CHARRA-PM: An Attestation Approach Relying on the Passport Model

Antonio Marques[a] and Bruno Sousa[b]
DEI, CISUC, University of Coimbra, Portugal

Keywords: Remote Attestation, Passport Model, CHARRA, Challenge-Response, IoT.

Abstract: Attestation is a mechanism that is employed to verify the authenticity and integrity of the other(s) part(s), i.e., in hardware and/or software of a device. The remote attestation is the activity of verifying the authenticity and integrity of a target that provides evidence to a verifier over a network that should be accepted or denied as a result of this process. Classic authorization relies in the information provided by a device and gives permission for a specific operation. The attestation adds a new a layer of information, not only we need to know who the device is, but we also need to know if it is in good standing (i.e. performing according to its design) before authorization. This paper proposes the use of the Passport model, using the Challenge/Response development based on the architecture described by the IETF working group RATS - Remote Attestation Procedures Architecture. The elaborated Proof-of-Concept is designed and evaluated using docker containers and TPM software simulation.

1 INTRODUCTION

The wide range of low-cost embedded devices with the ability to command other devices on a network has grown enormously. The network to which these devices belong is called the IoT, which at the year 2022 registered a growth of 18% compared to the previous year (14.4 billion dollars) according to IoT Analytic (IoT Analytics, 2022).

Such devices have low cost, low consumption and easy integration. Their application is possible in several layers of the industry, such as the medical, automotive, smart cities sectors.

IoT devices generally do not have built-in security features or the ability to install or update software. That was a limitation without major problems when installed in isolated networks and not connected to the outside world. However, as technology advances, the IoT interconnectivity with the Internet's enterprise network grows (Kettani and Wainwright, 2019). Another point of observation is the manufacture of these devices on a large scale, where manufacturers want to produce quickly and at a lower cost without safety, security concerns. On the other hand, we have the installation of devices without proper precautions. Due to this, IoT systems are becoming a favourite target for cyber attacks (Martin et al., 2019).

Remote Attestation (RA) allows to find which device is not following the network's policies, or not functioning as intended and was possibly the target of an attack. Thus, RA supports the collection of evidences regarding the behaviour of a device, which can be used as input to build the reputation or to set trust levels regarding the device usage.

CHARRA-PM, built on top of CHARRA (Fraunhofer, 2021), introduces the Relying Party component in the challenge-response remote attestation process. The relying party entity, within the passport model is specified in the RATS architecture (Birkholz et al., 2022b), as an entity that can make decisions about communication, access, and permissions of devices. Available implementations like CHARRA do not include the support of the Relying Party, which is a relevant element, upon the need to apply policies.

The contribution of CHARRA-PM is threefold: 1) implementation of a relying party contributing to the support of an architecture for remote attestation; 2) A complete remote attestation process allowing the enforcement of policies, and 3) Open-source implementation available (Marques, 2022).

The paper is organized as follows, section 1 introduces the paper, section 2 introduces relevant background and related works. Section 3 introduces the proposed approach for attestation, section 4 describes the evaluation methodology and the achieved results. Section 5 concludes the paper.

[a] https://orcid.org/0000-0002-1416-9558
[b] https://orcid.org/0000-0002-5907-5790

Marques, A. and Sousa, B.
CHARRA-PM: An Attestation Approach Relying on the Passport Model.
DOI: 10.5220/0011623600003405
In *Proceedings of the 9th International Conference on Information Systems Security and Privacy (ICISSP 2023)*, pages 349-356
ISBN: 978-989-758-624-8; ISSN: 2184-4356

2 ATTESTATION BACKGROUND AND STATE OF THE ART

2.1 IETF Remote Attestation - RATS

The objective of RATS is to propose a framework (architecture and standardized data formats) for a secure and reliable connection between IoT devices. The structure currently in place consists of a device - Attester - that produces reliable information - Evidence - about itself, transmitting it to another entity responsible for validating the information received - Verifier. The Relying Party is responsible for receiving the verification result to make decisions about communication, access, and permissions.

The verifier, when evaluating the evidence, or the Relying Party, when evaluating the attestation results, verify that the claims values correspond to their evaluation policy. Such verification can include the equality comparison with a reference value, or if it is in a range delimited by reference values, or if it belongs to a set of reference values.

