On Verification of Remote Computing on Potentially Untrusted Nodes

Hiroki Masuda, Kentaro Kita, Yuki Koizumi, and Toru Hasegawa
Graduate School of Information Science and Technology, Osaka University
Email: {h-masuda, k-kita, ykoizumi, t-hasegawa}@ist.osaka-u.ac.jp

***Abstract*—Verifying remote computing environments, such as computing nodes in fog and edge computing, has gained considerable attention. This poster extends an existing remote attestation method so that it can verify that obtained results are generated by trusted computing nodes as well as remote computing nodes are trusted.**

I. Introduction

Remote computing is a useful approach to several applications like autonomous vehicles. Though remote computing used to be concentrated on clouds, it is shifting towards network edges, i.e., cloud to fog and edge computing, and ultimately edge routers [1]. Such computing environments may contain untrusted computing nodes.

To realize reliable computing, it is indispensable to verify two factors: One is that computing nodes are trusted and the other is that obtained results are truly generated by the nodes. *Remote attestation* [2], which verifies a computing node, is one of the promising approaches. However, existing methods can verify the former but cannot do the latter.

This poster develops a program execution attestation method by extending an existing remote attestation method so that it verifies not only that computing nodes are trusted but also that obtained results are generated by the trusted computing nodes. The key contributions of the paper are twofold: First, we realize an attestation method that verifies returned results as well. Second, we generalize the attestation method so that it is applicable to computing environments where nodes delegate a part of the requested program to other nodes.

II. Computing model, Threats and Countermeasures

1) Computing Model: Nodes that request a program and are requested to execute a program are referred to as *requesting* and *computing nodes*, respectively. A requesting node, RN_P, requests a computing node, CN_P, to execute a program, P, and CN_P returns the result of P, R_P, to RN_P. P may consist of several programs $\{P_1,\cdots,P_n\}$, and CN_P can delegate their execution to $\{CN_{P_1},\cdots,CN_{P_n}\}$, as shown in Fig. 1. In this case, CN_P behaves as both a computing and a requesting node. CN_P receives the results of the delegated programs,$\{R_{P_1},\cdots,R_{P_n}\}$, from the computing nodes, finalizes the execution of P by using the results, and returns R_P to RN_P. Note that programs $\{P_1,\cdots,P_n\}$ may also consist of several programs and $\{CN_{P_1},\cdots,CN_{P_n}\}$ can delegate them to other computing nodes.

978-1-7281-2700-2/19/$31.00 2019 © IEEE

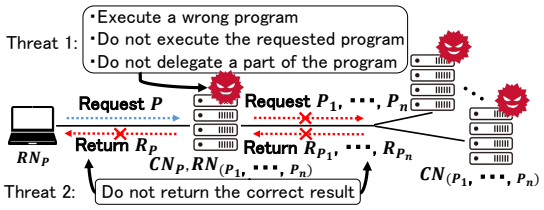

Fig. 1. Computing model and threats

2) Threats: Threats in the computing model are twofold: First, a computing node may not execute a requested program correctly. This malicious behavior includes that the computing node executes a wrong program and it neither executes the requested program nor delegates a part of the program to other computing nodes. The second threat is that a computing node may not return correct results. Even if the computing node executes a requested program correctly, there is still a risk that it may return a wrong result.

3) Countermeasures: To cope with the aforementioned threats, RN_P must verify two factors: One is that *CN_P is a trusted computing node* and the other is that *an obtained result R_P is truly generated by the computing node CN_P*. Note that the word "trusted" represents that an entity conforms to protocols and it does not behave maliciously. In this case, verifying the first factor ensures that CN_P correctly executes a requested program P and correctly generates a result R_P for P.

To verify the first factor, we adopt *remote attestation* [2], which is a technique to verify that a remote computing node is trusted. To verify the second factor, we extend an existing remote attestation method so that it verifies that R_P is generated by a trusted node as well.

Note that our computing model enables computing nodes to delegate the execution of a part of a requested program. Each computing node that delegates other computing nodes to execute programs verifies them as a requesting node. This allows us to focus on the verification process between one requesting node and one computing node.

III. Remote Attestation

Before describing our proposal, we explain remote attestation based on a trusted platform module (TPM). A TPM is a tamper-resistant security chip. It has capabilities of cryptographic hash and public-key cryptography, including key-pair generation, encryption, decryption, and digital signature. It also has

registers, referred to as platform configuration registers (PCRs), each of which stores a hash value.

TPM-based remote attestation assumes that the following three elements are trusted: The hardware of a node, the TPM on the hardware, and the first program like a boot program, which is installed to a ROM device of the hardware. The fundamental idea behind remote attestation is that a node is trusted if all programs, including a boot program, a BIOS, a boot loader, and an OS, that have been executed on the node are trusted. Remote attestation generates verification information, which consists of hash values of programs that have been executed on a node, and verify all the generated hash values are the same as those of trusted programs. Note that hash values of trusted programs are computed in advance and they are assumed to be known information.

