Standard-Based Remote Attestation: The Veraison Project

Standard-Based Remote Attestation: The Veraison Project

Lorenzo Ferro[1,*,†], Antonio Lioy[1,*,†]

[1]Politecnico di Torino, Corso Duca degli Abruzzi, 24 | 10129 Torino, ITALY, Dip. Automatica ed Informatica

Abstract

Given the trend towards softwarized and distributed infrastructures, there is an increasing need to demonstrate the integrity of their components as a basis to evaluate their trustworthiness. To this aim, evidence about the current state of a component must be generated and provided to an external party that verifies it. This is a complex task because various architectures and proof types exist, and each scenario requires a custom strategy for verification. The current solutions are developed for specific contexts, resulting in a lack of standardisation and interoperability. Veraison is a standard-based open-source software that aims to address this issue enhancing consistency when developing an attestation framework. Based on the RATS architecture proposed by IETF, it reduces the effort necessary by offering a set of components easily adaptable to different use cases. This paper analyzes Veraison and compares it to existing integrity verification systems, to suggest possible applications and further developments.

Keywords

Remote Attestation, Veraison, Trusted Platform Module, TPM, Attesattion Verification Service, Trusted Computing Group, IETF, Cloud Security, IoT Security, RATS

1. Introduction

With the advent of Cloud Computing, we are now used to perform computations in the cloud, rather than locally. Similarly, data is often no longer stored locally but in the cloud. While this simplifies many operations, it comes with risks related to the administrators of the cloud infrastructure and its multi-tenancy nature. There is therefore a clear need to verify the trustworthiness of the cloud platform where computation is performed.

On another ground, the increasing adoption of embedded systems, cyber-physical systems, and Internet of Things (IoT) devices poses other risks. They are related to manufacturers that rarely prioritize security over other aspects, such as cost, size, and power [1]. Additionally, while linking these devices to the Internet offers numerous advantages, it concurrently exposes them to several attacks [2, 3]. Given the difficulty to protect IoT devices, an alternative approach is to focus on detecting whether the device has been compromised or not. A similar approach can be used also for the cloud, to verify the correct configuration and operation of the remote environment.

ITASEC 2024: The Italian Conference on CyberSecurity

*Corresponding author.

†These authors contributed equally.

lorenzo.ferro@polito.it (L. Ferro); antonio.lioy@polito.it (A. Lioy)

0009-0009-3286-7366 (L. Ferro); 0000-0002-5669-9338 (A. Lioy)

A common solution to address both problems is the generation of an integrity proof for the target system, which is a verifiable and unforgeable statement about its correct configuration. This allows a remote node to verify the state of the system. Remote Attestation (RA) is a method for assessing the integrity of a computational node by an external entity [4, Sec. 7] [5]. Several organizations actively work to standardize the RA procedure. Initially, this was performed mostly by the Trusted Computing Group (TCG) [6] but recently also the IETF and ETSI are paying a lot of attention to this subject.

The complex process of verifying and comparing integrity proofs against reference values is typically carried out by a Verification Service. The Verification Service will need up-to-date data for the process of appraising the evidence presented to it. This requires the establishment of trust relationships between the Verification Service and authoritative sources, from which the information is obtained. The necessity of attestation includes different classes of devices, starting from the IoT ones up to large computing servers. The substantial difference between these devices suggests that each deployment requires a custom service, thus introducing a substantial software barrier and raising the cost and complexity of a RA system.

Veraison [7] is an open-source project to develop software components that can be employed to construct Attestation Verification Services. Veraison (pronounced "ver-ayy-sjon") refers to the stage in winemaking when grapes begin the ripening process and there is need to regularly verify when they are ready for picking. This term was selected for the project given its alignment with the project's purpose (verification of integrity attestation, not of grapes). Veraison's adaptable structure enables easy adjustment to contextual requirements, offering customizable Verification Services.

The scope of this paper is to describe the current status of Veraison, highlighting its strong points and open issues. We will first explain Veraison's architecture, and then discuss its potential applications, outline prospects, and explore directions for development and extension. The adaptability of Veraison opens up numerous applications. Additionally, we performed a comparative analysis between Veraison and other systems performing similar functions.

Paper structure. The rest of this paper is structured as follows. Section 2 is an introduction to the fundamental concepts of the framework. Section 3 introduces the Veraison architecture, explaining its base functionalities. Section 4 makes a comparison with other solutions for attestation. Section 5 gives ideas for applications and possible future works. Section 6 contains the conclusions.

2. Background

2.1. Remote Attestation

The Remote ATTestation ProcedureS (RATS) [8] is a security protocol designed by the IETF to authenticate and verify the integrity of a remote platform. RA concepts were introduced two decades ago and have sparkled several applications [5]. The importance of RA has surged in recent times due to its many applications, such as IoT and 5G/6G networks. In turn this has pushed for development of suitable secure hardware, open-source implementations, and standardization initiatives.