A concrete example would be the need for a device - Attester - to prove that it can convey with another device. For that, it has to provide evidence from its system that must be validated by a trusted entity - Verifier. This last entity will appraise the evidence and return an attestation result to prove the attester's state.

Upon completion of all checks regarding the assessment policy, values are accepted as input to determine attestation results when evaluating evidence or as input to a relying party when evaluating the attestation results. A Relying Party can be any equipment with software capable of performing authorizations on the network, such as a router, switch, or access point responsible for admitting certified devices to the network.

For CHARRA-PM it is relevant to consider the Remote Attestation Procedures Architecture (Birkholz et al., 2022b) and the Reference Interaction Models for Remote Attestation Procedures (Birkholz et al., 2022a) that enable the remote attestation on the RATS architecture. The RATS conceptual flow considers the following aspects (Birkholz et al., 2022b).

The RATS specification (Birkholz et al., 2022a) specifies concepts that are related to the information produced, requested and exchanged between entities. The most relevant include the **Claim** which includes the structure of evidence and other artefacts in the information chain. This can be a statement or a value pair. The **Evidence** which is a set of claims generated by the Attester to be evaluated by a Verifier. Evidence can include configuration data, measurements, telemetry, or inferences. The **Attestation Result** is the output generated by a verifier containing information about an Attester, where the Verifier guarantees the validity of the results.

2.2 Reference Models

The RATS architecture documents two reference models for the Challenge/response operation (Birkholz et al., 2022b; Birkholz et al., 2022a): Passport Model and Background-check model.

The **Passport-Model** is an analogy to the model of issuing passports by a nation to an individual and its use at an immigration counter. So, in this immigration counter analogy, the citizen is the Attester, the passport issuing agency is the Verifier, the passport application and identification information (e.g. birth certificate) is the Proof, the passport is a Result Certificate, and the immigration desk is a Relying Party. In this model an Attester transmits the Evidence to a Verifier, who compares it against its assessment policy or verifies the requested data. The Verifier then returns an Attestation Result to the Attester, which does not consume the Attestation Result but can cache it. The Attester can then present the Attestation Result and possibly additional Claims to a Relying Party, who then compares this information against its own assessment policy.

The **Background-check model** has this name because of the similarity of how employers and organizations perform background checks. When a prospective employee submits information about previous education or experience, the employer will contact the respective institutions or employers to validate such information provided. So, in this analogy, a potential volunteer is an Attester, the organization is the Relying Party, and the reporting organization is a Verifier. In this model an Attester transmits Evidence to a Relying Party, who forwards it to a Verifier. The Verifier compares the Evidence against its assessment policy and returns an Attestation Result to the Relying Party. The Relying Party then compares the Attestation Result with its own evaluation policy.

The challenge/response model only allows the validation that an attester meets the conditions to convey with another entity. This would be enough in some cases to implement a model with these two entities. However, using a third entity whose function would be to validate that the information generated by the Verifier is trustworthy and, from there, grant access permission to resources or other entities provides us with an increase in security.

For a more robust level of security, the Relying

Party may require the Verifier to provide information about itself that the Relying Party can use in assessing the Verifier's reliability before accepting the Attestation Results. In order to implement a trust model that fully utilizes the "trust anchor" concept, it is necessary to have implemented the "Relying Party Owner".

The Passport Model has less network traffic in comparison to the Background-check Model and the possibility to cache the attestation result (for some time) the Attestation Result in case of no communication with a Relying Party.

2.3 CHARRA

The CHAllenge-Response based Remote Attestation with TPM 2.0 - CHARRA is a proof-of-concept implementation of the Remote Challenge/Response Attestation.CHARRA conforms to the RATS architecture description (Fraunhofer, 2021). CHARRA was written in C language, implementing libraries for accessing data on TPM chips and transporting this data using protocols suitable for use in more restricted environments such as te IoT.

The implementation was entirely developed using software, not being linked to a specific hardware platform for code execution. Docker containers are using the official repository of the TPM2 Software community. Each container uses a TPM 2.0 chip simulator, developed by IBM and a C library for interaction with the TPM (TCG, 2022b).

