since the attestation process is not limited by the handshake period. A significant advantage is that the attestation can be triggered on-demand to verify runtime integrity, which provides continuous verification throughout the session lifetime. The attestation frequency can also be adaptively adjusted based on risk assessment. However, since authentication and attestation operate separately, identity and device state may not be cryptographically linked. Then a session-specific key binding is required (e.g., using OSCORE protocol) to establish this cryptographic association.

Table II presents the recommended handshake type for each use case introduced in Section I, along with the corresponding security requirements. Intra-handshake attestation is particularly suitable for secure onboarding (Use case 1) and device mobility with gateway handoff (Use case 5). These use cases demand strong time efficiency and minimal round-trip latency. For secure onboarding, establishing trust before the session starts and cryptographically binding authentication with attestation are essential. Similarly, for gateway handoff, such binding is highly recommended to ensure seamless trust continuity. Since these events typically occur only once per session, continuous attestation or on-demand availability is not required. In contrast, update verification (Use case 2) and runtime/periodic integrity checks (Use case 4) generally require multiple attestations over the device's lifetime. The Verifier may initiate attestation after detecting a software update or based on a predefined schedule. These scenarios prioritize long-term trust assurance rather than rapid and early trust establishment, making post-handshake attestation a more appropriate fit. Software version synchronization (Use case 3) can be addressed using either intra- or post-handshake attestation. For instance, version checking during the initial connection prevents outdated devices from joining collaborative tasks. Alternatively, devices that have already joined the group may become outdated after the initial handshake; in such cases, they can update their software to synchronize with the group and subsequently request a post-handshake attestation to regain group membership.

Computational overheads are similar for intra- and post-handshake attestation, while communication overheads are higher for post-handshake attestation due to the additional round trips required for evidence exchange after the handshake.

In summary, intra-handshake attestation is preferred when fast and early trust establishment is required, such as during initial onboarding or mobility events. Post-handshake attestation is better suited for long-term assurance and scenarios requiring attestation at arbitrary times, such as periodic checks or software updates.

## VI. Related Work

Both intra-handshake attestation and post-handshake approaches have been proposed in academia and industry for different secure channel establishment protocols. For example,

| Property required | Use case 1: Secure Onboarding | Use case 2: Update verification | Use case 3: Software version synchronization | Use case 4: Runtime or periodic integrity checks | Use case 5: device mobility and gateway handoff |
|---|---|---|---|---|---|
| time efficiency | ✓ | ✗ | △ | ✗ | ✓ |
| minimal round-trip | ✓ | ✗ | ✗ | ✗ | ✓ |
| continuous assurance | ✗ | △ | △ | △ | ✗ |
| authentication binding | ✓ | △ | ✗ | ✗ | △ |
| pre-session trust establishment | ✓ | △ | ✗ | ✗ | △ |
| on-demand availability | △ | ✓ | ✓ | ✓ | ✗ |
| recommended handshake type | Intra | Post | Intra/Post | Post | Intra |

TABLE II: Comparison between different use cases. Symbols: ✓= Strongly required, ✗= not required, △= optional.

AuRA [10] is an intra-handshake attestation approach that is specifically designed for constrained IoT devices. Moreover, Ott *et al.* [11] proposed a post-handshake attestation approach over Transport Layer Security (TLS) protocol. From practical deployments, an example is SCONE [12], which is a closed-source solution and uses a post-handshake protocol with the TLS exporters. Open source deployed solutions include Intel's RA-TLS protocol, which is shown to be vulnerable to replay attacks [13].

Additionally, there are ongoing standardization efforts for attested EDHOC as well as attested TLS protocol in the IETF. Attested EDHOC protocol [5] is a lightweight protocol that specifies how to integrate attestation with the EDHOC authentication protocol. Other lightweight protocols have been proposed for the integration of remote attestation with key negotiation and distribution [14]. Examples of standardization efforts on attested protocols for TLS include TLS-attest [3] and TLS-EA [4]. The former uses intra-handshake attestation and presents two design options, namely remote attestation only and remote attestation with PKI Infrastructure (PKIX). The latter uses post-handshake attestation with exported authenticators based on RFC 9261 [15]. The core idea is that the challenger sends an authenticator request, and the Attester sends an authenticator containing the evidence.