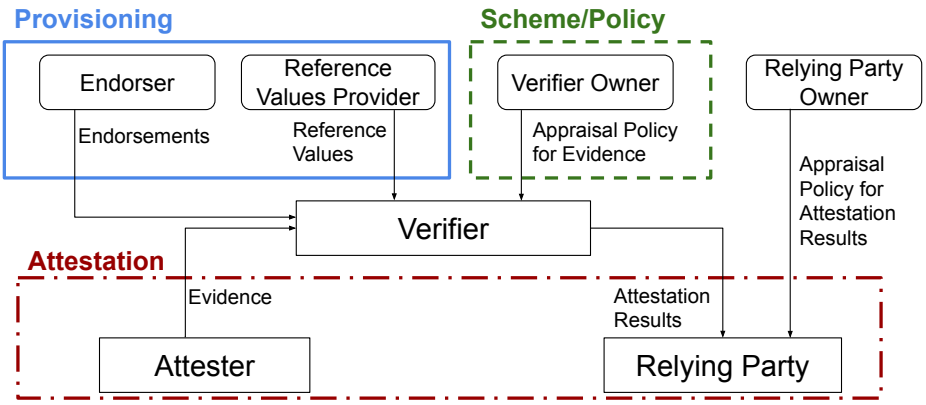

Figure 1: RATS Summary Schema [8].

The Root of Trust is a trusted element, typically hardware-based, that initiates and verifies the integrity of a system's boot process and of other critical operations. RATS uses secure hardware that is inherently trusted, composing the Root of Trust. This hardware allows a remote system, called the Verifier, to check the integrity of another system, called the Prover or Attester with more assurances. This evaluation requires creating a cryptographic signature or measurement of the prover's software, hardware, and configuration. The Verifier compares this measurement with a pre-established reference, often referring to a known-good configuration, to check if the Prover has been compromised or altered (Fig. 1). The main objective of RATS is to instil confidence in remote devices or systems, guaranteeing their integrity and safeguarding them against tampering. Their security objectives encompass identifying unauthorized modifications, guarding against malware, and offering proof of the trustworthiness of the remote system. This enhances security in various scenarios, including remote device management, secure bootstrapping, and establishing secure communication.

It is necessary to introduce some roles in the RA context [9]. The *Attester* is the entity to be verified. The *Relying Party* requires information about an Attester's status to trust it. The *Verifier* appraises the integrity of an Attester. The Reference Values are provided by a *Reference Value Provider*. Also some artefacts, in the attestation context, require an explanation. A *Claim* is a piece of asserted information. A set of Claims is called *Evidence*. *Reference Values* are a set of values on which a set of Claims may be compared as part of applying an Appraisal Policy for Evidence. Evidence is used to reveal operational status, health, configuration, or construction that has security relevance. The Verifier evaluates Evidence to determine its relevance, compliance, and timeliness. Claims need to be collected reliably to prevent the Target Environment from providing deceptive information to the Attesting Environment. An *Endorsement* is a secure statement that an Endorser vouches for the integrity of the device's various capabilities. For example, if the device utilizes hardware-based signing capabilities, an Endorsement could be represented by a manufacturer certificate. This certificate signs a public key, and the corresponding private key is exclusively known within the hardware of the device. The Verifier generates the *Attestation Result* as the attestation output. The Verifier is also capable of validating Evidence or Attestation results using *Appraisal policies*.

For example, it may be essential to obtain reliable reports containing identity and version details of connected hardware and software in a network. In this scenario, a specified integrity level may be required, necessitating claims for verification. The verification of those claims with RA prevents potentially harmful access by vulnerable or compromised devices. A single component, inherently trusted, (root of trust) securely records integrity proof of the system. This is a trustworthy device and enables trustworthiness assessments for other components through a series of operations. Signed measurements by these components serve as evidence either supporting or refuting trustworthiness claims upon evaluation.

RA comes in two main types: Static and Dynamic. Static RA involves assessing the integrity of software and firmware on a device at a specific moment, typically during boot-up or system initialization. It is a one-time process, offering a snapshot of the device's state, but it cannot identify changes occurring after the measurements are taken. On the other hand, Dynamic RA entails the ongoing monitoring of the device's state over time, utilizing a combination of software and hardware-based measurements. This approach provides a more real-time perspective of the device's condition, capable of detecting changes that may occur during its operation.

Within our systems, various methods are employed for evidence gathering. Measured Boot is one method for gathering hashes related to system boot operations. Evidence collection may continue after boot time. For this purpose, Linux offers the Integrity Measurement Architecture (IMA) [10]. IMA embeds hooks within the kernel to permit the generation and aggregation of hash values for each file accessed, before its use for operations. If a hardware root-of-trust is present, the collected values are extended on it to enhance the security of the list.

