Scaling Intel SGX

- **Requirements**
  - Expand SGX from client to server, with the same software development experience and APIs
  - SGX applications should be platform agnostic
- **Challenges**
  - 1. Keys used for attestation and sealing are derived from per CPU socket secrets
  - 2. Memory residing in EPC, physically attached to one socket, need to be securely accessible from another socket
  - 3. Supporting large protected memory
    - Reduce the performance overhead of the MEE
    - Reduce the memory storage overhead for memory integrity and replay protection tree
Addressing challenge 1 – Creating a single identity

- For sealing and attestation
  - Single Socket: A key request will result in the same answer on all the logical processors in the socket
    - Sealing keys, Report keys, Signing keys
  - Multiple Socket requirement
    - Keys available to the enclave must be consistent even when a process is scheduled on different sockets (solutions: generation of platform keys, shared among sockets)
    - Attestation keys (generated in the platform) should be registered with the attestation infrastructure
    - When a socket is moved into a different platform, it must not continue to use the source platform’s keys
    - Prior to exposing existing platform keys to a new socket, that new socket must be vetted by the infrastructure that certified the platform with an attestation key
Addressing challenge 1 – Creating a single identity

- Multi-Socket Life-cycle stages
- Platform establishment
  - Generates platform keys, along with an authenticated manifest that describes all the devices with access to those keys
- Platform Registration
  - Makes use of the manifest to register platform and its components with a registration service
  - The Registration Service authenticates the manifest and the devices referenced in it
  - The Registration Service generates the certificates necessary for attestation key provisioning
  - After registration, these certificates allow the existing provisioning infrastructure to recognize the new platform in the same way as it does a standard client
Addressing challenge 1 – Creating a single identity

- Multi-Socket Life-cycle stages
- Platform establishment
  - Generates platform keys, along with an authenticated manifest that describes all the devices with access to those keys
- Components
  - HW Key1/Key2
  - Platform Keys
  - Encrypted PlatKeys
Addressing challenge 1 – Creating a single identity

- Multi-Socket Life-cycle stages
- Platform registration
  - DCAP recap

Figure 2: Single Socket Provisioning

Figure 3: Quote Certificate Chain
Addressing challenge 1 – Creating a single identity

- Multi-Socket Life-cycle stages
- Platform registration
  - Makes use of the manifest to register platform and its components with a registration service
  - The Registration Service authenticates the manifest and the devices referenced in it
  - The Registration Service generates the certificates necessary for attestation key provisioning
  - After registration, these certificates allow the existing provisioning infrastructure to recognize the new platform in the same way as it does a standard client
Addressing challenge 1 – Creating a single identity

- Platform registration
  - The Intel Trusted Firmware generates platform manifest, including platform root keys (such as platform provisioning root key), the information of sockets in the platform
  - Signed by the sockets’ private key
  - The registration service verifies the private key and the platform configuration, generates the certificate for platform PCKs, and sends to the PCK certificate service

*Figure 3: Multi-Socket Provisioning*
Addressing challenge 1 – Creating a single identity

• Key exchange between sockets
  – Long-lived Mater Comms Keys
    • Diffie-Hellman key exchange, using the unprotected memory as a channel
    • Each socket negotiate keys with the adjacent sockets and the Master Socket (appointed by the BIOS)
  – On every reset
    • Master Comms Keys are used to establish encryption/MAC keys between sockets
    • Using these keys, processors exchange and verify consistency of memory configurations and information found in the Platform Info structure
Memory coherency architecture

- Addressing challenge 2
  - 2. Memory residing in EPC, physically attached to one socket, need to be securely accessible from another socket
Scalable SGX memory protection

- Challenges
  - 3. Supporting large protected memory
    - Reduce the performance overhead of the MEE
    - Reduce the memory storage overhead for memory integrity and replay protection tree
- To support large memory
  - Increase the side of on-die storage used for the top-level anti-replay counters (linearly)
  - Add more levels to the anti-replay tree (latency)
Scalable SGX memory protection

- What benefits does the integrity replay tree provide?
  - SGX does not rely on encryption to provide separation between enclaves and regular memory or separation between enclaves (encryption with the same key)
  - Separation/isolation/access control (conducted by the PMH on each TLB miss) are orthogonal from physical memory protection (i.e., memory encryption)
  - The MEE provides protection against physical attacks, but does not help in preventing software attacks

- If the MEE is replaced by (MK)TME
  - What kind of software attacks need to be prevented?
  - What kind of physical attacks need to be prevented (not covered by the white paper)?
Scalable SGX memory protection

- If the MEE is replaced by (MK)TME
  - What kind of software attacks does MEE provide protection against?

