# AuRA: Remote Attestation over EDHOC for Constrained Internet-of-Things Use Cases

Yuxuan Song, Geovane Fedrecheski, Mališa Vučinić, Thomas Watteyne

# AuRA: Remote Attestation over EDHOC for Constrained Internet-of-Things Use Cases

Yuxuan Song, Geovane Fedrecheski, Mališa Vučinić, Thomas Watteyne
Inria, France
first.last@inria.fr

***Abstract*—Remote Attestation (RA) is a security process that verifies the integrity and trustworthiness of a remote device's software and hardware. While RA for high-end devices is well-developed, RA in constrained IoT environments remains incomplete. Existing embedded RA mechanisms focus on local evidence generation and verification, but lack a complete process that includes a secure attestation channel. This paper introduces AuRA, a lightweight RA solution that builds upon the newly standardized Ephemeral Diffie-Hellman over COSE (EDHOC) protocol. AuRA specifies how to transport existing attestation mechanisms in parallel with network authentication. We evaluate AuRA on the nRF5340 microcontroller running at 64 MHz. This implementation has a memory footprint of 6,665 B of RAM and 17,163 B of flash. The device completes Remote Attestation by exchanging three EDHOC messages with a verifier entity, of sizes 42 B, 59 B and 223 B. This allows the device to prove that it is running the right hardware and software in only 5.51 s, consuming as little as 88 mC of charge.**

## I. Introduction

The rapid growth of the Internet of Things (IoT) has led to a vast network of interconnected embedded devices, creating new opportunities but also significantly increasing the risk of cyberattacks. To mitigate risks, only legitimate devices should be allowed to access a network. The traditional network access policies are based on the identity of the device, through a verification process known as "authentication". However, consider an IoT device with maliciously altered firmware. Such a device still possesses valid authentication credentials, such as a digital certificate and the corresponding private key, but they are under the control of the attacker. This allows the compromised device to authenticate with the network and start sending bogus data. To prevent such threats and to ensure that only devices with verified and trustworthy software and hardware configurations are allowed to join the network, a solution is to implement remote attestation.

Remote attestation [1] is a security service that verifies the integrity and trustworthiness of the device. It enables a device to generate evidence of its software and hardware state, which it sends to a remote entity for verification. The process typically relies on hardware isolation techniques such as the Trusted Execution Environment (TEE), which prevent the code residing in the *non-secure* world from accessing the code residing in the *secure* world. An attestation service is implemented within the secure world on a device. The code residing in non-secure world can then invoke the attestation service and request evidence of its state. The evidence is typically a token signed by a private key that the non-secure world cannot access.

There are many ways of generating evidence [2]–[4]. Conveying the evidence to a remote party for verification and ensuring it is *fresh* is another challenge. Existing research integrates remote attestation frameworks with Transport Layer Security (TLS) for secure data transport in non-constrained environments [5]. However, constrained environments face greater challenges due to limited data rates (from kbps to 2 Mbps), restricted maximum transmission units (e.g. up to 256 B), and the necessity for low energy consumption. To our knowledge, no studies to date have proposed transport mechanisms for performing remote attestation in constrained environments.

To address this challenge, this paper proposes AuRA, a novel, complete, lightweight remote attestation process specifically designed for constrained embedded systems. Its contribution is three-fold:

1) A first hardware-agnostic attestation and authentication transport framework for low-power constrained IoT devices. We promote our approach for standardization within the Internet Engineering Task Force (IETF).
2) A proof-of-concept implementation available under an open-source 3-clause BSD license.
3) An evaluation of the framework on the nRF5340 ARM Cortex-M33 microcontroller, measuring message sizes, memory footprint, and energy consumption. We also compare the token sizes with the state-of-the-art in the non-constrained world. We demonstrate the feasibility of AuRA in resource-limited environments.

The remainder of this paper is organized as follows: Section II presents the preliminaries. Section III reviews the related work. Section IV describes our proposal and is followed by Section V which discusses the evaluation results. Section VI concludes the paper.

