Supporting Intel® SGX on Multi-Socket Platforms

Simon Johnson, Raghunandan Makaram, Amy Santoni, Vinnie Scarlata

Intel Corporation

{simon.p.johnson, makaram.raghunandan, amy.santoni, vincent.r.scarlata}@intel.com

ABSTRACT

Intel® Software Guard Extensions (Intel® SGX) was initially released on client platforms and later extended to single socket Intel® Xeon® E3 Processor-based servers (code-named Cascade Lake). Various Cloud Service Providers (CSPs) are demonstrating the value of using Intel SGX-based Trusted Execution Environments (TEE) to create a new paradigm of Confidential Cloud Computing. As developers have become familiar with the capabilities of the technology, the applicability of this capability in the cloud has been tested. Intel has made several enhancements in upcoming dual-socket 3rd Gen Intel® Xeon® Processor-based server platforms (code-named Ice Lake) to deliver Intel SGX for Data Center.

This paper describes the additional enhancements that are essential to deliver a user-programmable Trusted Execution Environment that scales to cloud usages and improves security on multi-Socket platforms without compromising performance.

1 Introduction

In introducing Intel® SGX [1], Intel has changed the way applications provide protection to secrets without having to fully rely on the underlying and more privileged platform software. Usages that benefit from this type of protection are wide ranging, including protection of fingerprint matching, password manager vault protection, blockchain wallets, smart-contract verification, AI/ML algorithms, and a variety of other key protection usages.

In recent years as applications become more dependent on services from the cloud and the network edge, more and more data are being processed on platforms that do not necessarily belong to the data owner or the service provider. This places cloud and network edge operators in a unique situation, where they want to host other parties' workloads, but do not want access to those workloads or their data. Most of these hardware owners would like to tell their customers that their workloads can be hosted without requiring trust in the CSP – that they won't reveal data unintentionally through security breaches.

The birth of the confidential computing paradigm, using Trusted Execution Environments to provide enhanced protection to data and the CSP's reputation is unsurprising and has resulted in new usages such as federated learning, privacy preserving analytics and secure multi-party computations coming to the fore.

To deliver this paradigm, a Trusted Execution Environment that scales *and* performs is required. Using desktop PC's and mobile chips only delivers the initial potential in concept. TEE's that can scale to many concurrent instances, running on many cores on platforms with more than one CPU socket, are required for todays' most complicated data exchange and privacy usages.

1.1 Scaling Intel SGX – Requirements and Challenges

To address this need, Intel looked to expand Intel SGX from client to server. The goal in this expansion was that the software development experience and API remained unchanged so that Intel SGX enabled application software, which is platform agnostic, would just work whether it's running on Intel® Core® i7 client or an Intel Xeon Processor server.

To achieve this outcome, there was a need to address challenges that were specific to Intel SGX.

Keys used for client attestation and sealing are derived from per CPU Socket secrets. The platform needed to consist of multiple CPU sockets that are cryptographically coherent and also with a single identity.

Memory residing in protected ranges, physically attached to one socket, needed to be securely accessible from another socket.

At the same time these challenges were being explored, the advent of containerizing whole applications inside an Intel SGX enclave [2] [3] started to arise as a serious usage scenario. This challenged the view that applications would only use small amounts of protected memory, consequently this addressed two other significant challenges:

Reduce the performance cost that the Memory Encryption Engine [4] introduces to workload owners.

Reduce the memory storage overhead costs that platform owners need to pay for the memory integrity and replay protection tree.

Additionally, several ways were explored to help reduce both the size of the integrity tree and the impact on performance created by the additional memory access of the integrity tree. Balancing these questions results in a performance vs security trade-off discussion.

2 Creating a Single Identity

2.1 Overview

On single socket platforms, such as standard clients, Intel SGX functionality and security properties are provided completely by the one socket. Each socket ships with per-part unique keys built into the HW. Intel SGX instructions allow enclaves to access keys derived from these HW keys to help protect secrets or securely communicate between enclaves. Unique signing keys can also be derived. Using these signing keys, along with certificates issued by Intel, 3rd parties can remotely authenticate that they are communicating with a genuine Intel SGX-capable processor socket. The signing keys can be used to provision attestation keys to the platform or be used directly for attestation. These keys are common across all logical processors on a single socket. A key request will result in the same answer on all the logical processors in the socket. Operating system schedulers rely on this coherency. Losing this property would require complex operating system changes.

