Delegating Verification for Remote Attestation using TEE

Takashi Yagawa
University of Tsukuba
Tsukuba, Ibaraki, Japan
yagawa@osss.cs.tsukuba.ac.jp

Tadanori Teruya
AIST
Koto-ku, Tokyo, Japan
tadanori.teruya@aist.go.jp

Kuniyasu Suzaki
Institute of Information Security
Yokohama, Kanagawa, Japan
suzaki@iisec.ac.jp

Hirotake Abe
University of Tsukuba
Tsukuba, Ibaraki, Japan
habe@cs.tsukuba.ac.jp

***Abstract*—Remote Attestation (RA) can check the status of target IoT devices and cloud platforms from remote locations. In RA, a user is presented with evidence by the target to evaluate its status, verified using a dedicated service or tool. However, the increase in IoT devices and the expansion of microservices make the responsiveness of RA challenging. As a solution, we propose the verification delegation to easily increase the number of verification services. The verification delegation enables guaranteed verification on any platform through the Trusted Execution Environment (TEE). This paper implements delegated verification service using Intel Software Guard Extensions (SGX) as TEE. The evaluations of the implementation show that our proposal is scalable and practical with excellent runtime performance.**

1. Introduction

Modern service platforms, including cloud computing and IoT, are becoming increasingly important. However, users of both platforms need help because the devices hosting them are physically inaccessible or difficult to access, resulting in difficulties assessing whether they are properly operated.

Remote attestation (RA) [11] is a means of remotely verifying the authenticity and integrity of target devices and programs. An IoT device or a cloud platform to be verified is called an attester, which presents evidence for legitimacy. The evidence includes information that only trusted security hardware components can derive as a basis for authenticity. That component is the Trusted Execution Environment (TEE), for example, Trustzone-M in IoT devices or Intel SGX in cloud platforms. Users can verify the evidence to confirm the authenticity and integrity of the attester.

The content of evidence depends on the specifications of the device and RA. Dedicated verification services and tools are often provided since the verification content must also be tailored to these specifications. However, the increase in the number of IoT devices and the growing adoption of microservices pose challenges to the responsiveness of verification services. In edge computing, many IoTs execute RA to TEEs in the cloud [9], [14]; in Function as a Service, each small program requires RA unless a dedicated design [7], [27]. These increase the number of verification requests, which, if concentrated on a few verification servers, can result in delayed responses or no longer accepted requests [20]. Furthermore, since these are real-time critical, the geographic remoteness of the verification servers may not satisfy the requirement.

Because verification services need to be operated by a limited number of trusted institutions, there are limits to what can be solved by scaling their servers. Users could perform verification themselves, but they would need to obtain a separate endorsement of authenticity and then perform verification that meets the RA's specifications. That is very costly for most users who want to use RAs easily.

In the context of IoT, Swarm Attestation [8], [26] could alleviate this problem. Swarm Attestation is a technology that can collectively verify devices in the same network, contributing to the scalability of RA for IoT devices. In addition, it is often profitable for IoT device deployers to build dedicated verification services for many IoT devices. However, in the case of TEE on a cloud platform, these cannot apply because requests from many users are concentrated on a single online verification service.

Therefore, primarily targeting TEE on cloud platforms, our research aims to implement a verification service that maintains responsiveness even when verification requests increase. To achieve it, we propose a verification delegation process that allows for the secure deployment of verification services on servers managed by a third party. The verification delegation process is independent of the verification program and can be applied to any TEE as well as IoT device verification service. Figure 1 shows an overview of the verification delegation process. A trusted party issues certificates to the vetted verification program in TEE in the process on a third-party server. The corresponding program and the signature key are managed in TEE on the third-party server. TEE guarantees the authenticity and integrity of the program and data, even on an untrusted server. Therefore, the verification delegation process can quickly increase the number of third-party servers verifying evidence of RA without increasing the number of trusted entities. When a user uses it, he or she can determine if the result is trustworthy by verifying the signature given to the verification result. Users can also select from among them those that are available or close in the distance to improve the response time of RA.

As a proof of concept, we have implemented an RA with Intel Software Guard Extensions (SGX) [15] that implements the verification delegation process and uses delegated verification services. Its performance evaluation shows that the overhead due to verification delegation is small and practical. In addition, security analysis for this implementation shows that verification delegation is straightforward and robust.