## II. Background

### A. Remote Attestation

Remote attestation is a security service that verifies and confirms the integrity and trustworthiness of a remote device or system. A standardized way of doing remote attestation involves three entities [1].

- The *Attester* provides reliable evidence of its current state.

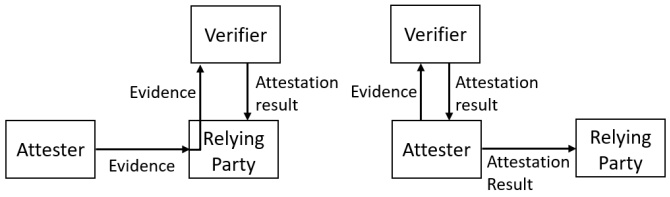


Fig. 1: RATS background-check model (left) and RATS passport model (right) [1].

- The *Verifier* evaluates the evidence and produces attestation results.
- The *Relying Party* consumes the attestation results to take application-specific actions.

IETF RATS (Remote ATtestation procedureS[1]) is a standardization working group that defines two attestation models: the background-check model and the passport model (Fig. 1). In the background-check model, the Attester sends evidence to the Relying Party, which consults the Verifier for evaluation and receives the attestation result. In the passport model, the Attester sends evidence directly to the Verifier, receives the attestation result, and then passes it to the Relying Party. A trade-off involved is that when the Attester faces connection constraints, the background-check model is preferred.

### *B. Hardware Requirements for Evidence Generation*

Hardware-based attestation mechanisms typically rely on the dedicated security co-processors and trusted execution environments (TEEs).

The Trusted Platform Module (TPM) [6] is a widely used coprocessor for attestation [7], [8]. TPM uses Platform Configuration Registers (PCRs) that cannot be overwritten but can only be extended through a cryptographic hash combining new measurements with previous PCR values. During attestation, the PCR values are signed with an attestation key to produce verifiable evidence.

TEEs, such as ARM TrustZone [9] and Intel SGX [10], partition the processor into secure and normal worlds, ensuring hardware isolation and secure memory encryption. Several studies focus on them [11], [12].

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

Ephemeral Diffie-Hellman over COSE (EDHOC) [13] is a newly standardized protocol that enables two parties to perform lightweight authenticated key exchange. EDHOC was developed by the IETF Lightweight Authenticated Key Exchange (LAKE) working group[2], with the goal of addressing resource limitations of constrained IoT devices and networks. The EDHOC protocol involves two parties: an Initiator and a Responder. They obtain authenticated session keys after completing the EDHOC handshake (Fig. 2).

In EDHOC `message_1`, the Initiator sends a supported authentication method using either a signature or static Diffie-Hellman key (representing authentication by a signature or a Message Authentication Code – MAC – respectively). The message also includes a selected cipher suite, an ephemeral public key (`G_X`), a connection identifier (`C_I`), and an optional External Authorization Data (EAD), `EAD_1`. EDHOC supports the integration of external security applications by transporting EAD through its designated field in the protocol.

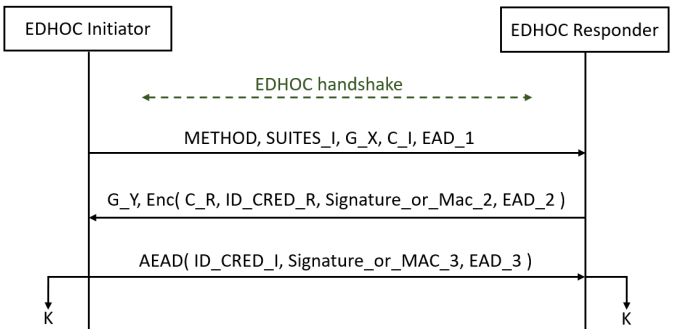


Fig. 2: The Ephemeral Diffie-Hellman over COSE (EDHOC) protocol [13].