Establishing a single coherent software environment on multi-socket platforms creates several new platform requirements.

1. Keys available to the enclave must be consistent even when a process is scheduled on different sockets. Additionally, this requirement should not be met by allowing a socket's hardware keys (effectively its identity) be exposed outside that socket. Under this restriction a new set of platform-wide shared keys must be established.

2. Keys generated in the platform and used as a foundation for attestation must be registered with the attestation infrastructure to be recognized during attestation key provisioning.

3. When a socket is moved into a different platform, it must not continue to use the source platform's keys.

4. 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.

2.2 Multi-Socket Life-cycle Stages

Deploying a multi-socket platform has two new events that occur as part of the platform bring-up: Platform Establishment and Platform Registration.

Platform Establishment generates new platform-wide keys along with an authenticated manifest that describes all the devices with access to those keys. Next, *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.

2.2.1 Platform Establishment

During Platform Establishment, sockets configure the platform with common keys and secure inter-socket communication.

The processor socket contains unique, per-part hardware keys, called the Intel SGX root keys.

Figure 1 shows the Intel SGX data protection model when 2 sockets are used. Rather than encrypting user data with the hardware keys in the sockets, derivatives of new "platform" root keys are used. Data can then be accessed regardless of which physical socket. To ensure the platform keys are accessible after the platform is reset, they are encrypted and stored in persistent storage (ex. flash). Each socket encrypts its own copy of the platform keys using its hardware keys. This ensures that if any socket fails, the remaining sockets can still access the platform keys.

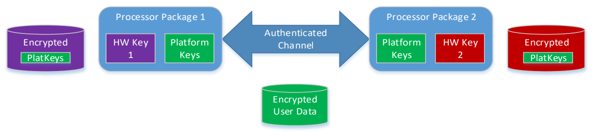

Figure 1: Multi-socket Intel SGX Keys

Sockets use unprotected memory as a communication medium to negotiate secure session keys (see section 2.2.3). Using these session keys, the sockets compare/agree on configurations and generate the new platform keys. If a physical link connects two sockets, they will also negotiate and configure a set of link protection keys.

2.2.2 Platform Registration

The Registration Service bridges the attestation provisioning gap between client provisioning model and multi-socket platform provisioning.

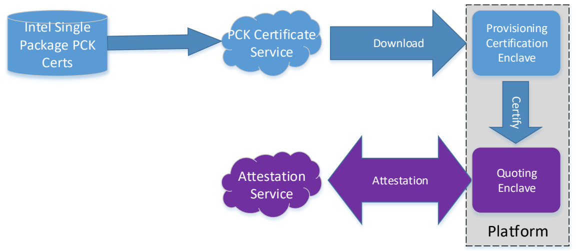

Figure 2: Single Socket Provisioning

Figure 2 depicts the single socket server attestation model based on Intel® Data Center Attestation Primitives (DCAP) services [5]. Each socket has a unique TCB-based identity key called the Provisioning Certification Key (PCK). Intel published certificates for all PCKs in a PCK Certificate Service. During server or VM initialization, this certificate is downloaded by platform software. In most cloud deployments, the CSP will host a local PCK certificate caching service that will download certificates from the Intel® PCK Certificate Service in bulk and then make them available to CSP servers without any runtime communication with Intel. The Provisioning Certification Enclave uses the PCK to certify Attestation Keys created by Quoting Enclaves on the platform.

Attestation Services, customers, or other challengers of the platform can request Quotes from the platform, which now have a certificate chain back to Intel.

The PCK is derived from the Intel SGX Provisioning Root key, which is a per-part unique value present in HW. Each processor has a Provisioning Root Key and a Sealing Root Key. Intel uses its knowledge of the Provisioning Root Key to derive the socket's PCK and issues a PCK certificate for the processor.