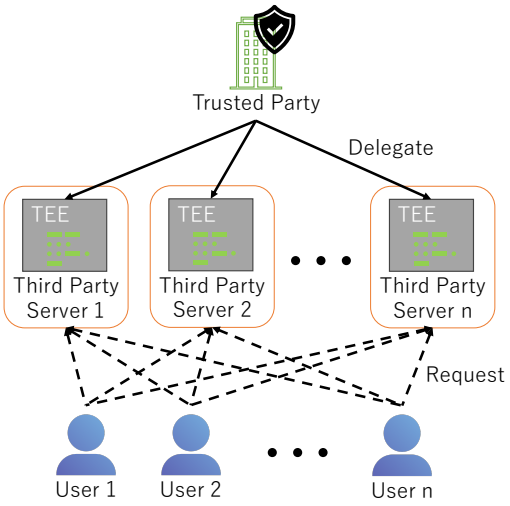

Figure 1. Overview of our proposal: the solid arrow indicates the execution of the delegation process by a trusted party. The dashed arrow indicates that the user can arbitrarily select one server to verify the evidence of RA.

In summary, the contributions of our study are as follows:

- We present a secure delegation process for easily building a verification service to improve the responsiveness of RA. (Section 4)
- The verification server supporting the delegation process demonstrates that the delegation process is practical. In our implementation, we modified an existing verification server and used Intel SGX as the TEE. (Section 5)
- Based on that implementation, we performed a security analysis and performance evaluation. Furthermore, we discuss the challenges to making the delegation process more practical. (Section 6 and 7)

2. Background

2.1. Remote Attestation

Remote Attestation [11] allows the user to verify the authenticity and integrity of the target device and application before remote access. That is completed by verifying the evidence of the target device. The Root of Trust is divided into three types: hardware-based, software-based, or hybrid-based [10]. In hardware-based, hardware modules such as TPMs are typical. Although the protection is strong, RA specifications are hardware-dependent, and updating them is difficult. In software-based environments, the root of trust is established by assuming the conditions under which an attack is possible and performing scans to make it impossible. While it has the advantage of being hardware-independent, it must be carefully configured to prevent remote attacks. In hybrid-based, the Root of Trust consists of minimum hardware and software rooted in them. Its advantage is that the software part can be updated; for example, online updates can fix TEE vulnerabilities. RA has been studied as a security countermeasure for IoT devices without rich protection functions. The Root of Trust can be used with TPMs, TEEs, or defense mechanisms [20]. RA is also used in some TEEs. TEE is an extension of the CPU that protects the confidentiality of programs and data in memory. The specifications vary from CPU vendor to CPU vendor. The Root of Trust for TEE is a hardware key burned into the CPU.

2.2. Intel Software Guard Extensions (SGX)

Intel SGX is a TEE for Intel CPU that guarantees the confidentiality and integrity of programs and data in memory through instruction set extensions and dedicated hardware. The protected area containing programs and data is called the enclave and is stored in the Enclave Page Cache (EPC), a memory area CPU allocates at boot time. Direct access to EPC is denied by the CPU, even from OS or hypervisor. The contents of EPC are decrypted only during CPU computation, and no system calls are allowed in an enclave. For developers, Intel distributes an SGX SDK for the operation of the enclave. An enclave has two identities: MRSIGNER and MRENCLAVE. The former indicates the creator of the enclave and is the SHA-256 hash value of the public key corresponding to the signature given to the enclave. The latter is a SHA-256 hash value for enclave attributes and content. This value will also change if the environment changes, such as the execution platform changes. These values are used by RA and SGX functions for authenticity and integrity checks.

Currently, Intel has adopted an RA method called ECDSA Attestation [19] for SGX. The evidence generated by ECDSA Attestation is called Quote, which contains hardware information, security version information, MRENCLAVE and MRSIGNER of the target enclave, and the trust chain from Quote to Intel CA via the hardware key in CPU. ECDSA Attestation also allows third parties other than Intel to build their own attestation infrastructure. To support this, Intel distributes a program and tools called Intel SGX Data Center Attestation Primitives (DCAP) [1], [13]. DCAP provides Quote Verification Library (QVL) and Provisioning Certificate Cathing Service (PCCS) to allow third parties to verify Quote. QVL is the program for verifying Quote. PCCS is a cache server that third parties can build and use. The cache data is provided by Intel Provisioning Certificate Service (PCS). PCS distributes the data and certificates required for verification.