In EDHOC `message_2`, the Responder replies with its ephemeral public key (`G_Y`), and an encrypted plaintext that contains its connection identifier (`C_R`), a credential identifier, an authentication item (either a signature or a MAC) and an optional `EAD_2`. The exchanged connection identifiers (`C_I` and `C_R`) serve as references to correlate the messages in an EDHOC session. EDHOC also allows the full authentication credentials to be omitted within the protocol messages. Instead, the Responder and the Initiator may send credential identifiers (`ID_CRED_R` and `ID_CRED_I`) to enable the retrieval of authentication credentials and keys.

EDHOC `message_3` is a ciphertext containing the Initiator's credential identifier, an authentication item (signature or MAC), and an optional `EAD_3`. After creating `message_3`, the Initiator derives the shared secret. Upon receiving `message_3`, the Responder also derives the shared secret, completing the authentication handshake. This shared secret is then used to secure subsequent application messages.

## III. Related Work

Remote attestation for low-power IoT devices focuses on minimizing hardware dependencies and creating lightweight, compatible mechanisms for constrained environments.

Some lightweight remote attestation mechanisms focus on monitoring or capturing the memory state to generate evidence. RealSWATT [2] is a software-based attestation approach that periodically scans memory to generate evidence, enabling continuous attestation to promptly detect any compromise or malicious activity. LAPE [3] provides lightweight control-flow attestation for resource-constrained IoT devices. It generates evidence by monitoring function calls within the firmware, which the Verifier compares against a pre-stored control-flow graph to detect any unauthorized deviations.

Some attestation mechanisms involve additional hardware-feature designs. SIMPLE [14] designs a software-based memory isolation technique called the Security MicroVisor (S$\mu$V).

[1] https://datatracker.ietf.org/wg/rats/
[2] https://datatracker.ietf.org/wg/lake/

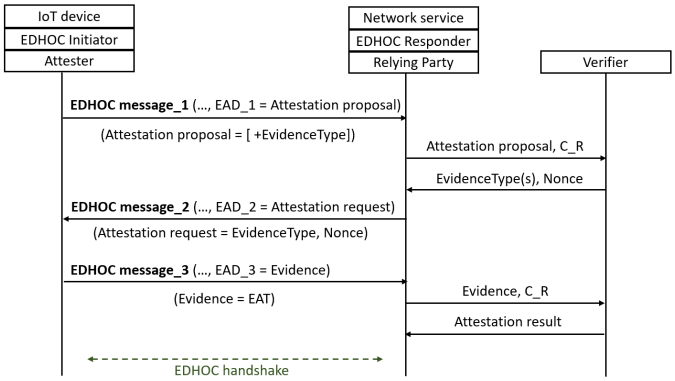


Fig. 3: AuRA enables a constrained IoT device to securely prove to a verifier that it is running the right hardware and software.

It isolates the memory for the Trusted Computing Module (TCM) from untrusted software, ensuring that no unauthorized code can access the protected regions. CHARM [4] integrates Hardware Performance Counters (HPCs)-based monitoring to generate vectors representing the device state, which are analyzed by a machine learning classifier on the verifier side to detect malicious behavior. Another alternative is the use of Physical Unclonable Functions (PUFs), which can be defined as functions embedded into the hardware. PUF provides unique, unclonable responses to specific challenges, making it ideal for generating tamper-resistant identifiers and have been explored in various studies [15] as part of attestation mechanisms. In AuRA, we adopt the mechanism that examines memory and hashes the firmware image to generate evidence representing the device state.

For remote attestation incorporated with secure transport mechanisms, Ott *et al.* [5] integrated remote attestation into the TLS handshake via a post-handshake process, allowing devices to verify their integrity after establishing a secure communication channel. While this approach completes the attestation process with TPM, it only supports evidence generation for microcontrollers without an attestation channel. Feng *et al.* [16] introduced a combined attestation and authentication mechanism that verifies both software integrity and device identity. However, this method relies on PUFs, which are uncommon in many constrained devices and pose implementation challenges.

