www.openswarm.eu

Call: HORIZON-CL4-2022-DATA-01

Type of action: RIA

Grant agreement: 101093046

Deliverable 2.4: Footprint of a Secure Firmware Management Solution for Embedded Devices

Work Package 2: Orchestration of Collaborative Smart Nodes

Task Lead: INRIA

WP Lead: ADI

This project has received funding from the European Union's Horizon Europe Framework Programme under Grant Agreement No. 101093046.

Funded by the European Union

Document information	
Author(s)	Mališa Vučinić, Yuxuan Song
Reviewers	Genki Miyauchi, Thomas Watteyne
Submission date	31-Oct-2024
Due date	31-Oct-2024
Type	Report
Dissemination level	Public

Document history

Date	Version	Author(s)	Comments
5 July 2024	01	Mališa Vučinić	preview
31 October 2024	02	Mališa Vučinić, Yuxuan Song	deliverable

Table of contents

Executive Summary

This document details the results achieved in the context of Task 2.3 and specifically the development of a secure firmware management solution for embedded devices. The goal of a secure firmware management OpenSwarm module is to enable the verification of the state of firmware on each of the swarm devices before admitting those devices to the network. This goal can be achieved through a security service called "remote attestation", which is described in this Deliverable. Remote attestation requires a swarm device that needs to be attested to contain a trusted module that can attest other non-trusted parts of the system. This module is called the root of trust and can be implemented through hardware isolation techniques that are present on constrained hardware used in the context of the OpenSwarm project. The root of trust performs measurements over the non-secure parts of the system and signs them with a private key that is not accessible from the outside, producing what is termed the "evidence". This Deliverable gathers the results achieved within the context of the OpenSwarm project that relate to how the evidence is communicated to a remote entity that has the ability to verify the evidence, i.e. the remote attestation protocol. The effort within this Task focused on standardizing the remote attestation protocol within the Internet Engineering Task Force (IETF) standardization body and specifically within the Lightweight Authenticated Key Exchange (LAKE) working group. The proposed protocol, under standardization in the IETF, is implemented in the OpenSwarm environment and the implementation is benchmarked in terms of its memory and communication footprint. In the appendix, we also detail a proposed technique to securely generate a key that can be used for signing the evidence by employing only the SRAM memory initialization values, which is known as the SRAM Physically Unclonable Function (PUF).

Introduction

As Internet-of-Things devices become ubiquitous, they are increasingly targeted by cyberattacks, making it crucial to verify their software and hardware configurations before they are admitted into a network. Remote attestation provides a way to verify device integrity and trustworthiness. The remote attestation process involves three entities: the Attester, which generates evidence of its state; the Verifier, which evaluates this evidence and generates the attestation results; and the Relying Party, which acts based on the attestation results.

Traditional protocols are not practical for low-power IoT devices due to memory and energy constraints. This deliverable proposes a lightweight remote attestation integrated into a newly standardized authenticated key exchange protocol Ephemeral Diffie-Hellman over COSE (EDHOC), designed for resource-limited environments.

The deliverable is organized as follows. We first present the state of the art on the topic of remote attestation. We then propose performing remote attestation in parallel with network access authentication and discuss the newly standardized EDHOC protocol. We use EDHOC to piggyback remote attestation information during network access authentication. We then specify the remote attestation protocol operating over EDHOC and the corresponding data structures in detail, and propose it for standardization within the IETF. We then implement the solution within the DotBot firmware ecosystem and benchmark it in terms of 1) message overhead, 2) memory footprint, and 3) energy consumption. We conclude the deliverable by presenting, in the appendix, a method for securely extracting a secret key from SRAM memory which can be used for the generation of a private signing key used for remote attestation.

Key Performance Indicators

The Key Performance Indicator (KPI) defined for Task 2.3 is shown in Table 1. It covers the number of protocol stacks the secure firmware management solution is running on. The remote attestation solution proposed for standardization in the IETF and developed in the context of the OpenSwarm project is implemented as a stand-alone project in the C programming language within the DotBot firmware ecosystem. As such, it can directly be used for securely managing the firmware of DotBots but also for any nRF5340-based microcontroller robotic platform, such as the CapBot robot developed by KU Leuven.

Table 1. KPI for Task T2.3

Task	Key Performance Indicator	Target	Achieved
T2.3	Number of protocol stacks the secure firmware management is running on	At least 2	2

Technology Readiness Level (TRL)

Before the start of the OpenSwarm project, the secure firmware management module was formulated (**TRL 2**). Currently, the solution developed in the context of Task 2.3 is under standardization within the IETF as an individual submission, as explained in Deliverable D2.3 Section Standardization. Inria has delivered an implementation of this secure firmware management solution, and it is available under an open-source 3-clause BSD license as part of the DotBot firmware project [1]. Inria has completed the experimental proof of concept. We conclude that the TRL of the secure firmware management solution is currently **TRL 3.**

Table 2. Technology readiness level of the secure firmware management solution.

Technology Developed	Start TRL	Current TRL
Secure firmware management solution for DotBots	TRL 2	TRL 3

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

List of Publications

- Standards-based Remote Attestation for Internet-of-Things Swarms. Yuxuan Song, Mališa Vučinić, Thomas Watteyne. IEEE International Conference on Robotics and Automation (ICRA), Breaking Swarm Stereotypes workshop, Yokohama, Japan, 17 May 2024.
- Remote attestation over EDHOC. Yuxuan Song. IETF Internet Draft. draft-song-lake-ra-00. March 2024.
- TMVS: Threshold-based Majority Voting Scheme for Robust SRAM PUFs. Sara Faour, Mališa Vučinić, Filip Maksimovic, David C. Burnett, Paul Muhlethaler, Thomas Watteyne, Kristofer Pister. IEEE Symposium on Computers and Communications (ISCC), Paris, 26-29 June 2024.
- Enhancing Security for Constrained IoT Devices with Lightweight Remote Attestation. Yuxuan Song, Mališa Vučinić, Thomas Watteyne. Extended Abstract at Journée thématique sur la sécurité des piles réseau et de leurs implémentations, Orléans, France, 30 Sep 2024.
- Remote Attestation over EDHOC for Constrained Internet-of-Things Use Cases. Yuxuan Song, Geovane Fedrecheski, Mališa Vučinić, Thomas Watteyne. In preparation for IEEE International Conference on Communications (ICC) 2025.

State of the Art

The state-of-the-art in remote attestation (RA) is characterized by the integration of advanced hardware and software technologies, and standardization efforts.

Hardware-based attestation relies on trusted computing architectures. The current widely used architectures are Trusted Platform Modules (TPMs) [1], ARM TrustZone [2], Intel Software Guard Extensions (SGX) [3], and Physically Unclonable Function (PUF) [4]. TPMs are hardware modules that enable secure storage and computation, which provides a root of trust for computing systems. With TPM, the attestation schemes can ensure that the attestation response is trustworthy. Intel SGX and ARM TrustZone have dedicated processor architectures with a hardware-enforced isolated execution environment, which guarantees the trusted execution of the attestation protocol. PUF can be defined as a function that is embedded into the hardware. PUF provides a unique response to specific input and challenge, making it ideal for creating an unclonable and tamper-resistant identifier. These hardware technologies provide capabilities for acting as a trust anchor and providing attestation evidence for verification. With hardware-based remote attestation, it is possible to achieve the highest security guarantees. However, it requires specialized hardware that is expensive to deploy on many devices, especially not suited for low-power constrained IoT devices.

Software-based remote attestation schemes usually do not require delicate hardware security components or access to specialized hardware. SWATT [5] is a software attestation system that uses a simple challenge-response strategy. SWATT computes a checksum of the memory using a pseudo-random number generator (PRNG) to iterate over the memory unpredictably. The attester receives a seed for the PRNG from the verifier, ensuring that every memory address is accessed at least once to prevent attackers from storing unexpected values. The protocol relies on strict timing checks, where any delay caused by injected malicious code is detectable. However, this dependency on precise timing may be a problem in networks with unpredictable delays. Pioneer [6] is similar to SWATT but designed for legacy devices with more processing power and memory. It improves on SWATT by including additional information in the checksum, such as the program counter, data pointer, and jump locations, to detect memory copy and illegal jump attacks. Pioneer operates in a two-step challenge-response protocol, where it first computes a checksum of the attestation functions and the checksum code, and also includes a challenge generated by the verifier. The next step is to hash an arbitrary piece of code in the memory being attested. The verifier checks both the checksum and the hash to confirm the trustworthiness. Several other software-based remote attestations are built over SWATT, and are improved in different aspects. For example, Shaneck et al. [7] propose a protocol that improves the strict time constraint of SWATT by making the attestation code different for every attestation to avoid replay attackers. Others are Proactive [8], memory filling over SWATT by AbuHmed et al. [9] and USAS [10]. However, not relying on special hardware limits the security guarantees with limited tamper resistance, which is more vulnerable against active attackers. Software-based remote attestation has been abandoned in the most recent remote attestation proposals because they rely strongly on assumptions and are considered insecure.

Hybrid attestation bridges the gap between software and hardware by leveraging security architectures to achieve the necessary guarantees to provide secure remote attestation. A recently published paper by Roman et al. [11] presents a new hybrid remote attestation by using PUFs and hash-based signatures for low-end IoT devices. They have a low-cost Root of Trust based on the combination of a PUF and read-only memory, in which the secret keys are reconstructed using the PUF. Other research groups such as Eldefrawy et al. [12] and Brasser et al. [13] also rely the attestation schemes on the existence of a minimal read-only hardware-protected memory. SMART [12], TrustLite [14], and TyTan [13] focus on the protection of secret keys and access control. They use ROM to store the data such as the keys and attestation codes that cannot be tampered with, and MPU controls access to the critical data. VRASED [15] proposes the first specified hybrid remote attestation architecture. They also use ROM to store the secure data, and they create 7 input signals and 1 output signal by FPGA to guarantee 7 high-level security properties. Many architectures are proposed based on VRASED. For example, RATA [16] is a hybrid attestation protocol over VRASED. The protocol made further expansion over VRASED to be against Time-Of-Check-Time-Of-Use (TOCTOU) related attacks, which perform illegal binary modifications on the system. They create a secure logging mechanism where a set of CPU signals are monitored to detect the memory modifications. Hybrid attestation is widely used to remain both inexpensive and secure.

There are also some standardization works on remote attestation. A working group in the IETF named Remote ATtestation procedureS (RATS) has standardized the structure and entities to do remote attestation [17]. An Internet-Draft in IETF describes a general way to perform remote attestation over Transport Layer Security (TLS) [18].

To overcome the scalability challenges of remote attestation, researchers explore the design of collective remote attestation, also called swarm attestation. Collective remote attestation is well-suited for IoT devices which are expected to work collaboratively in a swarm of large quantities. The majority of swarm attestation schemes follow a collective attestation mechanism. The verifier collects the security state of each device in the swarm. The collective remote attestation schemes differ on several aspects, whether they consider tree-based or distributed aggregation techniques, central or decentralized structure, and static or dynamic networks. SEDA [19] and SANA [20] are typical examples that the devices are tree-based and static during the attestation. The attestation requests and attestation results are propagated under a parent-child relationship. PADS [21] and SALAD [22] are for dynamic networks, the device acts as an attester and also a verifier to attest other devices they interact with. The messages are collected from the verifiers in the swarm up to the root of the verifier. Some collective remote attestations do not aggregate all the state information of each node, they set the neighbor nodes as supervisors. For example, LICAPA [23] is a newly published paper that detects a device under physical attack with the timestamps signed by other recently attested devices. Another example is by Kuang et al. [24], where they report the compromised nodes instead of transferring the attestation result.

Standards-based Remote Attestation for Internet-of-Things Swarms

The following work was published as "Standards-based Remote Attestation for Internet-of-Things Swarms. Yuxuan Song, Mališa Vučinić, Thomas Watteyne. IEEE International Conference on Robotics and Automation (ICRA), Breaking Swarm Stereotypes workshop, Yokohama, Japan, 17 May 2024."

Summary

Remote attestation is a security service to verify and confirm the integrity and trustworthiness of each robot in the swarm. We developed an efficient method of doing remote attestation in parallel with the network access authentication, utilizing a newly standardized key exchange protocol named EDHOC. We demonstrate the feasibility of this method through benchmarks on the hashing time of 520 kB under constrained scenarios, resulting in times of 780 ms on the nRF52840 and 728 ms on the nRF5340.