CHARRA implements the secure challenge/response flow to validate evidence collected by an attester but does not return the attestation result back to the attester.

The challenge/response model can be used when we want to ensure that a system has started correctly and only runs authentic software.

Despite the compliance with the IETF RATS architecture, CHARRA has some limitations regarding the implementation of the Challenge/Response Model. First, what is the purpose of doing the attestation only between the Attester and the Verifier? Second, when establishing a new communication with a Relying Party, how can the Attester verify whether or not he is trustworthy? Third, if no response is provided in the challenge/response model there is no remote attestation.

CHARRA-PM, herein proposed, aims to overcome these limitations, by providing a complete and open-source implementation (Marques, 2022) of the remote attestation architecture, with the introduction of the Relying Party entity that contributes to the enforcement of policies regarding devices.

2.4 Related Work

SHeLA (Rabbani et al., 2019) is a model of at least three layers, with a Root Verifier that acts as the "owner" of the attestation network and has associated high computing power. The Root Verifier communicates with the edge verifiers through the network. The Edge Verifiers are devices with significant computing power and storage, in comparison to the low-end devices - Swarm Nodes. Edges verifiers have a permanent connection to root verifiers. The SHeLA model assumes that Edge devices are trusted entities supporting secure hardware that the root verifier can attest. Swarm nodes (the provers) are low-end devices that communicate using particular wireless technologies like Zigbee, Wifi or Bluetooth.

SARA (Dushku et al., 2020) proposes using asynchronous protocols for group attestation, resulting in the non-interruption of all devices simultaneously. SARA uses a publisher/subscriber model based on the Message Queue Telemetry Transport (MQTT) or Advanced Message Queuing Protocol (AMQP). The use of queue managers and the historical attestation are an advantage, in comparison to SHeLA.

CHARRA-PM has the advantage of following a standardized approach for remote attestation, without introducing demanding requirements in terms of computation. SHeLA, in this regard introduces limitations due to the high cost of Root and Edge Verifier.

3 CHARRA-PM PASSPORT MODEL

The motivation for choosing the Passport Model was primarily due to the minimization of traffic processed by the Relying Party, compared to the one observed in the Background-check model. On the other hand, we isolated the Relying Party that centralizes claims and responses as described in the Background-check model, significantly reducing the Relying Party's work, processing and network connections.

Adding a trusted third party in the Challenge/Response model aims to validate the attestation result, sent by the Verifier to the Attester that forwards it to the Relying Party. This process can decide the type of permission/role the Attester can have on the network, like limiting or blocking communication.

We created a new proof of concept: the CHARRA-PM (Marques, 2022), taking advantage of the development already carried out in CHARRA, including a new device/entity, the Relying Party, and the entire process necessary for the signing, transfer and validation of the Resulting Certificate.

As stated previously, the contribution of CHARRA-PM is the addition of the Relying Party entity, leading to a complete implementation of the Remote Attestation architecture. In addition, RP also contributes to the policy enforcement. The addition of a third party can be thought of in two situations: functioning as a gateway or firewall for granting privileges depending on the attestation result or even an end device that would only establish communication upon a valid attestation result.

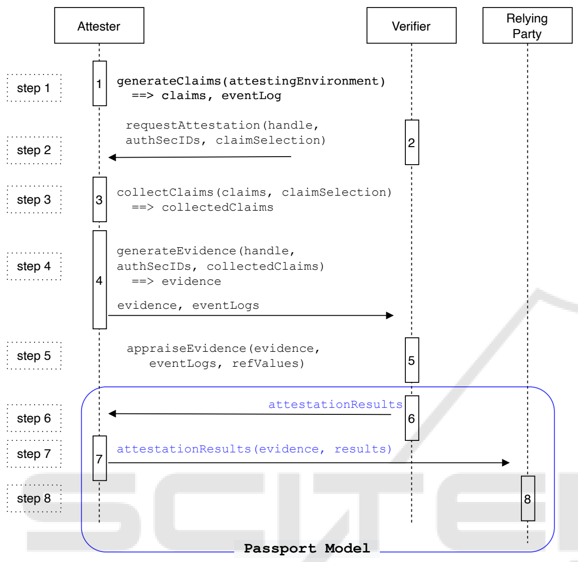
Attester
Verifier
Relying Party
step 1
step 2
step 3
step 4
step 5
step 6
step 7
step 8
1
2
3
4
5
6
7
8
generateClaims(attestingEnvironment) ==> claims, eventLog
requestAttestation(handle, authSecIDs, claimSelection)
collectClaims(claims, claimSelection) ==> collectedClaims
generateEvidence(handle, authSecIDs, collectedClaims) ==> evidence
evidence, eventLogs
appraiseEvidence(evidence, eventLogs, refValues)
attestationResults
attestationResults(evidence, results)
Passport Model