On a multi-socket platform (Figure 3), during Platform Establishment, platform root keys are generated randomly. Intel HW provides a platform manifest containing the platform root keys along with the information on the sockets that participated in establishing this platform root key, Different sockets come together in the platform when the customer deploys the platform in their data center. A registration step is required for the Registration Service to be able to derive the keys and this platform will use and in turn issue certificates for them.

The Registration Service evaluates the sockets and configuration of the platform present in the manifest. If the Registration Service approves of the platform as a trustworthy Intel SGX environment, it uses copies of the Platform Provisioning Root Keys enclosed in the manifest to create certificates for the PCKs derived from the platform's Provisioning Root Key. Once created, the PCK certificates are delivered to the PCK Certificate Service.

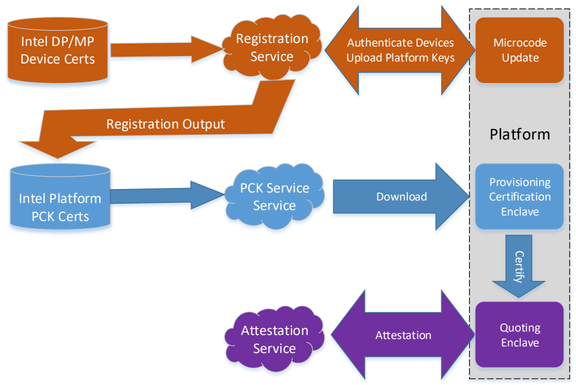

Figure 3: Multi-Socket Provisioning

Since the Platform Provisioning Root Keys are exposed to the Registration Server, an Intel SGX platform can only have one Registration Server for a platform at a time. Otherwise there would be a requirement to attest to both Registration Servers. The only way to change Registration Servers is to delete the platform keys and create a new instance that can be registered with the different Registration Service.

2.2.3 Key Exchange

Platform establishment requires secure communication between sockets before inter-socket HW encryption over UPI is enabled.

To establish these connections, sockets use unprotected memory as a channel to conduct signed Diffie-Hellman based key negotiations using their PRKs. The resulting keys are the long-lived Master Comms Keys. All sockets negotiate keys with their adjacent sockets and a BIOS appointed coordinator socket called the Master Socket. Some steps are taken by the master socket on behalf of the entire platform.

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.

If inconsistencies are found, Intel SGX is not enabled.

3 Performant Memory Protections

3.1 Memory Protection Performance

In order to expand the amount of memory covered by any one Memory Encryption Engine (MEE) [4], there are two methods which can be used to increase the size of memory protected.

1. Increase the amount of on-die storage used for the top-level anti-replay counters
2. Add more levels to the anti-replay tree

Expanding the amount of on-die storage only increases the amount of protected memory linearly. Adding a level on the other hand increases the protected memory by a factor of 8 (using the current design). To support large memory EPC sizes, this consumption of memory is onerous.

Numerous studies have shown that workloads that make significant use of Intel SGX memory have their performance impacted. The reason for this is the additional bandwidth required to fetch the various levels of the integrity tree protecting the Intel SGX memory.

While caching and other techniques can be used to address some of this performance loss, it was concluded that it was best to take a different approach to multi-socket memory in order to significantly boost the performance of Intel SGX on these types of platforms. Removing the additional memory access for the integrity tree reduces the performance impact to a small latency on each memory operation due to encryption.

The next few sections discuss the security benefits of using an integrity replay tree and what additional changes are required in order to ensure critical benefits are still maintained with the new approach.

3.2 Physical Memory Protections are Orthogonal from Access Controls

Implementation of Intel SGX only relies on the use of an inline memory encryption engine to provide protection to code/data as it leaves the CPU socket. It does not use encryption to provide separation between enclaves and regular memory (when resident in caches for instance) nor does it rely on encryption to provide separation between enclaves - they are all encrypted using the same key. The Intel SGX architecture implements several access controls, which help to prevent unauthorized access to cache lines that are resident inside the socket in decrypted form. This is designed to ensure only the specific enclave the page has been assigned to can access the cache lines; other enclaves and untrusted system SW cannot access those lines. Changing the physical memory protection scheme does not change these access controls.