Also, a Trusted Execution Environment (TEE) [11] could be employed to enhance the system's security. A TEE is a secure area in the primary processor. It ensures that data or code loaded inside it is not accessed or modified by unauthorized entities.

2.2. Trusted Platform Module

The Trusted Platform Module (TPM) [12], as defined by the Trusted Computing Group [6], serves as a secure root of trust, so a secure starting point on which the system security is based. This component integrates various physical security measures to ensure a specified degree of resistance against tampering. A TPM is capable of attesting the platform identity, securely storing the platform history and providing a report of its values. The system state can be recorded by the TPM through the use of a specific set of registers called Platform Configuration Register (PCR) [13, Sez. 6]. The PCRs serve as internal storage within the TPM, representing the host system's software and hardware configuration history. Each PCR has a different scope, the first ones are used for the services launched at boot. These registers have only two operations: "reset" and "extend". Extend operates by hashing the previous value stored in it, combined with the digest of new data. The resulting hash becomes the updated value of the PCR. In formula: $\mathrm{PCR}_{\mathrm{new}} = h(\mathrm{newData}, \mathrm{PCR}_{\mathrm{old}})$. These registers undergo an exclusive reset only during a platform reset, which is initiated after a reboot or hardware signal. It is possible to retrieve the PCR values and use them for verification. During the RA procedure, the PRCs values are retrieved and sent along with the platform data. This is done to give a guarantee of the collected data integrity.

3. Architecture

Veraison [7] is a collection of libraries and tools, that aims to enhance consistency when developing a verification service. The Veraison's architecture (Fig. 2) is structured as follows. It comprises diverse features, including the capability to provide endorsements and trust anchors while also conducting verifications. Veraison Trusted Services (VTS) backend provides core services to the Verification and Provisioning frontends. The key-values (KV) store is the Veraison storage layer. It is used for both endorsements and trust anchors, it is possible to use different methods by configuration.

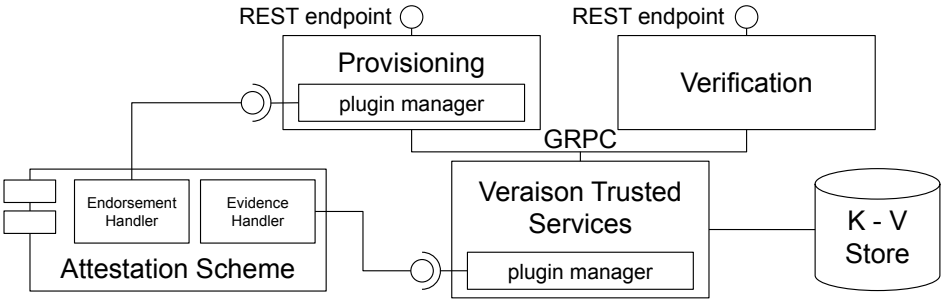

Figure 2: Veraison Summary Schema.

3.1. Configuration

The configuration for every Veraison component is encapsulated within a ParamStore, housing parameter definitions. Each parameter definition outlines the parameter type, employed in validation, along with its configuration path. The overall configuration for a Veraison deployment is managed by a Config, which aggregates ParamStores for the components utilized in that specific deployment. ParamStores are generated and initialized with parameter definitions specific to their corresponding components. Subsequently, these ParamStores are handed over to the Config, which takes ownership and populates them with values retrieved from external sources. The provided values undergo validation against the defined parameters. Following this, the ParamStores are transmitted to the Init() methods of the components (Fig. 3). Veraison services leverage Viper [14] for configuration purposes. Viper is an open-source library for configuration solutions in Go applications. The implemented config loader in this context introduces a validation layer atop Viper, enabling pre-processing of the configuration obtained by Viper before its utilization. It ensures comprehensive validation before proceeding with further processing.

3.2. Provisioning

Providing endorsements and trust anchors to the Veraison service is essential. All information is transmitted in the Coincise Reference Integrity Manifest (CoRIM) format [15]. CoRIM is a signed CBOR-formatted document (COSE) [16]. Concise Binary Object Representation (CBOR) [17] is a data format that offers adaptive encode type and message size, without the need for