QVL verifies that (1) the trust chain from Quote to Intel CA has not been tampered with, (2) the key used for it has not expired, (3) the information provided in Quote meets the criteria, (4) the appropriate QE is used, and (5) the target enclave is as expected. However, only (5) must be performed by the user due to its nature. All other verifications are performed using data that the verification server can obtain from PCS/PCCS. The data used for each verification item are (1) the PCK certificate, (2) the PCK certificate and the revocation list applicable to the intermediate CA used for the certificate, (3) the MRSIGNER of the Intel QE, and (4) the latest security version information of the CPU and PCE. Since all of these data are signed by Intel, an attacker cannot forge or alter them.

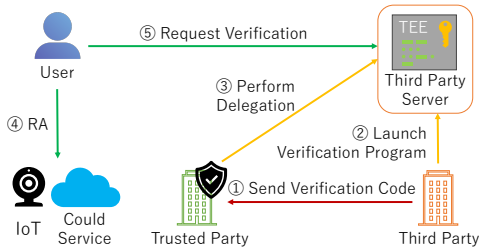

Figure 2. This describes the process overview of the delegation process. The red arrow indicates the examination phase, the yellow arrow is the delegation phase, and the green arrow is the verification phase.

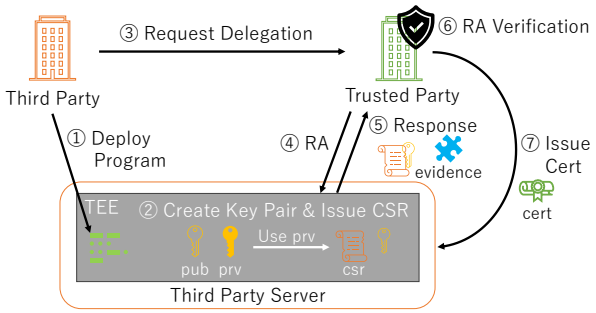

Figure 3. The trusted party delegates verification to a third-party server. The third party is a person or organization that intends to build the delegated verification service. Prv is a delegation key, which can be verified with pub; the public key is included in the CSR.

3. Adversarial Model and Assumption

Our proposed process involves three entities: Trusted Party, Third Party Server, and User.

A trusted party is unconditionally trusted and must be a minimal organization. It also functions as a certification authority that issues certificates for validation delegation. The proposal assumes that a trusted party's behavior and the data it provides are always correct and robust.

A third-party server is a general-purpose server with a TEE managed by a third party other than a trusted party; only the programs executed and data stored inside the TEE are guaranteed confidentiality and integrity. There are multiple third-party servers worldwide, and multiple third parties can increase their number. In our proposal, we do not trust the entire third-party server; we only trust the programs in the TEEs identified by the RA.

The user validates RA evidence using a third-party server for RA on any cloud platform or IoT. The user also verifies the signature of its evidence verification results. At this time, it is assumed that the user knows the issuer certificate of a trusted party. There will be numerous users worldwide.

The adversary's goal is for users to execute RA on vulnerable or rogue platforms without detection. That is possible, for example, if the cloud vendor can generate fraudulent evidence or tamper with evidence. It can also be caused by a malicious or lazy verifier who fails to perform proper verification. We assume an adversary can manipulate any program on the platform where SGX is installed, including OS, BIOS, and hypervisor, using administrative privileges. The adversary can also eavesdrop, delete, and falsify packets in an arbitrary network. Furthermore, the adversary can build and use a new rogue verification server that performs improper verification. However, side-channel attacks, such as timing attacks, are not considered. Also, availability-related attacks, such as denial-of-service attacks, are not considered.

4. Design

In this section, we describe our proposal for the delegation process. First, we present the overall flow in Section 4.1, and then describe the detailed process for each phase.

4.1. Process Overview

Figure 2 illustrates an overview of the delegation process. The delegation process is divided into three phases: the examination phase, the delegation phase, and the verification phase. In the examination phase, (1) a trusted party examines whether the sent verification code is legitimate. The delegation phase ensures that the validation can be performed securely on (2) the validation program invoked by the third party through (3) the vetting of a trusted party. In the verification phase, (4) a user executes RA against other platforms and (5) verifies its evidence using the delegated verification service on the third-party server. Note that these phases do not have to be executed consecutively.

4.2. Examination Phase

A third party first creates the program code for verification and submits it to a trusted party. The trusted party examines the program code that was sent. Specifically, the trusted party must check to see if the verification program has items that should be verified, if there are any known vulnerabilities, and if there is any built-in malware. These are difficult to screen rigorously, but existing tools can alleviate the problem. We discuss it in more detail in 7.1. If the verification program passes the review, the trusted party builds and signs the verification program.

