DSP Lab

Secure Java Native Interface using Intel SGX

Group Members :

Clindo Devassy K
Subhadeep Manna

Guidance By:
Marcel Blöcher
Malte Viering

Intel SGX(Quick Look)

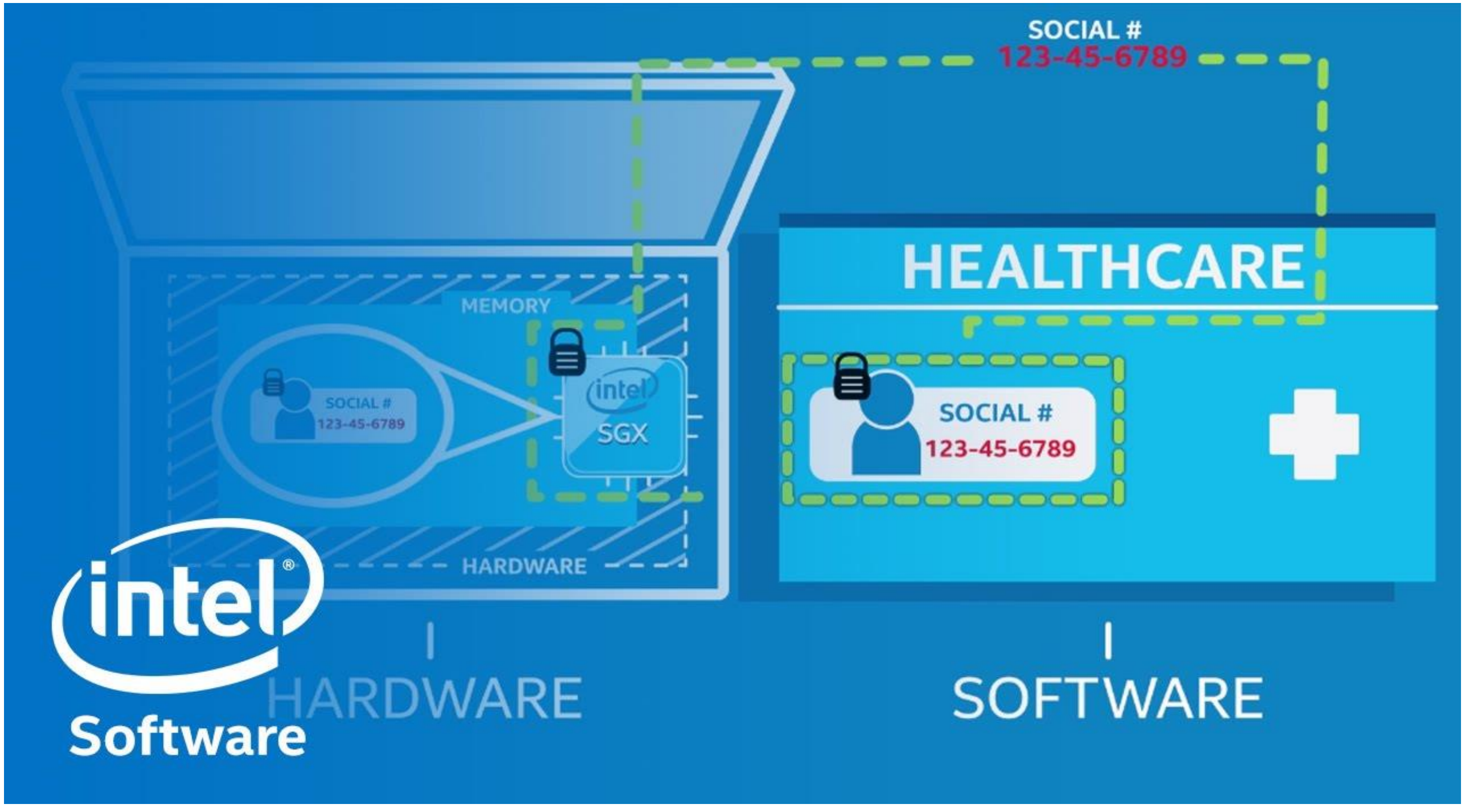
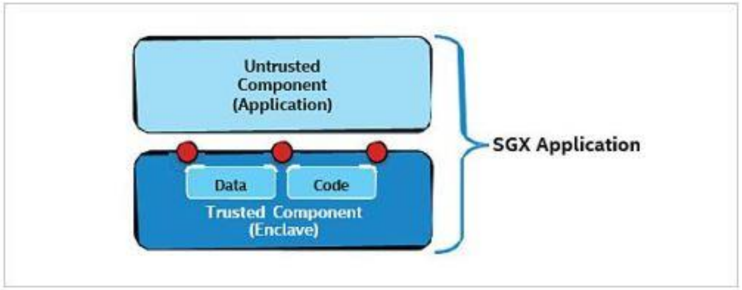
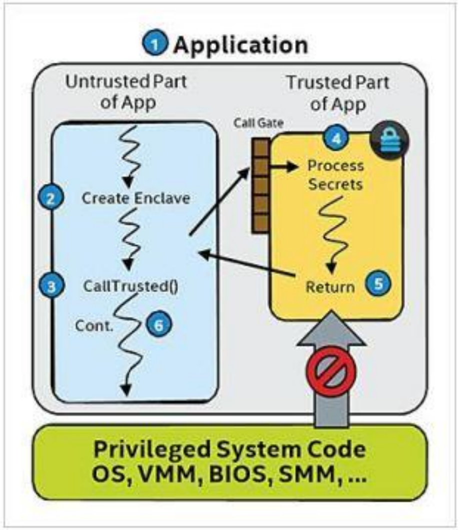


Image Source:https://software.intel.com/en-us/sgx/details

Securing App using Intel SGX (Quick Recap)