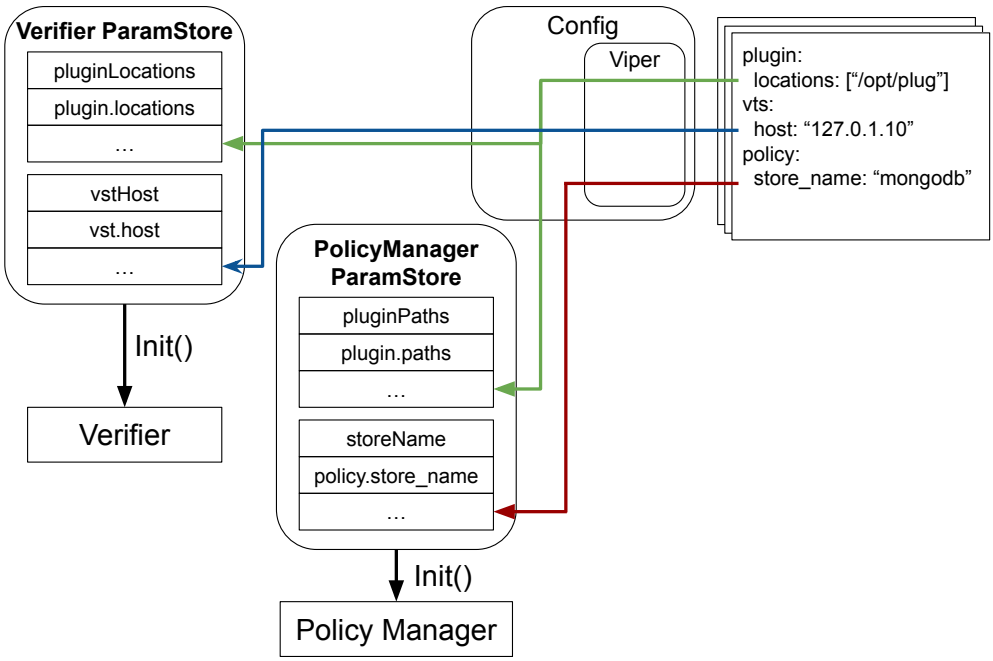

Figure 3: Veraison verifier configuration schema.

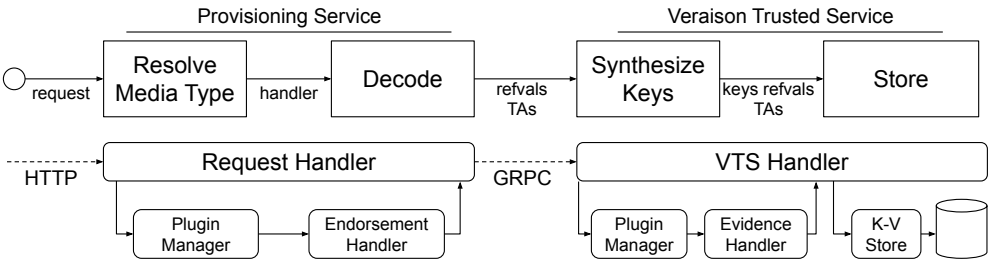

Figure 4: Veraison provisining schema.

negotiation. The underlying data model is an extended version of JSON [18]. The representation of composite devices or systems involves a combination of concise module identifiers (CoMID) and concise software identifiers (CoSWID). These are seamlessly bundled within a CoRIM document. All the received CoRIM data is treated, decoded, and stored (Fig. 4). The Provisioning service is in charge of decoding and the Veraison Trusted Service synthesizes the keys and stores the provisioned data.

3.3. Verification

The Verifier receives an attestation token and produces an attestation result, indicating whether the token is well-structured and has been verified against provisioned endorsements (Fig. 5). The majority of the intricate processes are carried out by the Veraison Trusted Services (VTS)

component (Fig. 6). This component can function within the same process as the Verifier for integrated deployment or operate in a separate process, possibly on a distinct node, to isolate high-trust services. VTS undertakes the verification of the token against the trust anchor, extracts evidence, retrieves associated endorsements, and initializes an attestation result. It populates the Trust Vector by assessing the evidence against the endorsements in a manner specific to the attestation scheme. The resulting attestation is then sent back to the Verifier. In case a policy has been registered as part of the attestation scheme, the Verifier applies it to the attestation, enabling the Trust Vector to be updated in a customized manner.

The VTS has several responsibilities. Firstly, it manages schemes that contain the knowledge of parsing a specific token to extract evidence. It also oversees the validation of tokens against a trust anchor and the evaluation of evidence against endorsements to populate the Trust Vector. The Trust Vector is a method to indicate the trust degree of a component. In addition to this, there is a trust anchor store, responsible for maintaining provisioned trust anchors, such as keys and certificates, which are used to validate received tokens. Furthermore, there is an endorsement store, which is tasked with maintaining endorsements provisioned from the supply chain. Endorsements may serve as "golden values" against which claims from the evidence can be verified. They can also include additional claims associated with, but not part of, the token.

An Entity Attestation Token (EAT) [19] provides a set of attested claims. These claims describe the state and characteristics of an entity. This entity could be a device, such as a smartphone, IoT device, or network equipment. The claims set included in an EAT is used by a relying party, server, or service to determine the type and degree of trust placed in the entity. EATs come in two forms. They can be either a CBOR [17] Web Token (CWT) or a JSON [18] Web Token (JWT). Both of these token types feature attestation-oriented claims. EAT Atteestaion Result (EAR) [20] is used by a Verifier to encode the result of the appraisal over an Attester's evidence. The verifier gives back a signed attestation result as an EAT document.

