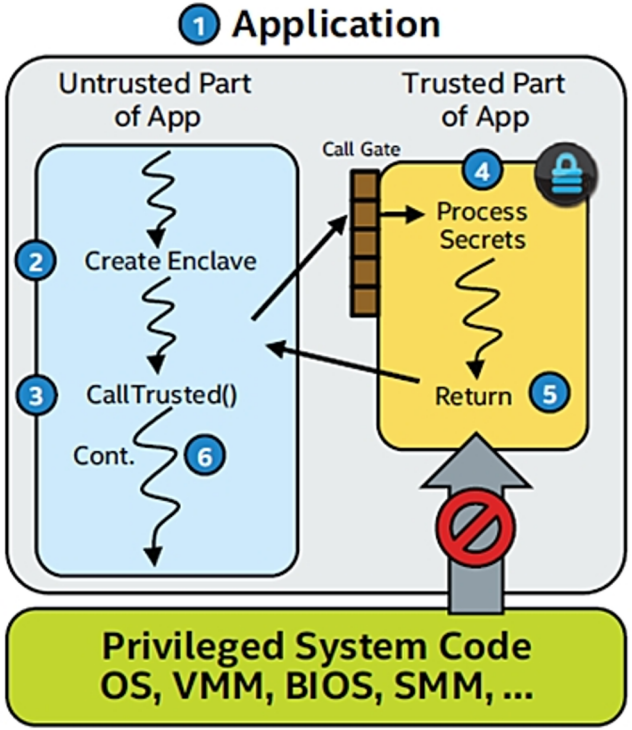
Andromeda: Enabling Secure Enclaves For The Android Ecosystem

Dimitris Deyannis, **Dimitris Karnikis**, Giorgos Vasiliadis & Sotiris Ioannidis

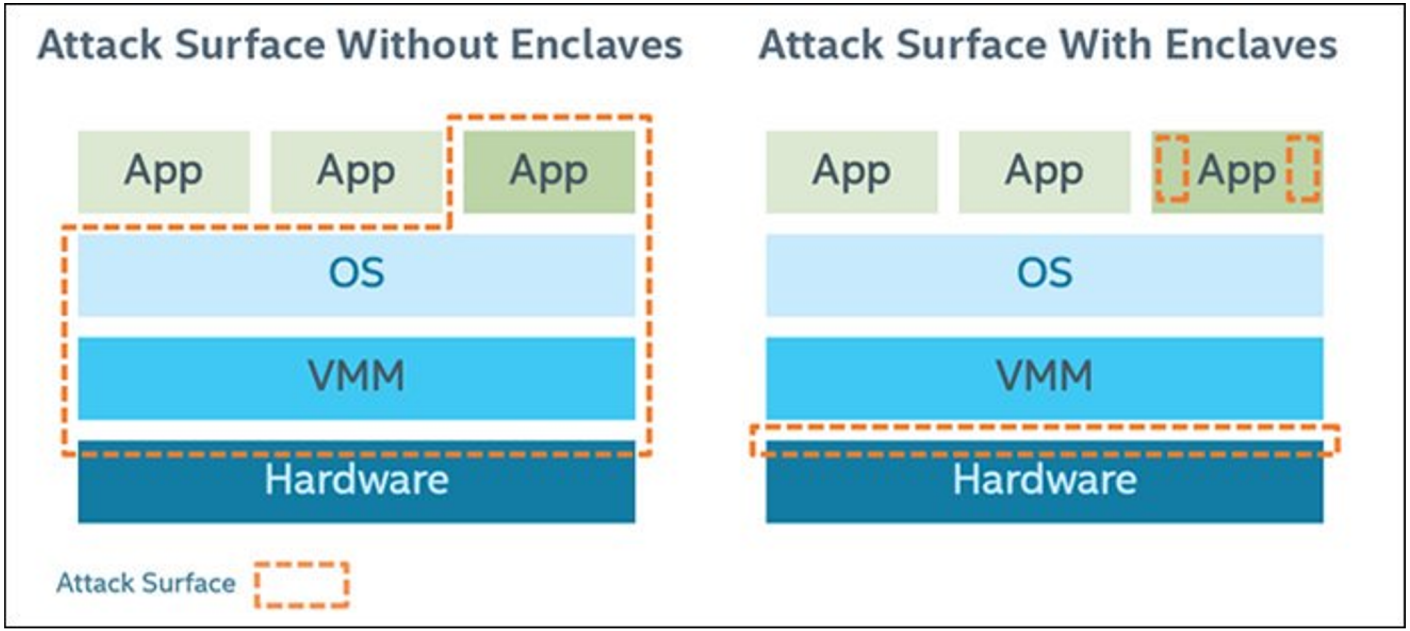
d.ntegiannis@sphynx.ch, **dkarnikis@gmail.com**, gvasil@ics.forth.gr & sotiris@ece.tuc.gr