Anyone can use the verification program created in this way. Since the trusted party knows the program's identifier, it can detect if the program has been tampered with in the delegation phase.

4.3. Delegation Phase

Figure 3 shows the flow for verification delegation. (1) At first, a third party launches the verification program that passes the examination phase on a third-party server. (2) The third-party server then generates an asymmetric key pair in the same TEE and issues a Certificate Signing Request (CSR). (3) Afterward, the third party requests a trusted party to perform RA for the verification program running in TEE. (4) The trusted party accordingly executes RA to it. In response to this request, (5) the third-party server returns with the CSR, along with evidence of the authenticity and integrity of the verification program. (6) The trusted party verifies the received evidence and, (7) if there are no problems, issues a certificate using the received CSR. Since then, the signature key has been called the delegation key, and the corresponding certificate is called the delegation certificate in our paper.

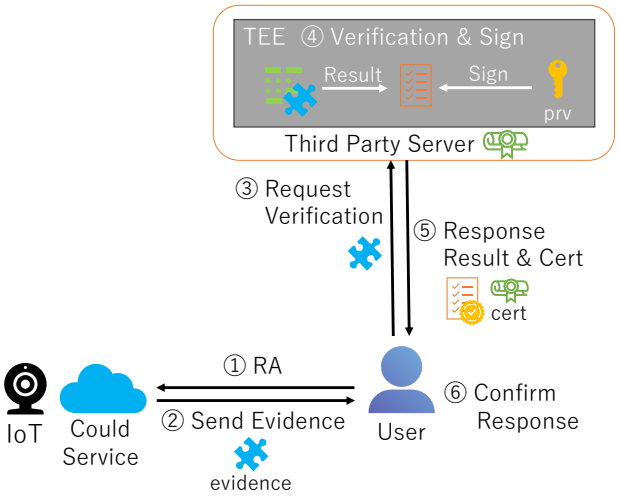

Figure 4. A user requests verification of RA evidence from a third-party server. If the delegation phase has already been executed, the user can use the cert to verify the signature given to the result.

4.4. Verification Phase

Figure 4 shows the verification phase using a delegated verifier. (1) First, A user executes RA against an external cloud service or IoT and (2) receives the evidence about it. (3) The user uses the delegated verifier to verify the evidence. (4) The delegated verifier executes the verification program and signs the verification results with the Delegation Key. Both of these are executed within TEE. (5) The delegated verifier then returns the signed verification result and delegation certificate and issuer certificate to the user. (6) The user checks the verification result and verifies the certificate chain from the signature with the result to a trusted party. As described above, users can check the operation of the verification program only with signature verification.

5. Proof of Concept

This section presents a proof of concept using Intel Software Guard Extensions (SGX) as a TEE for verified program protection. In the verification phase, we assume a user executes RA to the enclave running on a different SGX platform.

5.1. Implementation

We implemented a proof-of-concept verification service by modifying Intel's publicly available Quote Verification Service (QVS) and protecting it with gramine (formerly Graphene-SGX). QVS is a REST API server for QVL, and validation by QVL is performed by sending a quote using the POST method. Gramine is a tool for applying SGX without code modification by including libOS in the enclave. A container running QVS was built and signed with Gramine Shielded Containers (gsc) to apply gramine. The user can use the protected QVS container (gsc-qvs) no differently than a normal QVS.

In the examination phase, a trusted party verifies the code and behavior of QVS and uses gsc to make QVS compatible with SGX. At this time, a trusted party uses its own private key to sign the enclave. A trusted party adds its MRENCALVE to the allowlist of validation program identifiers. This allowlist, maintained only by a trusted party, is used in the delegation phase.

In the delegation phase, a trusted party performs a gramine-compliant RA on the gsc-qvs deployed on a third-party server. At this time, gsc-qvs provides not only a quote but also a CSR corresponding to the delegation key. The OpenSSL EVP function is used for key generation and CSR issuance. However, the current implementation is limited to issuing CSR in the enclave, and tying this to the RA in the gramine is a future task. The RA is the responsibility of the trusted party, which can check if the deployed gsc-qvs have been vetted by using the allowlist of validation program identifiers. If the verification succeeds, the trusted party signs the CSR with its private key and issues a delegation certificate. Then, a trusted party's self-certificate and the delegation certificate are sent to a third-party server. SGX does not have to protect these certificate chains.