Introduction

To mitigate the risk of cyberattacks on robotic swarms, only legitimate robots should be admitted into the swarm. Let's imagine a robot in which an attacker has altered the firmware, but which still has valid authentication credentials such as a digital certificate and the corresponding private key. This allows it to still authenticate to the swarm operator. *How to ensure that only robots with verified and trustworthy firmware are allowed to join the swarm?*

The solution is to do a remote attestation. An attestation service running on a robot enables it to generate evidence about the state of its software and hardware. By sending this evidence to a remote entity called Verifier, the swarm operator can assess whether the robot is compromised. For a swarm of robots, the remote attestation procedure should be both energy and bandwidth efficient. A lightweight security protocol named Ephemeral Diffie-Hellman over COSE (EDHOC) [25] is standardized recently. EDHOC is a highly compact and efficient protocol that enables authenticated key exchange in constrained scenarios (see Deliverable D2.3 for more details on EDHOC). In this paper, we present a lightweight method to perform the remote attestation using EDHOC. The attestation procedure is integrated into the EDHOC handshake session. An evaluation is achieved on a swarm robotic platform to prove the feasibility of this method. We benchmarked the time to hash the bytes in firmware image size on the nRF52840 microcontroller and nRF5340 microcontroller. The results are 780 ms on the nRF52840 and 728 ms on the nRF5340 for hashing 520 kB, which shows that this method can be deployed even on the constrained platforms.

A Primer on Remote Attestation

Remote attestation is a security process that can help the swarm central server establish a level of trust in the robot before allowing the robot to join the swarm. A standardized way of doing remote attestation involves three entities. The Attester provides reliable evidence about the state of itself. The Verifier evaluates the evidence and produces attestation results. The Relying Party consumes the attestation results to execute application-specific actions. The architecture for doing remote attestation is shown in Figure 1. The Relying Party conveys the evidence from the Attester to the Verifier. The Verifier then sends back the attestation result to the Relying Party.

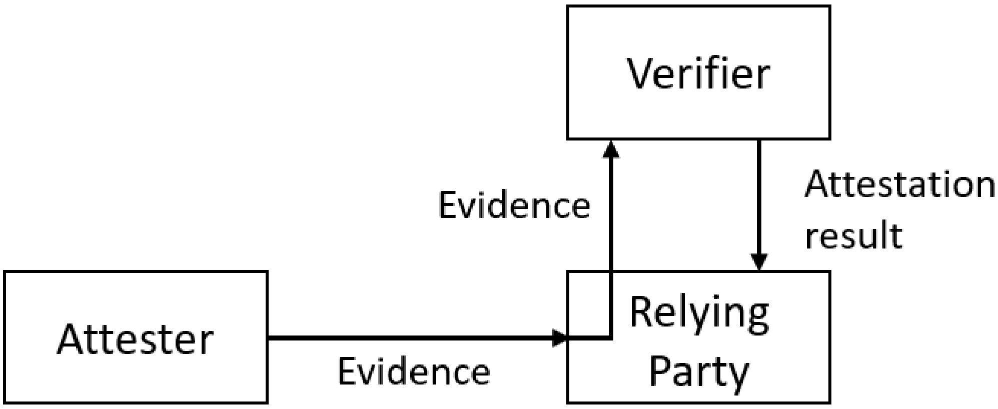

Figure 1. Remote attestation architecture.

Ephemeral Diffie-Hellman over COSE (EDHOC) Protocol

EDHOC [25] was recently standardized in the IETF. It is a key exchange protocol based on ephemeral elliptic curve Diffie-Hellman (ECDH) keys. The execution of EDHOC takes place between an Initiator and a Responder. The protocol consists of three mandatory messages (message 1, message 2, and message 3) and an optional fourth message (message 4). EDHOC also allows external security applications to be integrated in the handshake by defining the External Authorization Data (EAD) field. EADs are sent in dedicated fields of the EDHOC messages: EAD 1, EAD 2 and EAD 3.

Fedrecheski et al. [26] have compared EDHOC with Datagram Transport Layer Security (DTLS) protocol 1.3, a widely used protocol for key exchange on the Internet. Their study comprises several metrics, including message footprint, time, energy, and memory.

Results indicate that, in constrained IoT networks, EDHOC outperformed DTLS in all metrics. Compared to DTLS, EDHOC exhibits a 7× reduction in time-on-air, a 1.44× reduction in handshake duration, a 2.79× reduction in energy consumption, and a 4× reduction in flash and RAM memory usage.

Remote Attestation over EDHOC

We propose to do remote attestation in parallel with network access authentication. In that setting, we propose to map the EDHOC Initiator to the Attester, and the EDHOC Responder to the Relying Party. The Verifier is hosted on the central swarm server. The Attester and the Relying Party communicate by transporting messages within EDHOC's External Authorization Data (EAD) fields. The Attester sends an attestation proposal in EDHOC's EAD 1 field, which contains all the supported evidence types by the Attester. The evidence type indicates the format type and an appropriate profile of the evidence. For example, an evidence type indicates that the evidence is formatted as CBOR Web Token (CWT), with a profile of Platform Security Architecture (PSA) claims [27]. If the Verifier can support at least one of the provided evidence types, the Relying Party signals to the Attester an attestation request in EDHOC's EAD 2 field. The Verifier specifies a selected evidence type and a Nonce to guarantee the message freshness.

The Attester conveys the evidence to the Relying Party in EDHOC's EAD 3 field. The evidence is defined as an attestation token [28], an attested claims set. The information about the state of robot software, such as the active firmware image or the software configurations are carried within the evidence. The value of the measurement must be a hash of 32, 48, or 64 bytes. This hash, together with other useful information is then signed or symmetrically protected with a MAC to construct the evidence. The evidence is then sent to and evaluated by the Verifier.

Evaluation

To evaluate the feasibility of remote attestation in a constrained swarm, we benchmark the execution of a SHA-256 hash function on the DotBot swarm robot (Figure 2). DotBot is a cm-scale micro-robot that is developed for education and research purposes within the scope of the OpenSwarm project. All DotBot developments are done under open-source hardware and software licenses.

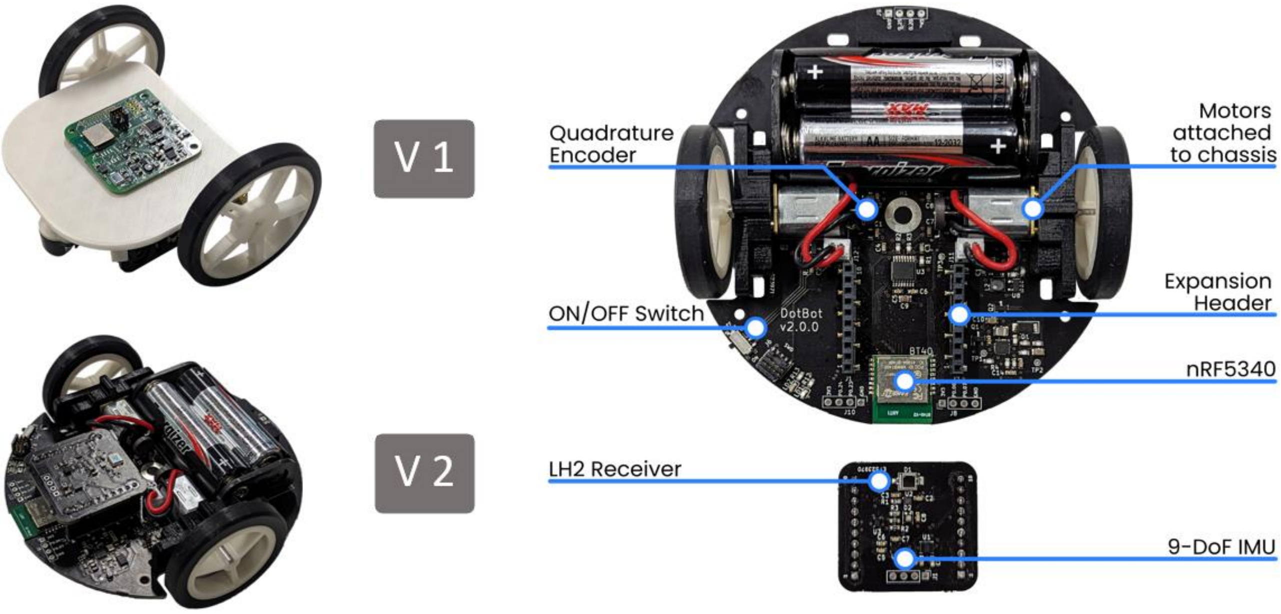

Figure 2. DotBot platform.

The first version runs on the nRF52840 microcontroller and the second version runs on the nRF5340 microcontroller. We evaluate the time to hash a string of bytes about the same size as a firmware image. The maximum size of a firmware image is 520 kB. As shown in the results in Figure 3, the time to hash increases linearly with the number of bytes. We can see that the hash of 520 kB takes approximately 780 ms on the nRF52840, approximately 728 ms on the nRF5340. This result shows the feasibility of hashing the entire firmware image even on constrained platforms. This does not need to be done in real-time; attestation is expected to be done only a handful of times throughout the lifetime of the robot.

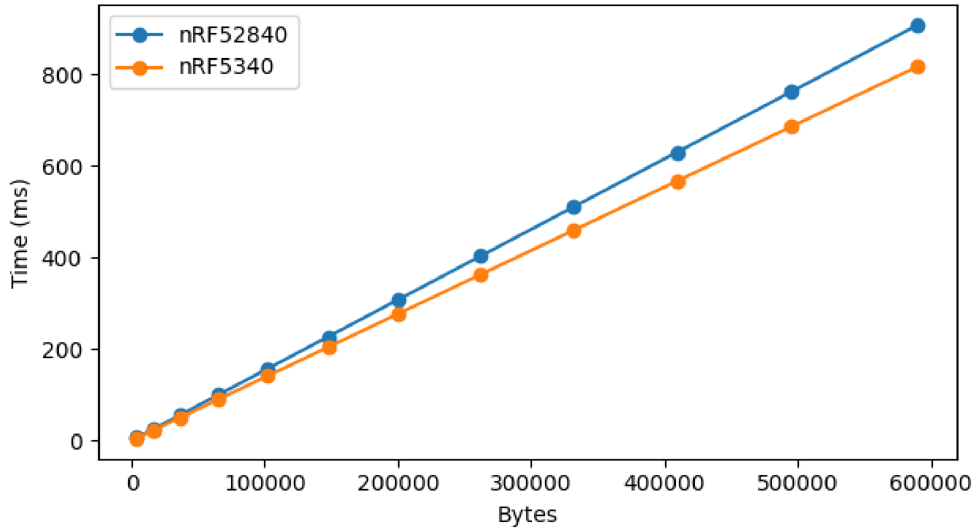

Figure 3. Benchmark of a software implementation of SHA-256 on a constrained swarm robotic platform DotBot.

Specification of Remote Attestation over EDHOC

The following work was published as an Individual Internet-Draft within the IETF as "Remote attestation over EDHOC. Yuxuan Song. IETF Internet Draft. draft-song-lake-ra-00".

Summary

This section specifies how to perform remote attestation as part of the lightweight authenticated Diffie-Hellman key exchange protocol EDHOC, based on the RATS architecture.

Introduction

The IETF working group RATS has defined an architecture [17] for remote attestation. The three main roles in the RATS architecture are the Attester, the Verifier, and the

Relying Party. The Attester generates the evidence concerning its identity and integrity, which must be appraised by the Verifier for its validity. Then, the Verifier produces the attestation result, which is consequently used by the Relying Party for the purposes of reliably applying application-specific actions.

One type of interaction model defined in the RATS architecture is called the background-check model. It resembles the procedure of how employers perform background checks to determine the prospective employee's trustworthiness, by contacting the respective organization that issues a report. In this case, the employer acts as the Relying Party, the employee acts as the Attester and the organization acts as the Verifier. The Attester conveys evidence directly to the Relying Party and the Relying Party forwards the evidence to the Verifier for appraisal. Once the attestation result is computed by the Verifier, it is sent back to the Relying Party to decide what action to take based on the attestation result. This specification employs the RATS background check model.

One way of conveying attestation evidence is the Entity Attestation Token (EAT) [28]. It provides an attested claims set that describes the state and the characteristics of the Attester, which can be used to determine its level of trustworthiness. This specification relies on the EAT as the attestation evidence.

EDHOC [25] is a lightweight authenticated key exchange protocol for highly constrained networks. In EDHOC, the two parties involved in the key exchange are referred to as the Initiator (I) and the Responder (R). EDHOC supports the transport of external authorization data, through the dedicated EAD fields. This specification delivers EAT through EDHOC. Specifically, EAT is transported as an EAD item.

Typically, the Attester incorporates an internal attestation service, including a specific trusted element known as the "root of trust". Root of trust serves as the starting point for establishing and validating the trustworthiness appraisals of other components on the system. The measurements signed by the attestation service are referred to as the Evidence. The signing is requested through an attestation API. How the components are separated between the secure and non-secure worlds on a device is out of the scope of this specification.

Terminology

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, RFC2119, RFC8174 when, and only when, they appear in all capitals, as shown here.

The reader is assumed to be familiar with the terms and concepts defined in EDHOC [25] and RATS [17].

Problem Description

This section describes how to perform remote attestation over the EDHOC protocol according to the RATS architecture. Remote attestation protocol elements are carried within EDHOC's External Authorization Data (EAD) fields. More specifically, the specification supports the RATS background check model. It describes how the Attester (EDHOC Initiator) and Relying Party (EDHOC Responder) complete the EDHOC handshake complemented with remote attestation protocol elements.

