Intel SGX usage to seal sensitive data using fuses of CPU

Problem Statement

In case of no TPM device available on edge-node or if TPM does not fit requirements, we do not use encryption to store the data on device and store keys on the device in plain text. We need to find possible ways to use hardware/software approaches to tight our keys with hardware we deploy EVE-OS onto. This way we can avoid using of data in case of stealing of disk.

Motivation

In order to enforce security we can use Seal function of Intel SGX. Intel SGX do not provide non-volatile storage unlike the TPM, but it propose the way to encrypt the data using the key derived from fuses of the CPU.

Proposed Solution

The SDK provided by Intel uses AES-GCM, more precise Rijndael128GCM to encrypt the input data. Information about the key is available in SGX Programming Reference 5.4.1: EGETKEY instruction returns a 128-bit secret key from the processor specific key hierarchy.

So the main idea here is that without non-volatile encrypted memory we can seal the data using key derived from processor fuses and store it in un-encrypted storage. And we will able to unseal the data only using the same CPU.

Inside the prototype, which is based on samples from Intel SDK I implement the logic of sealing of device key: I remove stored device key after the first reboot to store only sealed key on disk. Then I unseal the key into in-memory directory and bind this directory instead of on-disk config partition.

Few drawbacks/comments

1. The key derivation function also uses the signature of enclave application the we load (MRSIGNER is set by default). The idea here is to also be tight on application that was used to seal the data and to support future versions signed with the same key. As we build open-source project we can put the key in the repository, but any application signed by the publicly available key will be able to unseal the key on the CPU. It may be the problem if one day we will allow to use SGX on edge-node to un-trusted applications. Not the case for now, but should be considered.
2. Intel SGX is deprecated on customer (non-Xeon) CPUs
3. There are two kernel modules available: in-tree and out-of-tree. Out-of-tree module do not expect additional features to be available on platform, so I choose it for prototyping. But out-of-tree module is deprecated and it come with additional limits for platforms.
4. It is not clear, is the sdk and utility which app uses (psw) are open-sourced fully. During build sdk uses pre-build libraries which limit usage on non-libc distributions (only Ubuntu and RHEL/CentOS officially supported). So usage with musl-based environment looks limited (in prototype I prepared container based on Ubuntu).
5. Using the single object encryption approach leads to significant changes in the EVE logic. It is not possible to use boot measurement, since the startup is done after the kernel/system is started.