Poster: A Novel Framework for User-Key Provisioning to Secure Enclaves on Intel SGX

Takanori Machida, Dai Yamamoto, Ikuya Morikawa, Hirotaka Kokubo, Hisashi Kojima
Fujitsu Laboratories Ltd., Kawasaki, Japan
Email: {m-takanori, yamamoto.dai, morikawa.ikuya, kokubo.hirotaka, hkojima}@jp.fujitsu.com

***Abstract*—Intel Software Guard Extensions (Intel SGX) protects user software from malicious software by maintaining the confidentiality and integrity of the software executed in secure enclaves on random access memory. However, the confidentiality of its stored executable is not guaranteed. Therefore, secret information, e.g. user keys, should be provided to the secure enclaves via appropriate secure channels. In this paper, we propose a novel framework for user-key provisioning to secure enclaves on Intel SGX.**

I. Introduction

Intel Software Guard Extensions (Intel SGX) [1], [2], [3] is a function included in modern Intel processors and under contemporary focus as one of the hardware-based isolation technologies that protects user software from malware for several years. Intel SGX reserves parts of random access memory (RAM) for "secure enclaves". In secure enclaves, stored data are encrypted with processor internal keys and verified with message authentication codes in the processor. Since the trust boundary of Intel SGX is the processor package, confidentiality and integrity of user software executed in the secure enclaves is ensured, and user software is protected from malware outside these enclaves, even if such malware has OS root privileges.

Beginning in 2017, many researchers published more than a hundred of papers related to Intel SGX and proposed various applications for it, e.g. digital rights management (DRM) [3], system logs [4], and power grid protection [5]. These applications often include secret information such as keys for encryption, decryption, authentication, and signing, which are predefined by an independent software vendor (ISV). Although these keys (user keys) are protected in secure enclaves on RAM, an attacker can get the user keys with binary analysis of SGX executables stored not in RAM but in disc storage before executions of SGX user software; SGX executables have no confidentiality[1]. Consequently, how we set user keys into user software executed in secure enclaves on RAM becomes one of the challenges for Intel SGX. In this paper, we introduce conventional methods of provisioning user-keys to secure enclaves on Intel SGX, and propose a novel framework. Our framework utilizes *sealing*, a function of Intel SGX, and supports high security at a low cost.

[1]The integrity of the executable is maintained with a certificate attached to the executable.

II. Intel SGX

Section I showed the prime function of Intel SGX: isolated execution of user software. This section explains other functions: sealing and remote attestation.

Sealing/Unsealing

Software, executed in a secure enclave, sometimes generates ephemeral data. We might need to reuse the data in the next execution. The sealing function encrypts such data with seal keys, which depend on the processor key, which in turn is based on a symmetric cryptographic scheme. Sealing enables us to transfer data securely from the secure enclave to storage. Unsealing regenerates the seal keys and decrypts the encrypted data. There are two types of seal keys. One is derived from the processor key and hash value of user software executed in the secure enclave on RAM, i.e. it is unique for each software installation. Hence, only when that same software is executed on the same processor as used in the encryption, can data be decrypted correctly. The other seal key is derived from the processor key and hash value of the ISV public key[2] that ISV sets for user software when it is compiled. If ISV sets the same ISV public key into different software, the data can be decrypted correctly since the same seal key can be regenerated. The latter is utilized for sharing sensitive data among software on the same processor.

Remote Attestation

Remote attestation verifies whether software in secure enclave is executed on a genuine Intel processor or not. This function uses a group signature based on the processor key. A verifier requests a certificate, signed with a member secret key derived from the processor key, to the SGX user software (prover) via networks. The Intel attestation verification service verifies the trustworthiness of the certificate anonymously using the group public key. The verifier can obtain the result by asking the service. Remote users can establish secure communication with SGX software by attaching an ephemeral public key to the certificate.

III. User-Key Provisioning to SGX Enclaves

A. Conventional Methods

A challenge of Intel SGX is to provision user keys securely to secure enclaves on RAM since SGX executables have no

[2]The public key is essentially used for detecting forgeries of executables.

confidentiality. The first possible solution is the obfuscation of the user keys inside the executables. Although this can be demonstrated at low cost, it cannot keep the user keys completely confidential.

Another solution is to use the remote attestation function. As mentioned in Section II, remote attestation enables users to establish a secure communication channel with an SGX application deployed on a remote platform. The users can send their keys securely to the SGX application through the channel. The disadvantages of this method are that using the third party (Intel) services occasions a cost, and inevitably requires network communication, i.e. the method cannot be used in offline environments.