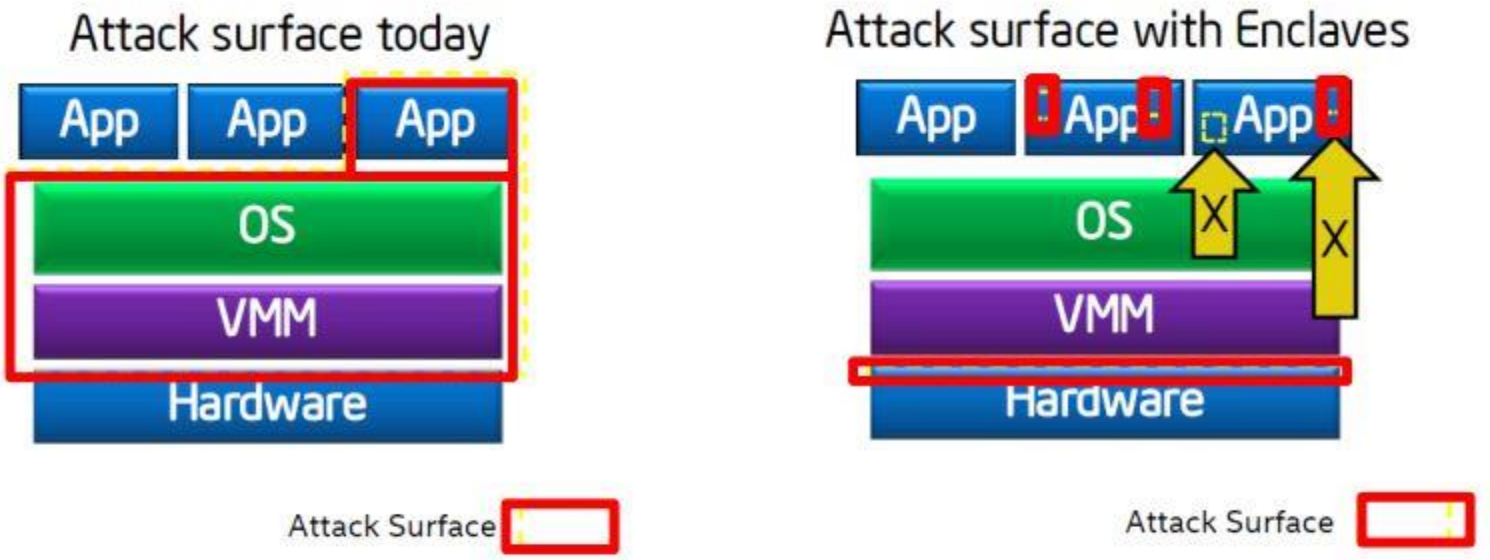

Untrusted Component (Application)
Data
Code
Trusted Component (Enclave)
SGX Application



- Application code can be put into an enclave.
- Intel® provides SGX Software Development Kit (SDK).
- Currently available only for C/C++.

Image Source:https://software.intel.com/en-us/sgx/details

Securing App using Intel SGX (Quick Recap)



- Application gains ability to defend its own secret.
- Reduced Attack Surface.

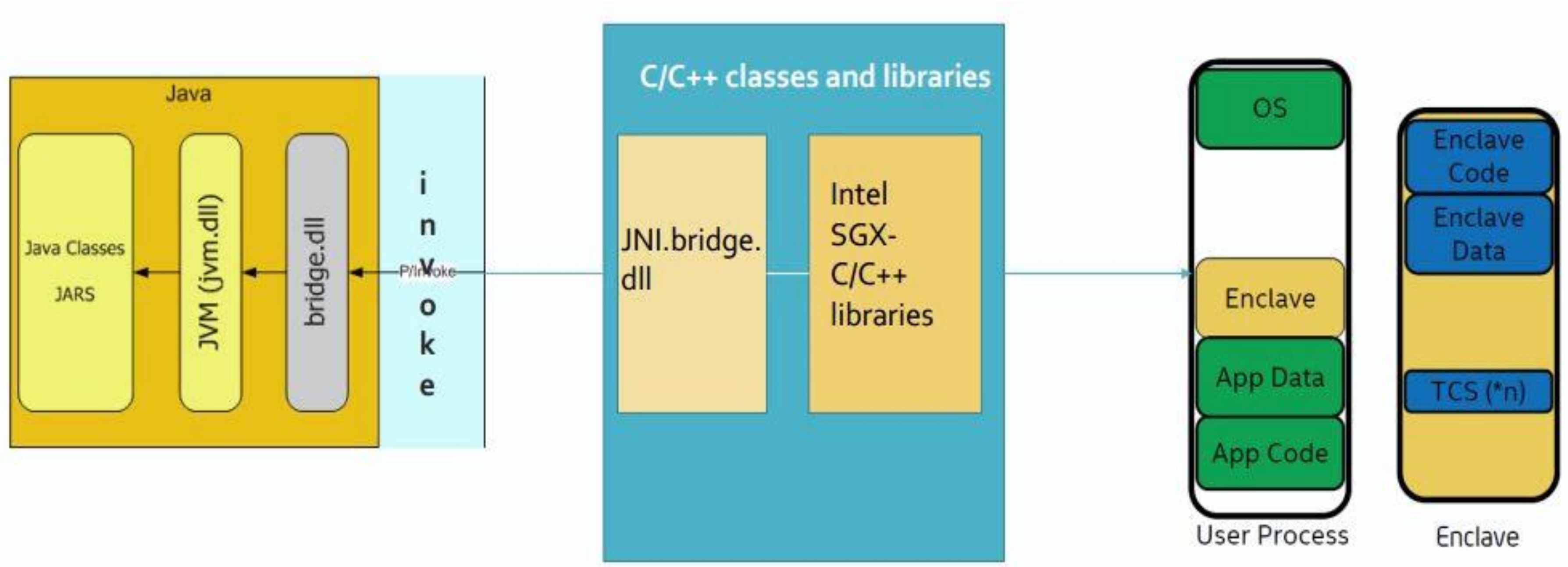
Image Source:https://software.intel.com/en-us/sgx/details

Java Client-Server App (Intel SGX and JNI)

Problem Statement

- To implement a server side Application using SGX and Java.
- It accepts arithmetic expressions(secrets) from Java clients.
- Puts them into the enclave.
- Evaluates them with the code residing in the enclave.
- Return the result the back to the client.