To our knowledge, no existing research integrates the attestation mechanisms with an authentication channel specifically designed for hardware-agnostic, resource-limited devices.

## IV. AuRA: Lightweight Remote Attestation over EDHOC

We propose AuRA, which runs **Au**thentication in parallel with **R**emote **A**ttestation, using the External Authorization Data (EAD) fields of the EDHOC protocol. The detailed message flow is shown in Fig. 3.

### A. Attestation Protocol

The constrained IoT device acts as the Attester. The network service acts as the Relying Party; it is responsible for granting network access based on the attestation result. A trusted web server acts as the Verifier, which evaluates the evidence from the Attester. The EDHOC session is established between the Attester and the Relying Party.

To ensure evidence freshness and prevent replay attacks, a unique Nonce is generated by the Verifier and returned with the evidence to validate the session's integrity.

During the EDHOC handshake, three EAD fields (`EAD_1`, `EAD_2`, and `EAD_3`) are exchanged for remote attestation. `EAD_1` (*Attestation Proposal*) lists the evidence types supported by the Attester. The evidence type specifies the encoding method and the content format of the evidence and is identified by an integer registered under the Internet Assigned Numbers Authority (IANA). The Relying Party forwards the *Attestation Proposal* to the Verifier along with the EDHOC session connection identifier (`C_R`). The Verifier binds the received `C_R` with a randomly generated Nonce to uniquely identify the attestation session, enabling simultaneous attestations from multiple devices. The Verifier then selects a compatible evidence type, and responds with the selected type and Nonce. Based on these inputs, the Relying Party generates `EAD_2` (*Attestation Request*), and sends it to the Attester, specifying the chosen evidence type and Nonce. Upon receiving the request, the Attester generates the evidence and returns it in `EAD_3` (*Evidence*).

The privacy considerations for remote attestation align with those of EDHOC's EAD fields. `EAD_1` does not provide resistance to active or passive attackers due to the absence of authentication. `EAD_2` provides confidentiality via encryption but remains vulnerable to active attackers. `EAD_3` offers robust protection against both, securing the attestation evidence exchanged between Initiator and Responder.

### B. Evidence Generation

The *Evidence* is formatted as a signed attestation token structured as `COSE_Sign1`, a cryptographically signed piece of data that ensures the integrity of the device. The Attester signs the payload of the attestation token with its private key, using the signature algorithm specified in the header of `COSE_Sign1`. The Verifier uses the corresponding public key to validate the signature, which is prepared in a provisioning step before the attestation process. In AuRA, this attestation token follows the IETF "Entity Attestation Token" [17]. The details of the evidence generation process are provided in Section V-B. The attestation evidence includes information on the device's software and hardware state, particularly the active firmware image of the microcontroller.

Our proposal AuRA addresses the gaps in existing remote attestation solutions by integrating a lightweight hardware-agnostic attestation framework into EDHOC, specifically for constrained networks. The *Evidence* is securely transmitted in EDHOC's third message, encrypted using AEAD (Authenticated Encryption with Associated Data), ensuring evidence protection within an authenticated channel.

### C. Standardization

The AuRA proposal outlined in this section is promoted for standardization within the IETF [18].

## V. Evaluation

### A. Setup

To evaluate AuRA, we use the DotBot[3], an open-source micro-robot supporting wireless communication and indoor localization. The DotBot is a robotic platform designed for research and education, based on the Cortex-M33 nRF5340 microcontroller operating at 64 MHz.

The DotBot communicates wirelessly with a gateway over Bluetooth Low Energy (BLE) at 2.4 GHz. The gateway, built on the nRF52840 platform, acts as an intermediary by receiving the data from the DotBot and forwarding it to the computer via a USB connection, using the UART protocol. The computer executes the DotBot software written in Python, which we refer to as the "controller". The controller handles all processing tasks and high-level operations, including swarm coordinating commands via a web server. The evaluation setup is shown in Fig. 4.