In the verification phase, the user performs an RA against Enclave on a different SGX platform than the third-party server. The user sends a quote to gsc-qvs' REST API on the third-party server for verification. gsc-qvs verify the Quote with QVL and then sign the verification result with a delegation key. gsc-qvs then responds to the user with the signed verification result, the delegation certificate, and the issuer's certificate. The user can verify the signature with the certificates to confirm that a legitimate verification program has worked.

5.2. Security Analysis

The delegation phase may be subject to man-in-the-middle attacks. A malicious third party may request RA from the third-party server to issue a fraudulent delegation certificate. However, during the verification phase, users use CA certificates to verify the delegation certificate, so a delegation certificate issued by a non-trusted party will be detected.

In the verification phase, the cloud vendor may try to execute an illegal RA by replacing the QE and Quote items with malicious ones. However, this is only possible if the verification is legitimate since the verification confirms the MRSINGER of the QE. Since a delegated verifier without legitimate verification will not pass the examination of a trusted authority, the user can detect a rogue delegated verifier by using a certificate. After the review, verification programs cannot be tampered with, even by using administrative privileges.

In addition, the key that signs the verification results is generated inside the TEE so that no one can steal it. Since only the examined verification program can handle the private key corresponding to a certificate, the user can be sure that a valid verification has been performed using the certificate.

The administrator of the delegated verifier may also conduct a replay attack that returns previous verification results using an unprotected program. In our proposal, this is prevented by using nonce. The verification program signs verification results containing nonce, which allows users to detect past verification results.

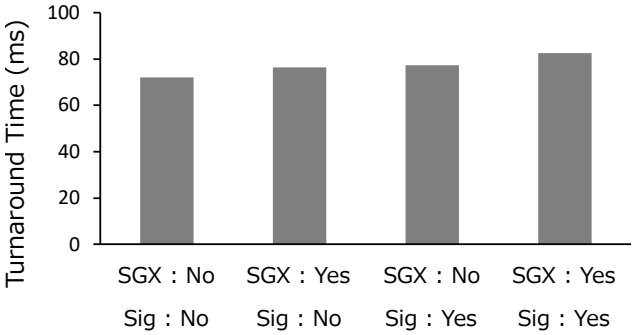

Figure 5. Compare the turnaround time in four cases for evaluating verification overhead. SGX: indicates whether the verification program is running on Enclave. Sig: indicates whether a delegation signature is attached to the verification result.

Although availability is not considered in this research, it is difficult to become a single point of failure because the number of verifiers can be easily increased. Therefore, the system is more resistant to DoS attacks than conventional systems.

6. Evaluation

This section presents the results of evaluation experiments using the implementation of section 5.1. The evaluation environment of our platform has Intel(R) Xeon(R) Silver 4314 CPU, 64GB RAM, Ubuntu22.04 OS, 6.2.0-36-generic kernel. SGX SDK version is 2.22.100.3 and SGX PSW version is 1.19.100.3-jammy1.

We used k6 for our evaluation. The scenario posts a quote serialized in base64 to the validation API. Measurements were taken for 10 seconds and the average value was used.

6.1. Runtime Overhead

The additional overhead compared to the original ECDSA Attestation is signing the verification result with the delegation key. Although a delegation phase requires time to review the verification program and issue the certificate, this is done before a verification phase and does not represent a runtime overhead.

Figure 5 shows the time between posting a quote and receiving a response when only one user existed. We compared four cases: SGX enabled or not, with or without a delegation signature. The measurement results show that the overhead due to SGX is about 4 ms and the overhead due to signatures is about 5 ms. In addition, the signature overhead remains the same even when processing within an enclave. The total overhead with our proposal is 10 ms, which is tolerably tiny enough to be acceptable in practice.

6.2. Effect of Simultaneous Connections

We evaluated how changing the number of simultaneous connections would change the turnaround time for a request to a single third-party server. Specifically, we measured it by varying the number of users from 1 to 15 with the k6 option. Figure 6 shows native qvs versus our implementation of gsc-qvs. Both graphs show the same

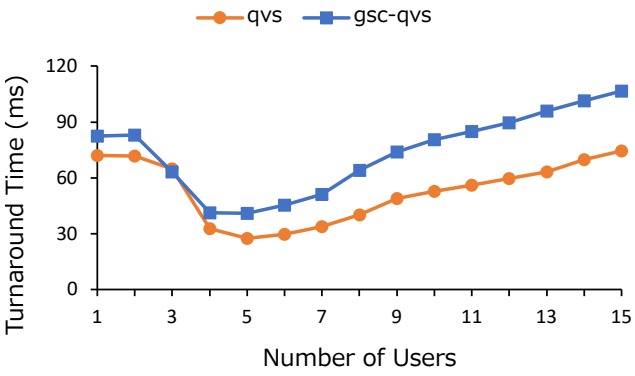

Figure 6. Measurement of turnaround time relative to the number of users. The orange line shows the performance of native qvs and the blue line shows the performance of gsc-qvs with SGX and delegated signatures.