We should also note that Intel SGX was not designed with only a single form of memory protection in mind. Indeed, if you look at how the CPU enumerates the protected memory regions through CPUID.SGX_LEAF (EAX=12h), then you can see that there is sufficient flexibility built in.

3.2.1 What software attacks does the MEE provide protection against?

While the MEE provides enhanced protections against hardware-based attacks on memory, its architecture also provides additional protections in the SW domain as shown in Table 1.

Research was also placed into looking at providing a regular form of memory encryption known as Intel® Total Memory Encryption (Intel® TME) and adding a separate key for the Intel SGX memory range. This would resolve a number of these issues particularly the Reset and EPC Reclaim based attacks. The largest remaining concern would be memory aliasing issues as these would allow simple SW based replay and integrity attacks.

3.3 Memory Aliasing

Aliasing occurs when two addresses can map to a single physical data location. In some cases, like page tables, this is entirely allowable, and the Intel Architecture relies on this trick to allow remapping of operating system pages in application address spaces. However, when dealing with security, an alias could be used to bypass the Intel SGX memory range checks that occur as part of the access control mechanisms used for enforcing isolation of enclave memory from other software.

Attack	Description	Resulting loss
Reset	The attacker forces a reboot of the platform and forces the previous memory range protections to be dropped. Secrets can now be retrieved from uninitialized memory.	Confidentiality
Aliasing/ EPC Replay	Attacker configures a second system address to map to a protected memory location. Ability to replay memory of arbitrary data allows code injection into enclave.	Execution Control / Confidentiality
EPC Reclaim	If platform has ability to tear down protections post use (without reset) EPC protected secrets would be exposed.	Confidentiality
DIMM Config. Attacks	SW attacks DIMM configuration settings to prevent writes from becoming persistent.	Integrity / Confidentiality

Table 1: MEE Attack Protection

The memory encryption engine with its integrity/anti-replay scheme caught changes to Intel SGX memory pages that may have occurred through an alias when the line was correctly consumed. The Intel TME engine does not have these protections, so actions need to be taken to prevent SW aliases in the Intel SGX memory region.

Given the complexity of server platforms, there are several sources of aliasing. These include memory configuration architecture itself and a Reliability Availability and Serviceability (RAS) feature known as sparing.

There are two types of alias that could occur:

1. Aliasing within EPC region (inside-in)
2. Aliasing from outside into the EPC region (outside-in)

3.3.1 Inside-In Aliasing

In the case of inside-in, two system addresses (SA3 & SA4 in Figure 4) in the Intel SGX protected region of memory alias to the same physical DRAM location. This allows a malicious enclave running in the protected region to bypass enclave ownership checks to gain access to any other enclave resident in the EPC page affected.

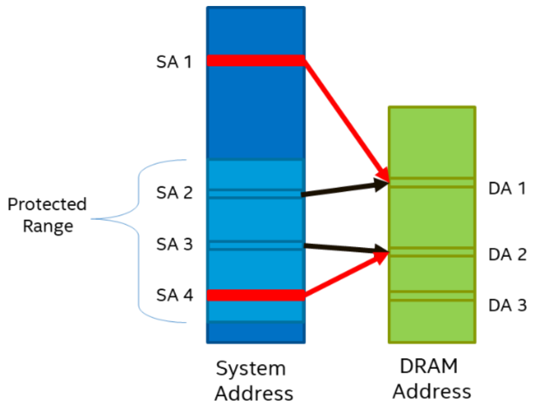

Figure 4: Aliasing

3.3.2 Outside-In Aliasing

In the case of outside-in aliasing a system address (SA1) outside the protected region memory aliases with another system address (SA2) to the same region of protected memory, effectively bypassing the Intel SGX EPC range check protections. This case can be used by untrusted software outside the enclave to manipulate any enclave resident in the EPC page affected.