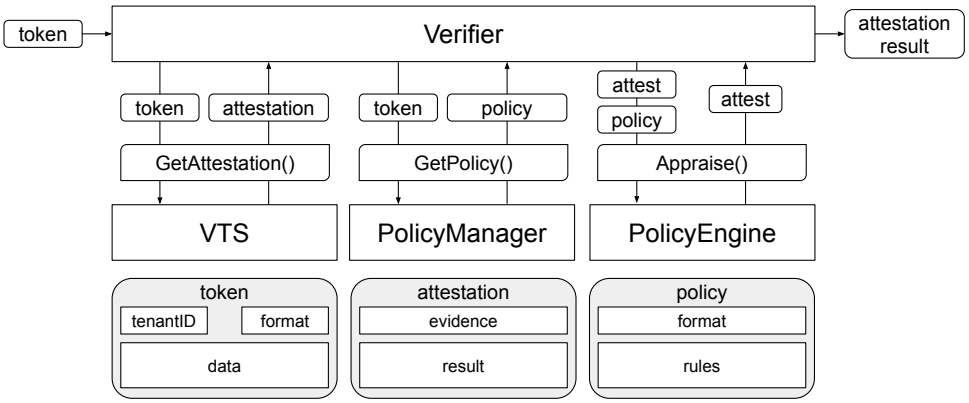

Figure 5: Veraison Verifier Schema.

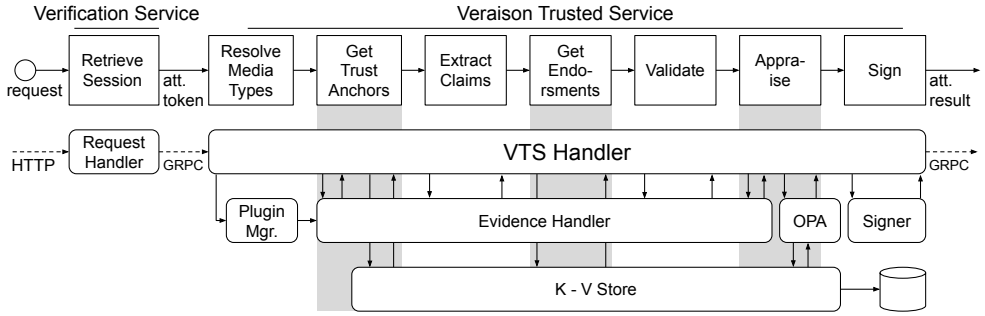

Figure 6: VTS verification Schema.

4. Comparison with related works

Various works have been proposed to implement RA process but all of them are currently restricted to a specific environment (e.g. cloud or IoT) or procedure (e.g. static or dynamic RA)

Keylime [21] is a RA framework developed by the MIT's "Lincoln Laboratory" security research team. It was introduced to the academic community in December 2016. Currently, it exists as a project under the auspices of the "Cloud Native Computing Foundation" (CNCF) [22]. Keylime offers an open-source solution for establishing hardware-rooted cryptographic identities for cloud nodes. Additionally, it facilitates regular attestation to oversee the system integrity of these nodes.

Currently, Keylime [21] stands out as the primary framework for RA. It employs a straightforward architecture to facilitate essential RA functionalities. Keylime consists of an Agent responsible for generating a quote, which is then verified for validity by a Verifier. Additionally, a register is included in the system to handle the mapping between registers and verifiers and manage endorsement provisioning. Keylime provides a command line management tool called the Tenant, designed for overseeing agents. Its functionalities encompass tasks such as adding or removing agents from attestation, validating the Endorsement Key (EK) certificate against a certificate store, and retrieving the status of a given agent.

Also, several main service providers have developed their own attestation schema for containers to guarantee their correct deployment. Google Cloud [23] requires an attestation for each container to trust its deployment and furnish data. The attestation required is based on binary attestation on the container's image to scan for vulnerabilities. Instead of dynamic attestation of the container at runtime, the attestation process is limited to vulnerability assessment on the container image. This process is based on multiple open-source components, the first step is creating the container image and its push on a repository. Kritis [24] blocks the container deployment by enforcing security policies using Grafeas [25] as a metadata server for container images. Scorecard [26] and Voucher [27] perform checks on the container's image looking for vulnerabilities releasing an attestation and storing it into Grafeas. Kritis checks the attestation result stored on Grafeas to allow or deny the container's deployment.

Amazon proposes a similar schema for container attestation. Before the deployment, a vulnerability scan on container images, using Enhanced Scanning [28], is performed and an

attestation result is released. Amazon has a service for further protecting and securely processing highly sensitive data with the usage of a TEE [29]. It is possible to attest enclaves to prove their identity and build trust with an external service. The external service can authenticate the measurements contained within the attestation document in comparison with the values in the access policy [30]. This assessment helps determine whether to authorize the enclave's access to the requested operation.