B. Our Framework

Our proposed method realizes complete confidentiality at low cost. Our framework for user-key provisioning to secure enclaves consists of two phases: the provisioning phase and the operation phase.

Provisioning Phase*:* This phase provides user keys to secure enclaves. Once the phase is completed, SGX applications using the secure enclaves can utilize the user keys securely. The only requirement of our framework is that devices where an SGX executable dedicated for the provisioning phase is executed are not compromised by malware during this phase. A situation satisfying the requirement is that the devices have not yet been used in real-world environments connected to public networks, e.g. have only been used in environments like a manufacturing factory that makes the devices, or a closed network for administrators. The procedure for the provisioning phase is as follows (also as shown in Fig. 1).

(p-1) The dedicated executable including user keys is set to a device by an ISV.

(p-2) An administrator executes the executable in a secure enclave.

(p-3) The executable performs the sealing function in order to encrypt the user keys with a seal key of the second type mentioned in Section II, and saves the encrypted user keys in storage.

(p-4) The dedicated executable with the user keys must be removed from the device since SGX executables have no confidentiality.

Operation Phase*:* In this phase, the SGX applications can utilize the user keys sealed in the provisioning phase. The procedure for the phase is as shown below and in Fig. 1.

(o-1) The ISV provides the device with executables of the SGX applications that have the ISV public key identical to the one included in the dedicated executable used in the provisioning phase.

(o-2) An application user or administrator executes the executables.

(o-3) The executables read the encrypted user keys from storage, and perform the unsealing function that decrypts the encrypted user keys with the seal key derived from the processor key and the identical ISV public key.

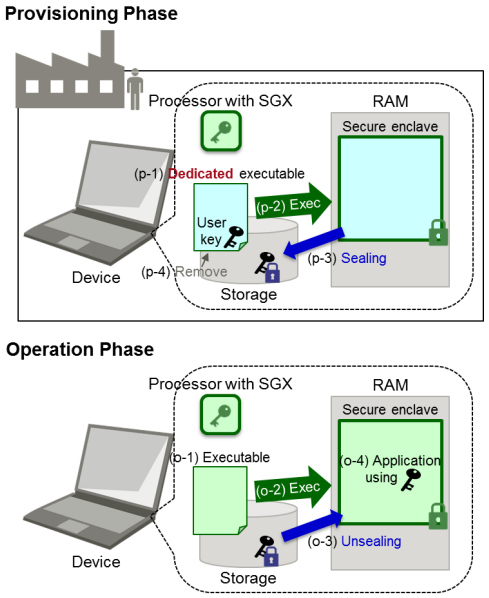

Fig. 1. Our framework of user-key provisioning to secure enclaves on Intel SGX

(o-4) The user key are utilized in the performance of the SGX applications, such as DRM and biometric authentication.

Our framework is not considered for cloud applications because, in the cloud, it is difficult to demonstrate the provisioning phase in a secure environment for ISV; there is a potential risk that secrets included in the executable dedicated for the provisioning phase fall prey to eavesdropping by insiders at a cloud provider. Consequently, our framework is suitable for using on edge or endpoint environments because administrators on these environments can manage the dedicated executable without exposing it to attackers. We implemented a prototype of our framework using a commercial Intel CPU (Intel Core i7 6700K) and evaluated its feasibility.

IV. Conclusion

In this paper, we proposed a novel framework for user-key provisioning to secure enclaves on Intel SGX. Our framework achieved both security and cost on edge and endpoint environments compared to conventional methods.

References

[1] F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar, "Innovative Instructions and Software Model for Isolated Execution," Intel Corporation, White Paper, Aug. 14, 2013.

[2] I. Anati, S. Gueron, S. Johnson, and V. Scarlata, "Innovative Technology for CPU Based Attestation and Sealing," Intel Corporation, White Paper, Aug. 14, 2013.

[3] M. Hoekstra, R. Lal, P. Pappachan, V. Phegade, and J. del Cuvillo, "Using Innovative Instructions to Create Trustworthy Software Solutions," Intel Corporation, White Paper, Aug. 14, 2013.

[4] V. Karande, E. Bauman, Z. Lin, and L. Khan, "SGX-Log: Securing System Logs with SGX," in *AsiaCCS 2017*, 2017, pp. 19–30.

[5] F. Campanile, L. Coppolino, S. D'Antonio, L. Lev, G. Mazzeo, L. Romano, L. Sgaglione, and F. Tessitore, "Cloudifying Critical Applications: A Use Case from the Power Grid Domain," in *PDP 2017*, 2017, pp. 363–370.