Eliminating such aliases are therefore critical to ensuring that software integrity and anti-replay protections.

3.4 Preventing Aliasing

3.4.1 Outside PRM → Inside PRM

Memory used in servers uses a special form of memory that has extra storage for metadata which are used for features which improve the Reliability, Availability and Serviceability of the platform. This is typically known as Error Correct Code memory (ECC-memory). To resolve the outside-in aliasing issue, there is a need to repurpose one of the bits in the DRAM ECC metadata to indicate whether the line previously written was an Intel SGX specific line.

In the cases where a regular memory access is used to read a line that was previously written via secure path, a fixed value is returned. This would prevent attacks which rely on knowing just cipher text values. In the case where a secure memory read encounters a line was previously written in insecure mode, the system determines that the enclave is under attack. This will result in the disabling of Intel SGX. Note that the HW checks for outside-in aliases is performed after RAS ECC detection and correction. This ensures that DRAM soft and hard errors corrected by ECC will not lead to disabling of Intel SGX.

Read Secure	Written Secure	Returns
No	No	Cache line
No	Yes	Fixed Value
Yes	No	Fixed Value & Poison
Yes	Yes	Cache line

Table 2: Detecting Outside-In Attacks

3.4.2 Inside PRM → Inside PRM

Unfortunately, there is no hardware that can be added to support the run-time detection of inside-in aliases. Instead, during system boot, all the locations inside the Intel SGX EPC need to be checked to ensure they do not overwrite each other. Conveniently a piece of trusted FW runs during the Intel SGX initialization process which undertakes certain memory configuration and lock checks before Intel SGX can be enabled. This FW is used to perform checks to ensure there are no aliases within the PRMRR region.

3.4.3 Comparison of Memory Protections

In summary, you can now see with the changes that have been put in place for datacenter security, SW adversaries remain outside the trust boundary for Intel SGX whilst balancing performance required for many scalable server workloads.

Protect Memory from	Attack Vector	Client SGX	Scalable SGX
Loss of Confidentiality	SW	Yes	Yes
Loss of Integrity		Yes	Yes
Anti-Replay		Yes	Yes
Loss of Confidentiality	HW	Yes	Yes*
Loss of Integrity		Yes	No

Anti-Replay		Yes	No

Table 3: Memory Protections Comparison

*The final thing to note is that Scalable SGX for data center use TME which relies on AES-XTS mode for confidentiality. The cryptographic scheme used can only mitigate a class of HW attacks where the adversary can only see the cipher text once and not while the system is changing the data.

3.4.4 Coherent Memory Protections

Multi-socket Intel Xeon Processor-based servers use a high-speed interconnect for keeping memory coherent between the sockets. While this link is proprietary, it is exposed externally on the socket and is therefore open to attack.

4 Memory Coherency Architecture Overview

In Figure 5 you can see a basic block diagram of the major components involved in memory coherency in a two-socket system with a single Intel® Ultra Path Interconnect (UPI) Link between them.

When a core on the first socket is referencing a memory location and it fails to locate that line in its local cache, the reference is passed on to the local Caching Agent. These agents understand which physical memory locations are attached to which CPU socket (via BIOS configuration process) and forwards the request to the relevant UPI interconnect. Memory locations that are considered secure are sent with a secure attribute set. The UPI link agent forwards requests with the secure attribute to UPI Crypto Engine (UCE) for protection prior to transmission. Non-secure requests are forwarded in plain text to the appropriate queue for transmission. UCE will provide confidentiality, integrity and replay protection for the secure requests and data using an AES counter-mode based scheme before transmission.