Intel Software Guard eXtensions (SGX)

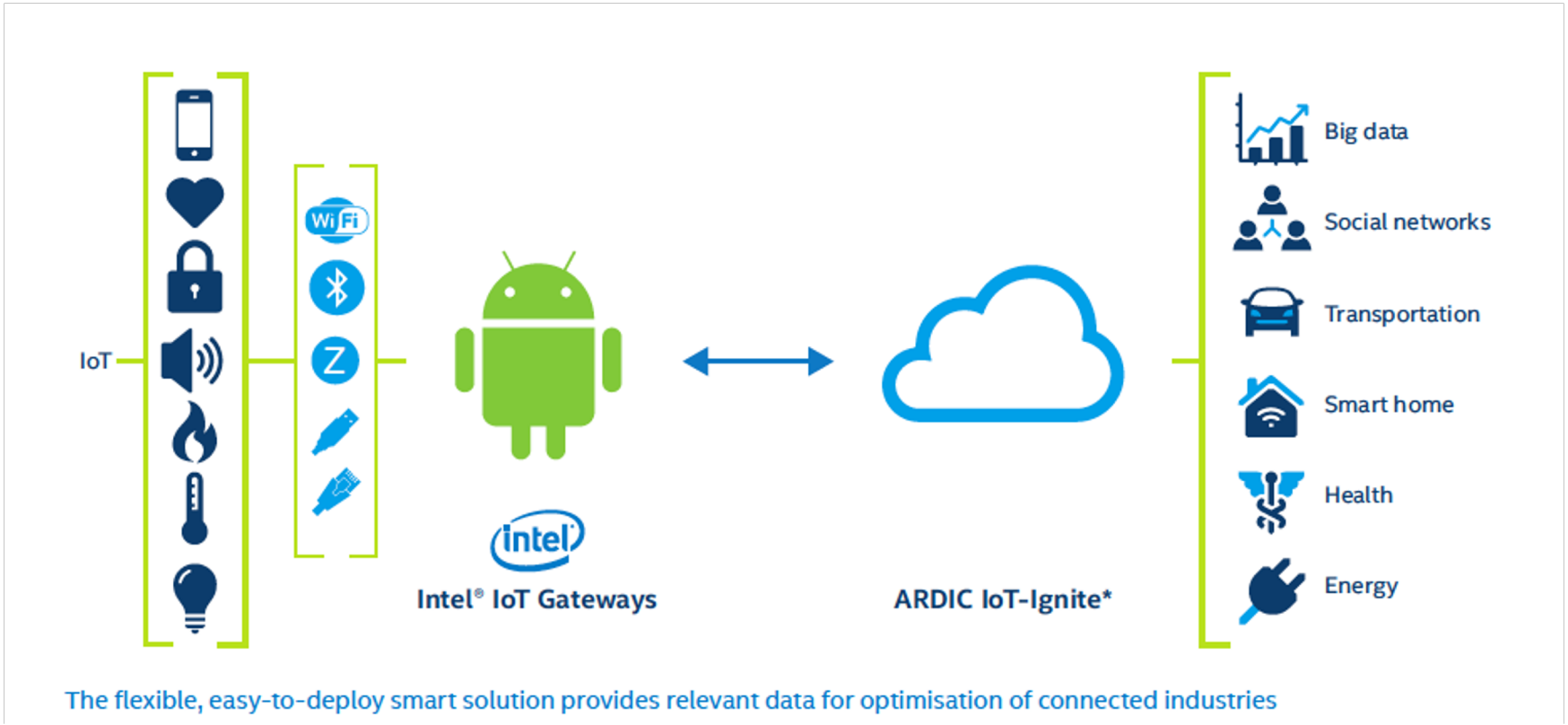
- Introduced with the 6th generation Intel processors (Skylake)
- Reverse sandbox
- Offers user-space secure enclaves
- Application level granularity
- Remote attestation