Figure 1: CHARRA-PM data Flow.

As identified in section 2.3, several limitations were identified regarding the challenge/response model implemented in CHARRA. CHARRA-PM adds functionalities to further enrich the compliance with the remote attestation architecture. First, the attestation result, validated by the Verifier, is sent to the Attester, as per documented in (Birkholz et al., 2022a). Second the attestation result is employed for policy enforcement. That is the attestation result proves that the Attester has requirements (valid or not) to present to another peer to gain some level of access. Third, the Relying Party reviews the submitted certificate by any policy it has or through a Relying Party Owner (RPO). This policy can be a true or false check, for example. Most importantly, the Relying Party trusts the Verifier and the attester.

Figure 1 helps us understanding the generation and data exchange flow between the Passport model parts. Assuming that the communication between the parties is carried out safely and securely, as recommended by the RATS documents, in this case, using the DTLS-RPK or DTLS-PSK. The collection of information and communication between the parties involved is summarized in two steps: Challenge/Response and the Passport Model.

3.1 Challenge/Response Steps

When the Attester starts up, it produces claims about its startup and operational state. Event logs track the statements produced by providing a trail of critical security events in a system, as per step 1 in Figure 1.

The Challenge/Response remote attestation procedure is initiated by the Verifier sending a remote attestation request to the Attester. This request includes an identifier (e.g., in the form of a nonce), authentication secret IDs and claims selection, corresponding to step 2 in Figure 1. In the Challenge/Response model, the handle is composed in the form of a virtually unguessable nonce (e.g., a strong random number). The nonce generated by the Verifier ensures that the Evidence is up to date and prevents replay attacks. The *authSecID* is a key sent by the Verifier to the Attester to sign the Attestation Evidence. Each key is uniquely associated with an Attestation Environment in the Attester. As a result, a single Authentication Secret ID identifies a single Attestation Environment.

The Attester collects Claims based on the Claim Selection submitted by the Verifier. If Claim Selection is omitted, all Claims available in the Attester must be used to create the corresponding evidence by default. For example, in a boot health assessment, the Verifier may only ask for a subset, such as Evidence on BIOS/UEFI and firmware, and does not include other information about the current running environment. This corresponds to step 3 in Figure 1.

With the Collected Claims, Handle, Authentication Secret IDs, the Attester produces the evidence and signs it. It digitally signs the Handle and Claims collected with a cryptographic secret identified by the Authentication Secret ID. It is done once per Attestation Environment, which is identified by the specific Authentication Secret ID. The Attester communicates the signed evidence and all accompanying Event Records back to the Verifier, as depicted in step 4 of Figure 1.

The implementation of CHARRA behaves differently, using the TPM private key of the Attester and sending the public key along with the message to the Verifier. CHARRA-PM modifies this behaviour due to the fact of communicating the Attestation result to the Relying Party.

When the Verifier receives the Evidence and Event Logs, it starts the Evidence assessment process, validating the signature, received nonce against sent nonce, and received claims. The assessment procedures are application-specific and can be conducted by comparing values, using reference measures, and producing the Attestation Results. This is done in step 5 of Figure 1.

3.2 Passport Model Steps

After producing the attestation, the Verifier signs it with its private key and sends it, through a secure channel, to the Attester, as illustrated in step 6 of Figure 1. The Attester must immediately send the received certificate to be evaluated by the Relying Party, as per step 7 in Figure 1. The Attester cannot process, verify or change the received certificate. At most, it can keep until it expires.