<table>
<thead>
<tr>
<th>Attacks</th>
<th>Description</th>
<th>Resulting loss of</th>
<th>Prevented also by TME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset attack</td>
<td>The attacker forces a reboot of the platform and forces the previous memory range protection to be dropped. Secrets can now be retrieved from uninitialized memory.</td>
<td>Confidentiality</td>
<td>yes</td>
</tr>
<tr>
<td>EPC reclaim</td>
<td>If platform has ability to tear down protections post use (without reset), EPC protected secrets would be exposed. [1]</td>
<td>Confidentiality</td>
<td>yes</td>
</tr>
<tr>
<td>Aliasing/EPC replay</td>
<td>Attacker configures a second system address to map to an EPC page. Ability to replay memory of arbitrary data allows code injection into enclave.</td>
<td>Integrity/Confidentiality</td>
<td>no</td>
</tr>
<tr>
<td>DIMM Config. Attacks</td>
<td>SW attacks DIMM configuration settings to prevent writes from becoming persistent.</td>
<td>Integrity/Confidentiality</td>
<td>no</td>
</tr>
</tbody>
</table>

[1] EPC content should be cleared by EREMOVE?
Scalable SGX memory protection

• If the MEE is replaced by (MK)TME
  – What kind of attacks does MEE provide protection against?

• Clarification of aliasing attacks[3]
  – The EPCM inside SGX helps to provide protection against simple software-only memory aliasing attacks
  – Aliasing may still happen in server platforms, e.g., due to the Reliability Availability and Serviceability (RAS) feature known as sparing
  – MEE catches changes to the memory pages through the integrity/anti-replay tree, which is not present on TME
Scalable SGX memory protection

- **Outside-in aliasing**
  - SA1 and SA2
  - Untrusted software outside the enclave can manipulate an enclave resident in the EPC page affected
  - Resolved by repurposing 1 ECC bit to indicate whether the line previously written was an SGX line

- **Inside-in aliasing**
  - SA3 and SA4
  - A malicious enclave can gain access to any other enclave in the EPC page affected
  - No runtime detection; checked by the trusted FW on system boot to ensure no aliases within the PRMRR region
Scalable SGX memory protection

- Summary

<table>
<thead>
<tr>
<th>Protection memory from</th>
<th>HW or SW</th>
<th>Client SGX (SGX1)</th>
<th>Scalable SGX (SGX2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loss of Confidentiality</td>
<td>Software</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Loss of Integrity</td>
<td></td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Anti-Replay</td>
<td></td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Loss of Confidentiality</td>
<td>Hardware</td>
<td>yes</td>
<td>yes*</td>
</tr>
<tr>
<td>Loss of Integrity</td>
<td></td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Anti-Replay</td>
<td></td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

[*] TME uses AES-XTS mode, so it mitigate the HW attacks where the adversary only sees the ciphertext once, and not while the system is changing the data – unclear
Platform Configuration

- Platform setup
  - BIOS: normal (e.g., prepares the platform hardware); then triggers the Intel FW module
  - Intel FW module: runs on each socket, and
    - Check platform configuration
    - Calculates/decrypts program keys in HW
    - Memory alias checking to prevent inside-in memory scenario
  - More details
    - Establishment
      - Key exchange between sockets
      - The sockets create a signed manifest of the pairing and platform keys (for registration verification) – returned to BIOS
      - Encrypted key structure for platform keys, communication keys for each pairing, and platform configurations
Platform Configuration

• Platform setup
  – More details
  • Reboot
    – BIOS determines every socket has a key structure
    – During boot, platform keys, session keys, UCE key, TME key are decrypted and configured
  • Adding a socket
    – BIOS discovers a new socket and sends the ADD Request to the registration service
    – If approved, the service creates a Platform Membership Certificate
    – After the next reset, existing socket can verify the certificate and share the previously established platform keys with the new socket
Discussion

• Is the Intel FW module trusted?
  – No secret
  – If it is manipulated by the attacker
    • Inside-in aliasing is possible

• What if the encrypted platform key is replaced (physical attack)
  – We assume the secret key of all sockets is protected, otherwise no protection is possible
  – Replay/downgrade: use a leaked encrypted platform key (limited security threat)

• Protection against software attacks
  – Strong

• Protection against physical attacks
  – Weak