Assumptions

The details of the protocol between Relying Party and Verifier are out of the scope. The only assumption is that the Verifier outputs a fresh nonce and that same nonce is passed on to the EDHOC session. That is where the link between the two protocols comes in. The remainder, such as the evidence type selection is just the negotiation. In general, the Verifier is supposed to know how to verify more than one format of the evidence type. Therefore, the Verifier MUST send back at least one format to the Relying Party. We assume in this specification that the Relying Party also has knowledge about the Attester, so it can narrow down the type selection and send to the Attester only one format of evidence type.

The Attester should have an explicit relation with the Verifier, such as from the device manufacturer so that the Verifier can evaluate the Evidence that is produced by the Attester. The authentication between the Attester and the Relying Party is performed with EDHOC [25] and defines the process of remote attestation using the External Authorization Data (EAD) fields defined in EDHOC.

The Protocol

Overview

EDHOC Initiator plays the role of the RATS Attester. EDHOC Responder plays the role of the RATS Relying Party. An external entity, which is out of the scope of this specification, plays the role of the RATS Verifier.

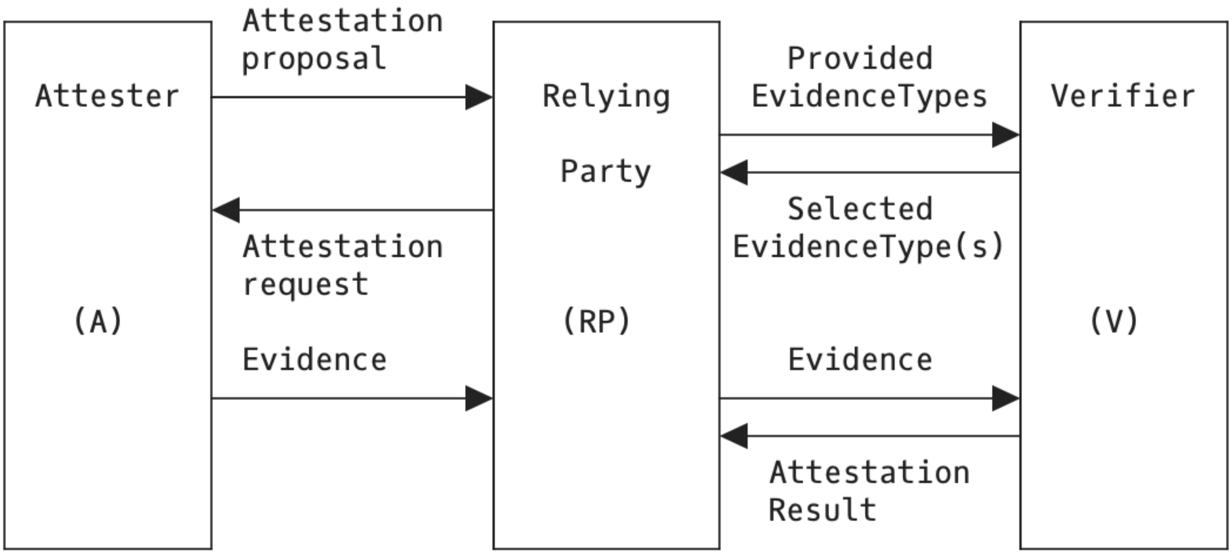

Figure 4. Overview of message flow. EDHOC is used between A and RP. Remote attestation proposal and request are sent in EDHOC External Authorization Data (EAD). The link between V and RP is out of the scope.

The Attester and the Relying Party communicate by transporting messages within EDHOC's External Authorization Data (EAD) fields.

External Authorization Data 1

In EAD_1, the Attester transports the Proposed_EvidenceType object. It signals to the Relying Party the proposal to do remote attestation, as well as which attestation claims the Attester supports. The supported attestation claims are encoded in CBOR in the form of a sequence.

The external authorization data EAD_1 contains an EAD item with

ead_label = TBD1

ead_value = Attestation_proposal, which is a CBOR byte string:

Attestation_proposal = bstr .cbor Proposed_EvidenceType

```
Proposed_EvidenceType = (
    content-format:     [ + uint]
)
```

where

content-format is an array that contains all the supported evidence types by the Attester in decreasing order of preference.

There MUST be at least one item in the array.

content-format is an indicator of the format type (e.g., application/eat+cwt with an appropriate eat_profile parameter set).

The sign of ead_label MUST be negative to indicate that the EAT item is critical. If the receiver cannot recognize the critical EAD item, or cannot process the information in the critical EAD item, then the receiver MUST send an EDHOC error message back.

External Authorization Data 2

In EAD_2, the Relying Party signals to the Attester the supported and requested evidence types. In case none of the evidence types is supported, the Relying Party rejects the first message_1 with an error indicating support for another evidence type.

EAD_2 carries the Selected_EvidenceType object. Similarly to EAD_1, Selected_EvidenceType object is encoded in CBOR.

The external authorization data EAD_2 contains an EAD item with

ead_label = TBD2

ead_value = Attestation_request, which is a CBOR byte string:

Attestation_request = bstr .cbor Selected_EvidenceType

Selected_EvidenceType = (

content-format: uint,

nonce:bstr

)

where

content-format is the selected evidence type by the Relying Party and supported by the Verifier.

nonce is generated by the Verifier and forwarded by the Relying Party.

External Authorization Data 3

As a response to the attestation request, the Attester calls its local attestation service to generate and return the serialized EAT [28] as Evidence.

The external authorization data EAD_3 contains an EAD item with

ead_label = TBD3

ead_value is a serialized EAT.

Example Remote Attestation Flow

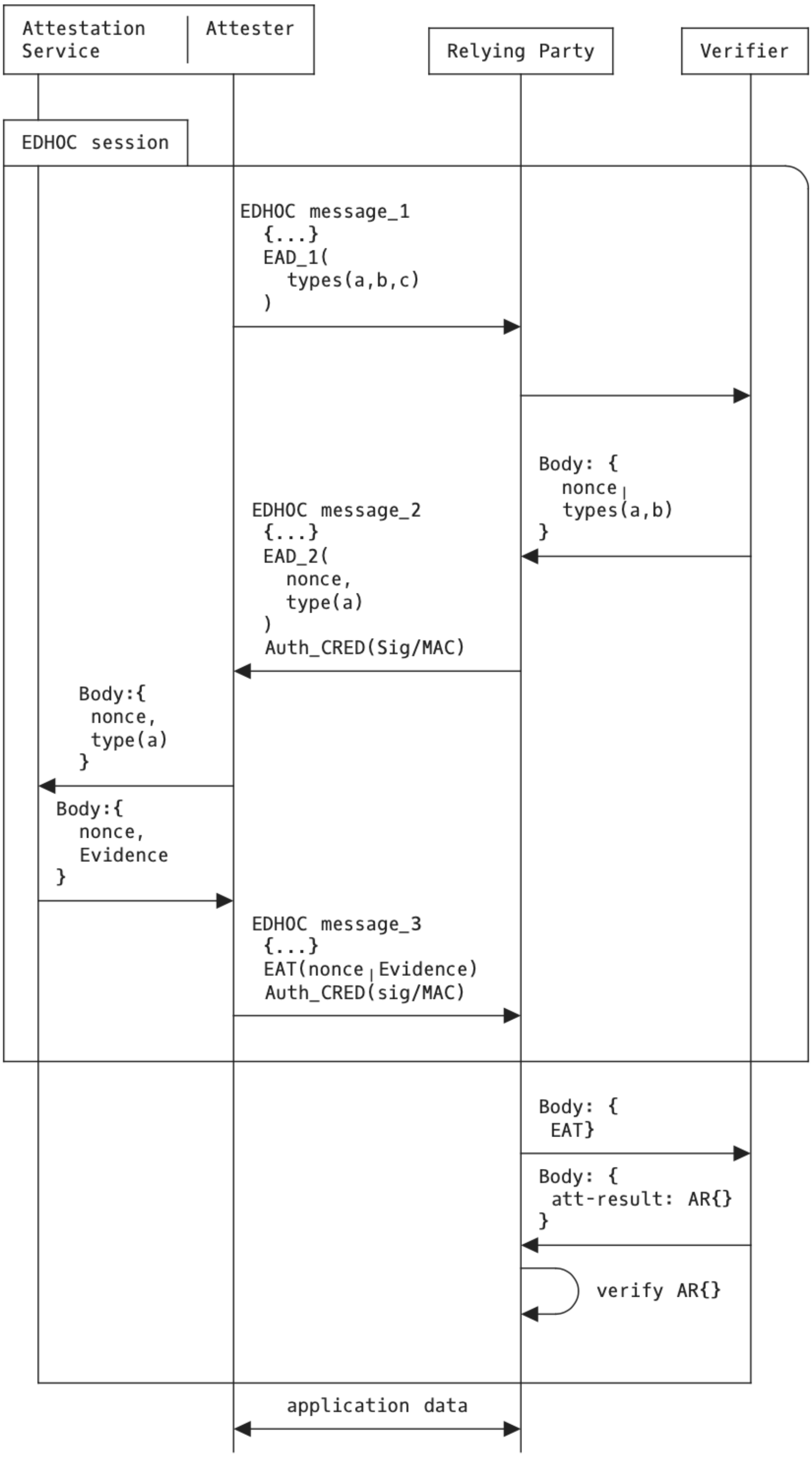

Figure 5. Example remote attestation flow.

Figure 5 illustrates an example remote attestation flow in background-check model.

Example: Firmware Version Attestation

The goal in this example is to verify that the firmware running on the device is the latest version, and is neither tampered or compromised. A device acts as the Attester, currently in an untrusted state. The Attester needs to generate the evidence to attest itself. A gateway that can communicate with the Attester and can control its access to the network acts as the Relying Party. The gateway will finally decide whether the device can join the network or not depending on the attestation result. The attestation result is produced by the Verifier, which is a web server that can be seen as the manufacturer of the device. Therefore it can appraise the evidence that is sent by the Attester. The remote attestation session starts with the Attester sending EAD_1 in EDHOC message 1.

An example of the EAD_1 in EDHOC message_1 could be:

[60,61,258]

If the Verifier and the Relying Party can support at least one evidence type that is proposed by the Attester, the Relying Party will include in the EAD_2 field the same evidence type, alongside a nonce for message freshness.

(258, h'a29f62a4c6cdaae5')

The Evidence in EAD_3 field is an Entity Attestation Token (EAT) [I-D.ietf-rats-eat], with the measurements claim formatted in CoSWID [RFC9393]. The components of the Evidence are presented in Figure 6.

```
{
/eat-nonce/                  10: h'a29f62a4c6cdaae5',
/ueid/                       256: 'aaabbcc',
/measurements/               273: [
  /CoAP Content-Format ID/         [ 258,
  /evidence in CoSWID/               {
                                       0: 'tagID'            /tag-id/
                                       12: 0                 /tag-version/
                                       1: "DotBot firmware"  /software-name/
                                       2: {                  /entity/
                                           31: "Attester"    /entity-name/
                                           33: 1             /role, must be "tag-creator"
                                          },
                                       3: {                  /evidence/
                                           17: [             /file/
                                                 {
                                                  24: "partition0-nrf52840dk.bin",    /fs-na
                                                  7: [                                /hash
                                                      1,                              /alg S
                                                      h'06294f6806b9c685eea795048579cfd02a0
                                                     ]                                /hash
                                                 }
                                               ]
                                          }
                                     }
                                   ]
                             ]
}
```

Figure 6. The attestation evidence, as specified by CoSWID (RFC 9393).

The information above serves as the payload of the COSE object. The complete resulting COSE object is shown in Figure 7.

```
18([
    /*protected header*/
    h'a10127',

    /*unprotected header*/
    {},

    /*payload*/
    h'A30A48A29F62A4C6CDAAE5190100476161616262636319011181821901 02A50045746
    16749440C00016F446F74426F74206669726D7761726502A2181F6841747465737465 72
    18210103A11181A218187819706172746974696F6E302D6E72663532383430646B2E626
    96E07820158200 6294F6806B9C685EEA795048579CFD02A0C025BC8B5ABCA42A19EA0EC
    23E81A',

    /*signature*/
    h'd4100901f4c3e51312c3110c6ddc8dcf7f68d8f5d3791c19133f2f0ac158c1f5ee6ed
    afe9d7c3d6eb3d2d197f82e733d375fdda9fb258b304961dfc38558950d'
])
```

Figure 7. COSE object wrapping the attestation evidence.

which is encoded as shown in Figure 8.

```
D28443A10127A05890A30A48A29F62A4C6CDAAE51901004761616162626363190111818
2190102A500457461674944OC00016F446F74426F74206669726D7761726502A2181F68
417474657374657218210103A11181A218187819706172746974696F6E302D6E7266353
2383430646B2E62696E078201582006294F6806B9C685EEA795048579CFD02A0C025BC8
B5ABCA42A19EA0EC23E81A5840D4100901F4C3E51312C3110C6DDC8DCF7F68D8F5D3791
C19133F2F0AC158C1F5EE6EDAFE9D7C3D6EB3D2D197F82E733D375FDDA9FB258B304961
DFC38558950D
```


Figure 8. Encoding of the attestation evidence as a COSE object.