The Attestation results can contain a Boolean value indicating compliance or non-compliance with a verifier's assessment policy, dispensing with the use of a Relying Party Owner to provide an additional evaluation policy (Birkholz et al., 2022b). The Relying Party confers greater confidence in the model, as it is responsible for verifying whether the attestation result is reliable and valid, deciding the level of access it will give (or not) to the system.

When the relying party receives the certificate, it starts evaluating the information, verifying the signature and the value(s) received. At this time, and depending on the policy, it will give privileges to the Attester according to the results of the attestation, as per step 8 in Figure 1.

In the stages where information is exchanged between parties, it is carried out through a secure communication channel.

3.3 Implementation Details

The implementation of CHARRA-PM was based on CHARRA. For instance, we reused CHARRA code, such as calling connection and cryptographic functions. CHARRA-PM introduced new functions, developed to handle the Passport Model functionality. It should be noted that it was not a development from scratch but an improvement to the existing code with the endorsement of developer Michael Eckel (Eckel and Riemann, 2021). The CHARRA-PM was coded as Proof of Concept (PoC) to the Passport Model as described in RATS documentation (Birkholz et al., 2022b) and the code is available on GitHub as Open-Source (Marques, 2022).

3.3.1 Establishing a CoAP Session with DTLS

This section will present and explain the piece of code functions that handle CoAP sessions using DTLS PSK and RPK.

```
coap_session = charra_coap_new_client_session_psk (ctx,
LISTEN_RP, port_rp, COAP_PROTO_DTLS, tls_psk_identity,
( uint8_t *)dtls_psk_key, strlen(dtls_psk_key))
```

Listing 1: Function to set up PSK CoAP connection.

`charra_coap_new_client_session()` function listed in List 1, is responsible for establishing a channel through the CoAP protocol using DTLS-PSK with Pre-Shared key. The shared key is known to all parties involved (`dtls_psk_key`). The other fields identify the Relying Party, the DTLS protocol (COAP_PROTO_DTLS), the shared key, and its size.

```
charra_coap_setup_dtls_pki_for_rpk(&dtls_pki,
dtls_rpk_private_key_path, dtls_rpk_public_key_path,
dtls_rpk_peer_public_key_path,
      dtls_rpk_verify_peer_public_key) ;
```

Listing 2: Function to set up RPK CoAP connection.

Function `charra_coap_setup_dtls_pki_for_rpk()` listed in List 2, is responsible for establishing a channel through the CoAP protocol using DTLS-RPK with Raw Public Keys. Here the public key of the target device must be known. That is, the Attester knows the public key of the Verifier and the Relying Party; Verifier knows the Attester's public key; the Relying Party knows the public key of the Attester and the Verifier. The advantage of this approach is related with the fact of sharing a pre-shared key between the involved entities.

3.3.2 Receipt of the Certificate by the Relying Party

This is where the Relying Party call functions responsible to appraise the Attestation Result - `attestetionResult`.

```
01: /* Reading CoAP data*/
coap_get_data_large(in, &data_len, &data,
&data_offset, &data_total_len);

02: /* convert from CBOR to data */
charra_unmarshal_attestation_passport(data_len, data, &
      att_result);

03: /* Validating signature */
charra_verify_att_result(verifier_public_key_path,
att_result.attestation_result_data,
att_result.attestation_signature,
att_result.attestation_signature_len);

04: /* mbedtls function related to signature verify */

mbedtls_pk_parse_public_keyfile(&peer_public_key,
      peer_public_key_path);
charra_crypto_hash(MBEDTLS_MD_SHA256, att_result,
      att_result_len, hash);

mbedtls_pk_verify( &peer_public_key, MBEDTLS_MD_SHA256,
      hash, 0, signature, sig_size );
```

Listing 3: Receive and appraise Attestation Results.

Listing 3 shows excerpt of code with several steps:

Step 01. Reading the data received via CoAP.

Step 02. Reverting the CBOR encoding in data that can be processed.

Step 03. Verifies the validity of the received signature using the Verifier's public key. This process will generate a new hash using SHA256 that will be employed with the public key to verify the validity of the signature.

Step 04. Functions called by step 03, coming from the C *mbedtls* library used to read the public key, generate the hash of the `attestationResult` and validate the signature.