We map the remote attestation entities that we present in Section IV in the background-check model to the DotBot infrastructure. The DotBot acts as the Attester, and the DotBot controller software implements the Relying Party (RP). A web server acts as the Verifier. The Verifier implements a policy that accepts different firmware versions along with their corresponding hashes. We implement the background-check model, illustrated in Fig. 4.

### B. Evidence Format Size Comparison

In AuRA, we define the evidence format as a token, specifically following the structure of the Entity Attestation Token (EAT), as standardized by the IETF RATS [17].

EAT supports various profiles that specify implementation choices across different aspects of the token, offering flexibility for diverse application needs. Two widely adopted options for generating evidence as an attestation token are:

1) The Arm's Platform Security Architecture (PSA) Attestation Token [19], a specific profile of the Entity Attestation Token (EAT).
2) An EAT with the *"measurements"* claim formatted using Concise Software Identification Tags (CoSWID [20]).

We compare the token formats by listing their required attributes and minimum byte sizes.

CoSWID does not specify a fixed size range for text-type attributes. Since CoSWID serves as the format for *measurements* claims in EAT, and the "ueid" in EAT specifies a minimum size of 10 bytes for text, we apply the same minimum size of 10 bytes for text attributes within CoSWID. As shown in Tables I and II, the minimal byte requirement for PSA Attestation Token is 195 B, whereas EAT with CoSWID-formatted *measurements* claims requires 90 B. These numbers are not extracted from a real implementation, rather a theoretical comparison with the minimum size requirements.

| Attribute | | Size (Bytes) |
|---|---|---|
| psa-nonce | | 32 |
| psa-instance-id | | 33 |
| psa-profile | | 33 |
| psa-implementation-id | | 32 |
| psa-client-id | | 1 |
| psa-software-components | measurement-value | 32 |
| | signer-id | 32 |
| **Total** | | 195 |

TABLE I: Entity Attestation Token profiled in Arm's Platform Security Architecture (PSA) Attestation Token.

| Attribute | | | | Size (Bytes) |
|---|---|---|---|---|
| eat-nonce | | | | 8 |
| ueid | | | | 7 |
| measurements | tag-id | | | 10 |
| | tag-version | | | 1 |
| | software-name | | | 10 |
| | entity | entity-name | | 10 |
| | | role | | 1 |
| | evidence | fs-name | | 10 |
| | | hash | hash-alg-id | 1 |
| | | | hash-value | 32 |
| **Total** | | | | 90 |

TABLE II: Entity Attestation Token with the *"measurements"* claim formatted in Concise Software Identification Tags.

We conclude that PSA is more suitable for high-end devices, as it includes a greater number of attributes to detail the various hardware and software component IDs. Additionally, the PSA Attestation Token requires a TrustZone-enabled processor with a CPU capable of secure and non-secure modes, which necessitates additional configuration not standard in all embedded devices. In contrast, CoSWID in EAT is more appropriate for low-end devices, where fewer unique IDs and minimal overhead are needed, and where generation imposes no specific hardware requirements. For constrained IoT use cases, we opt for EAT with CoSWID-formatted *measurements* claims due to its lightweight structure and compatibility with all low-resource environments.

### C. Memory Footprint

We evaluate the static memory footprint of our proposal and compare it to the case of the EDHOC protocol without remote attestation.

In our evaluation, we use the lakers library[4] in Rust, integrated as a static library within the DotBot project. We configure lakers with its authentication method 3, which uses static Diffie-Hellman keys for both parties, and cipher suite 2, relying on Elliptic-Curve Diffie-Hellman (ECDH) on the P-256 curve, the SHA-256 hashing algorithm, and the AES-128-CCM authenticated encryption scheme. We developed a C library[5] specifically integrated with the DotBot framework for the attestation component, including the preparation and processing of the attestation items.