The Relying Party (co-located with the gateway) then treats the Evidence as an opaque byte string and sends it to the Verifier. Once the Verifier sends back the Attestation Result, the Relying Party can be assured on the version of the firmware that the device is running.

Footprint of a Remote Attestation Solution for DotBots

The following work is under preparation for submission as "Remote Attestation over EDHOC for Constrained Internet-of-Things Use Cases. Yuxuan Song, Geovane Fedrecheski, Mališa Vučinić, Thomas Watteyne. In preparation for IEEE International Conference on Communications (ICC) 2025".

Setup

To evaluate the proposal, we use the DotBot, an open-source micro-robot supporting wireless communication and indoor localization. We use DotBot in its second version, based on 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. An overall view of the evaluation setup is illustrated in Figure 9.

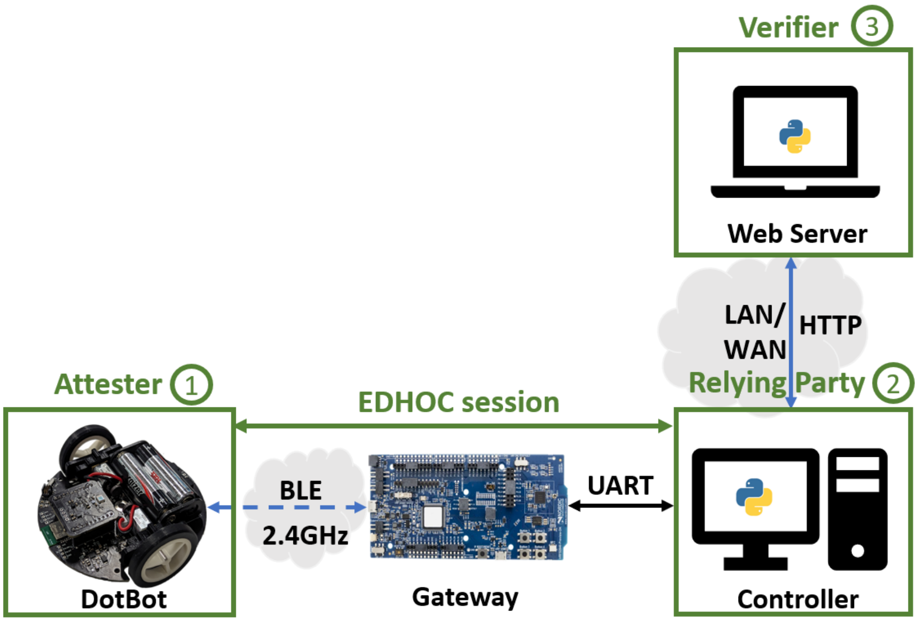

Figure 9. Evaluation setup in the background-check model.

We map the remote attestation entities 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 Figure 9.

Flash and RAM Memory Footprint

We evaluate the 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 2 in Rust, integrated as a static library within the DotBot project. We use 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[2] specifically integrated with the DotBot framework for the attestation component, including the preparation and processing of the attestation items. Our measurements (see Figure 10) show that our baseline, the EDHOC protocol, consumes 15,236 bytes of flash memory. When the attestation process is integrated over EDHOC, the total flash memory consumption increases to 17,163 bytes, which is an increase of 1,927 bytes (12.6%) for the attestation-related operations. In terms of RAM, EDHOC alone requires 5,185 bytes. When attestation is added, the RAM consumption increases to 6,665 bytes, representing an increase of 1,480 bytes (28.5%) for the attestation-related operations. As a reminder, the DotBot version 2 runs on the nRF5340 microcontroller, which provides 512 KB of flash memory. The total flash memory usage for EDHOC with remote attestation represents 3.35% of the available flash, and the 1,927 bytes used for attestation operations consume 0.38%. The RAM usage of EDHOC with remote attestation accounts for over 5.2% of the total 128KB RAM available on the nRF5340, and the 1,480 bytes for attestation represents 1.16% of the total RAM. We can conclude that the memory usage of both flash and RAM can be considered acceptable given the memory capacity of the nRF5340 microcontroller.

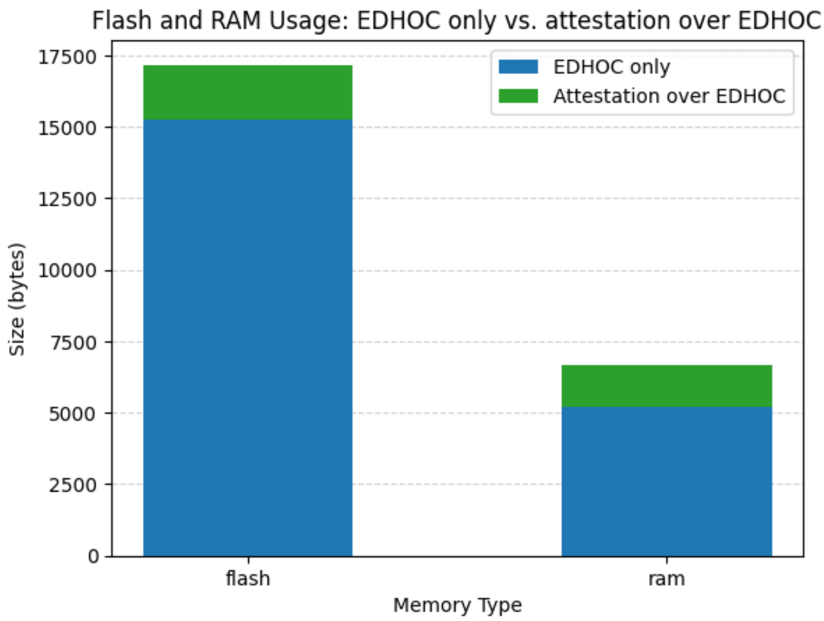

Figure 10. Flash and RAM footprint of the remote attestation implementation.

2 https://github.com/ysong02/DotBot-firmware/tree/main/drv/attestation

Message Size

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, which acts as the Relying Party. These logs captured the precise communication overhead. The results are shown in Figure 11, where the blue section indicates the message size for EDHOC alone, and the green section represents the message size of EAD items carrying remote attestation information.

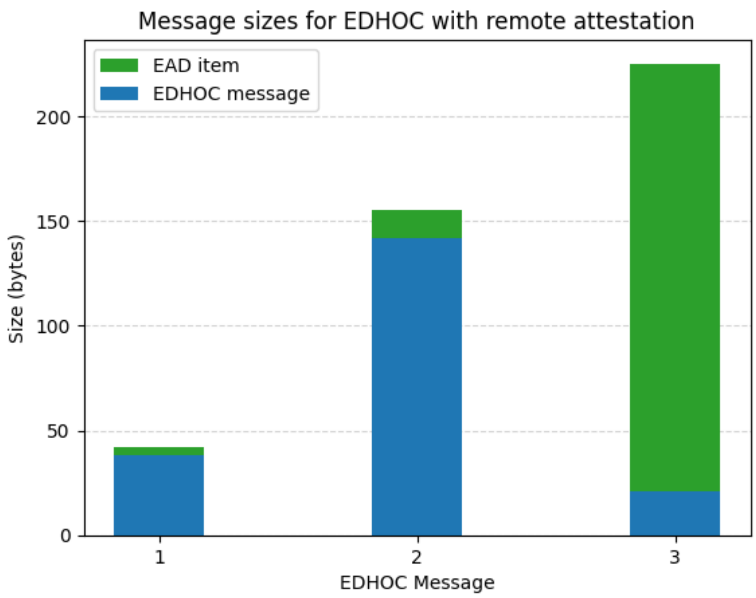

Figure 11. Message sizes for the remote attestation over EDHOC.

The first message that initiates the EDHOC protocol has a size of 38 bytes. The EAD 1 carries an array indicating the provided evidence types with 5 bytes, bringing the total to 43 bytes. In this evaluation, the second message includes an ephemeral key, with an encrypted credential and MAC. The 100-byte credential makes the second EDHOC message larger, with the EDHOC message contributing 142 bytes. The EAD 2 includes a nonce and an integer, which increases the total message size by 13 bytes to 155 bytes. The third EDHOC message is encoded in 21 bytes, which is much shorter due to the use of ID CRED instead of the real credential. EAD 3 carries the attestation evidence over EDHOC message 3, adding 204 bytes, resulting in a total of 225 bytes. The message sizes are acceptable given our BLE physical layer, as the radio supports a maximum transmission unit of 255 bytes. For other physical layers, such as LoRA or IEEE 802.15.4, message 3 would need to be fragmented.

Evidence Format Size Comparison

In our proposal, we define the format of the evidence as a token. More specifically, we follow the structure that is defined as Entity Attestation Token (EAT) from the IETF RATS working group. EAT allows various profiles to indicate the choices made for different implementation options for all aspects of the token. Two options are widely used for generating evidence as an attestation token:

1) The Arm's Platform Security Architecture (PSA) Attestation Token, a specific profile of the EAT.

2) An EAT with the measurements claim formatted using Concise Software Identification Tags (CoSWID).

Table 3. Entity Attestation Token profiled in Arm's Platform Security Architecture (PSA) Attestation Token.

<table>
<tr><td colspan="2">Attribute</td><td>Size (Bytes)</td></tr>
<tr><td colspan="2">psa-nonce</td><td>32</td></tr>
<tr><td colspan="2">psa-instance-id</td><td>33</td></tr>
<tr><td colspan="2">psa-profile</td><td>33</td></tr>
<tr><td colspan="2">psa-implementation-id</td><td>32</td></tr>
<tr><td colspan="2">psa-client-id</td><td>1</td></tr>
<tr><td rowspan="2">psa-software-components</td><td>measurement-value</td><td>32</td></tr>
<tr><td>signer-id</td><td>32</td></tr>
<tr><td colspan="2">Total</td><td>195</td></tr>
</table>

Table 4. Entity Attestation Token with the measurements claim formatted in Concise Software Identification Tags.

<table>
<tr><td colspan="4">Attribute</td><td>size (Bytes)</td></tr>
<tr><td colspan="4">eat-nonce</td><td>8</td></tr>
<tr><td colspan="4">ueid</td><td>7</td></tr>
<tr><td rowspan="8">measurements</td><td colspan="3">tag-id</td><td>10</td></tr>
<tr><td colspan="3">tag-version</td><td>1</td></tr>
<tr><td colspan="3">software-name</td><td>10</td></tr>
<tr><td rowspan="2">entity</td><td colspan="2">entity-name</td><td>10</td></tr>
<tr><td colspan="2">role</td><td>1</td></tr>
<tr><td rowspan="3">evidence</td><td colspan="2">fs-name</td><td>10</td></tr>
<tr><td rowspan="2">hash</td><td>hash-alg-id</td><td>1</td></tr>
<tr><td>hash-value</td><td>32</td></tr>
<tr><td colspan="4">Total</td><td>90</td></tr>
</table>

We compare the two token formats below, detailing all mandatory attributes in each format and their corresponding minimum byte requirements. CoSWID does not specify sizes for attributes of type text. Since CoSWID is the format for measurements claim in EAT, and the "ueid" in EAT defines text with a minimum size of 10 bytes, we apply the same minimum size of 10 bytes for text attributes in CoSWID. We can see from Table 3 and Table 4 that the minimal byte requirements for PSA are 195 bytes, while EAT with the measurements claim formatted in CoSWID requires 90 bytes. We conclude that PSA is more suitable for high-end devices, as it includes more attributes to describe the different IDs for hardware and software components. In contrast, CoSWID in EAT is more preferable for low-end devices that have fewer distinct IDs and require minimal overhead. We choose the option of the EAT with measurements claim formatted in CoSWID for our implementation targeting constrained IoT use cases because of its lightweight structure.

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. First, the Otii Arc power profiler[3] was employed to supply power to the

[3] https://www.qoitech.com/otii-arc-pro/

DotBot and to measure energy consumption during the protocol execution. This tool measures current consumption over time. Additionally, the Otii Arc can read signals from a digital pin, which allowed us to monitor timing throughout the handshake process. We used the Saleae Logic Analyzer[4] for precise timing measurements, with a sample rate of 500 MS/s and the capability to handle digital signals with a maximum frequency of 100 MHz. The logic analyzer captures the timing of the EDHOC handshakes, including the attestation process on the DotBot. We show the setup in Figure 12.