On the other part, Azure Confidential Containers (ACC) are more related to the RA concept: they are not limited to a vulnerability assessment but introduce more guarantees. It is based on a Trusted Execution Environment (TEE) [31], allowing the relying party to verify that the service is running in a TEE before processing sensitive data. If other technologies have been employed to enhance container security, they could be included in the report. For example, AMD Secure Encrypted Virtualization (SEV) is a security feature that allows the encryption of a virtual environment at the hardware level. Inside the attestation report, the AMD SEV hardware report could also be inserted and used as part of the attestation flow.

The solutions used by the main service providers do not implement standard RA procedures, they just provide assurances regarding the deployed container. Performing analysis before the deployment gives no information regarding the container state at runtime. The Azure solution controls better the container lifetime, as execution into a trusted environment reduces the risk of compromise and allows fast detection of attacks. The only solution that is comparable with Veraison is Keylime because implements similar concepts but specialized in the cloud scenario. Keylime is favoured for its stability, making it a more widely utilized option. Keylime is a solution for cloud environment making it hard to adapt to different contexts. Although Veraison is a relatively newer project, it has garnered increased attention for its versatility. Numerous features have been integrated into Veraison, and several more are currently in development. The framework's flexibility makes it easier to add features and contribute independently.

5. Possible applications

The Veraison project includes many related works and libraries to handle different standards and that can be used standalone. Among various options, standard libraries are employed to manage message formats. The software's flexibility allows for the incorporation and utilization of alternative libraries in place of the standard ones. This flexibility allows for future adjustments to new standards and provides the option to accommodate multiple standards.

Presently, only a limited number of attestation schemas are implemented. The attestation for the ARM Confidential Compute Architecture (CCA) on the cca-ssd platform and the attestation for the ARM Platform Security Architecture (PSA) on the psa-iot platform are both supported. Parsec [32] is an open-source project that offers a universal API for hardware security and cryptographic services. It creates a layer of abstraction to separate workloads from specific platform details, facilitating cloud-native delivery in data centers and at the edge. Implementations for both CCA and TPM hardware-backed attestation are available for Parsec. Enact [33] is an open-source solution that aims to monitor the health of a system. The main target for Enact is IoT devices for industrial or automotive fields. Veraison also encompasses TPM-based attestation implementations for EnactTrust security. The implementation of a new attestation scheme

involves establishing protocols for provisioning endorsements and determining procedures for processing evidence tokens.

As Veraison is a relatively new project, there is room for incorporating additional functionalities and improving existing ones. Currently, the integration with the TEE is at a basic level, not many of them are supported. Given the difference between different TEE architectures, it is important to give a starting point to support attestation in the main ones. The flexibility of Veraison gives a natural disposition to support various and different TEEs.

DICE [34] is a schema for the attestation of a layered architecture, each level depends on the attestation result of the previous one. This provides a comprehensive result that describes the system's state and connects all layers to the root of trust. Presently, Veraison incorporates Open DICE [35] which is a standard designed by Google. OpenDICE aims to simplify the effective implementation of DICE with a focus on quality and confidence. Its objectives include enhancing consistency for both DICE implementers and attestation verifiers. However, it should be noted that the current implementation lacks support for specific DICE architectures.

In recent years, the ascent of cloud computing [36] has underscored the need to uphold the integrity of virtual nodes. Consequently, certifications regarding the integrity of virtual machines or containers have become imperative [37, 38]. Preventing the disclosure of information about other virtual environments within the system is now a mandatory requirement. The verification process must be independent, and the verification system should accommodate such attestation. To facilitate the verification of virtual environments, potential modifications to Veraison are being considered. Various solutions have been proposed to tackle this challenge, with virtual machines typically employing a virtual TPM for simplicity. Meanwhile, containers, that share the kernel with the host, necessitate alternative solutions.

Another popular field regards IoT devices, and their diversity in architecture and functionalities makes challenging the creation of a unified verification framework. Starting with Veraison's components is possible to furnish a verification framework that has great adaptation to the device's peculiarities. To achieve this goal some work needs to be done, to support a larger number of architectures' basic functionalities. Into some fields are required devices swarms that are interconnected devices that operate in large numbers. Performing attestation on them may be time-consuming due to the high number of requests that a Verifier has to perform. The concept of swarm attestation [39] addresses this issue by implementing a distributed attestation. The Verifier has to perform one request and the devices propagate it to another part of the swarm collecting all the required evidence. This schema is not yet supported in Veraison but some present libraries favour its deployment. Finding the right compromise between security and performance is necessary since those devices usually have low capabilities.