[3] https://github.com/DotBots/DotBot-firmware

[4] https://github.com/openwsn-berkeley/lakers/

[5] https://github.com/ysong02/DotBot-firmware/tree/main/drv/attestation

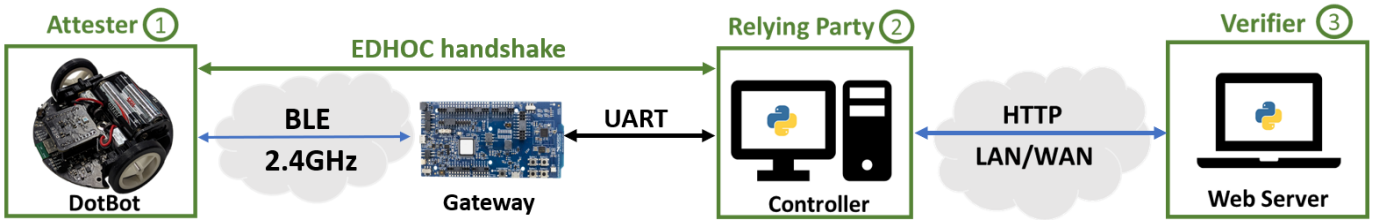


Fig. 4: Setup for evaluating AuRA in the "background-check" model.

TABLE III: AuRA memory footprint.

| | Flash | RAM |
|---|---|---|
| AuRA | 1,927 B | 1,480 B |
| EDHOC | 15,236 B | 5,185 B |
| **Total** | **17,163 B** | **6,665 B** |

TABLE IV: AuRA message sizes.

| | message 1 | message 2 | message 3 |
|---|---|---|---|
| AuRA | 4 B | 13 B | 202 B |
| EDHOC | 38 B | 46 B | 21 B |
| **Total** | **42 B** | **59 B** | **223 B** |

Table III shows that our baseline, the EDHOC protocol, consumes 15,236 B of flash memory. When integrated with the attestation process, flash memory consumption rises to 17,163 B, which is an increase of 1,927 B (12.6%). In terms of RAM, EDHOC alone requires 5,185 B. With attestation, the RAM consumption increases to 6,665 B, an increase of 1,480 B (28.5%) for the attestation-related operations.

The DotBot features the nRF5340, which has 1 MB of flash memory and 512 kB of RAM. The total flash memory usage for EDHOC with remote attestation is 1.64% of available flash, with attestation operations contributing 0.18%. The RAM usage of remote attestation over EDHOC represents 1.27% of the total RAM, attestation accounting for 0.28%. We can conclude that the memory usage of both flash and RAM can be considered acceptable given the memory capacity of the nRF5340 microcontroller.

### D. Message Sizes

We evaluate the communication overhead of remote attestation over EDHOC by measuring the message sizes for each of the three EDHOC messages, along with the additional payload introduced by the attestation items (EAD items). The results were obtained directly from the log files generated by the controller (Relying Party). These logs captured the precise communication overhead. The results are shown in Table IV.

The first message that initiates the EDHOC protocol has a size of 38 B. The `EAD_1` adds 4 B for evidence type information, totaling 42 B. The second message, including an ephemeral key, an encrypted credential identifier and MAC, is 46 B. `EAD_2` adds 13 B (nonce and an integer), resulting in a total of 59 B. The third EDHOC message is 21 B. `EAD_3` adds 202 B of attestation evidence, bringing the total to 223 B. These sizes are acceptable for our BLE physical layer, which supports a maximum transmission unit of 255 B[6]. For other physical layers, such as LoRA with a maximum payload size of 255 B depending on the Spreading Factor, Coding Rate and other configurations, or IEEE 802.15.4 with a maximum payload size of 127 B, message 3 would need to be fragmented.

[6] https://docs.nordicsemi.com/bundle/ps_nrf5340/

### E. Energy Consumption

In this section, we evaluate the energy consumption and time required for performing remote attestation over EDHOC.