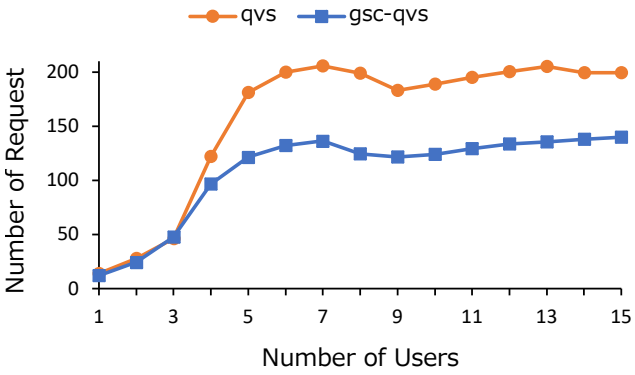

Figure 7. Measurement of the number of requests processed per second relative to the number of users. The orange line shows the performance of native qvs and the blue line shows the performance of gsc-qvs with SGX and delegated signatures.

transitions, but gsc-qvs has a longer turnaround time due to overhead. 1 to 4 users have a faster turnaround time. This seems to be because qvs and gsc-qvs perform asynchronous processing using multiple threads. At 8 users, the turnaround time increases, and from there, it grows almost linearly. Figure 7 shows the number of requests processed per second relative to the number of users. From the 9th user, the number of requests remains almost flat, indicating that the number of requests that can be processed has reached its limit. Therefore, we conclude that many requests caused linear delays.

If more real-time performance is required, or if more requests need to be processed, a third party can easily scale out the verification server using our delegation process. This is discussed in section 7.3.

7. Discussion

This section discusses practical issues regarding the proposed process.

7.1. Verification Code Review

In the examination phase, reviewing a verification code assures proper verification is performed on the program. However, verification programs vary for each security hardware component and its RA specifications. The re-

sulting increase in the amount of code that needs to be reviewed can burden a trusted party.

Therefore, it is desirable to restrict how the verification code is written so that tools can be used to check the verification code. For example, a restriction to separate verification items by function would allow testing for elements that must be satisfied per function.

7.2. Updating Delegation Certificate

Delegation certificates are issued in the same number as the number of deployed verification programs. A trusted party should properly update or revoke all of them. For example, when the TCB of a TEE is renewed, a trusted party revokes the previous certificate as necessary and must perform a new delegation phase.

As for this job, it is desirable for the verification program to periodically check for updates. If there are updates, a new delegation phase is performed and a new delegation certificate is distributed. Implementing this function is a future work.

7.3. Scalability

Since SGX supports multiple threads, the verification service can scale up. However, as indicated in Section 6.2, as RA requirements increase, scale-up alone may not be sufficient to sustain performance. In addition, in cases where real-time performance is required, the servers must be geographically close to each other to reduce communication delay.

Our work improves scalability by facilitating the scale-out of the verification service. The examination phase can automatically perform everything from RA to certificate issuance. It is much faster and easier than borrowing a normal certificate issuance process that requires CA vetting.

However, downloading an endorsement from an online service during verification may cause a bottleneck. When using an endorsement, it is desirable to prepare a cache server like PCCS in SGX or perform caching when the verification service is idle.

7.4. Physical Security

Physical attacks are expected since third-party servers are installed in various locations. In fact, many side-channel attacks have been reported [12], [17]. However, TEE can take countermeasures against them by updating its microcode [16]. Remote attestation can check the microcode updates, so the delegation phase is the countermeasure to physical security.

However, additional countermeasures may be required depending on the type of TEE. In the case of the Scalable SGX we used, the delegation server needs to be cage-boxed, as it does not guarantee integrity against physical attacks [18].

7.5. Implementation on Other TEE

Although we implemented the proof of concept in SGX, our proposed method can also be applied to other TEEs. This section discusses the feasibility of implementation in Intel TDX, AMD SEV-SNP, and Arm TrustZone, which are currently used in practice.

Intel Trust Domain Extensions (TDX) is a new TEE that provides memory protection on a per-virtual machine basis called Trust Domain (TD). Its RA is appropriated from SGX DCAP, and users can see the TD MESUREMENT as the TD's identifier. Therefore, our proposed method can be applied by a trusted party distributing a virtual machine image that includes the verification service.