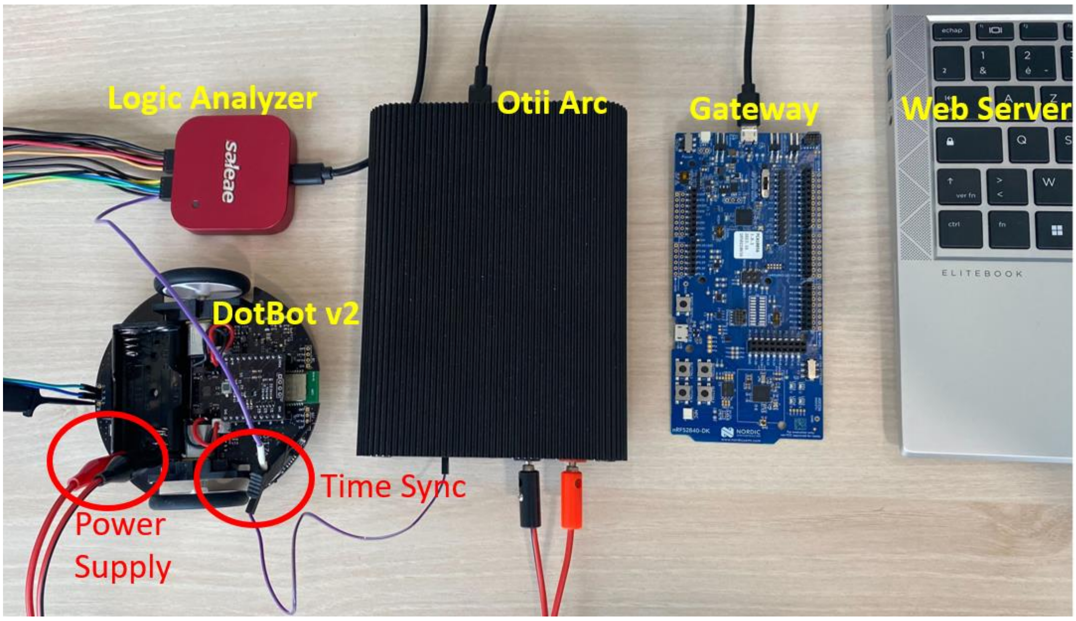

Figure 12. The energy consumption setup.

We collect the energy data using the Otii Arc software alongside another file for the General-Purpose Input/Output (GPIO) signal. To ensure the accurate synchronization between timing and energy measurements, we connect both the Otii Arc and the Saleae Logic Analyzer connect to the same GPIO pin on the DotBot. We present the results in Figure 13.

4 https://www.saleae.com/

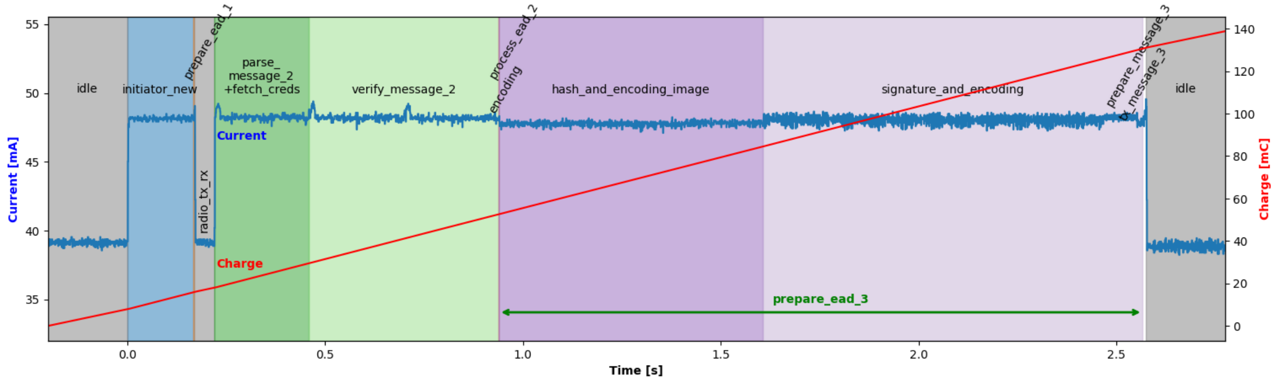

Figure 13. Energy benchmark of the remote attestation process.

The results indicate that completing the EDHOC handshakes, along with both authentication and attestation processes, takes a total of 2.57 seconds and consumes 138.8 mC of energy.

In terms of timing consumption, the most time-consuming operation occurs during the signing and encoding of the signature, which takes 0.96 seconds. This duration is attributed to the use of Ed25519 for signature generation, which includes both the computation of a fixed-size hash input on which Ed25519 operates and the signing time for the hash. It is noteworthy that this signature generation is performed only once for each attestation token, thereby mitigating its impact on overall feasibility.

The second most time-intensive operation is the hashing and encoding of the image, which takes 0.67 seconds. All other processes are completed in under 0.5 seconds, with specific durations as follows: verifying message 2 takes 0.48 seconds, parsing message 2 and fetching credentials takes 0.24 seconds, and initiating a new process for the initiator takes 0.17 seconds. The preparation of EAD 1 and the processing of EAD 2 are notably time-efficient, taking only 4 microseconds and 1 microsecond, respectively. We can conclude, as expected, that the processing is dominated by the cryptographic operations.

Conclusion

This document gathers the results achieved in the context of the Task 2.3 on a secure firmware management solution for OpenSwarm. Particularly, the security of firmware management is achieved through the process known as remote attestation, which we propose to perform during network access authentication of the robots in a swarm.

We first overview the state of the art on remote attestation in swarms by summarizing the works on hardware- and software-based approaches, as well as hybrid ones. We also overview the standardization activity in this area, particularly the work of the IETF RATS working group, which we use as a baseline of our proposal.

We then outline the basics of our approach in a work that we published at the ICRA 2024 Workshop on Breaking Swarm Stereotypes. We present the EDHOC protocol, which we use to piggyback the remote attestation information, and then detail the different attestation models: the background check and the passport model.

We continue by providing a detailed specification of the remote attestation protocol over EDHOC, published as an independent Internet-Draft submission to the IETF LAKE working group. This specification details the data structures used in the protocol as well as how they are carried by EDHOC.

We then implement the proposed attestation scheme for DotBots, a robotic platform developed at Inria in the context of the OpenSwarm project, and benchmark it over the following metrics: message size, memory footprint, and energy consumption. Our results show that performing remote attestation is feasible even for the constrained IoT use cases that we consider in the scope of the OpenSwarm project.

Appendices

TMVS: Threshold-based Majority Voting Scheme for Robust SRAM PUFs

The following work was published as "TMVS: Threshold-based Majority Voting Scheme for Robust SRAM PUFs. Sara Faour, Mališa Vučinić, Filip Maksimovic, David C. Burnett, Paul Muhlethaler, Thomas Watteyne, Kristofer Pister. IEEE Symposium on Computers and Communications (ISCC), Paris, 26-29 June 2024.". It presents a method for extracting a key from SRAM memory of a swarm device that can be used for signing the attestation evidence during the remote attestation procedure.

Summary

Using a Physically Unclonable Function (PUF) for extracting a secret key from the unique submicron structure of an integrated circuit, rather than storing it in non-volatile memory, provides notable advantages such as physical unclonability and tamper resistance. SRAM PUFs offer the advantage of reusing memories that already exist in many resource-constrained embedded devices, by leveraging the unique start-up values of memory cells. However, certain SRAM cells may exhibit bit-flipping due to noise, temperature and voltage fluctuations. In this section, we propose a straightforward yet effective method to mitigate PUF error rates. This method uses a majority voting decoder to rectify errors only in selected SRAM patterns, determined by a predefined threshold. Our scheme is simpler than other error correction codes (ECCs) based solutions. Even for the simplest ECCs, our method can provide better leftover entropy, e.g., more than 3x improvement, at the cost of worse error correction capability and memory overhead. TMVS can reach much smaller bit-error rate values compared to existing solutions based on majority voting, e.g., 360x improvement. TMVS has also minimal pre-processing runtime and cost with respect to existing pre-processing techniques like preselection and hardening.

Introduction

All keyed cryptographic primitives rely on the secrecy of the key used to encrypt or sign a given message. Conventional secure key storage schemes use non-volatile memory (NVM) technologies to store cryptographic secrets. This method introduces additional manufacturing process steps, costs, footprint while decreasing security because of possible readout and cloning attacks. Physical Unclonable Functions (PUFs) have appeared as a promising solution for secret key storage, boosting superior performance, lower cost, and tamper resistance.

Due to submicron process variations inherent in manufacturing, every transistor within an integrated circuit has slightly distinct physical properties. These properties can be measured, and since the process variations are uncontrollable, they result in features that cannot be replicated, even by the manufacturer. Therefore, a PUF serves as a silicon biometric for an electronic device, which can generate a unique, repeatable and unpredictable response to an applied stimulus or challenge. These properties have made PUFs very useful in secure key generation and storage. The Arbiter PUF [29] and Ring Oscillator PUF [30] were proposed based on delay measurements. The Butterfly PUF [31], D Flip-Flop PUF [32], Buskeeper PUF [33] and SRAM PUF [34] were proposed relying on start-up values of memory cells.

SRAM PUF is the only type of PUFs that can be completely implemented in software without requiring a dedicated circuit. This PUF leverages the unpredictable power-up value of SRAM cells to provide the PUF response, with the corresponding address serving as the challenge. This unpredictability arises from inherent random process variations in the relative strengths of the two cross-coupled inverters within each cell, caused by random-dopant fluctuations, line-edge roughness and other factors [35]. Consequently, upon power-up, some SRAM cells initialize to logical '0' while others to logical '1'. This unique pattern of '0's and '1's across each SRAM can then be used for secure key generation.

The reliability of the SRAM cells is one of the challenges of SRAM PUF. The response of SRAM PUF is noisy because some SRAM cells are unstable, which means that their values are not always the same during multiple power-up cycles. The power-up state in these cells is highly sensitive to random noise, temperature, and voltage variations. To generate secret keys, the response for each SRAM cell should be highly reliable under different operating conditions, with a bit-error rate (BER) close to zero.

We classify the existing methods used to solve the problem of robustness on SRAM PUFs into software-based and non software-based approaches. Non software-based approaches either select stable SRAM cells during testing, e.g. multiple measurements and power resets of the SRAM at different voltage/temperature conditions [36], or strengthen the response of SRAM cells by applying reverse burn-in aging [36]. However, software-based approaches are mainly based on error correction codes (ECCs), e.g. BCH code, whether as a base solution [37], or as a complementary technique [38]. ECCs correct the erroneous bits and reduce the bit-error rate to a specific level. Based on these two main classes, combinations of different approaches have also been exploited [39].

Motivation. We aim to develop an efficient solution to extract a reliable secret key from an SRAM PUF. By “efficient”, we refer to hardware resources, cost, and runtime of the implementation, in large networks of resource-constrained devices. We need our solution implementation to be as compact as possible, without the need to perform pre-processing steps as in non software-based approaches. All of these steps can add cost and time overhead that is unacceptable for many applications, including large networks of wireless sensor nodes. Additionally, we want the new approach to be as low-complexity as possible compared to ECC-based solutions. This is a key requirement that makes our work fundamentally different from other SRAM PUF implementations.

Contribution. A naive majority voting decoder produces bit 1 if the sequence of bits has more ‘1’ bits, and bit 0 if it has more ‘0’ bits. Although this decoder has low complexity, it fails to decrease the PUF error rate, since a single noisy cell can change the majority decision when a pattern has close number of zeros and ones. In our approach, we exclude these SRAM patterns causing error decoding from PUF, and choose only the patterns with small or large number of ones according to a threshold. The selected patterns, even when noisy, undergo the majority voting decoder and produce highly reliable bits. This new method neither requires knowing the positions of the most stable

SRAM cells, nor using complex ECC techniques. The contribution of this paper is threefold:

1) We develop a new comprehensive methodology, called TMVS, to increase the SRAM PUF reliability by selecting certain SRAM patterns to be used as PUF.

2) We theoretically analyze the approach by deriving the probability of decoding error and the probability of finding a pattern that respects the determined threshold.

3) We discuss the proposed approach with respect to existing solutions in the literature.

Related Work

Several mechanisms have been proposed in the literature to reduce cell instability and increase the reliability of SRAM PUFs.

Error Correction Codes (ECC). ECCs such as the repetition code, BCH, Reed-Muller, or Golay codes [37] are used to construct fuzzy extractors and reduce the BER to a specific level so that the SRAM array can be directly used as a PUF. A powerful ECC provides strong error-correcting capability but also leads to an excessively large hardware overhead, which is inadequate in some hardware resource-limited applications.

Preselection. Highly stable cells are selected during testing and used as PUF cells, to ensure high reliability. A stable cell has a high probability of having the same power-up value. The overall hardware overhead is negligible since only the address of the selected cell requires to be stored in NVM. However, testing cells requires multiple measurements and power resets at different voltage/temperature conditions [36], [40], during the enrollment of a key, which is an unacceptable overhead for many applications. Even when some methods operate under a single voltage/temperature condition, these methods require either precise control of the power-off duration of the power supply [41], posing implementation challenge for advanced CMOS technology; or regulation of the voltage ramp rate [42], [43], an option dependent on chip design. Moreover, some complex circuits such as adjustable regulators could be needed.