The computer field that has in the last years experienced the most significant growth is machine learning [40]. The machine learning process is heavily based on finding the correct dataset to train the artificial intelligence models. Collecting data is essential, with a constant focus on safeguarding user privacy. New modalities to collect data and model training techniques have been developed. An effective way to train a model concerning user privacy is the Federated Learning [41]. This technique consists of collecting user data is used to train a local model. Periodically the local model is used with the global model, anonymously and so preserve user privacy. This technology has been also adopted by Google for the development of the GBoard [42, 43]. Google collect anonymous data regarding the typing on the GBoard keyboard and

use it to train the language model. A problem in this scenario is to check the integrity of the platform that furnishes the data. Multiple compromised devices may lead to incorrect training of the model. The verification for those devices should give minimum information about the integrity without disclosing information regarding the user.

Some scenarios may require multiple Verifiers, each one attesting some proprieties of the system. In this case is necessary to disclose to each Verifier only the data required to attest its part. At the end of the attestation process may be required to coordinate all the Attestation Results and produce a unique one. With Veraison, it is possible to implement different entities that in a coordinated way attest to the same platform.

Other scenarios may include a unique Verifier, that after a comprehensive attestation of the system, produces multiple results. This is useful if there are multiple Relaying Parties and each one depends only on some characteristic of the system. Here may be unnecessary a full disclosure of all the Attestation Results. With Veraison is possible to furnish a subset of Claims to each Relaying Part. This allows to guarantee the necessary level of trustworthiness without excessive information disclosure.

Zero-knowledge proofs [44, Sec. 2] involve verifying assertions without disclosing the actual information. During this procedure, a Prover presents evidence of their assertion to a Verifier, who assesses the validity of the proof without gaining any extra knowledge. Various zero-knowledge proof systems cater to distinct applications. Since the zero-proof concept is still in its early days, robust standardization initiatives are crucial. Adjustments to Veraison could be made to support the zero-knowledge paradigm.

6. Conclusions

This paper presented the Veraison software, beginning with an overview of its architecture and the standards it employs. A comparison with a more established project, Keylime, provided insights into the features and development state of Veraison. Various possible applications were explored to demonstrate its versatility in various scenarios. Since RA concepts are applied to different contexts, having a versatile RA framework is important. Veraison is a very promising foundation for developing custom solutions spanning various fields, with different requirements and capabilities. Since RA is a growing research and application field, we expect the importance of Veraison to increase in the near future.

Acknowledgments

This work was partially supported by the projects SERICS (PE00000014) under the NRRP MUR program funded by the European Union - NextGenerationEU, and by the Smart Networks and Services Joint Undertaking (SNS JU) under the European Union's Horizon Europe research and innovation programme with Grant Agreement No.101139198 (iTrust6G project). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the EU or SNS-JU. Neither the EU nor the granting authorities can be held responsible for them.

References

[1] A. Francillon, Q. Nguyen, K. B. Rasmussen, G. Tsudik, A minimalist approach to Remote Attestation, in: 2014 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden (Germany), March 24-28, 2014, pp. 1–6. doi:10.7873/DATE.2014.257.

[2] A. Cui, S. J. Stolfo, A Quantitative Analysis of the Insecurity of Embedded Network Devices: Results of a Wide-Area Scan, in: 26th Annual Computer Security Applications Conference, Austin (TX, USA), December 6-8, 2010, p. 97–106. doi:10.1145/1920261.1920276.

[3] D. Canavese, L. Mannella, L. Regano, C. Basile, Security at the Edge for Resource-Limited IoT Devices, sensors 24 (2024). doi:10.3390/s24020590.

[4] D. Feng, Trusted Computing Principles and Applications, De Gruyter, 2017.

[5] A. Sprogø Banks, M. Kisiel, P. Korsholm, Remote Attestation: A Literature Review, CoRR abs/2105.02466 (2021). doi:10.48550/arXiv.2105.02466.

[6] Trusted Computing Group, 2024. https://trustedcomputinggroup.org/.

[7] Veraison Project, 2024. https://github.com/veraison.

[8] H. Birkholz, D. Thaler, M. Richardson, N. Smith, W. Pan, Remote ATtestation procedureS (RATS) Architecture, RFC 9334, 2023. doi:10.17487/rfc9334.

[9] R. W. Shirey, Internet Security Glossary, Version 2, RFC 4949, 2007. doi:10.17487/rfc4949.

[10] R. Sailer, X. Zhang, T. Jaeger, L. Van Doorn, Design and implementation of a TCG-based integrity measurement architecture, in: USENIX Security symposium, San Diego (CA, USA), August 9-13, 2004, pp. 223–238.

[11] M. Sabt, M. Achemlal, A. Bouabdallah, Trusted execution environment: what it is, and what it is not, in: IEEE Trustcom/BigDataSE/Ispa, IEEE, Helsinki (Finland), 2015, pp. 57–64. doi:10.1109/Trustcom.2015.357.

