September 20, 2018

A Cpu-Instruction-Based Asymmetric Signing/ Decryption Mechanism For Secure Handling Of Asymmetric Keys

Uday Savagaonkar

Andrew Honig

A CPU-INSTRUCTION-BASED ASYMMETRIC SIGNING/DECRYPTION MECHANISM FOR SECURE HANDLING OF ASYMMETRIC KEYS

ABSTRACT

A method and system are disclosed for allowing a central processing unit (CPU) to perform signing/decryption operations securely. The system includes the CPU, which embeds an asymmetric private decryption key called CPU Decryption Key (CDK). A public key corresponding to the CDK, known as CPU Encryption Key (CEK) is published by the CPU vendor, and comes with a vendor-signed certificate. The CPU exposes two instructions - IMPORT_KEY and USE_KEY, which point to memory locations for storing decrypted keys, wrapped keys, and data. The disclosed mechanism provides a high level of security in cloud environments by providing a secure key delivery to the signer and protecting the signer. In addition, it involves low cost when compared to hardware security modules(HSM).

BACKGROUND

A data-center environment is required to handle a number of asymmetric signing/decryption keys. These keys include SSL signing keys, RSA decryption keys, as well as authentication token (e.g., OAUTH2) signing keys. Generally, in data-center environments, these keys are completely handled in software, making these keys vulnerable to various software as well as hardware attacks. An attacker that has root-level access to the computers handling the signing/decryption operation can easily identify these keys within the system memory by scraping the memory. Additionally, attackers that have physical access to computer which handle the signing/decryption operation can steal the keys using cold-boot or similar hardware attacks. An adversary possessing these keys seldom needs anything else to get access to security-sensitive resources. Consequently, asymmetric signing/decryption keys, if compromised, can

pose significant security risk, as such keys are generally used as bearer credentials. Hardware security modules (HSMs) today provide comprehensive solution for both secure key delivery as well as secure signing operation. HSMs typically come with a mechanism for securely importing keys, and then utilizing these keys in a hardware-protected environment that is strongly protected against both software and hardware attacks. However, HSMs with reasonable performance are extremely expensive.

DESCRIPTION

A system and a method are disclosed for allowing a central processing unit (CPU) to perform signing/decryption operations securely, as illustrated in FIG. 1. The system includes the CPU, which embeds an asymmetric private decryption key called CPU Decryption Key (CDK). The CDK can be an RSA or an EC-IES private key. A public key corresponding to this CPU private key, known as the CPU Encryption Key (CEK) is published by the CPU vendor, and comes with a vendor-signed certificate. In one example, the CEK may be used to encrypt keys or data transmitted to the CPU by any third party entity. The CPU exposes two instruction set extensions - IMPORT_KEY and USE_KEY, for secure handling of keys.

The IMPORT_KEY instruction takes two memory addresses as parameters - one memory address is an input and the other memory address is an output. The input/output parameters could also be CPU registers, instead of memory locations. The input parameter points to a memory location that contains the Target Asymmetric Key (TAK), which is encrypted using the CEK. For example, the third party entity may encrypt the TAK using the CEK and share it with the CPU. The TAK is the key with which a data-center operator or signer intends to perform the signing/decryption operation. The IMPORT_KEY instruction decrypts the encrypted TAK using the CDK, which is embedded in the CPU itself, and wraps it using a symmetric key called the

CPU Wrapping Key (CWK). This wrapped key is written to the location pointed to by the output memory parameter. The CWK may or may not be ephemeral, i.e., it may change from boot-to-boot.