Due to space limitations, we describe how to generate verification information securely in the next section.

IV. Program Execution Attestation

1) Platform of Computing Nodes: The platform of computing nodes is based on hypervisor-based virtualization, such as Xen and Hyper-V, and each program is executed on a dedicated virtual machine (VM). We refer to such dedicated VMs as computing nodes, hereafter. We assume that VMs are perfectly isolated, and they cannot access the hypervisor and other VMs. Computing nodes support the virtualized TPM (vTPM) technology [3], and hence each VM has its exclusive vTPM.

2) Overview of Program Execution Attestation: The proposed program execution attestation method is based on TPM-based remote attestation. While TPM-based remote attestation generates verification information regarding a computing node, the proposed method does verification information regarding a computed result as well as a computing node. By using the two types of verification information, the program execution attestation method verifies the two factors discussed in Section II-3.

3) Verification Information for Computing Nodes: Since the platform of a computing node consists of a hypervisor and a VM that executes a requested program P, verification information for a computing node also consists of two types of information about them. If both the hypervisor and the VM are verified, the computing node is trusted. The first verification information is made of hash values of all programs from the initial boot program of the hardware of the computing node to the hypervisor. The second verification information is made of hash values of all programs from the initial boot program to the OS of the VM and P. The hash values are securely stored to TPMs or vTPMs as a hash chain, which comprises superimposed hash values. While the first type of verification information is stored on a TPM, the second type is stored to a vTPM.

A hash chain is generated on a PCR in a TPM/vTPM with the *PCRExtend* command, which executes $V_{PCR} \leftarrow H(V_{PCR}||M_I)$. V_{PCR} is the value of a PCR, M_I is the hash value of a program I, the operator $||$ represents bitwise concatenation, and H is a cryptographic hash function. Each program is responsible for computing the hash value of its next program and storing it to the TPM/vTPM with the *PCRExtend* command. We assume that the initial program is trusted and a trusted program correctly computes and stores the hash value of the next program. Under this assumption, the hypervisor and the VM can be regarded as a trusted platform if all the hash chains are the same as that of the trusted one.

4) Verification Procedure: Next, this subsection explains the verification procedure when RN_P requests CN_P to execute P, which is summarized in Fig. 2. At the side of CN_P, it generates the verification information as well as the result R_P. Next, CN_P generates a return value $A = (R_P, V_{PCR}, V_{vPCR})$ and asks the TPM on the node to make a signature for A, $Sign_{k_s}(A)$, with its secret key k_s. Note that the certificate of the public key corresponding to k_s is opened to the public by the vendor of the TPM and everyone can verify the signature.

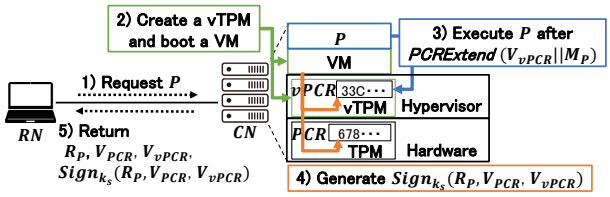

Fig. 2. Generation of Verification Information

When RN_P receives A, it first verifies the integrity of A with $Sign_{k_s}(A)$ and it next compares the hash chains V_{PCR} and V_{vPCR} with those of trusted ones. These two steps ensure that CN_P is a trusted node and R_P is truly generated by CN_P with executing P.

V. Related work

Verifiable computation schemes also enable requesting nodes to verify the correctness of computation requested to computing nodes without TPMs [4]. Most of these schemes, however, are unpractical in terms of computation and storage cost because they rely on complex cryptographic schemes, such as fully homomorphic encryption. Moreover, they do not assume the computing model described in Section II, where computing nodes can also delegate their computation to other computing nodes.

VI. Conclusion

This poster proposes a program execution attestation method to verify that a computing node correctly executes a requested program and the result is truly generated by the computing node.

Acknowledgements

This work has been supported by JSPS KAKENHI Grant Number 18K11263.

References

[1] M. Król and I. Psaras, "NFaas: Named function as a service," in *Proceedings of ACM ICN*, Sep. 2017.

[2] R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn, "Design and implementation of a TCG-based integrity measurement architecture," in *Proceedings of USENIX Security Symposium*, Aug. 2004.

[3] S. Berger, R. Cáceres, K. A. Goldman, R. Perez, R. Sailer, and L. van Doorn, "vTPM: Virtualizing the trusted platform module," in *Proceedings of USENIX Security Symposium*, Aug. 2006.

[4] X. Yu, Z. Yan, and A. V. Vasilakos, "A survey of verifiable computation," *Mobile Networks and Applications*, vol. 22, pp. 438–453, May 2017.