The development was based on the CoAP protocol with endpoints for communication with the FETCH method. This means that each endpoint in the code is associated with a *handler* function that handles the information received.

The running environment chosen for this PoC was Docker containers. Three docker containers Attester, Verifier and Relying Party, run the same docker image and map a volume in a local folder on the host with the binaries. The docker container image was based on the official *tmp2software* repository (TCG, 2022a).This docker implements the TPM 2.0 Simulator Development Environment simulating a TPM environment.

3.4 Data Structures

A Proof of Concept to validate CHARRA-PM has developed steps 6, 7 and 8 of figure 1. This section aims to show and explain the PoC running process using screenshots.

The PoC consists of three different devices, each running its isolated environment in a Docker container sharing the same internal network. When the Attester starts, two CoAP endpoints ar createed using the FETCH method: **attest** and **result**. The endpoint **attest** is where the Verifier requests evidence and the endpoint **result** is where the Verifier will send the attestation result back to the Attester. The Relying Part also configures an endpoint where the Attestation Result, from the Attester, will be received.

When the Verifier starts, it creates a CoAP connection with the Attester, in this case using the IP address 192.168.1.2. Then it creates an attestation request indicating what it needs (claims) as a response. Then creates the CoAP message (encoded in CBOR), sends it and wait for the Attester's response.

4 EVALUATION AND RESULTS

4.1 Evaluation Scenario & Methodology

The evaluation scenario includes the Verifier, the Attester and the Relying Party components, running as docker containers. Each component has specific CoAP endpoints that are exported to be used other components/entities.

To assess the performance of the CHARRA-PM PoC, we consider the time required to execute each step in the attestation process. This measurement also allowed us to quantify the impact of the additional steps that were introduced with CHARRA-PM.

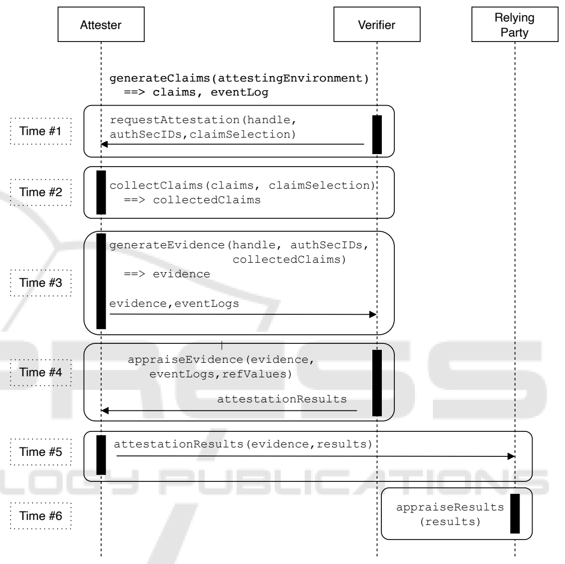
Attester
Verifier
Relying Party
generateClaims(attestingEnvironment) ==> claims, eventLog
Time #1
requestAttestation(handle, authSecIDs,claimSelection)
Time #2
collectClaims(claims, claimSelection) ==> collectedClaims
Time #3
generateEvidence(handle, authSecIDs, collectedClaims) ==> evidence
evidence,eventLogs
Time #4
appraiseEvidence(evidence, eventLogs,refValues)
attestationResults
Time #5
attestationResults(evidence,results)
Time #6
appraiseResults (results)

Figure 2: Times measurement interactions.

Looking the figure 2 helps us understand which process steps were measured. Each step was measured by capturing the time registered by the CPU clock before and immediately after execution. For the time measurement, the *clock()* function of the *time.h* C library language was our choice. This function returns the number of elapsed clock pulses since it was called. To get the number of seconds used by the CPU, one needs to divide by the number of CPU ticks per second (*CLOCKS_PER_SEC*). *CLOCKS_PER_SEC* is the macro responsible for storing the number of pulses per second of the machine's processor on which the program is being executed (Puttnies et al., 2020).

For different executions the times go up or down regardless of the use of encryption using the DTLS protocol (PSK or RPK).