## VII. Conclusions

This paper presents arguments in the context of ongoing IETF standardization efforts concerning the design choices between intra- and post-handshake attestation mechanisms. We present and propose the intra- and post-handshake attestation approaches for IoT scenarios based on existing and emerging IETF standards. Our finding is that intra-handshake attestation is suitable for use cases that need a fast attestation before the session establishment, and may be required only once during the session lifetime, such as onboarding and gateway handoff scenarios. As for post-handshake attestation, it fits well when periodic or on-demand attestation is required, such as periodic attestation for long-lived, low-power IoT devices or in IoT swarms that need to synchronize software versions before coordinated operations or after firmware updates.

## Acknowledgements

This document is issued within the frame and for the purpose of the OpenSwarm project. This project has received funding from the European Union's Horizon Europe Framework Programme under Grant Agreement No. 101093046. Views and opinions expressed are however those of the author(s) only and the European Commission is not responsible for any use that may be made of the information it contains. Muhammad Usama Sardar was funded by DFG grant 389792660 as part of TRR 248 – CPEC.

## References

[1] R. Chataut, A. Phoummalayvane, and R. Akl, "Unleashing the Power of IoT: A Comprehensive Review of IoT Applications and Future Prospects in Healthcare, Agriculture, Smart Homes, Smart Cities, and Industry 4.0," *Sensors*, Aug. 2023.

[2] H. Birkholz, D. Thaler, M. Richardson, N. Smith, and W. Pan, *Remote ATtestation procedureS (RATS) Architecture*, Internet Engineering Task Force (IETF) Std. RFC9334, 2023.

[3] H. Tschofenig, Y. Sheffer, P. Howard, I. Mihalcea, Y. Deshpande, A. Niemi, and T. Fossati, "Using Attestation in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)," 2025, work in Progress.

[4] T. Fossati, M. U. Sardar, K. T. Reddy, Y. Sheffer, H. Tschofenig, and I. Mihalcea, "Remote Attestation with Exported Authenticators," 2025, work in Progress.

[5] Y. Song and G. Selander, "Remote Attestation over EDHOC," 2025, work in Progress.

[6] C. Weinhold, M. U. Sardar, I. Mihalcea, Y. Deshpande, H. Tschofenig, Y. Sheffer, T. Fossati, and M. Roitzsch, "Separate but Together: Integrating Remote Attestation Into TLS," 2025.

[7] G. Selander, J. P. Mattsson, and F. Palombini, *Ephemeral Diffie-Hellman Over COSE (EDHOC)*, Internet Engineering Task Force (IETF) Std. RFC9528, 2024.

[8] G. Selander, J. P. Mattsson, F. Palombini, and L. Seitz, *Object Security for Constrained RESTful Environments (OSCORE)*, Internet Engineering Task Force (IETF) Std. RFC8613, 2019.

[9] F. Palombini, M. Tiloca, R. Höglund, S. Hristozov, and G. Selander, *Using Ephemeral Diffie-Hellman Over COSE (EDHOC) with the Constrained Application Protocol (CoAP) and Object Security for Constrained RESTful Environments (OSCORE)*, Internet Engineering Task Force (IETF) Std. RFC9668, 2024.

[10] Y. Song, G. Fedrecheski, M. Vučinić, and T. Watteyne, "AuRA: Remote Attestation over EDHOC for Constrained Internet-of-Things Use Cases," in *IEEE Symposium on Computers and Communications (ISCC)*, 2025.

[11] S. Ott, M. Kamhuber, J. Pecholt, and S. Wessel, "Universal Remote Attestation for Cloud and Edge Platforms," in *International Conference on Availability, Reliability and Security*, 2023.

[12] S. Arnautov, B. Trach, F. Gregor, T. Knauth, A. Martin, C. Priebe, J. Lind, D. Muthukumaran, D. O'keeffe, M. L. Stillwell *et al.*, "SCONE: Secure Linux Containers with Intel SGX," in *USENIX Symposium on Operating Systems Design and Implementation*, 2016.

[13] M. U. Sardar, A. Niemi, H. Tschofenig, and T. Fossati, "Towards validation of tls 1.3 formal model and vulnerabilities in intel's ra-tls protocol," *IEEE Access*, vol. 12, pp. 173 670–173 685, 2024.

[14] L. Xia, W. Jiang, M. U. Sardar, H. Birkholz, J. Zhang, and H. Labiod, "Integration of Remote Attestation with Key Negotiation and Distribution," 2025, work in Progress.

[15] N. Sullivan, *Exported Authenticators in TLS*, Internet Engineering Task Force (IETF) Std. RFC9261, 2022.