AMD Secure Encrypted Virtualization - Secure Nested Paging (SEV-SNP) provides memory confidentiality and integrity protection, preventing virtual machines from being attacked by the hypervisor or other virtual machines. In that RA, address space measurements are provided as well as identification information about the virtual machine image. Therefore, the proposed method, like TDX, can be applied by trusted parties that distribute virtual machine images, including verification services.

Arm TrustZone includes TrustZone-A for Cortex-A CPUs and TrustZone-M for Cortex-M CPUs. Both are divided into Normal World, where the normal OS runs, and Secure World, a protected area with limited functionality. However, neither of them officially provides RA, which makes the application of our proposed method difficult. Using the RA proposed by the researcher may solve this problem.

VM-type TEEs have the advantage of requiring no program changes, but keep in mind that they have a larger TCB and consume more resources because they include the OS and kernel. Given that tradeoff, SGX, which can protect only the verification server, remains a reasonable option for our proposal.

8. Related Works

Proxy Signatures [22] is an early technology that increases the number of proxy signers through certificates. In our study, TEE and RA are used to achieve programmatic delegation of rights on a per-program basis. While it is possible to delegate verification authority on a per-server basis, our proposal is superior in that it trusts only a narrower range of trusted entities without increasing the number of trusted entities.

Swarm Attestation helps improve RA responsiveness in IoT. SEDA [8] is the first of its kind and improves the scalability of RA devices by validating IoT devices on the same network in batches. SHeLA [25] assumes the use of advanced verification edge nodes, which could apply to our proposal.

TM-Coin [23] is a blockchain of TCBs that allows any miner to become a verifier. SCRAPS [24] proposes an asynchronous many-to-many RA employing the PubSub model and using smart contracts to improve the scalability of verification. They assume that the verifier is honest by exploiting the properties of the blockchain. We make the verification server scalable while minimizing the number of trusted parties in the verification.

In addition to SGX, Intel TDX, AMD SEV-SNP, and RISC-V Keystone also provide RA capabilities as Remote Attestation for TEE [2], [4], [21]. There is also an independent RA service for TEE, Intel Trust Authority

(formerly Project Amber) [6], which is an online quote verification service for SGX and TDX. Microsoft Azure Attestation [5] is an RA service available for SGX and SEV-SNP on Microsoft Azure. Since only Microsoft manages the verification servers for this service, the number of verification servers cannot be increased for the convenience of third parties. Incorporating our proposed method can make it scalable. Tools for third parties to build their own verification services include DCAP, which is available for SGX and TDX, and VERAISON [3], which is designed to build verification servers for arbitrary TEEs. These are not intended to verify delegation from trusted institutions but can be used for our process implementation.

9. Conclusion

A proof of concept using SGX showed that our proposal's overhead is about 17 ms, which is practical. Also, the limit on the number of requests processed suggests that our delegation process is helpful in increasing RA.

As for future work, we enable the delegation process to be executed automatically and evaluate its performance. We will also show the generality of our proposal by implementing QVS or another verification program on other TEEs.

Acknowledgement

This work was supported by JST, PRESTO Grant Number JPMJPR21P6, JST CREST Grant Number JPMJCR21M3, JSPS KAKENHI Grant Number JP23H03373, and JST SPRING Grant Number JPMJSP2124, Japan.

References

[1] Intel(R) Software Guard Extensions Data Center Attestation Primitives. https://github.com/intel/SGXDataCenterAttestationPrimitives.

[2] Intel® Trust Domain Extensions. https://www.intel.com/content/dam/develop/external/us/en/documents/tdx-whitepaper-v4.pdf.

[3] Veraison. https://github.com/veraison.

[4] AMD SEV-SNP: Strengthening VM Isolation with Integrity Protection and More. *White Paper, January*, 53:1450–1465, 2020.

[5] Microsoft Azure Attestation, Jan. 2022. https://azure.microsoft.com/ja-jp/services/azure-attestation/.

[6] Intel® Trust Authority, Sep. 2023. https://www.intel.com/content/www/us/en/security/trust-authority.html.

[7] Fritz Alder, N Asokan, Arseny Kurnikov, Andrew Paverd, and Michael Steiner. S-FaaS: Trustworthy and Accountable Function-as-a-Service using Intel SGX. In *Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop*, pages 185–199, 2019.