Soft Decision Maximum-Likelihood (SDML). Similarly to preselection methods, multiple reliability measures are required in SDML to calculate individual bits error probabilities. SDML decoder uses these values to select the code word that was most likely transmitted [44]. It achieves the best possible error-correcting performance, but generally at a decoding complexity exponential in the number of key bits per block, and a large quantity of helper data stored in NVM. Even when SDML uses a single measure [10], ECCs will be still required.

Spatial Majority Voting (SMV). The idea of the SMV scheme [45] is about performing majority voting only on the subgroup of bits that constitutes the majority of total bits; e.g. for the group of bits '11100', the majority voting is applied only on the first three bits. However, this method is not sufficient by itself, and other techniques such as preselection or ECC are still needed to reach nearly 100% reliability.

Hardening. Reverse burn-in aging is applied to the SRAM cells to strengthen their response and harden their direction [36]. The cells aged with 0 (storing 0) will have a bias of 1, and the cells aged with 1 will lean towards 0 bias. This method somewhat improves the PUF stability, but is not sufficient on its own, and ECC is still required afterwards. Additionally, it is an expensive procedure in terms of tester runtime and cost. Recently, a new burn-in technique that can reach nearly 100% reliable SRAM PUF cells was explored [46]. However, it requires modifying the design of the SRAM cells, and this increases the design cost and time, and also limits occasional reuse of functional SRAMs as PUFs.

Preliminaries

In this section, we review SRAM PUF and introduce some metrics and statistical functions that will serve as the base for our approach construction. Later, we review the fuzzy extractor, a technique based on ECCs for improving SRAM PUF reliability using software only.

SRAM PUF

When an SRAM is initially powered up, each cell acquires a '0' or a '1' logic value. A well-known circuit design for an SRAM cell is the 6-transistor SRAM cell. In an ideal SRAM cell, if all transistors are identical, the cell is perfectly balanced. In the absence of an asymmetric electrical noise, such a cell has a random 50% probability of acquiring either a '0' or a '1' state at power-up. However, in nanometer-scale technologies, because of uncontrollable small random manufacturing variations, no two transistors in an SRAM cell are truly identical in practice. Therefore, when the SRAM memory has been powered up, the process variation along with the noise and environment variation will classify the cells into unstable and stable cells. Stable cells will have their preferred and consistent state, either a '0' or a '1'. Conversely, the final state of unstable cells is determined by the noise present in the circuit and by temperature and voltage variation.

Due to the noisy nature of SRAM PUF responses, two readings of the SRAM cells at start-ups will produce two different but closely related responses R and R'. The measure of closeness can be defined via the Hamming distance dH (R, R'); the number of bits that differ between two-bit strings. The raw PUF bits have a bit-error rate pe, and many techniques in the literature focused on increasing SRAM PUF reliability and decreasing pe. The target bit-error rate is 10^{-6}, to make PUF responses adequate for key generation.

$$\mathbf{bino}(k,n,p) = \binom{n}{k} p^k (1-p)^{n-k} \qquad (1)$$

The bits in an SRAM PUF are derived from physically distinct circuit components, thus, assuming that the PUF bits are independent and identically distributed (i.i.d) is reasonable [47]. However, some SRAM PUFs present dependency between physically neighboring memory cells [48]. Hence, bias and correlation between SRAM cells should be tested in advance for specific circuits and technology [49]. We denote by p the probability for a cell to have a '1' value, hence, perfectly unbiased SRAM memory cells will have p = 1/2. Assuming PUF bits are i.i.d, the Hamming weight of a bit string, defined as the number of non-zero bits, follows a binomial distribution, with the probability mass function given in (1), where n is the number of bits in the bit string, and k the number of occurring '1'. Furthermore, the Hamming distances between different raw bit strings result in also a binomial distribution [50]. The binomial distribution has a mean μ = np and standard deviation $\sigma = \sqrt{\mathrm{np}(1 - \mathrm{p})}$.

Fuzzy Extractor

PUF responses cannot be used directly as a key in a cryptographic primitive for two reasons: they are noisy and not uniformly distributed. To deal with the first issue, at least one of the presented techniques has to be used to generate reliable keys. Mitigating the second issue requires applying a privacy amplification or randomness extraction primitive, e.g. hash functions, and this is out of the scope of our study.

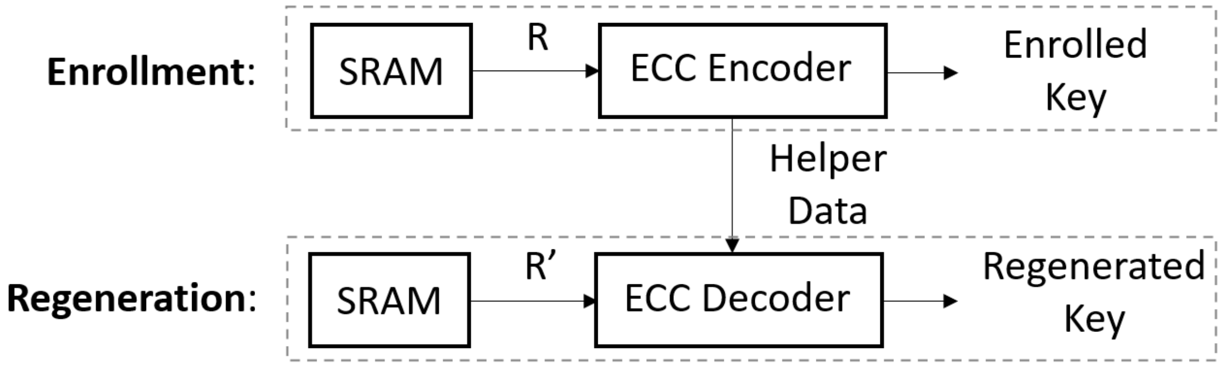

Figure 14. ECC-based key generator of SRAM PUF.

Fuzzy extractor is a conventional approach based on ECCs and is used to enhance the reliability of SRAM PUF and reach a target bit-error rate of less than 10^{-6}. Each SRAM PUF response R is a sequence of n bits. An ECC code C with parameters [n, k, t] is used to correct the noisy response R', where n is also the length of the code, k is the dimension of the input data, i.e. key bits, and t is the number of errors it can correct. The error correcting capability t is derived from the smallest Hamming distance d_{min} between any two different code words in C as $t = \lfloor (d_{min} - 1) / 2 \rfloor$.

There are two phases in a fuzzy extractor, as shown in Figure 14, enrollment and regeneration. During the enrollment, the ECC code performs the encoding by using a large amount of raw data from the SRAM array. The output of encoding is the secret key K and helper data W , where a code word CS is chosen at random from C and $W = CS \oplus R$. This approach is known as "code-offset construction". The helper data is public and can be stored in any NVM. During the regeneration phase, the user will exploit the helper data and the new regenerated (noisy) PUF data R' to recover the secret key.

$$P_{error} = \sum_{i=t+1}^{n} \binom{n}{i} p_e^i (1-p_e)^{n-i} \qquad (2)$$

The probability that a string of n bits has more than t bit errors is given in (2), where p_e is the raw bit-error rate, and once the number of errors exceeds t, the ECC cannot correct them, thus, the SRAM PUF fails to extract the correct key. Notice that the desired value of P_{error} determines the required error correcting capability t, and thus, the size of the code.

Different ECCs have been used to build robust SRAM PUFs. Using repetition code alone requires large PUF responses, using Golay code suffers high error probability, and BCH code has high decoding complexity. To mitigate these issues, Bosch et al. [37] proposed to use a concatenated code to reduce the size of the PUF response and the complexity for hardware implementation. These concatenated codes have as inner code a conventional error correcting code such as BCH, Reed-Muller, or Golay code, and as outer code a repetition code. Besides the hardware overhead, fuzzy extractors have another drawback: the helper data bits can leak information about the extracted secret key [51]. Thus, helper data should be carefully generated.

Threshold-based Majority Voting Scheme

In this section, we show the core idea behind our approach based on the majority voting. We explain how to choose the threshold, which is crucial in our method. We present our approach using a simple use case, then we describe the approach in a more general way. The goal of this approach, called TMVS, is to extract from an SRAM PUF a reliable key with an error probability of 10^{-6}, a value assumed to be adequate for any realistic application [37].

Majority Voting

The naive method to perform majority voting on a sequence of bits is to extract bit 1 if the sequence has more '1' bits, and bit 0 if it has more '0' bits. Only sequences with an odd number of bits are used in such an approach to prevent ties. Unfortunately, this approach increases the PUF error rate as shown in [45] instead of reducing the error rate. This refers to the fact that a single noisy cell can change the majority decision for a full pattern. When the number of '0' and '1' bits in an SRAM pattern is very close, this approach fails immediately. Take for example the pattern "01100", the majority of bits is 0, however, if the first bit flips, the majority will become 1, thus, a decoding error occurs. In fact, when an SRAM memory is unbiased $p = 1/2$, the Hamming weight of an SRAM pattern follows a binomial distribution with a mean equal to half of its length n. Therefore, many SRAM patterns have close number of zeros and ones, leading to a bad performance for the naive method.

When we exclude these SRAM patterns with Hamming weight close to n/2 from the PUF, the performance of the majority voting decoder increases significantly. Consequently, to decrease the error decoding probability, we should set a safety threshold on the Hamming weight of SRAM patterns selected to be used as PUF. SRAM patterns with a similar number of zeros and ones will be excluded from PUF according to a certain interval bounded by a lower and higher threshold. The way we choose these thresholds is detailed in the next section.

Threshold Choice

The number of errors ne occurring in a pattern of n i.i.d PUF bits with probability pe, follows a binomial distribution $bino(n_e, n, p_e)$ defined in (1). The binomial can be approximated by the normal distribution if both $np_e > 5$ and $n(1 - p_e) > 5$. Assuming $p_e = 0.15$, n should be larger than 33 to satisfy these conditions. This approximation allows us to apply the empirical rule, stating that around 68% of the error values are within one standard deviation (σe) of the mean (μe), 95% of the values are within two standard deviations (2σe) and 99.7% are within three standard deviations (3σe).

In the context of our study, this statistical rule aims at predicting the number of possible erroneous bits occurring within any PUF response for n cells with respect to a previous response. More precisely, we can deduce that the number of flipping bits between any two different SRAM PUF responses of the same cells, falls within the interval [μe – 3σe, μe + 3σe] with a probability of 0.997. Thus, the maximum number of possible flipping bits is ⌈μe + 3σe⌉ bits with a probability of 0.997, for a response of length n. This maximum bound will be denoted as $N_{e,max}(3\sigma e)$. Therefore, by taking a higher coefficient of σe, we can find the interval within which around 100% of the error values fall, thus, $N_{e,max}$ will be an accurate estimate for the maximum number of possible bit flips.

In our approach, we select SRAM patterns in a way that even if the maximum number of bit flips $N_{e,max}$ occurs, we are still able to extract exactly the same bits using majority voting. Hence, an SRAM pattern with Hamming weight far from np by a margin of $N_{e,max}$, can be decoded successfully. Consequently, only sequences with Hamming weight smaller than np – $N_{e,max}$ or larger than np + $N_{e,max}$ should be selected, these are respectively the lower and higher thresholds used in our approach.

Requirements. The diversity in chip designs leads to having different SRAM characteristics. Thus, in practical implementation of TMVS, an estimation of the worst-case BER pe and the average mean of bits p of the used SRAMs is required.

Basic Approach

Codebook. We start with a simple use-case of our approach where we choose a single code word S0 equal to a sequence of n zeros. Thus, the codebook C contains a single element. The Hamming distance between any SRAM pattern and S0 is equivalent, in this case, to the Hamming weight of that pattern.

Enrollment. An SRAM response R with n raw bits is selected to be used as PUF if the Hamming distance dH (R, S_0) satisfies one of the following conditions:

$$d_H (R, S_0) \leq \lfloor np \rfloor - N_{e,max} \quad (3)$$

$$d_H (R, S_0) \geq \lceil np \rceil + N_{e,max} \quad (4)$$

We define the lower threshold in (3) as TH_{low}, and the higher threshold in (4) as TH_{high}, and we write (3) and 4 as follows:

$$d_H (R, S_0) \leq TH_{low} \quad (5)$$

$$d_H (R, S_0) \geq TH_{high} \quad (6)$$

Satisfying the first condition (5) implies that the response R has a majority of '0' bits, and a '0' key bit can be extracted based on majority voting. For the second condition (6), a '1' key bit is extracted. These conditions are illustrated in Figure 15 with respect to dH (R, S_0) that follows a binomial distribution, for p = 1/2 . To extract a full key of length N_K , we should be able to find N_K SRAM patterns of length n each, satisfying (5) or (6). If the SRAM PUF memory starts at address x, then the first tested pattern will be the pattern starting at x and ending at x + n – 1. If the pattern was selected, then the next tested pattern starts at x + n, otherwise, if the pattern was discarded, we test the next pattern starting at x+1. Afterwards, the starting address of each eligible (selected) SRAM pattern is stored as helper data in any NVM.