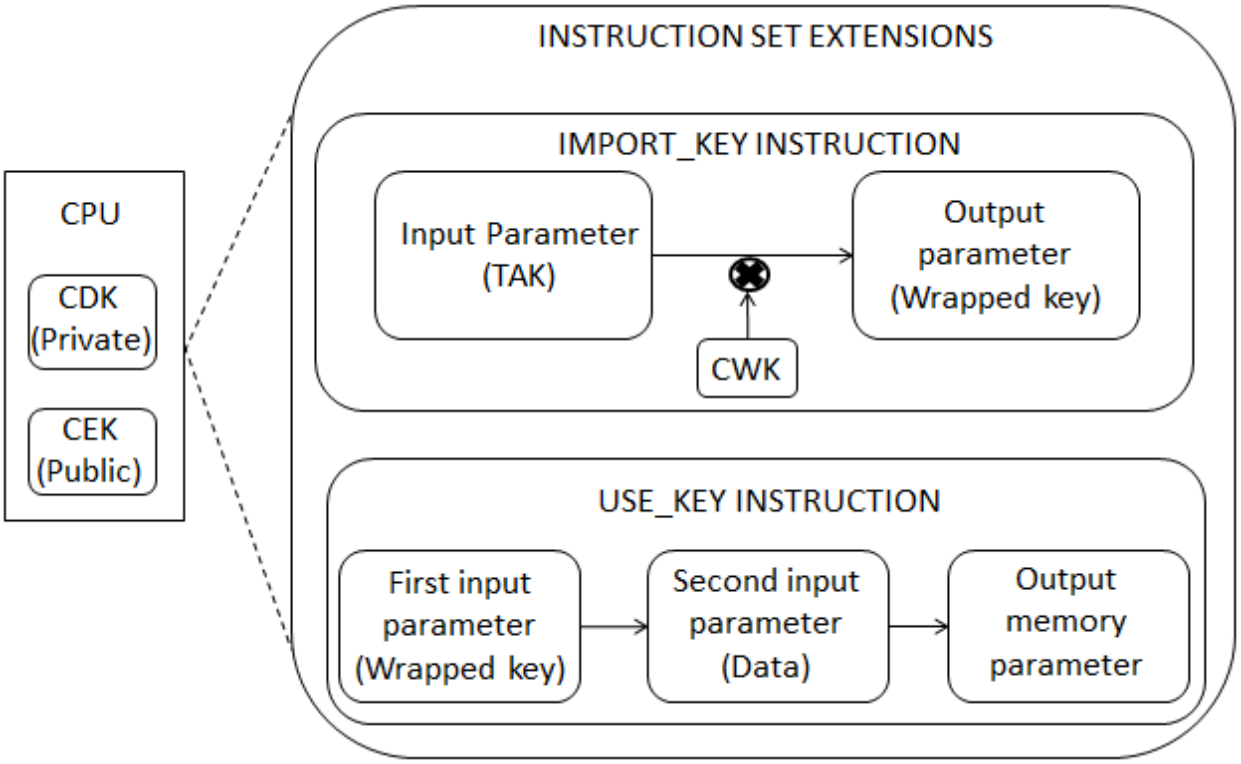

FIG. 1: System for allowing a CPU to perform signing/decryption operations securely

The USE_KEY instruction takes three memory addresses as parameters, wherein two memory addresses are input and the third memory address is an output. The first input parameter points to a memory location that contains the wrapped TAK, while the second memory parameter points to the memory location that holds the data that needs to be decrypted/signed. The USE_KEY instruction unwraps the wrapped TAK, signs/decrypts the data pointed to by the second input parameter, and writes the result to the output memory parameter, which may be accessed by the data-center operator or signer. Therefore, instruction set extensions (IMPORT_KEY and USE_KEY) provide secure delivery of keys from the third party to the signer.

Further, the multiple input memory parameters for these instructions could be coalesced into fewer memory parameters. For example, for the IMPORT_KEY instruction, only one memory parameter could be specified, and the other memory parameter could be inferred as a fixed offset (which could be zero) from the first memory parameter. Similarly, the memory parameters for the USE_KEY instruction could be coalesced into fewer instructions.

A single CDK/CEK pair could be shared by a group of CPUs. In such a scenario, CEK certificate is shared by the group of CPUs. Each CPU could have a unique CDK/CEK pair, and comes with its own CEK certificate. The IMPORT_KEY and USE_KEY instructions could either be interruptible or uninterruptible instructions.

For instance, when the instruction is interrupted, the partial computation could be discarded by the CPU. Alternatively, the partial computation could be wrapped using the CWK (or some other key) and handed back to the software. The software can then resume the interrupted operation using the wrapped partial results.

In a slight variation of this scheme, the CDK itself may not be embedded in the CPU. Instead, a symmetric key, called the CPU Symmetric Key (CSK) is embedded in the CPU, and the CDK is provided to the software wrapped with CSK. In such a scenario, the IMPORT_KEY instruction would first unwrap the CDK before using it to decrypt the TAK.

The disclosed mechanism provides a high level of security in cloud environments by providing a secure key delivery to the signer and protecting the signer. The mechanism can be used to harden asymmetric keys in datacenter operations, including cloud vendors, resellers, CPU vendors, third party security appliances to provide higher security to their customers. Further, the mechanism enables IT administrators to securely deploy and utilize machine certificates. In addition, it involves low cost when compared to hardware security modules

(HSM).