Times were computed for the *remote functions* - libraries calls and the *total execution time of the local* *functions*, which are part of the execution code.

Times were computed for the remote functions, which are called by external libraries, and the total execution time of the local functions that correspond to the functions of the programs.

Line 06 - accumulates the time spent in the function it is running on. This will be the total value of the running function;

To test different policies, two types of attestation were considered:

Good Attestation. The Attester sends a correct attestation to the Verifier. For instance the values in the PCR of TPM were correctly sent by the Attester, corresponding the to ones expected by the Verifier.

Bad Attestation. The Attester sends an incorrect attestation to the Verifier. In such case wrong values of the PCR were sent, thus not corresponding to the ones expected by the Verifier. Such type of attestation result denies the evidence, thus invalidating the passport.

To assess the functionality of the CHARRA-PM, besides the timing metrics, traces with the exchange CoAP messages were collected to verify communications between the entities Attester↔Verifier, and Attester↔Relying Party. the collection of such traces also aimed to check if the encryption mechanisms of DTLS-PSK and DTLS-RPK function properly.

4.2 Results

The measured results take into account the steps shown in figure 2. They are related to the execution of the CoAP protocol using DTLS PSK versus DTLS RPK and the overhead in the CHARRA-PM implementation. The total processing time of

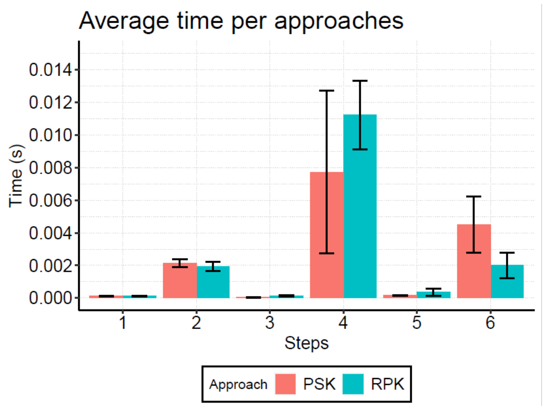

Figure 3: Average execution time per approaches.

CHARRA-PM implementation increases the execution time, as illustrated in Figure 3. This result was expected due to the additional interactions between the Verifier↔Attester, and between the Attester↔Relying Party. Besides the communication processes with the CoAP endpoints, CHARRA-PM introduced additional functionalities like the signature of the attestation.

Step 4 corresponds to the attestation signature, which introduces an higher impact, when compared to the remaining steps. The development of CHARRA does not consider the signature of the attestation, the data packaging time and its transmission to the Attester. This step, besides having the higher average times also have an higher variation, mainly due to the signature, packaging and transmission processes. Indeed, this is one of the steps that contributes to the distinction of performance between CHARRA and CHARRA-PM, as illustrated in Figure 4.

CHARRA-PM is the only approach implementing the appraising results, corresponding to step 6. In such step the Relying Party verifies the attestation result and enforces the policies as configured by the Relying Party Owner.

The employment of distinct approaches for DTLS has not a substantial performance distinction between DTLS-PSK and DTLS-RPK. The most clear distinction is on step 4, where the RPK as a asymmetric approach introduces higher times. Also, since the number of entities included in CHARRA-PM is higher the RPK also tends to present an higher impact, as demonstrated in Figure 4.

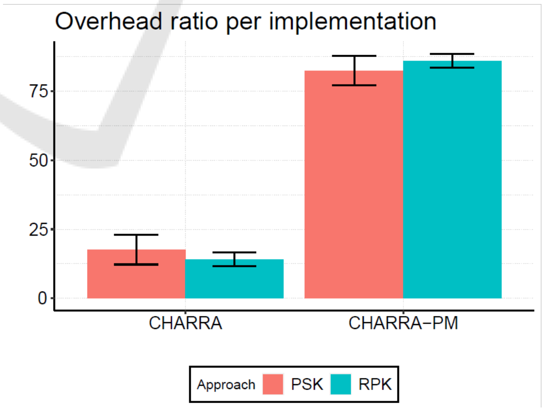

Figure 4: Overhead per Implementations.