For our measurements, we used a power profiler and a logic analyzer. We use the Otii Arc power profiler to supply power and measure energy consumption during protocol execution, monitoring current over time. The Otii Arc can read signals from a digital pin, allowing us to track timing throughout the handshake. The Saleae Logic Analyzer with a sample rate of 500 MS/s and a maximum frequency of 100 MHz, is used for precise timing measurements of the EDHOC handshake, including attestation operations on the DotBot.

We collect the energy data using the Otii Arc software along with another file for the General-Purpose Input/Output (GPIO) signal. To synchronize timing and energy measurements, both the Otii Arc and Saleae Logic Analyzer are connected to the same GPIO pin on the DotBot. Results are shown in Fig. 5.

Fig. 5 shows that completing the EDHOC handshake, including both authentication and attestation, takes 5.51 s and consumes 88 mC of charge, with an average current of 16 mA. Attestation accounts for 63% of the charge due to the signing and hashing operations for evidence generation.

TABLE V: Timing breakdown of attestation stages in Fig. 5.

| Stage Name | Duration (s) |
|---|---|
| initiator_new | 0.36 |
| prepare_ead_1 | 2.00e-6 |
| prepare_message_1 | 1.33e-3 |
| radio_tx_rx | 0.02 |
| parse_message_2+fetch_creds | 0.51 |
| verify_message_2 | 1.03 |
| process_ead_2 | 7.00e-6 |
| encoding | 1.90e-5 |
| hash_and_encoding_image | 1.53 |
| signature_then_encoding | 2.05 |
| prepare_message_3 | 0.01 |
| tx_message_3 | 2.3e-3 |

Based on Table V, the most time-consuming operation is the signature signing and encoding, which takes 2 s due to the use of Ed25519, involving both hashing and signing. This signature generation is performed only once per attestation token, reducing its impact on overall feasibility.

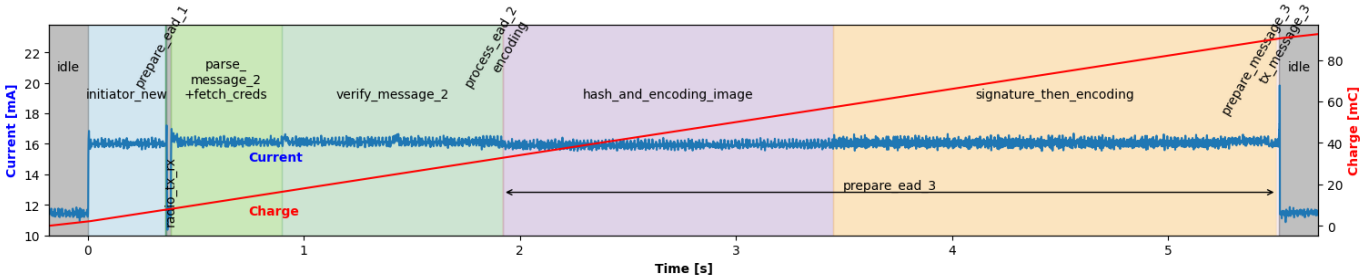


Fig. 5: Current consumption benchmark during attestation.

The second most time-consuming operation is hashing and encoding the firmware image, which takes 1.5 s. The input for the hashing process is the entire firmware image, which has a size of 520 kB. All other processes are completed in under 1 s, with specific durations as follows: verifying message 2 takes 1 s, parsing message 2 and fetching credentials takes 0.51 s, and initiating a new process for the initiator takes 0.36 s. Preparing `EAD_1` and processing `EAD_2` are notably time-efficient, taking only 2 $\mu$s and 7 $\mu$s, respectively.

Since we are using a software-based cryptographic backend, the results reflect a worst-case situation, where platforms with hardware accelerators could perform up to twice faster in the cryptographic operations. We can conclude, as expected, that the processing is dominated by the cryptographic operations.

