# Exploring Intel Software Guard Extensions (SGX)

Date: March 18, 2016

**Investigators:**

1. Praveen Keshavamurthy
   Masters in Computer Science
   Email: keshavamurthy.p@husky.neu.edu

2. Aboobacker Rizwan Payapura
   Masters in Computer Science
   Email: rizwan@ccs.neu.edu

**General Field:** The project falls in the realms of Cloud Security, where the resources are shared between many virtual machines. The adversary model which SGX targets is complete compromise of operating system, which is indeed a main threat model for cloud hosted applications.

**Keywords:** cloud security, software guard extensions(SGX), isolated containers, hardware security feature, resource sharing

**Project Description:** Intel SGX, which provides hardware security features is a major step towards a better security in cloud computing, but it is mostly untested by researchers. So the exact limitations and capabilities of this new technology is still not very clear. We will be exploring Intel SGX by adapting SQLiteCipher DB to run inside the secure container provided by Intel SGX technology.

**Project Responsibilities:**

1. Praveen Keshavamurthy: Lead Designer, Software Developer
2. Aboobacker Rizwan Payapura: Lead Researcher, Software Developer

**Total Budget:** $16200

**Deliverables:**
- SQLiteCipher Library adapted to run with Intel SGX.
- Analysis of the overhead due to Intel SGX by checking SQLiteCipher DB benchmarking measures.
- Guidelines  to develop an application with Intel SGX.

## Executive Summary
## Trusted Execution Environment – Intel Software Guard Extensions (SGX)

Secure remote computation, offloading processing to a remote computer owned and maintained by an untrusted party with some integrity and privacy guarantees, has remained an unsolved problem. This is mainly due to the hierarchical security model that aims only to protect the privileged code from untrusted code and does not fare well to protect user data from access by privileged code. Although complete homomorphic encryption solves the problem for a limited family of computations, it has an impractical performance overhead. Intel SGX is the latest technology which aims to solve the secure remote computation problem by leveraging trusted processors on remote computers.

Intel SGX is an architecture extension designed to increase the security of software through an “inverse sandbox” mechanism. At its root, Intel SGX is a set of new CPU instructions, using which legitimate software can be sealed inside an enclave (secured memory) and protected from attack by the malware, irrespective of the privilege level of the latter. It also provides a novel hardware assisted mechanism to allow secure remote attestation and sealing to application software. Intel SGX technology is the need of the hour for applications related to Data Rights Management (DRMs) and in designing a novel architecture for trusted cloud. As Intel SGX was available to select group of researchers until recently, there is a lack of current research and programs that evaluate the technology with respect to performance impacts on existing state-of-the-art solutions.

In this project, we dissect the security features of Intel SGX by adapting SQLiteCipher Library to use Intel SGX technology. We will benchmark the SQLite DB performance overhead due to the use of Intel SGX in secure software execution path. We will also come up with Intel SGX application development guideline document based on the experience we obtain by adapting SQLiteCipher library.

## Problem Statement

There is a lot of trust involved in a cloud infrastructure. In a security perspective 'trust' is a bad thing. But due to the complexity and performance issues many protocol is designed based on trust. For instance creating an app and hosting it in the cloud platform involves many level of trusts. The app developer needs to trust the Operating System (OS) in the cloud (which have high privilege access), the cloud provider, and worst of all other applications hosted in the same platform. Since the OS kernel operates in ring 0 (high privileged), the attacker can get a high privileged access and thus can manipulate the private data of other applications hosted in that platform, which operates in lower privileged user space (ring 3).

Intel Software Guard Extensions (Intel SGX) is designed for this purpose through an inverse sandbox mechanism. It solves the above problem by preserving the confidentiality and integrity of sensitive code and data of the applications without disrupting the ability of legitimate system software to schedule and manage the use of platform resources. Since applications create their own secure containers and the processor makes sure that the access to these containers is limited to only that application, the trusted components of the application are protected. Intel SGX provides a feature which allows remote user to verify that the application is set up securely on the untrusted infrastructure by  producing a signed attestation (rooted in the processor) that includes the measurement and other certification.

As part of this project, we are specifically trying to answer following key questions related to Intel SGX technology.

- What applications are suitable for using Intel SGX technology
- Guidelines for the design decisions to port an application to use Intel SGX
- How does Intel SGX impact the application performance

We will be adapting SQLCipher to use Intel SGX and based on the results, we are planning to come up with the answers to the above questions.

Why SQLCipher? SQLCipher is a specialized build of the SQLite database that performs transparent and on-the-fly encryption. Using SQLCipher, an application uses the standard SQLite API to manipulate tables using SQL. Behind the scenes the library silently manages the security aspects, making sure that data pages are encrypted and decrypted as they are written to and read from storage. Also the SQLite memory footprint ceils at 3 MB, irrespective of the size of the database file being accessed. This is in line with the Intel SGX enclave memory limitations i.e., max heap size per enclave is 96 MB and total Processor Reserved Memory ( PRM ) is 128 MB. All these features make SQLCipher an appropriate application to explore Intel SGX features.

## Work Accomplished

- Understanding the architecture of Intel SGX. As part of this activity, we have explored the following internal implementation of SGX:
    - SGX physical memory organization and layout using Enclave Page Cache (EPC).
    - Life cycle of an SGX Enclave and SGX thread.
    - EPC page eviction using version arrays.
    - SGX enclave measurement and versioning support.
    - Certificate based enclave identity using Signature Structures (SIGSTRUCT).
    - SGX software attestation using local and remote attestation with EGETKEY key sharing mechanism.