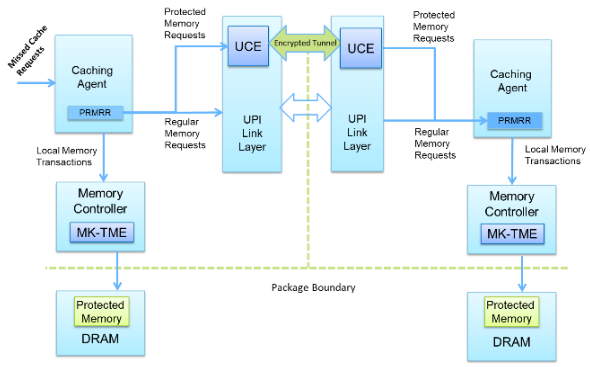

Figure 5: Memory Coherence Architecture

Upon receipt in the receiving socket, the UPI agent determines whether the packets making up the message need to be processed by UCE. After integrity check and decryption, the UCE will forward the message to its local Caching Agent with the secure attribute set. The local caching agent checks that memory requests for Intel SGX memory have the secure attribute set, before issuing memory references to the local memory controller for the secure memory.

Memory location cache lines are returned to the remote caching agent before being forwarded to the UPI for secure transmission back to the request originating socket.

4.1 Protecting Against Memory Coherency Attacks

Table 4 identifies the attacks that can occur on the memory coherency architecture.

Threat	Description
Eavesdropping	Attacker passively sniffs packets for Intel SGX memory resulting in confidentiality loss
Tampering	Attacker modifies Intel SGX packets to break the coherency model for the Intel SGX memory
Replay	Attacker forges/replays a Intel SGX packet to break the coherency model for the Intel SGX memory
Directory Corruption	Attacker corrupts the directory bits in DRAM corresponding to Intel SGX memory
Misconfiguration	Attacker manipulates configuration assets that affect the security of the Intel SGX memory.

Table 4: Intel SGX Memory Coherency Threats

To mitigate a number of these threats (i.e. Eavesdropping, Tampering and Replay), selecting the correct cryptographic protections is very important. In this case, a counter-mode based encryption scheme with integrity should suffice.

To prevent misconfiguration attacks each socket must check these configurations for consistency and lock the configuration before Intel SGX can be enabled.

5 Platform Configuration

The architectural enhancements described so far support the creation of a single key and the enabling of memory communications encryption. This section covers the configuration aspects of the architecture which need to occur during platform set-up.

The process overall is fairly simple. BIOS determines the enablement scenario the platform is in during boot and prepares the platform accordingly. Then, it triggers an Intel FW module that runs on each CPU socket. The FW module performs the following sequence:

3. Checks Platform configuration consistency and locked state across sockets
4. Calculates/decrypts and programs keys in HW used for identity, data protection, bus protection, and memory protection
5. Performs Memory Alias checking for Inside-in memory scenario

5.1 Supported Configuration Scenarios

The Intel FW supports the following main scenarios:

1. Establish New Platform
2. Reboot Old Platform
3. Add Socket to Platform

5.1.1 Establishing a New Platform

During Platform Establishment, a new platform instance is created along with a corresponding set of platform keys and material necessary to register the newly created instance.

Sockets start by establishing authenticated communication channels over unprotected memory. These channels are used to share the new platform keys. The sockets create a signed manifest of the pairings and platform keys.

Each socket creates an encrypted key structure containing a copy of the platform keys, the communication keys from each pairing, and platform configurations that are not permitted to change.

For each UPI link, sockets program link encryption keys and the platform is ready to boot with Intel SGX. New key structures and manifests are returned to BIOS for future use.

5.1.2 Rebooting a Platform

If the BIOS determines that every socket has a key structure with keys for the currently loaded microcode update, the BIOS instructs Microcode Update to conduct a standard Intel SGX boot.

During a standard boot, previously created platform root keys are decrypted and loaded; sessions between sockets are re-established; UPI UCE and memory encryption engine are configured with encryption keys; and Intel SGX is ready to be activated.

5.1.3 Adding a New Socket to a Platform

Once the platform instance has been established and secrets are provisioned, unverified devices must not have access to that instance's platform root keys.

Attestations on the platform are rooted in the Registration Service-issued PCK certificate. This is an assertion that the platform composition was evaluated and approved. The same Registration Service must ensure that that assertion remains true if the platform is using those keys.