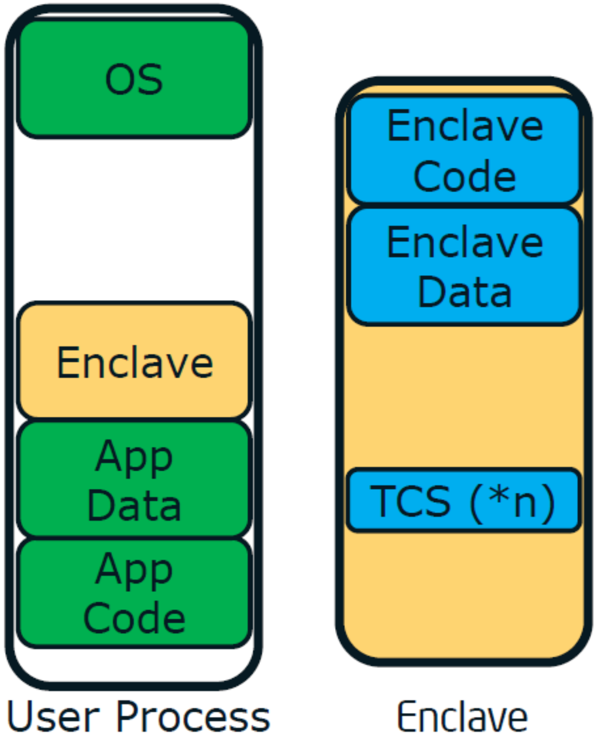
Java Implementation of Intel SGX



The Core of SGX Technology

Developer perspective of an SGX Enclave

Trusted execution environment embedded in a process



Intel® SGX generates a cryptographic log of all the build activities

- Content: Code, Data, Stack, Heap
- Location of Page within the enclave
- Security flags being used

Image Source:https://software.intel.com/en-us/sgx/details

The Challenge –
Provisioning Secrets to the Enclave

- An enclave is in the clear before instantiation.

- Secrets come from outside the enclave
  - Keys
  - Passwords
  - Sensitive data

- The enclave must be able to convince a 3rd party that it's trustworthy and can be provisioned with the secrets.(Trusted Computing)

- Subsequent runs should be able to use the secrets that have already been provisioned.

Developer perspective of an SGX Enclave

MRENCLAVE (“Enclave Identity”) is a 256-bit digest of the log represents the enclave’s software TCB

EREPORT: generates a cryptographic REPORT that binds MRENCLAVE to the target enclave’s REPORT KEY

EGETKEY provides the REPORT KEY to verify the REPORT

A Software TCB verifier should

- Securely obtain the enclave’s software TCB
- Securely obtain the expected enclave’s software TCB (Ex. Intel own Authentication Server)
- Compare the two values

Two ways Intel provides the functionality

Local Attestation.

The process by which one enclave attests its TCB to another enclave on the same platform.

Remote Attestation.

The process by which one enclave attests its TCB to another entity outside of the platform.

Development of a SGX Application.

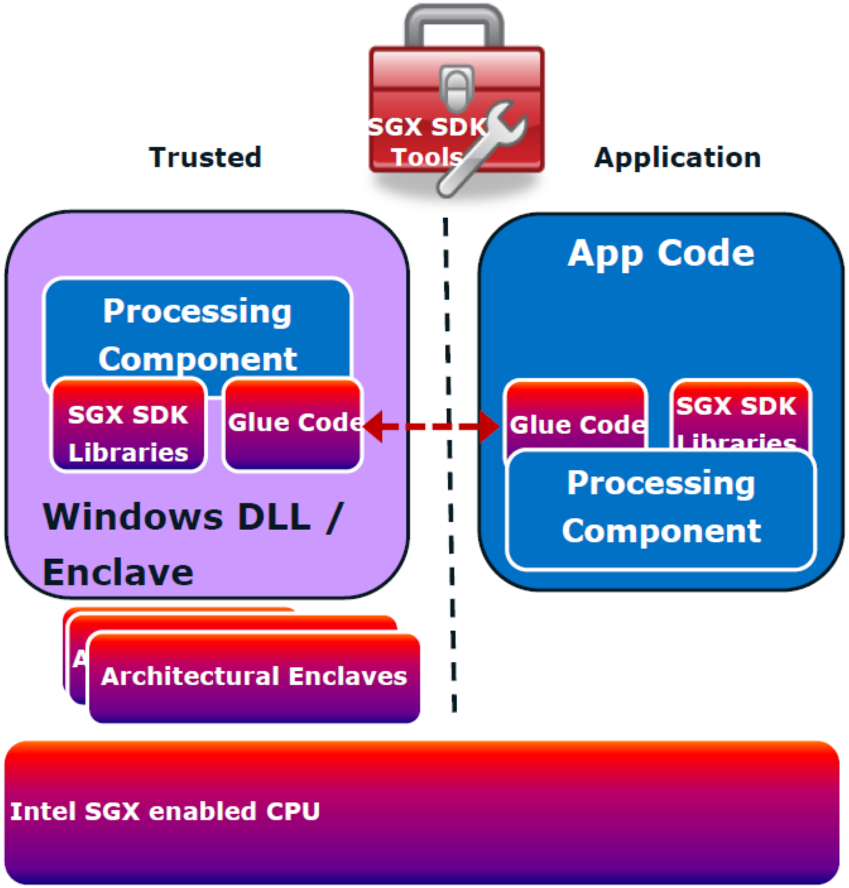


Image Source:https://software.intel.com/en-us/sgx/details

From a developers perspective:

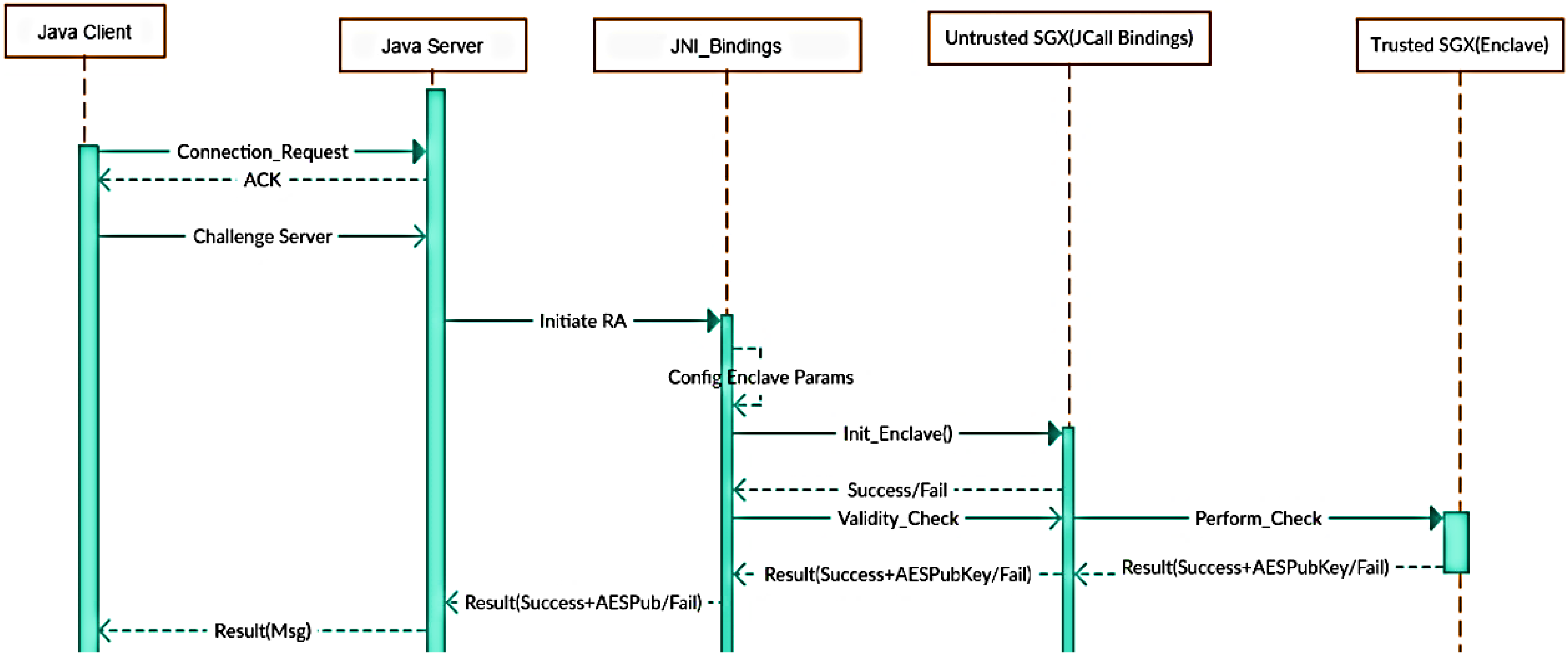
1. Create trusted and untrusted part of the application.
2. Configure enclave parameters.
3. Define calls to a enclave(ECALLS)
4. Define calls from enclave(OCALLS)
5. Initialize enclave(s).
6. Add data and secrets.