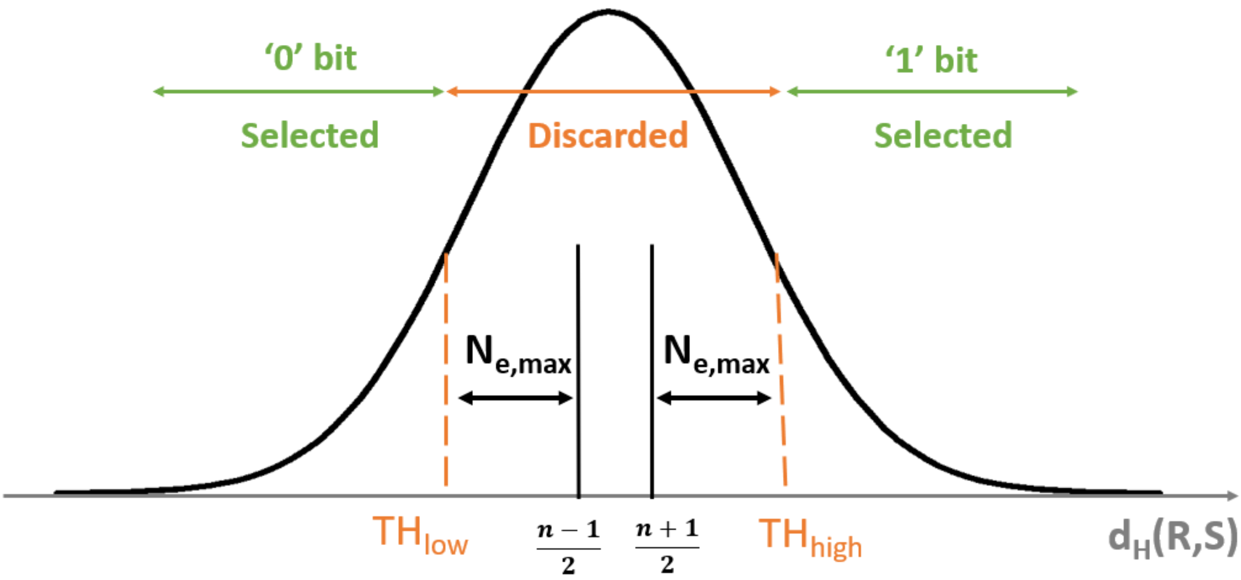

Figure 15. Illustration for the selection region determined by thresholds on the Hamming distance d_H(R,S) distribution.

Regeneration. To reconstruct the full key of length N_K, the helper data is used to identify the addresses of the selected patterns. A new SRAM PUF response R' of a selected pattern is noisy with respect to its value obtained during enrollment. For each selected pattern, we calculate the Hamming distance d_H (R', S_0), then we apply the majority voting decoder according to the following rule:

d_H (R', S_0) ≤ ⌊np⌋ → bit key = '0' (7)

d_H (R', S_0) ≥ ⌈np⌉ → bit key = '1' (8)

The first case (7) implies that the majority of bits in a pattern remain '0', so a '0' key bit is extracted. Otherwise, a key bit of '1' is generated. We can notice that even if the maximum number of bit flips $N_{e,max}$ occurs, the worst case dH (R', S0) will be equal to (n−1)/2 or (n+1)/2, with p = 1/2. These values can be correctly decoded as well.

General Approach

The previously described use case uses a single code word S0. This implies that during enrollment, a large number of SRAM patterns will be discarded before finding a pattern satisfying (5) or (6). Thus, the probability of selection for a pattern is very small, and a large SRAM size is required..

To increase the probability of selection for a pattern, more code words should be considered, so that the patterns excluded for a certain code word, could be selected with respect to another code word. Therefore, when a codebook C has multiple code words, the same approach described previously applies with a small difference during enrollment: for each SRAM pattern, the Hamming distance is calculated with respect to different code words, and the first code word satisfying (5) or (6) is assigned to it, as shown in Figure 16. Consequently, the helper data includes both the address of the selected pattern and the index of its corresponding code word.

In fact, the larger the codebook, the larger the helper data size, because each code word index requires at least log2 (nb. of code words) bits to be stored. Thus, a trade-off exists between the size of required SRAM PUF, quantified using the probability of selection, and the size of generated helper data. Since we need to generate reliable key bits with an error probability of 10^{-6}, depending on the raw PUF error rate pe, it becomes an optimization problem to choose the best codebook and thresholds in terms of hardware resources, number of SRAM bits required, performance, etc.

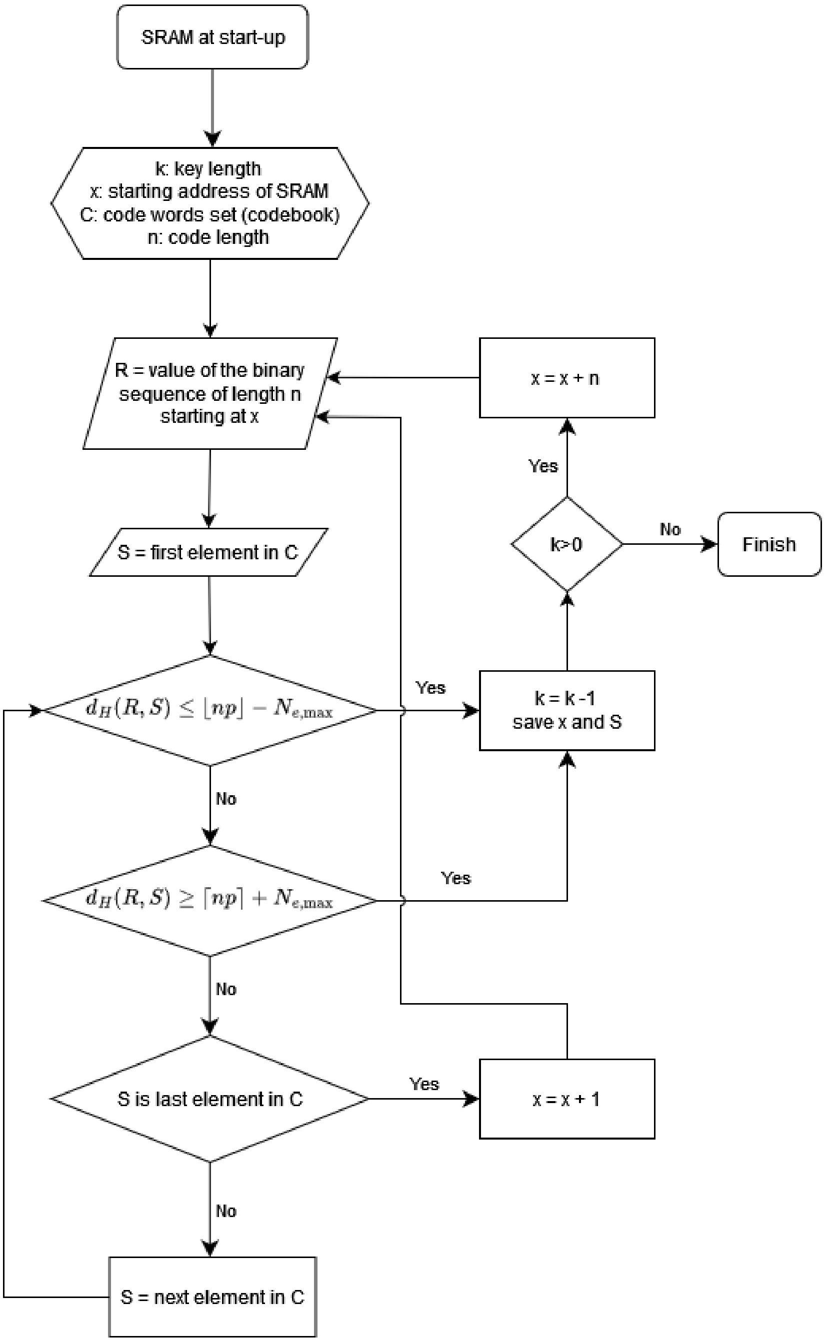

Figure 16. Flowchart of general TMVS.

Discussion

We compare the performance of TMVS to existing techniques presented as part of Related Work. We discuss a few criteria including decoding complexity, memory overhead, entropy loss, bit-error rate (BER), enrollment time, and cost.

Non Software-Based Approaches. Approaches such as preselection and hardening would either require multiple measures under all possible conditions, controlling the power supply, or applying reverse burn-in techniques. These requirements lead to a longer time and higher cost required for enrolling each PUF. Therefore, this could be very challenging when enrolling millions of devices in production lines. Our approach requires an estimation of the worst-case BER and the average mean of bits (bias). These required parameters are not specific for every single cell as required for other approaches. In our case, an estimation over several chips is sufficient, thus no expensive characterization effort is required. The estimation can be done using stochastic concentration theory [52]. The goal of having these parameters is to avoid pessimistic threshold values, leading to a decrease in the performance of TMVS.

ECCs. Error correction codes are software-based approaches that do not require expensive and time-consuming pre-processing steps. However, an ECC implementation usually requires significant memory overhead. For example, to generate a 128-bit key with a targeted key error rate ≤ 10^{-6}, the BCH coding in [37] requires 26.9 raw response bits to generate 1 reliable bit if the raw response bits exhibit errors with pe = 0.15, and it requires 3.68 raw PUF bits to generate 1 stable PUF bit if pe = 0.06 [53]. Additionally, powerful codes such as BCH and Reed-Muller have very complex decoders, thus, high power consumption and number of cycles occur when the decoder is implemented on general-purpose processors. Therefore, hardware implementations are designed to be as inexpensive, fast, and power-efficient as possible using error correction techniques [37]. This solution is expensive in terms of area, a cost that can be unacceptable for resource-constrained devices. TMVS has a very simple decoder compared to the aforementioned ECCs, where mostly XORing operations are used, thus, it cancels the need for expensive hardware implementations. Other codes such as the repetition codes are not complex as well, but they suffer from the large size of PUF response and helper data. To mitigate various ECCs limitations, concatenated codes were later developed and used [37], [51], with the repetition code as the outer code.

Repetition Code. To implement a repetition code of length n, a random bit b is repeated n times into an n-bit string and then XORed with an n-bit PUF response to produce the helper data. During the regeneration phase, the random bit b is recovered by XORing the helper data and the PUF response and then running a majority voting on the XORed result [50]. Similarly to TMVS, the repetition code has very low complexity compared to other ECCs, and only simple operations like XORing are used. Moreover, repetition codes are indeed more efficient than our method in terms of PUF error rate reduction. They can correct up to $(n-1)/2$ errors, which is possible in our method only if the enrolled pattern is identical to the chosen code word S, hence d_H (R, S) = 0, but this is not always the case. Our error correction capability depends on the Hamming distance between the selected SRAM pattern and the chosen code word during enrollment, it ranges between $N_{e,max}$ (maximum number of possible errors) and $(n-1)/2$. Overall, repetition code has low complexity and excellent error correction capability. However, repetition codes suffer a high entropy loss [51]. For an n-bit PUF response, the entropy loss is $n-1$ and there is at most one bit of leftover entropy. TMVS can achieve much smaller entropy loss, we show this through a toy example, and we omit the theoretical proof due to lack of space. Given an unbiased memory ($p = 1/2$), we fix a single code word S0 = 00000 and thresholds $TH_{low} = 1$ and $TH_{high} = 4$, assumed to be public data. According to (5)and (6), only a PUF response that differs from S0 with 0, 1, 4, or 5 bits is selected and used to derive a key bit. The total number of such response values is equal to:

$$\binom{5}{0} + \binom{5}{1} + \binom{5}{4} + \binom{5}{5} = 12$$

Considering these 12 possible patterns, an attacker needs to have log2(12) = 3.58 bits information about the raw SRAM data to guess the extracted key bit. Thus, the leftover entropy in our approach is equal to 3.58 for a single extracted key bit, which is about 3x improvement compared to repetition code. Furthermore, a tighter selection interval [T Hlow, T Hhigh] and longer code word provide higher entropy. In fact, concatenated codes described previously are not without high entropy loss risk since the repetition code constitutes their outer code [51]. The helper data generated in our method reveals less information about raw SRAM data, thus, higher leftover entropy values can be achieved. A helper data leaking information about raw SRAM data is undesirable in terms of key secrecy, however, it is useful for better error correction because it provides more information to the decoder to recover the occurred errors. This trade-off justifies how TMVS improves the leftover entropy at the cost of a worse error correction capability and more memory overhead. Our approach sacrifices a larger size of PUF response since we only select SRAM patterns that satisfy (5) or (6). Hence, when a large number of code words are used, fewer SRAM patterns are discarded, and vice versa. Thus, the size of helper data required to store code words is inversely proportional to the size of required raw SRAM bits. Besides code words, the memory addresses of selected patterns need to be stored also as helper data, and this requires (key length x address length) bits.