When a new socket is found, BIOS creates an Add Request identifying the platform and new socket. System SW sends the request to the Registration Service. If the request is approved, the service creates a Platform Membership Certificate, which other sockets in the system will use to authenticate the new socket. After the next reset, existing socket(s) verify the certificate and share the previously established platform keys with the new socket.

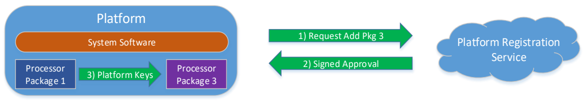

Figure 6: Flow for Adding New Socket

6 Summary

In this paper, the essential elements were described for Intel SGX and the data center architecture to support multi CPU-socket systems such as Intel Xeon Scalable Processor-based platforms, how it scales to over 1TB of protected memory and has minimal impact on performance. A method was described for creating a single identity for an Intel SGX platform from CPU HW that is manufactured separately through the creation of a registration process. In addition, a demonstration of how Intel SGX related memory coherency traffic is designed to be protected as it is transported between CPU sockets on high speed socket interconnect. Finally, it was demonstrated how memory protection from SW can be maintained and performance increased by adding additional checks in the memory architecture using ECC metadata.

Further information about Intel SGX can be found at http://software.intel.com/trustsgx .

7 References

[1] F. Mckeen, I. Alexandrovich, A. Berenzon, C. Rozas, H. Shafi, V. Shanbhogue and U. Savagaonkar, "Innovative Instructions and Software Model for Isolated Execution," in Hardware and Architectural Support for Security and Privacy, 2013.

[2] T. Chai-Che, D. E. Porter and M. Vij, "Graphene-SGX: A Practical Library OS for Unmodified Applications on SGX," in USENIX Annual Technical Conference, Santa Clara, CA, 2017.

[3] A. Baumann, M. Pienado and G. Hunt, "Shielding Applications from an Untrusted Cloud with Haven," in

11th USENIX Symposium on Operating Systems Design and Implementation, Broomfield, CO, 2014.

[4] S. Gueron, "A Memory Encryption Engine Suitable for General Purpose Processors," Intel, 2016.

[5] V. Scarlata, S. Johnson, J. Beaney and P. Zmijewski, "Supporting Third Party Attestation for Intel® SGX with Intel® Data Center Attestation Primitives," 2018. [Online]. Available: https://software.intel.com/sites/default/files/managed/f1/b8/intel-sgx-support-for-third-party-attestation.pdf.

[6] I. Anati, S. Gueron, S. P. Johnson and V. R. Scarlata, "Innovative Instructions for Attestation and Sealing," 2013. [Online]. Available: https://software.intel.com/en-us/articles/innovative-technology-for-cpu-based-attestation-and-sealing.

[7] E. Brickell and J. Li, "Enhanced privacy ID from bilinear pairing for Hardware Authentication and Attestation," International Journal for Information Privacy, Security and Integrity, vol. 1, no. 1, pp. 3-33, 2011.

[8] Trusted Computing Group, "Trusted Platform Module Main Specification (TPM1.2)," March 2011. [Online]. Available: http://www.trustedcomputinggroup.org/resources/tpm_main_specification.

[9] S. Johnson, V. Scarlata, C. Rozas, E. Brickell and F. Mckeen, "Intel(r) Software Guard Extensions: EPID Provisioning and Attestation Services," March 2016. [Online]. Available: https://software.intel.com/sites/default/files/managed/57/0e/ww10-2016-sgx-provisioning-and-attestation-final.pdf.

[10] Intel, "Intel(r) 64 and IA-32 Architectures Software Developers Reference Manual," May 2018. [Online]. Available: http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html.

[11] Intel, "Attestation Service for Intel® Software Guard Extensions (Intel® SGX): API Documentation," 2018. [Online]. Available: https://software.intel.com/sites/default/files/managed/7e/3b/ias-api-spec.pdf.

[12] Intel, "Performance Considerations for Intel(r) Software Guard Extensions Applications," 2018. [Online]. Available: https://software.intel.com/sites/default/files/managed/09/37/Intel-SGX-Performance-Considerations.pdf.