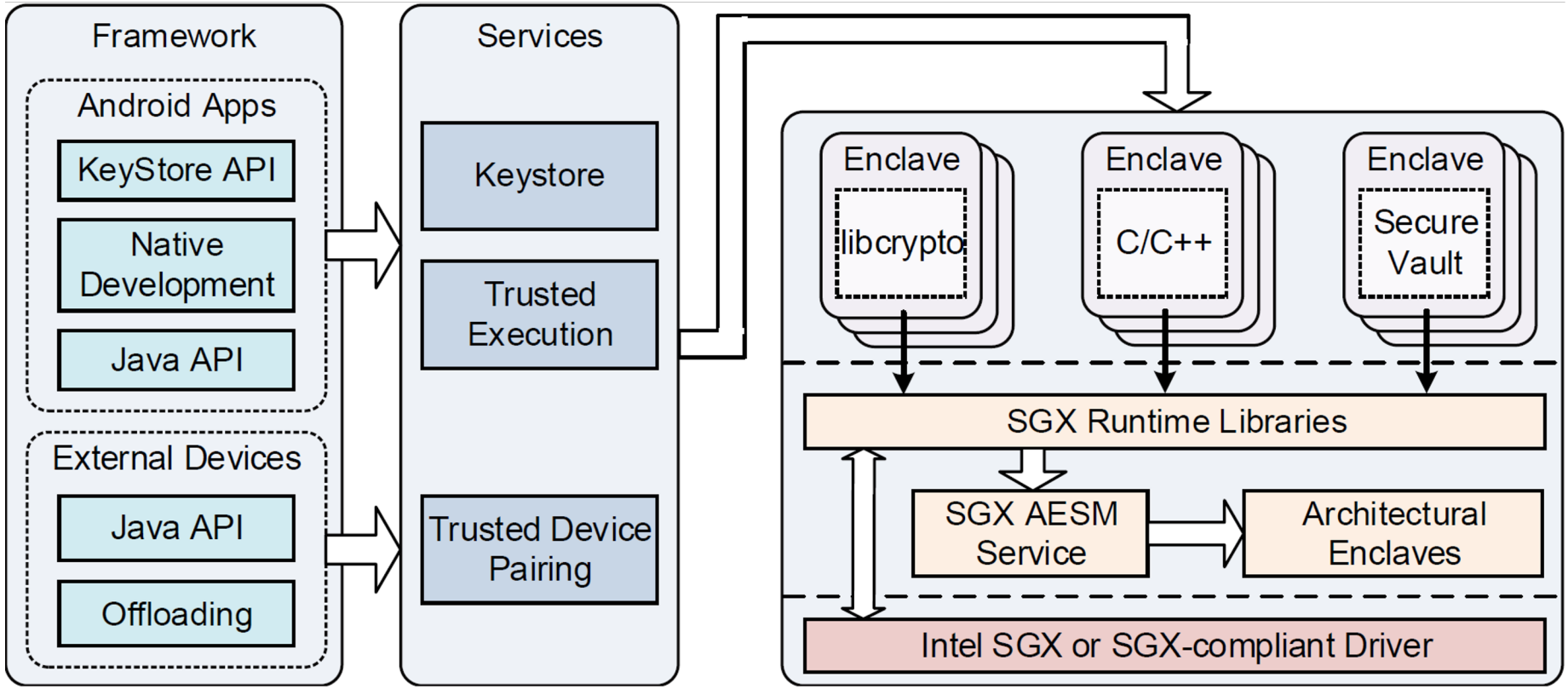
Intel Software Guard eXtensions (SGX)



Motivation


IoT
Wi Fi
Intel® IoT Gateways
ARDIC IoT-Ignite*
Big data
Social networks
Transportation
Smart home
Health
Energy
The flexible, easy-to-deploy smart solution provides relevant data for optimisation of connected industries