Putting Theory into Practice

+

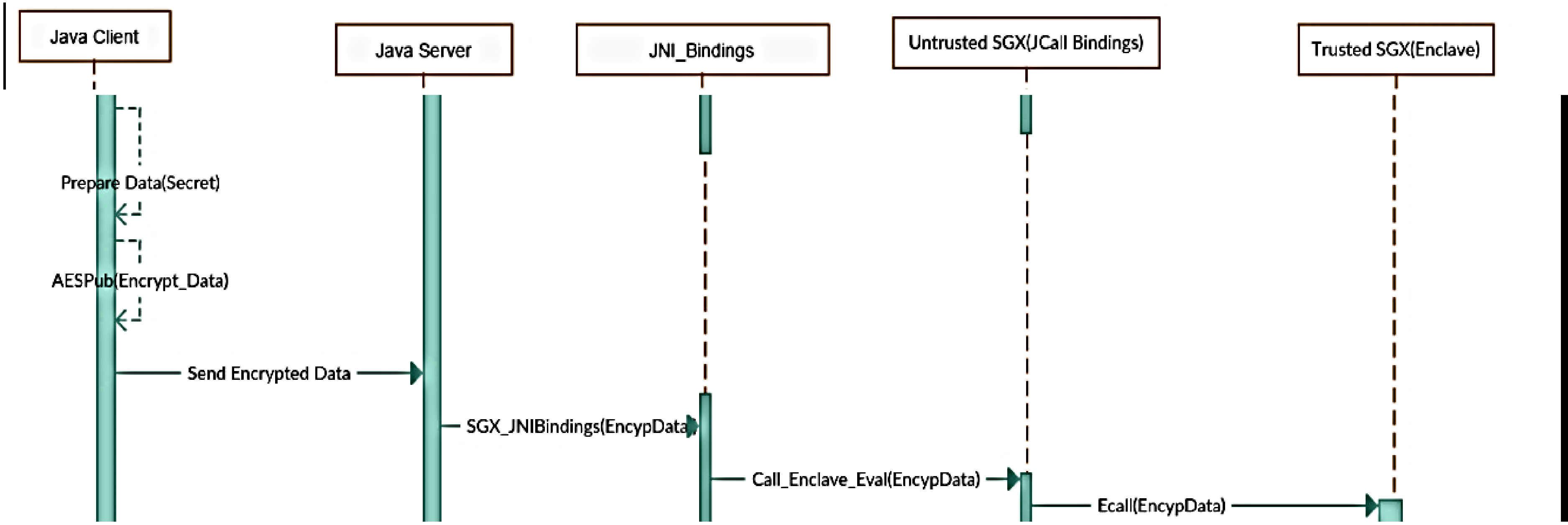
Implementation Sequence Diagram

1. Initialization of enclave and the key exchange with remote enclave



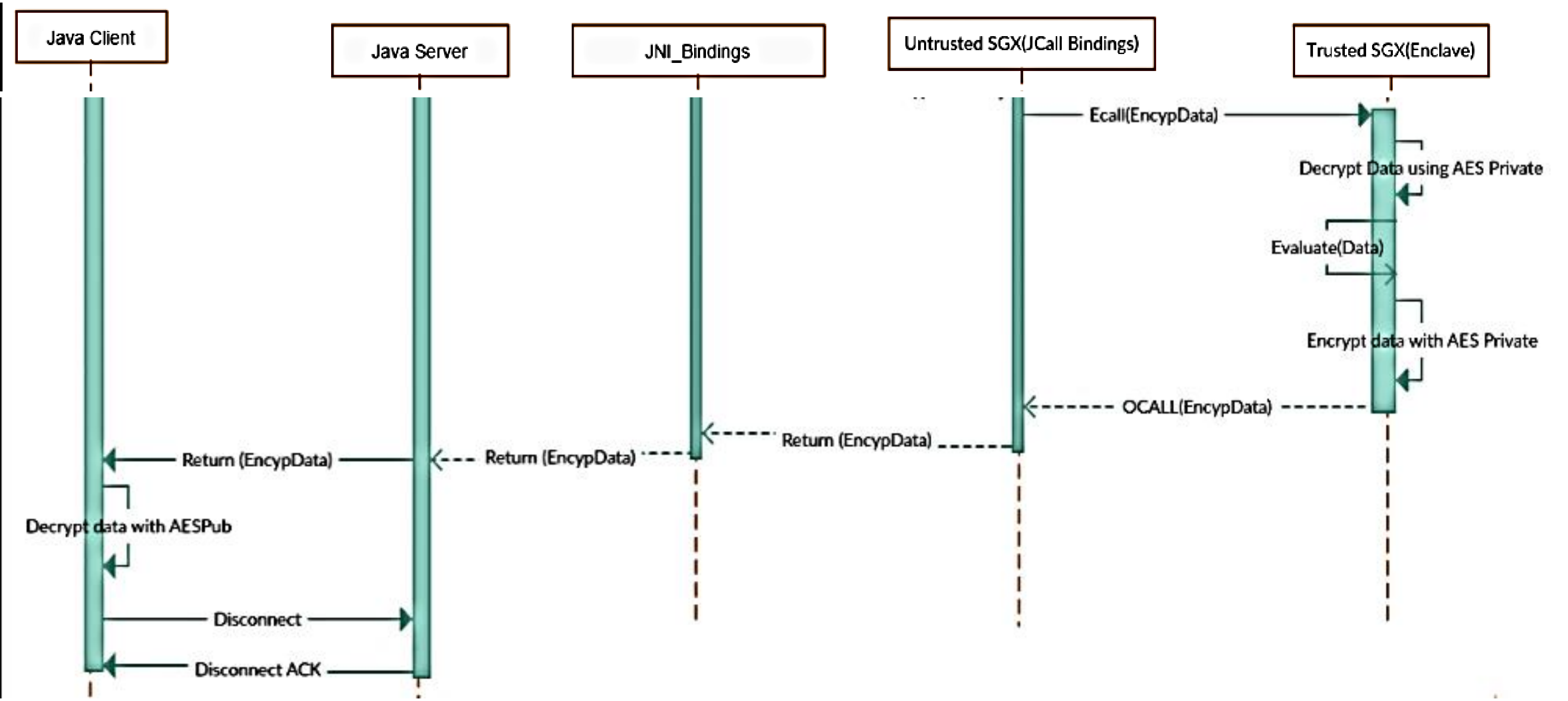
Implementation Sequence Diagram

2. Encryption and sending of data to remote enclave

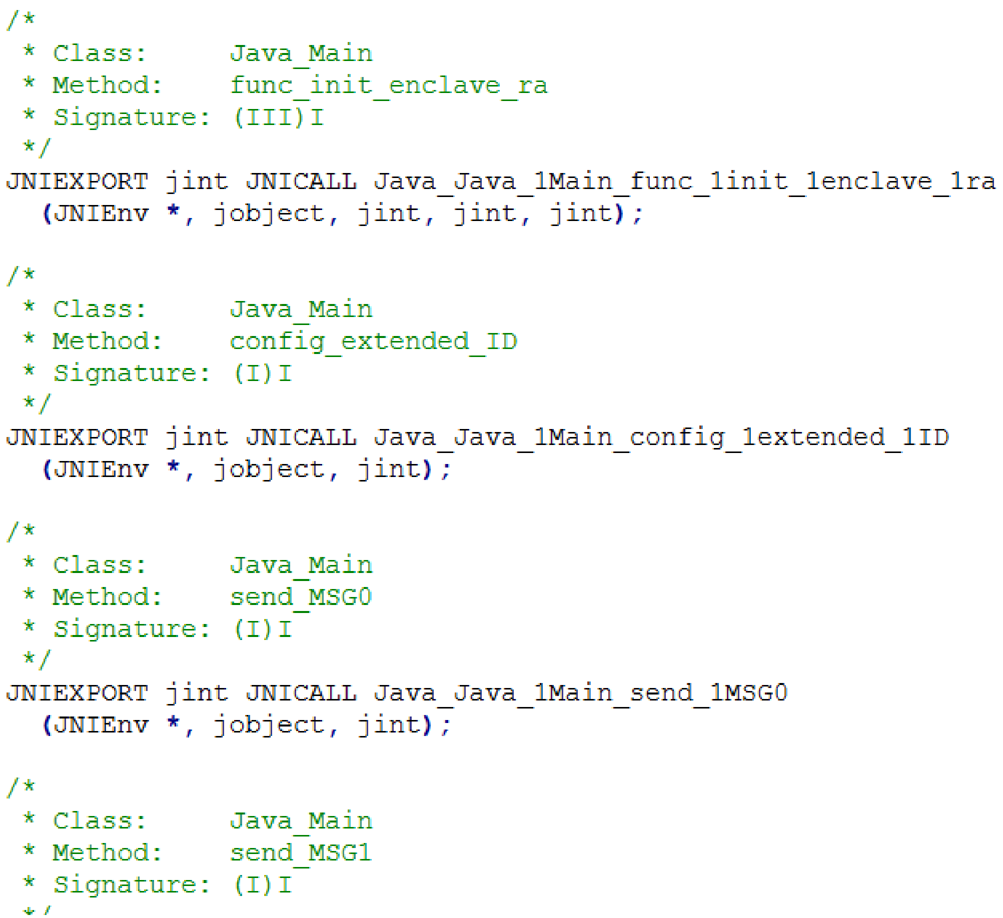


Implementation Sequence Diagram

3. Decryption, evaluation and sending back encrypted result to client from remote enclave



Using Java native interfaces(JNI) in SGX



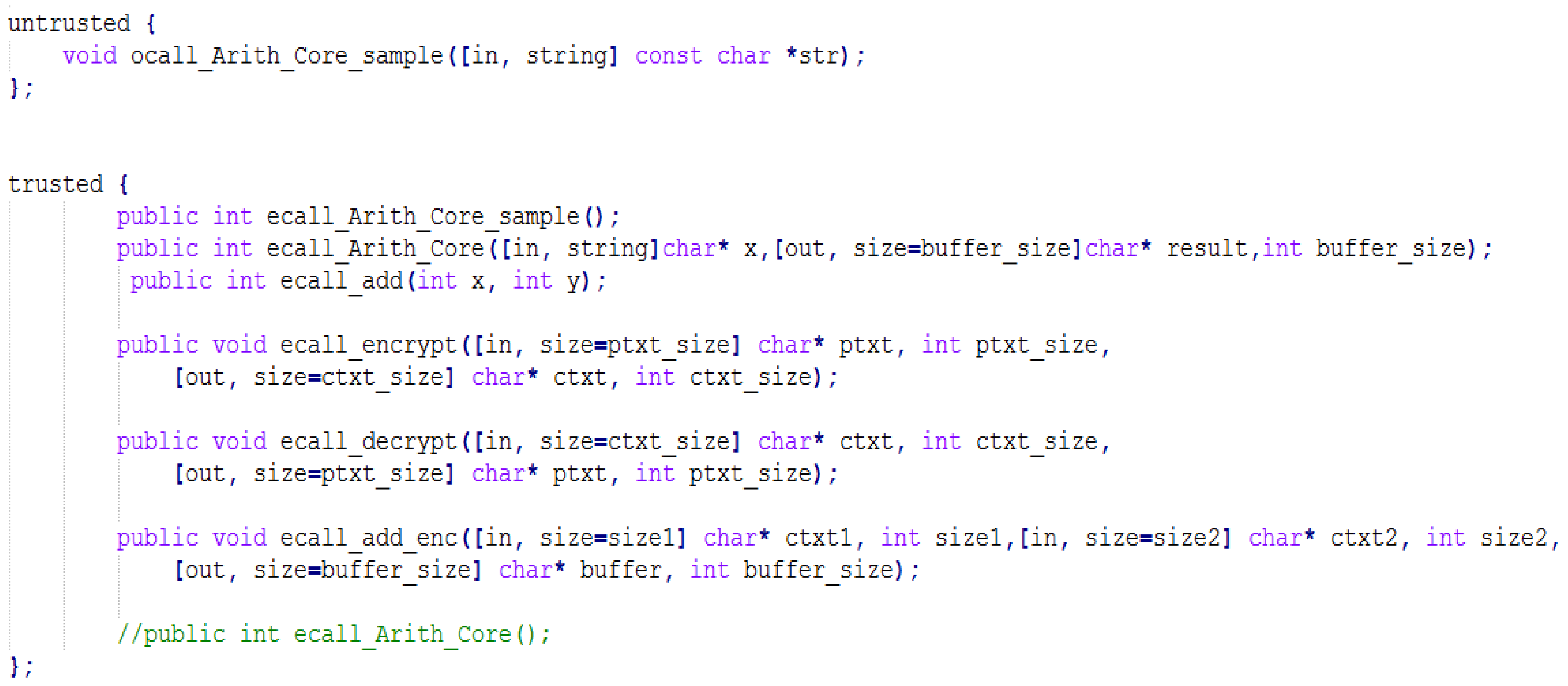
```
/*
 * Class:     Java_Main
 * Method:    func_init_enclave_ra
 * Signature: (III)I
 */
JNIEXPORT jint JNICALL Java_Java_1Main_func_1init_1enclave_1ra
  (JNIEnv *, jobject, jint, jint, jint);

/*
 * Class:     Java_Main
 * Method:    config_extended_ID
 * Signature: (I)I
 */
JNIEXPORT jint JNICALL Java_Java_1Main_config_1extended_1ID
  (JNIEnv *, jobject, jint);

/*
 * Class:     Java_Main
 * Method:    send_MSG0
 * Signature: (I)I
 */
JNIEXPORT jint JNICALL Java_Java_1Main_send_1MSG0
  (JNIEnv *, jobject, jint);

/*
 * Class:     Java_Main
 * Method:    send_MSG1
 * Signature: (I)I
 */
```