## VI. Conclusions

This paper introduces AuRA, the first complete, hardware-agnostic remote attestation framework integrated with the EDHOC authenticated channel, which is specifically designed for constrained IoT environments. Our approach is standards-based that aligns with the IETF RATS architecture and leverages the EDHOC protocol by IETF LAKE, ensuring compatibility with future IoT security frameworks. We demonstrate its lightweight nature, efficiency and feasibility of this approach through an evaluation on a low-power micro-robot using the nRF5340.

We are currently implementing a mutual attestation over the EDHOC protocol. Furthermore, we aim to integrate the remote attestation over EDHOC with swarm-based collective attestation mechanisms.

## 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.

## References

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

[2] S. Surminski, C. Niesler, F. Brasser, L. Davi, and A.-R. Sadeghi, "Realswatt: Remote software-based attestation for embedded devices under realtime constraints," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021.

[3] D. Huo, Y. Wang, C. Liu, M. Li, Y. Wang, and Z. Xu, "Lape: A lightweight attestation of program execution scheme for bare-metal systems," in *2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, 2020.

[4] D. Li Calsi and V. Zaccaria, "Interruptible remote attestation of low-end iot microcontrollers via performance counters," *ACM Trans. Embed. Comput. Syst.*, Sep. 2023.

[5] S. Ott, M. Kamhuber, J. Pecholt, and S. Wessel, "Universal remote attestation for cloud and edge platforms," in *Proceedings of the 18th International Conference on Availability, Reliability and Security*, 2023.

[6] Trusted Computing Group. (2024) Trusted platform module (tpm). [Online]. Available: https://trustedcomputinggroup.org/work-groups/trusted-platform-module/

[7] A. Dave, M. Wiseman, and D. Safford, "Sedat: Security enhanced device attestation with tpm2. 0," *arXiv preprint arXiv:2101.06362*, 2021.

[8] P. G. Wagner, P. Birnstill, and J. Beyerer, "Dds security+: Enhancing the data distribution service with tpm-based remote attestation," in *Proceedings of the 19th International Conference on Availability, Reliability and Security*, 2024.

[9] Arm Limited. (2024) Arm trustzone technology. [Online]. Available: https://developer.arm.com/documentation/PRD29-GENC-009492/latest/

[10] V. Costan, "Intel sgx explained," *IACR Cryptol, EPrint Arch*, 2016.

[11] Z. Zhang, "Enhancing iot security through trusted execution environments," in *2024 2nd International Conference on Image, Algorithms and Artificial Intelligence (ICIAAI 2024)*, 2024.

[12] R. Nagy, M. Bak, D. Papp, and L. Buttyán, "T-raid: Tee-based remote attestation for iot devices," in *International ISCIS Security Workshop*, 2021.

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

[14] M. Ammar, B. Crispo, and G. Tsudik, "Simple: A remote attestation approach for resource-constrained iot devices," in *2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS)*, 2020.

[15] A. Al-Meer and S. Al-Kuwari, "Physical unclonable functions (puf) for iot devices," *ACM Computing Surveys*, 2023.

[16] W. Feng, Y. Qin, S. Zhao, and D. Feng, "Aaot: Lightweight attestation and authentication of low-resource things in iot and cps," *Computer Networks*, 2018.

[17] L. Lundblade, G. Mandyam, J. O'Donoghue, and C. Wallace, *The Entity Attestation Token (EAT)*, Internet Engineering Task Force (IETF) Std. draft-ietf-rats-eat-31, 2024.

[18] Y. Song, *Remote attestation over EDHOC*, Internet Engineering Task Force (IETF) Std. draft-song-lake-ra-02, 2024.

[19] H. Tschofenig, S. Frost, M. Brossard, A. L.Shaw, and T. Fossati, *Arm's Platform Security Architecture (PSA) Attestation Token*, Internet Engineering Task Force (IETF) Std. draft-tschofenig-rats-psa-token-23, 2024.

[20] H. Birkholz, J. Fitzgerald-Mckay, C. Schmidt, and D. Waltermire, *Concise Software Identification Tags*, Internet Engineering Task Force (IETF) Std. RFC9393, 2023.