[12] Trusted Computing Group, TPM 2.0 Library specification: Architecture, 2019. https://trustedcomputinggroup.org/wp-content/uploads/TCG_TPM2_r1p59_Part1_Architecture_pub.pdf.

[13] S. Kinney, Trusted Platform Module Basics, Elsevier, 2006.

[14] Viper, 2024. https://github.com/spf13/viper.

[15] H. Birkholz, T. Fossati, Y. Deshpande, N. Smith, W. Pan, Concise Reference Integrity Manifest, 2022. https://datatracker.ietf.org/doc/draft-birkholz-rats-corim/.

[16] J. Schaad, CBOR Object Signing and Encryption (COSE), RFC 8152, 2017. doi:10.17487/rfc8152.

[17] C. Bormann, P. E. Hoffman, Concise Binary Object Representation (CBOR), RFC 8949, 2020. doi:10.17487/rfc8949.

[18] T. Bray, The JavaScript Object Notation (JSON) Data Interchange Format, RFC 8259, 2017. doi:10.17487/rfc8259.

[19] L. Lundblade, G. Mandyam, J. O'Donoghue, C. Wallace, The Entity Attestation Token (EAT), 2024. https://datatracker.ietf.org/doc/draft-ietf-rats-eat/.

[20] T. Fossati, E. Voit, S. Trofimov, EAT Attestation Results, 2023. https://datatracker.ietf.org/doc/draft-fv-rats-ear/.

[21] N. Schear, P. T. Cable, T. M. Moyer, B. Richard, R. Rudd, Bootstrapping and Maintaining Trust in the Cloud, in: 32nd Annual Conference on Computer Security Applications, Los Angeles (CA, USA), December 5-8, 2016, pp. 65–77. doi:10.1145/2991079.2991104.

[22] Cloud Native Computing Foundation, Website, 2024. https://www.cncf.io/.
[23] Google Cloud, 2024. https://cloud.google.com/?hl=en.
[24] Kritis, 2024. https://github.com/grafeas/kritis.
[25] Grafeas, 2024. https://grafeas.io/.
[26] OpenSSF Scorecard, 2024. https://securityscorecards.dev/.
[27] Voucher, 2024. https://github.com/Shopify/voucher.
[28] AWS, Enhanced Scanning, 2024. https://docs.aws.amazon.com/pdfs/AmazonECR/latest/userguide/ecr-ug.pdf.
[29] AWS, Nitro enclave, 2024. https://docs.aws.amazon.com/pdfs/enclaves/latest/user/enclaves-user.pdf.
[30] AWS, Nitro enclave attestation, 2024. https://docs.aws.amazon.com/enclaves/latest/user/set-up-attestation.html.
[31] Microsoft, Azure Container Instances (ACI), 2024. https://azure.microsoft.com/en-gb/explore/security.
[32] PARSEC project, 2024. https://parsec.community/.
[33] Enact Project, 2024. https://www.enacttrust.com/.
[34] Trusted Computing Group, DICE, 2024. https://trustedcomputinggroup.org/wp-content/uploads/DICE-Attestation-Architecture-r23-final.pdf.
[35] Google, Open DICE, 2024. https://github.com/google/open-dice.
[36] P. Mell, T. Grance, The NIST Definition of Cloud Computing, NIST SP800-145, 2011. doi:10.6028/NIST.SP.800-145.
[37] M. Souppaya, J. Morello, K. Scarfone, Application container security guide, NIST SP800-190, 2017. doi:10.6028/NIST.SP.800-190.
[38] M. Eder, Hypervisor- vs. Container-based Virtualization, in: Future Internet (FI) and Innovative Internet Technologies and Mobile Communications (IITM), Munich (Germany), July 25-26, 2016, pp. 1–7. doi:10.2313/NET-2016-07-1_01.
[39] S. Wedaj, K. Paul, V. J. Ribeiro, DADS: Decentralized Attestation for Device Swarms, ACM Transaction on Privacy and Security 22 (2019) 1–29. doi:10.1145/3325822.
[40] J. M. Zhang, M. Harman, L. Ma, Y. Liu, Recent advances on federated learning: A systematic survey, arXiv (2019). doi:10.48550/arXiv.1906.10742.
[41] B. Liu, N. Lv, Y. Guo, Y. Li, Recent advances on federated learning: A systematic survey, arXiv (2023). doi:10.48550/arXiv.2301.01299.
[42] Google, GBoard, 2024. https://play.google.com/store/apps/details?id=com.google.android.inputmethod.latin&sjid=14747850254852277844-EU&pli=1.
[43] Google, Federated Learning: Collaborative Machine Learning without Centralized Training Data, 2024. https://blog.research.google/2017/04/federated-learning-collaborative.html.
[44] V. Mulder, A. Mermoud, V. Lenders, B. Tellenbach, Trends in Data Protection and Encryption Technologies, Springer Cham, 2023.