Initialization of Enclave

Configuration of Enclave

Other function calls to Untrusted part

Enclave Description Language(EDL)



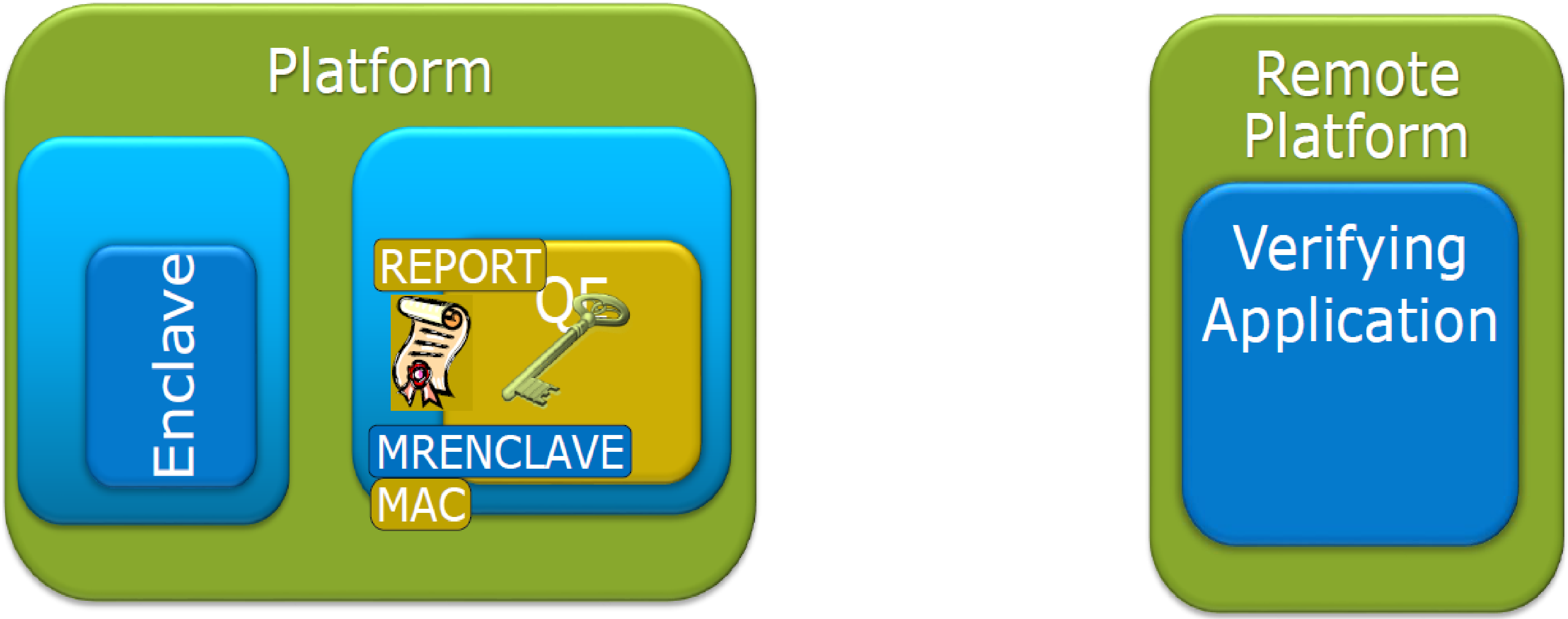
```
untrusted {
    void ocall_Arith_Core_sample([in, string] const char *str);
};


trusted {
        public int ecall_Arith_Core_sample();
        public int ecall_Arith_Core([in, string]char* x,[out, size=buffer_size]char* result,int buffer_size);
         public int ecall_add(int x, int y);

        public void ecall_encrypt([in, size=ptxt_size] char* ptxt, int ptxt_size,
            [out, size=ctxt_size] char* ctxt, int ctxt_size);

        public void ecall_decrypt([in, size=ctxt_size] char* ctxt, int ctxt_size,
            [out, size=ptxt_size] char* ptxt, int ptxt_size);

        public void ecall_add_enc([in, size=size1] char* ctxt1, int size1,[in, size=size2] char* ctxt2, int size2,
            [out, size=buffer_size] char* buffer, int buffer_size);

        //public int ecall_Arith_Core();
};
```

Rule of the Thumb:

- Trusted part contains all the ECALLS.
- Untrusted part contains all the OCALLS of the application.

How can a Remote Enclave be trusted?

By procedure of Remote Attestation offered by Intel SGX.