Spatial Majority Voting (SMV). Koeberl et al. [45] developed a method to reduce the PUF error rate based on majority voting, where only major bits (zeros or ones with the highest number) are used as PUF bits and undergo the majority voting decoder. The error correction capability t of their scheme is $t = (n-1)/4$, and other techniques such as preselection are required as a complement to extract a highly reliable key with an error rate of 10^{-6}. As explained in the previous paragraph, our t value ranges between $N_{e,max}$ (maximum number of possible errors) and $(n-1)/2$. Thus, for $N_{e,max}$ values larger than $(n-1)/4$, our approach achieves better error rates. For example, with $k = 17$, if the raw PUF has $pe = 0.15$, the error rate after SMV becomes $5.6 \cdot 10^{-3}$ [45]. However, in our method, p_e can be reduced to $1.53 \cdot 10^{-5}$ with $k = 17$ and $N_{e,max}(3.5\sigma e)$, which is a 360x improvement. TMVS can even achieve smaller error-rate values with an order of 10^{-6}. Therefore, no additional techniques would be required to extract a secret key.

Conclusion

We propose a new PUF processing technique called Threshold-based Majority Voting Scheme (TMVS). In this approach, we determine two thresholds that allow for a simple majority voting decoder to regenerate the correct key bits by sacrificing more raw PUF bits. We prove theoretically that our method can reduce the raw BER of SRAM PUF responses to 10^{-6}, even for a high raw BER of 15%, without relying on complementary solutions. We analytically compare TMVS to other well-known techniques used to enhance SRAM PUF reliability. We found that our approach has a smaller error correction capability and larger memory overhead compared to powerful ECCs, however, it leaks less information about the generated key, and it significantly decreases ECC complexity, resulting in a reduced system cost for applications where minimizing logic-cost is the primary design constraint. Additionally, our proposed method requires minimal effort for estimating some SRAM PUF characteristics including the worst-case BER, an acceptable requirement compared to other pre-processing methods with bit-specific expensive characterization efforts. In future versions of this approach, we intend to find the optimal codebook construction, and then evaluate TMVS experimentally on low-power and crystal-free single chip µicro motes (SCµM) [54]. We also aim to prove that TMVS is information-theoretically secure, taking into consideration biased SRAMs scenarios.

Bibliography

[1] W. Arthur, D. Challener and K. Goldman, "A practical guide to TPM 2.0: Using the new trusted platform module in the new age of security," *Springer Nature,* 2015.

[2] J. Yiu, "ARMv8-M architecture technical overview," ARM white paper, 2015.

[3] V. Costan and S. Devadas, "Intel SGX explained," *Cryptology ePrint Archive,* 2016.

[4] Y. Gao, S. F. Al-Sarawi and D. Abbott, "Physical unclonable functions," *Nature Electronics,* 2020.

[5] A. Seshadri, A. Perrig, L. Van Doorn and P. Khosla, "SWATT: Software-based attestation for embedded devices," in *IEEE Symposium on Security and Privacy*, USA, 2004.

[6] A. Seshadri, M. Luk, E. Shi, A. Perrig, L. Van Doorn and P. Khosla, "Pioneer: verifying code integrity and enforcing untampered code execution on legacy systems," in *Proceedings of the twentieth ACM symposium on Operating systems principles*, UK, 2005.

[7] M. Shaneck, K. Mahadevan, V. Kher and Y. Kim, "Remote software-based attestation for wireless sensors," in *Security and Privacy in Ad-hoc and Sensor Networks: Second European Workshop, ESAS*, Hungary, 2005.

[8] S. Ahn and K. Chong, "Requirements change management on feature-oriented requirements tracing," in *International Conference on Computational Science and Its Applications*, 2007.

[9] T. AbuHmed, N. Nyamaa and D. Nyang, "Software-based remote code attestation in wireless sensor network," in *GLOBECOM - IEEE Global Telecommunications Conference*, 2009.

[10] X. Jin, P. Putthapipat, D. Pan, N. Pissinou and S. K. Makki, "Unpredictable software-based attestation solution for node compromise detection in mobile WSN," in *2010 IEEE Globecom Workshops*, 2010.

[11] R. Román, R. Arjona and I. Baturone, "A lightweight remote attestation using PUFs and hash-based signatures for low-end IoT devices," *Future Generation Computer Systems,* 2023.

[12] K. Eldefrawy, G. Tsudik, A. Francillon and D. Perito, "Smart: secure and minimal architecture for (establishing dynamic) root of trust.," in *Ndss*, 2012.

[13] F. Brasser, B. El Mahjoub, A.-R. Sadeghi, C. Wachsmann and P. Koeberl, "TyTAN: Tiny trust anchor for tiny devices," in *Proceedings of the 52nd annual design automation conference*, 2015.

[14] P. Koeberl, S. Schulz, A.-R. Sadeghi and V. Varadharajan, "TrustLite: A security architecture for tiny embedded devices," in *Proceedings of the Ninth European Conference on Computer Systems*, 2014.

[15] I. D. O. Nunes, K. Eldefrawy, N. Rattanavipanon, M. Steiner and G. Tsudik, "VRASED: A Verified Hardware/Software Co-Design for Remote Attestation," in *28th USENIX Security Symposium (USENIX Security 19)*, 2019.

[16] I. D. O. Nunes, S. Jakkamsetti, N. Rattanavipanon and G. Tsudik, "On the TOCTOU problem in remote attestation," *arXiv preprint arXiv:2005.03873,* 2020.

[17] H. Birkholz, D. Thaler, M. Richardson, N. Smith and W. Pan, "Remote ATtestation procedureS (RATS) Architecture," *Internet Engineering Task Force,* vol. RFC 9334, 2023.

[18] H. Tschofenig, Y. Sheffer, P. Howard, I. Mihalcea and Y. Deshpande, "Using Attestation in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)," *Internet Engineering Task Force (IETF),* Vols. draft-fossati-tls-attestation-04, 2023.

[19] N. Asokan, F. Brasser, A. Ibrahim, A.-R. Sadeghi, M. Schunter, G. Tsudik and C. Wachsmann, "Seda: Scalable embedded device attestation," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015.

[20] M. Ambrosin, M. Conti, A. Ibrahim, G. Neven, A.-R. Sadeghi and M. Schunter, "SANA: Secure and scalable aggregate network attestation," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016.

[21] M. Ambrosin, M. Conti, R. Lazzeretti, M. M. Rabbani and S. Ranise, "PADS: Practical attestation for highly dynamic swarm topologies," in *International Workshop on Secure Internet of Things (SIoT)*, 2018.

[22] F. Kohnhauser, N. Buscher and S. Katzenbeisser, "Salad: Secure and lightweight attestation of highly dynamic and disruptive networks," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, 2018.

[23] Z. Wang and C. Sun, "LICAPA: Lightweight collective attestation for physical attacks detection in highly dynamic networks," *Pervasive and Mobile Computing,* 2024.

[24] B. Kuang, A. Fu, S. Yu, G. Yang, M. Su and Y. Zhang, "ESDRA: An efficient and secure distributed remote attestation scheme for IoT swarms," *IEEE Internet of Things Journal,* 2019.

[25] G. Selander, J. Preuss Mattsson and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)," *Internet Engineering Task Force (IETF),* vol. RFC 9528, 2024.

[26] G. Fedrecheski, M. Vučinić and T. Watteyne, "Performance Comparison of EDHOC and DTLS 1.3 in Internet-of-Things Environments," in *IEEE Wireless Communications and Networking Conference*, Dubai, 2024.

[27] 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),* Vols. draft-tschofenig-rats-psa-token-22, 2024.

[28] L. Lundblade, G. Mandyam, J. O'Donoghue and C. Wallace, "The Entity Attestation Token (EAT)," *Internet Engineering Task Force (IETF),* Vols. draft-ietf-rats-eat-25, 2024.

[29] J. W. Lee, D. Lim and B. Gassend, "A Technique to Build a Secret Key in Integrated Circuits for Identification and Authentication Applications," in *IEEE Symposium on VLSI Circuits (VLSI)*, 2004.

[30] B. Gassend, D. Clarke and M. Van Dijk, "Silicon Physical Random Functions," in *ACM Conference on Computer and Communications Security (CCS)*, 2002.

[31] S. S. Kumar, J. Guajardo and R. Maes, "The Butterfly PUF Protecting IP on Every FPGA," in *IEEE Hardware-Oriented Security and Trust (HOST)*, 2008.

[32] R. Maes, P. Tuyls and I. Verbauwhede, "Intrinsic PUFs from Flip-flops on Reconfigurable Devices," in *Benelux workshop on information and system security (WISSec)*, 2008.

[33] P. Simons and E. van der Sluis, "Buskeeper PUFs, a Promising Alternative to D flip-flop PUFs," in *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2012.

[34] J. Guajardo and S. S. Kumar, "FPGA Intrinsic PUFs and Their Use for IP Protection," in *Cryptographic Hardware and Embed. Sys.*, 2007.

[35] Y. Li, C.-H. Hwang and T.-Y. Li, "Process-variation Effect, Metal-gate Work-function Fluctuation, and Random-dopant Fluctuation in Emerging CMOS Technologies," *Transactions on Electron Devices,* vol. 57, 2009.

[36] S. K. Mathew, S. K. Satpathy and M. A. Anders, "16.2 A 0.19 pJ/b PVT-variation-tolerant Hybrid Physically Unclonable Function Circuit for 100% Stable Secure Key Generation in 22nm CMOS," in *IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, 2014.

[37] C. Bösch and J. Guajardo, "Efficient Helper Data Key Extractor on FPGAs," in *Cryptographic Hardware and Embedded Sys. (CHES)*, 2008.

[38] V. Van der Leest, B. Preneel and E. Van der Sluis, "Soft Decision Error Correction for Compact Memory-Based PUFs Using a Single Enrollment," in *International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2012.

[39] A. R. Korenda, F. Afghah and B. Cambou, "A Proof of Concept SRAM-based Physically Unclonable Function (PUF) Key Generation Mechanism for IoT Devices," in *2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, 2019.

[40] M. Yue, "A Two-Stage TMV SRAM PUF Preselection Method with Fewer ECC Resources," in *2024 IEEE 7th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, 2024.

[41] M. Liu, C. Zhou and Q. Tang, "A Data Remanence Based Approach to Generate 100% Stable Keys from an SRAM Physical Unclonable Function," in *IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, 2017.

[42] W. Wang, A. D. Singh and U. Guin, "A Systematic Bit Selection Method for Robust SRAM PUFs," *Journal of Electronic Testing,* vol. 38, no. 3, 2022.

[43] M. Gong, H. Zhang, C. Wang, Q. Tong and Z. Liu, "Design and implementation of robust and low-cost SRAM PUF using PMOS and linear shift register extractor," *Microelectronics Journal,* vol. 103, 2020.

[44] R. Maes, P. Tuyls and I. Verbauwhede, "Low-overhead Implementation of a Soft Decision Helper Data Algorithm for SRAM PUFs," in *Cryptographic Hardware and Embedded Sys. (CHES)*, 2009.

[45] P. Koeberl, J. Li and W. Wu, "A Spatial Majority Voting Technique to Reduce Error Rate of Physically Unclonable Functions," in *International Conference on Trusted Systems (INTRUST)*, 2013.

[46] K. Liu, X. Chen and H. Pu, "A 0.5-V Hybrid SRAM Physically Unclonable Function Using Hot Carrier Injection Burn-in for Stability Reinforcement," *Solid-State Circuits,* vol. 56, no. 7, 2020.

[47] S. Katzenbeisser, U. Kocabas and V. Rozic, "PUFs: Myth, Fact or Busted?," in *Cryptographic Hardware and Embedded Sys. (CHES)*, 2012.

[48] P. Koeberl, J. Li and R. Maes, "Evaluation of a PUF Device Authentication Scheme on a Discrete 0.13 um SRAM," in *International Conference on Trusted Systems (INTRUST)*, 2012.

[49] S. Faour, B. Korecic, M. Vučinić, F. Maksimovic, D. Burnett, P. Muhlethaler and T. Watteyne, "Single-Chip Motes and SRAM PUF: Feasibility Study," in *Workshop on Crystal-Free/-Less Radio and System-based Research for IoT (CrystalFreeIoT)*, 2024.

[50] C. Bohm and M. Hofer, Physical Unclonable Functions in Theory and Practice, Springer Science & Business Media, 2012.

[51] P. Koeberl, J. Li and A. Rajan, "Entropy Loss in PUF-based Key Generation Schemes: The Repetition Code Pitfall," in *IEEE Hardware-Oriented Security and Trust (HOST)*, 2014.

[52] Y. Wang and M. Orshansky, "Efficient Helper Data Reduction in SRAM PUFs via Lossy Compression," in *IEEE Design, Automation \& Test in Europe (DATE)*, 2018.

[53] J. Guajardo, S. S. Kumar and . G.-J. Schrijen, "Physical Unclonable Functions and Public-Key Crypto for FPGA IP Protection," in *IEEE Field Programmable Logic and Applications (FPL)*, 2007.

[54] F. Maksimovic, B. Wheeler, D. C. Burnett and K. Pister, "A Crystal-Free Single-Chip Micro Mote with Integrated 802.15.4 Compatible Transceiver, sub-mW {BLE} Compatible Beacon Transmitter, and Cortex M0," in *2019 Symposium on VLSI Circuits*, 2019.