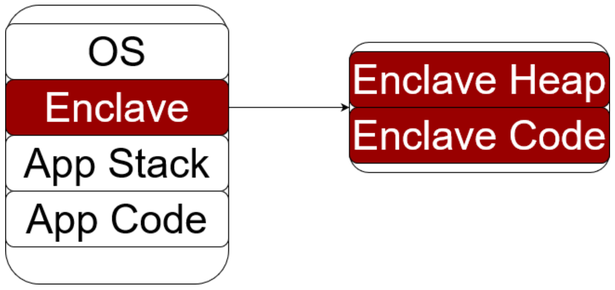
Intel SGX and Dynamic Memory Allocation

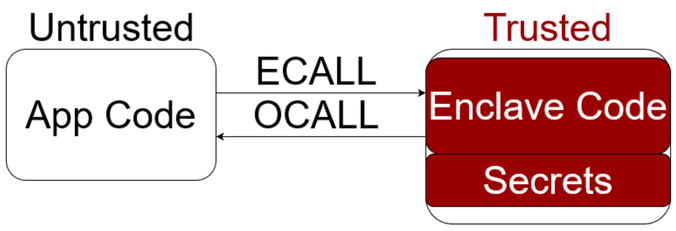
Edward DeMars
Computer Science
Indiana University 2021 UROC, Luddy School of Informatics Computing and Engineering, Research Mentor: Hongbo Chen

What is Intel SGX?

Intel Software Guard Extensions (SGX) is an instruction set extension that acts as a trusted execution environment (TEE) within a CPU. It works by creating enclaves, portions of encrypted memory that cannot be accessed from the outside except for in special controlled circumstances (e.g., by the trusted CPU) (Costan). This includes protecting the software from the operating system and programs running at higher privilege levels. The only way to access the information inside of an enclave, is for the enclave to let you access that information. This allows for the protection of data in use. Protection of data in storage and data in transit has been commonplace for years, but data in use has long been vulnerable, especially to privileged attackers. SGX helps to fix that. Since data cannot be accessed from outside the enclave, the data inside cannot be spied on while in use (Costan).



SGX programs are designed in two parts. 1) The trusted component, which contains the code to be run inside of the actual enclave that can access its secrets. 2) The untrusted component, which contains the code to instantiate the enclave, handle its interactions with the outside world, and anything else that does not need to be inside the enclave. The two can interact with each other through a series of programmer-defined functions known as ECALLs (untrusted to trusted) and OCALLs (trusted to untrusted). In general, the trusted component should be as small as possible, and its interactions with the untrusted portion kept to a minimum, in order to reduce the surface area vulnerable to attack. Enclaves are also attestable, both locally and remotely, allowing users to verify that they are communicating with a trusted party (Costan).



Why Use SGX?

As established previously, Intel SGX allows one to create a program that has a portion of itself encrypted, only able to be accessed by itself. Since SGX is hardware level and does not trust the operating system, hypervisor, or other programs, if the OS or other higher privilege program is compromised, the enclave is still secure. This provides benefits in a host of situations, but, due to how SGX handles remote attestation, has particular promise in remote computing, which often deals with sensitive information that the consumer does not want to be compromised. The prime example of this is in medical imaging, allowing doctors to have their patients' data analyzed on remote computers without risking their privacy. In addition to this, SGX is relatively easy to work with, providing programmers access to most tools they typically use, with the only unique burden being the separation of the program into two parts.

Dynamic Memory Allocation

Previously, the amount of memory allocated to an enclave was fixed at load time. Once the enclave was initialized, its memory was static in terms of size and page permissions, and this greatly limited enclave functionality. However, SGX 2 introduced dynamic memory allocation for enclaves, allowing them to be much more versatile (Xing). Now instead of declaring how much memory an enclave has from the beginning, developers set a max size for the heap and stack in the config file, along with other optional settings such as minimum sizes for the heap and stack or allowing the use of "reserved memory" for special functions such as just-in-time compilation.

SGX handles dynamic memory allocation like most other applications do. Developers can use methods such as malloc to dynamically allocate and deallocate memory in their programs, adding and removing items from the heap. SGX can also request additional memory for the heap (up to the defined max). It handles this like how most modern operating systems handle expanding the heap, with page faults (Xing).

1. Determine address of new EPC and exectue ENCLU[EACCEPT]

↓

2. Results in a #PF (page fault) causing SGX driver to commit a new EPC at address

↓

3. Enclave resumes at ENCLU[EACCEPT] which succeeds and accepts the new EPC

Example Program

To learn about, experiment with, and demonstrate Intel SGX and its dynamic memory management, I made a sample program in the guise of a school administration system, with the ability to add different classes and students that possessed classes. In this scenario, the trusted components were the student and class structures and related data, and the untrusted component was the user interface and related features, though I likely could have better separated this, as the only data that truly needed to be secret was the students' grades.

```
int new_student(int n)
{
    uint64_t page_size, aligned_size;
    page_size = getpagesize();
    aligned_size = ((n * sizeof(Student)) + page_size - 1) & ~(page_size - 1);
    printf("Aligned size %d\n", aligned_size);
    Student *address = (Student *)sgx_alloc_rsrv_mem(aligned_size);
```

Results and Analysis

While Intel SGX is an extremely useful tool, it has some serious drawbacks, even ignoring the many valid security and ethical concerns involved that are outside the scope of this project. By default, it is limited to C and C++, and while there are projects to remedy this, such as Apache Teaclave for Rust, it still can be a large drawback. Many of the frustrations I encountered came solely because of having to work in C, since I do not know C++ and did not have the time to learn it for this. In addition, while C can already be hard to debug, with gdb being hard to approach, SGX adds on top of this. Many bugs can come from going in and out of the enclave, which can be hard to find the root cause of. Due to SGX's unique nature, you cannot use normal debuggers, and I was never able to get sgx-gdb to work. It can also be difficult to figure out how one should split up their program between trusted and untrusted, though that would likely become easier with experience.

References

Costan, Victor, and Srinivas Devadas. "Intel sgx explained." IACR Cryptol. ePrint Arch. 2016.86 (2016): 1-118.

Xing, Bin Cedric, Mark Shanahan, and Rebekah Leslie-Hurd. "Intel® Software Guard Extensions (Intel® SGX) software support for dynamic memory allocation inside an enclave." Proceedings of the Hardware and Architectural Support for Security and Privacy 2016. 2016. 1-9.