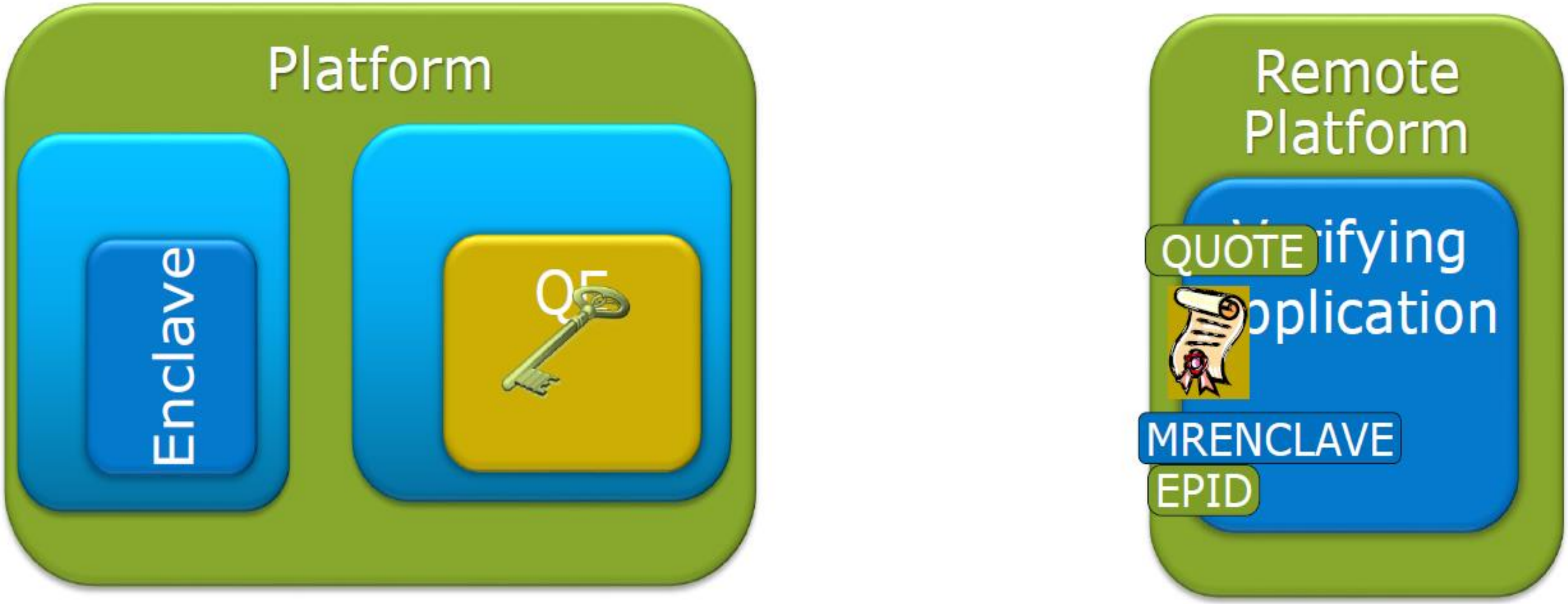


A verifying enclave becomes the Quoting Enclave.

Image Source:https://software.intel.com/en-us/sgx/details

How can a Remote Enclave be trusted?

Remote Attestation by Intel SGX.

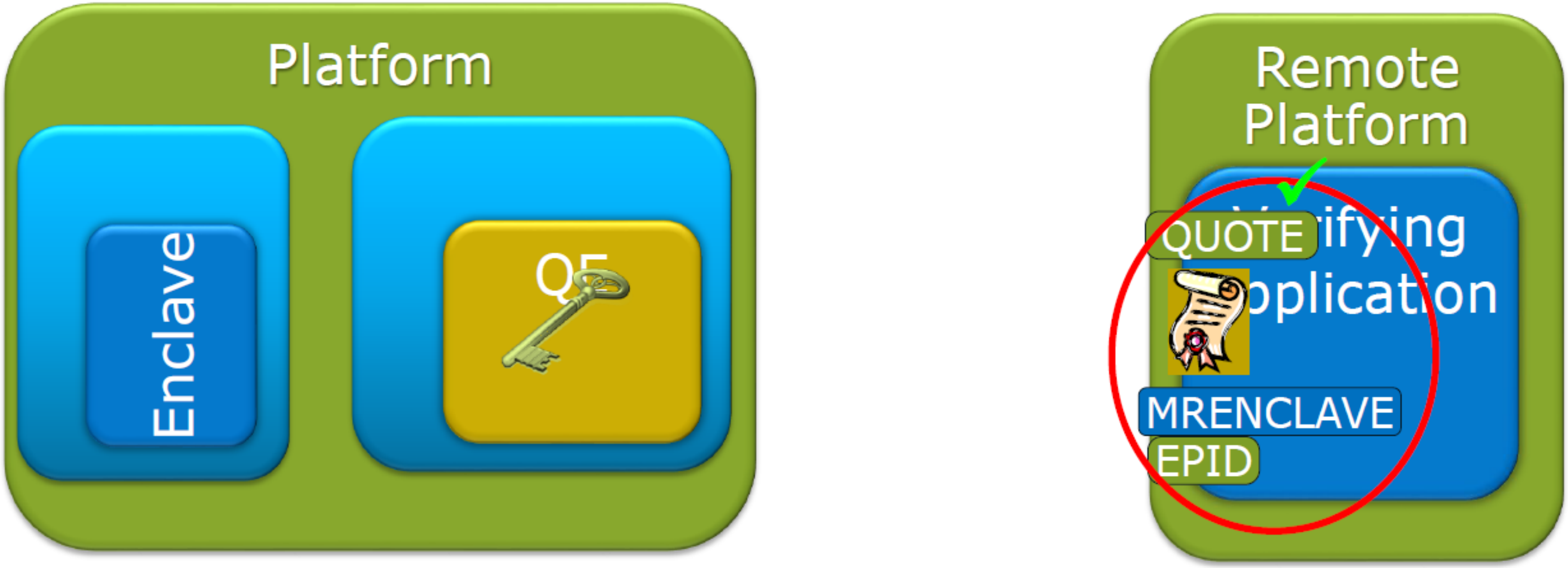


After verifying the REPORT the, QE signs the REPORT with the EPID private key and converts it into a QUOTE.

Image Source:https://software.intel.com/en-us/sgx/details

How can a Remote Enclave be trusted?

Remote Attestation by Intel SGX.



Remote platform verifies the QUOTE with the EPID public key and verifies MRENCLAVE against the expected value

Image Source:https://software.intel.com/en-us/sgx/details

Demo of the Application

Application Demo

Conclusion

- While Intel SGX provides new paradigm into securing application and its secrets using hardware enabled measures. It is highly platform and hardware dependent.
- Using JNI we can access C and C++ code which adds performance boost.
- However, JNI uses native languages which mean it has portability Issue.
- Code Debug could be one of the major problems for the developers who use JNI features in JAVA.

Thank You!