Another observable point is about the total overhead between CHARRA and CHARRA-PM as pictured in figure 4. The results put in evidence that CHARRA-PM introduces more overhead, as referred and illustrated. It should be noticed that this overhead is justified by the fact of involving a third entity - Relying Party to allow the application of policies.

The difference between the use of PSK and RPK is almost negligible, since there is not a pattern. In some steps the RPK takes longer (step 4, 5) in figure 3, while PSK takes longer in other step 6. RPK has the advantage of not requiring a full PKI infrastructure (Gonzalez Robles, 2015), and thus might be suitable to scenarios with low complexity, which also is inline with the PSK model.

5 CONCLUSIONS

This work presented the implementation of the CHARRA-PM with support for the Passport Model, This model uses a trusting third party to grant access to devices that have a certificate validated by a verifying party. The choice of the Passport Model over the Background-check Model was due to the interpretation of the number of requests and traffic that the Relying Party would have to support in this model, where the Relying Party is the centre of requests, possibly making it a more robust device.

CHARRA-PM can be used as a source of study and knowledge for implementations of domestic IoT networks, where the Relying Party can be software on the router. Tests show that it is necessary to investigate the delay in create the attestation results. We believe that the implementation should be improved, but as a PoC, the times are acceptable since it is one of the essential processes. As a next step, a compelling extension to this PoC would be the insertion of a policy provider for Attestation Results, the Relying Party Owner, adding more reliability and integrity without adding structural complications to the model.

ACKNOWLEDGEMENTS

This work was supported by the project ARCADIAN-IoT project, which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101020259.

REFERENCES

Birkholz, H., Eckel, M., Pan, W., and Voit, E. (2022a). Reference Interaction Models for Remote Attestation Procedures. Internet-Draft draft-ietf-rats-reference-interaction-models-05, Internet Engineering Task Force. Work in Progress.

Birkholz, H., Thaler, D., Richardson, M., Smith, N., and Pan, W. (2022b). Remote attestation procedures architecture. Internet-Draft draft-ietf-rats-architecture-15, Internet Engineering Task Force. Work in Progress.

Dushku, E., Rabbani, M. M., Conti, M., Mancini, L. V., and Ranise, S. (2020). SARA: Secure Asynchronous Remote Attestation for IoT Systems. *IEEE Transactions on Information Forensics and Security*, 15.

Eckel, M. and Riemann, T. (2021). Userspace software integrity measurement. ARES 21. Association for Computing Machinery.

Fraunhofer (2021). Fraunhofer-SIT/charra: Proof-of-concept implementation of the "Challenge/Response Remote Attestation" interaction model of the IETF RATS Reference Interaction Models for Remote Attestation Procedures using TPM 2.0. https://github.com/Fraunhofer-SIT/charra.

Gonzalez Robles, A. (2015). M2m and mobile communications: an implementation in the solar energy industry (dissertation). Technical report.

IoT Analytics (2022). State of iot 2022: Number of connected iot devices growing 18%.

Kettani, H. and Wainwright, P. (2019). On the top threats to cyber systems. In *2019 IEEE 2nd international conference on information and computer technologies (ICICT)*, pages 175–179. IEEE.

Marques, A. (2022). CHARRA Passport Mode (CHARRA-PM) - Source Code. https://github.com/aamarques/CHARRA-PM.

Martin, E. D., Kargaard, J., and Sutherland, I. (2019). Raspberry Pi Malware: An Analysis of Cyberattacks towards IoT Devices. *10th International Conference on Dependable Systems, Services and Technologies, DESSERT 2019*, pages 161–166.

Puttnies, H., Danielis, P., Sharif, A. R., and Timmermann, D. (2020). Estimators for time synchronization—survey, analysis, and outlook. *IoT*, 1.

Rabbani, M. M., Vliegen, J., Winderickx, J., Conti, M., and Mentens, N. (2019). SHeLA: Scalable Heterogeneous Layered Attestation. *IEEE Internet of Things Journal*.

TCG (2022a). Repository of metadata and scripts used to generate the container images used by the various tpm2-software projects. https://github.com/tpm2-software/tpm2-software-container.

TCG (2022b). The source repository for the trusted platform module (tpm2.0) tools. https://github.com/tpm2-software/tpm2-tools.