- Explored Intel SGX Software development kit (SDK) libraries to manage the enclaves.
    - sgx_t* - trusted libraries - Libraries used within enclave code
        - sgx_trts.lib - Intel® SGX internals - rand functions, dynamic enclave checks, instruction exception handlers
        - sgx_tstd.lib; sgx_tstdcxx.lib - standard C & C++ functionalities including STL support
        - sgx_tcrypto_opt.lib - Crypto functionality - AES-GCM, DH, ECC
        - sgx_tkey_exchange.lib
        - sgx_tservice.lib - Uses crypto and key exchange libraries. Supports APIs for data seal/unseal, trusted Architectural enclaves support, Elliptic Curve Diffie-Hellman (EC DH) library
    - sgx_u*- untrusted libraries - Libraries to be used from untrusted app code.
        - sgx_urts.lib - Provides functionality for applications to manage enclaves
        - sgx_uae_service.lib - Provides both enclaves and untrusted applications access to services provided by the AEs
        - sgx_ukey_exchange.lib - Untrusted key exchange library
        - sgx_status.dll - Provides functionality for applications to register Enclave Signing Key White List Certificate Chain.
        - sgx_capable.dll - Provides functionality for applications to check if the client platform is enabled for Intel SGX or to enable the Intel SGX device.

- Created a sample program which contains both trusted part (running inside the enclave) and untrusted part which can load the enclave and call the trusted code inside the enclave. This gave us much better insight on how to design the application for Intel SGX processor, by identifying trusted and the untrusted part of the program and using the signed enclave to run the trusted part code. Also we were able to explore the proxy and bridge functions which are generated for the ECALL/OCALL functions.

- Explored enclave configuration xml file to specify different configurable values like heap size, stack size, debugging mode etc. This helped us to find the limitations of the memory allocations in Intel SGX.
- Created Enclave definition language (.edl) file to specify the trusted and untrusted components of the program.
    - Explored tools available in the SDK for signing the enclave i.e., sgx_sign.exe, loading the signed enclave in the untrusted application, finding the maximum heap and stack memory used by an enclave.
    - Explored sgx_edge8r.exe tool - this is used to generate proxy/bridge functions based on enclave edl file.

- We were able to identify the following guidelines for using Intel SGX:
    - The SGX DLL should not depend on any runtime dlls. It can be linked to static SGX library, if required.
    - Max run time memory reserved for SGX enclaves is 128 MB. This brings in a restriction on number of active enclaves at any point of time in the target.
    - Max heap size per enclave is limited to 96MB. This brings in a limitation on the dynamic memory allocation to the enclaves.
    - The Intel SGX SDK supports only C and C++ language.
    - No support for Linux operating system. Platform software and SGX tools are available only for Windows and no SDK support for linux.

- In search for finding a cloud usecase for Intel SGX with all the above restrictions, we explored the following usecases in depth before finalizing on SQLite with SQLCipher extension:
    - Adapt Cloud Infrastructures to support Intel SGX.
        - Openstack: It is developed using python, which is not supported by Intel SGX SDK.
        - Elastic Building Block Runtime (EbbRT) library: Used in linux environment, which is not supported by Intel SGX.

    - Different database (DB) as these will be the building blocks for any application development using SGX in future.
        - In-memory DB (NoSQL Database): Not applicable as the reserved memory limit is 128 MB for Intel SGX.
        - Graph DB (Neo4J): Implemented in java, hence cannot be ported to SGX.
        - SQLite (SQL Database): This is lightweight library and contains an extension called SQLCipher which enables the database file to be encrypted. It is developed in C and can be built in Windows for our implementation. Hence this is a feasible usecase.

- TOR Network: Not a cloud use case. But it can be ported to Intel SGX since it is developed in C and available for Windows. It has been verified using SGX simulation. Now we could verify with the real SGX hardware.

- Data Center Applications: We looked into following distributed large data graph processing applications:
    - Apache Giraph: Implemented using java and used in Linux. Hence not applicable.
    - GraphLab PowerLab: Used in Linux. Hence not applicable.

- Kernel-based Virtual Machine (KVM): It is possible to provide QEMU-KVM support for Intel SGX so that it enables the community to use Intel SGX as part of virtual machines. But this involves a steep learning curve and a very good understanding of KVM which makes it difficult to finish with the available time frame.

## Work Remaining

The final objective is to port SQLite with SQLCipher extension to Intel SGX so that the database can make use of the enclaves to process the unencrypted data in the DB to secure it from any adversary even in a compromised operating system. We have the following milestones to accomplish the above goal:

- SQLCipher uses paging to keep the memory footprint low and all the cipher codec is implemented as part of this paging context. We need to understand the paging architecture as this should be implemented as part of trusted enclave code.

- Identify the trusted and untrusted components in the SQLCipher code. This will help us make a decision on what code need to be running inside the enclave. Considering the space limitation of enclave, the code running inside enclave must be minimal. The code that handles the unencrypted data of the DB must be running inside the enclave.

- We will have following code artifacts at the end of this project
    - A signed enclave dll, sqlitecipher_signed.dll - The logic related to decrypting and encrypting the page, sql query evaluation on the decrypted page will be part of this library.
    - An untrusted sqlitecipher dll, sqlitecipher.dll, which will be dependent on the signed dll. All the SQLCipher application APIs, enclave creation/deletion code and rest of the code of current SQLCipher library will be part of this dll.

- SQLCipher uses MinGW (Minimalist GNU for Windows) for the build environment. The build scripts should be adapted to make use of SGX dependent libraries rather than the OS library.

- We are planning to measure and compare the performance of the database ported to Intel SGX based on the following benchmarks:
    - Total number of single-record read/write in a given interval (ops/sec).
    - Total number of bulk read/write in a given interval (ops/sec).
    - Load on CPU and disk while executing complex and simple query.