[8] Nadarajah Asokan, Ferdinand Brasser, Ahmad Ibrahim, Ahmad-Reza Sadeghi, Matthias Schunter, Gene Tsudik, and Christian Wachsmann. SEDA: Scalable Embedded Device Attestation. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 964–975, 2015.

[9] Kassem Bagher and Shangqi Lai. Sgx-stream: A secure stream analytics framework in sgx-enabled edge cloud. *Journal of Information Security and Applications*, 72:103403, 2023.

[10] Alexander Sprogø Banks, Marek Kisiel, and Philip Korsholm. Remote Attestation: A Literature Review. *arXiv preprint arXiv:2105.02466*, 2021.

[11] Henk Birkholz, Dave Thaler, Michael Richardson, Ned Smith, and Wei Pan. Remote ATtestation procedureS (RATS) Architecture. RFC 9334, January 2023.

[12] Guoxing Chen, Sanchuan Chen, Yuan Xiao, Yinqian Zhang, Zhiqiang Lin, and Ten H Lai. SgxPectre: Stealing Intel Secrets from SGX Enclaves Via Speculative Execution. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 142–157. IEEE, 2019.

[13] INTEL CORP. Product brief Intel SGX Data Center Attestation Primitives (Intel SGX DCAP), 2019. https://www.intel.com/content/dam/develop/public/us/en/documents/intel-sgx-dcap-ecdsa-orientation.pdf.

[14] Cláudio Correia, Miguel Correia, and Luís Rodrigues. Omega: a Secure Event Ordering Service for the Edge. In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 489–501, 2020.

[15] Victor Costan and Srinivas Devadas. Intel SGX Explained. *IACR Cryptol. ePrint Arch.*, 2016(86):1–118, 2016.

[16] Shufan Fei, Zheng Yan, Wenxiu Ding, and Haomeng Xie. Security Vulnerabilities of SGX and Countermeasures: A Survey. *ACM Computing Surveys (CSUR)*, 54(6):1–36, 2021.

[17] Yeongjin Jang, Jaehyuk Lee, Sangho Lee, and Taesoo Kim. SGX-Bomb: Locking Down the Processor via Rowhammer Attack. In *Proceedings of the 2nd Workshop on System Software for Trusted Execution*, pages 1–6, 2017.

[18] Simon Johnson, Raghunandan Makaram, Amy Santoni, and Vinnie Scarlata. Supporting intel sgx on multi-socket platforms. *Intel Corp*, 2021.

[19] Simon Johnson and Vinnie Scarlata. Supporting Third Party Attestation for Intel SGX with Intel Data Center Attestation Primitives, 2018.

[20] William A. Johnson, Sheikh Ghafoor, and Stacy Prowell. A Taxonomy and Review of Remote Attestation Schemes in Embedded Systems. *IEEE Access*, 9:142390–142410, 2021.

[21] Dayeol Lee, David Kohlbrenner, Shweta Shinde, Krste Asanović, and Dawn Song. Keystone: An Open Framework for Architecting Trusted Execution Environments. In *Proceedings of the Fifteenth European Conference on Computer Systems*, pages 1–16, 2020.

[22] Masahiro Mambo, Keisuke Usuda, and Eiji Okamoto. Proxy Signatures: Delegation of the Power to Sign Messages. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 79(9):1338–1354, 1996.

[23] Jaemin Park and Kwangjo Kim. TM-Coin: Trustworthy management of TCB measurements in IoT. In *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 654–659. IEEE, 2017.

[24] Lukas Petzi, Ala Eddine Ben Yahya, Alexandra Dmitrienko, Gene Tsudik, Thomas Prantl, and Samuel Kounev. SCRAPS: Scalable Collective Remote Attestation for Pub-Sub IoT Networks with Untrusted Proxy Verifier. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 3485–3501, 2022.

[25] Md Masoom Rabbani, Jo Vliegen, Jori Winderickx, Mauro Conti, and Nele Mentens. SHeLA: Scalable Heterogeneous Layered Attestation. *IEEE Internet of Things Journal*, 6(6):10240–10250, 2019.

[26] Ioannis Sfyrakis and Thomas Gross. A Survey on Hardware Approaches for Remote Attestation in Network Infrastructures. *CoRR*, abs/2005.12453, 2020.

[27] Shixuan Zhao, Pinshen Xu, Guoxing Chen, Mengya Zhang, Yinqian Zhang, and Zhiqiang Lin. Reusable Enclaves for Confidential Serverless Computing. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 4015–4032, Anaheim, CA, August 2023. USENIX Association.