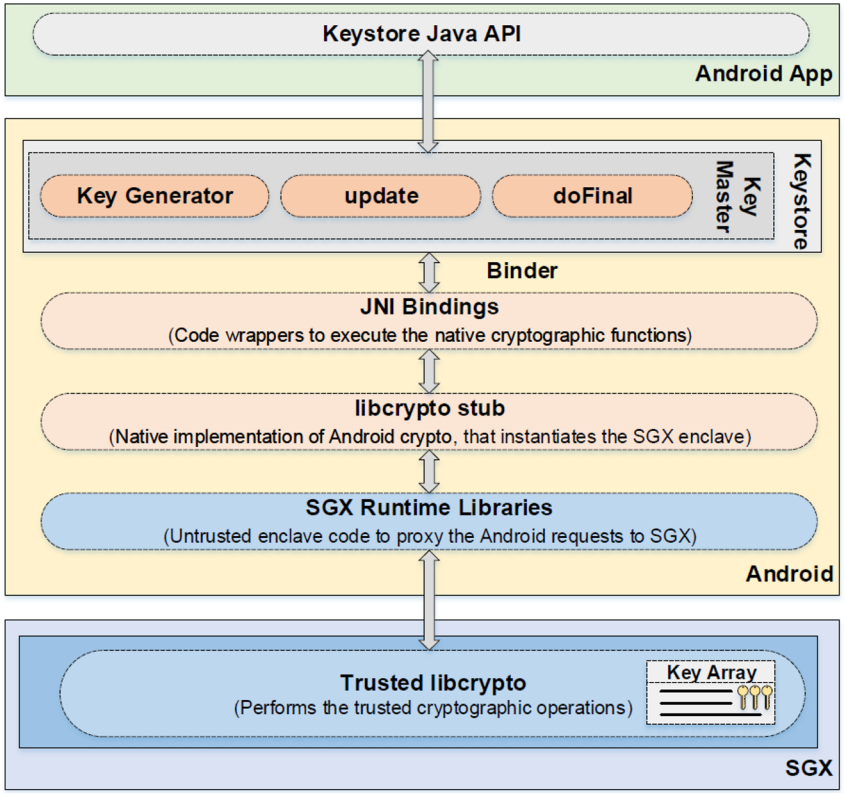
Architecture



Andromeda Framework

- Andromeda Keystore
- Native Development Kit
- Andromeda Java API
  - Secure Pairing API
  - Secure Execution
  - Secure Vault API

Andromeda Keystore



- Keys are stored **only** in SGX enclaves
- Encryption and decryption using the **default Keystore API**
- Can be used even by **legacy** apps **without any code modifications** or **recompilation**

Andromeda NDK

- Based on CrystaX NDK providing GCC-5.3 compatibility
- Cross-compiles C/C++ SGX-enabled applications for Android
- Development and compilation process similar to traditional SGX applications
- Developers must provide JNI bindings to link SGX enclaves with Java APKs

Secure device pairing

- Andromeda provides a Java-based API for secure pairing
- Allows external devices to securely pair with Andromeda-enabled devices
- External devices may not be equipped with TEE-enabled CPUs
- Upon pairing, devices are assigned with dedicated SGX enclaves
- Connection via Bluetooth or Wi-Fi

Secure device pairing process

1. Andromeda generates a dedicated SGX-enclave for the external device

2. The devices generate key-pairs and exchange public keys
    a. Andromeda generates and stores the private key in the enclave

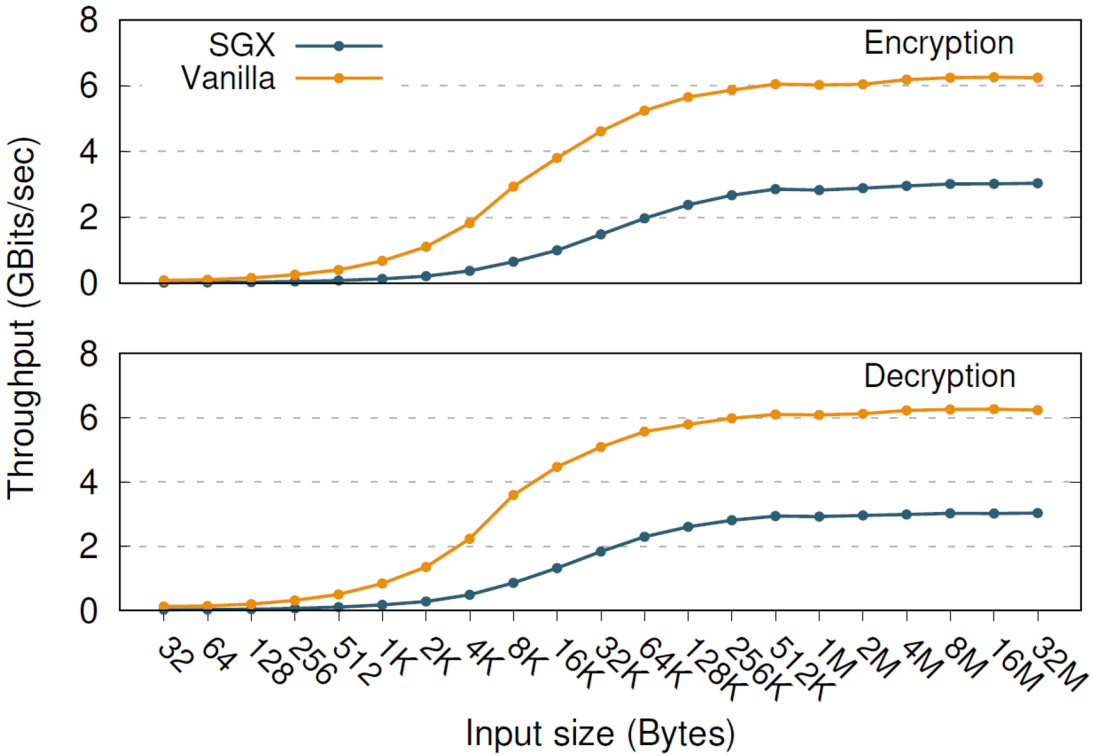
3. A session key is established between the devices
    a. Andromeda decrypts the data only within the enclave

4. The external device attests Andromeda
    a. Intel Remote Attestation used if available
    b. Alternative OTP-based attestation with the enclave-registered keys

Secure Vault

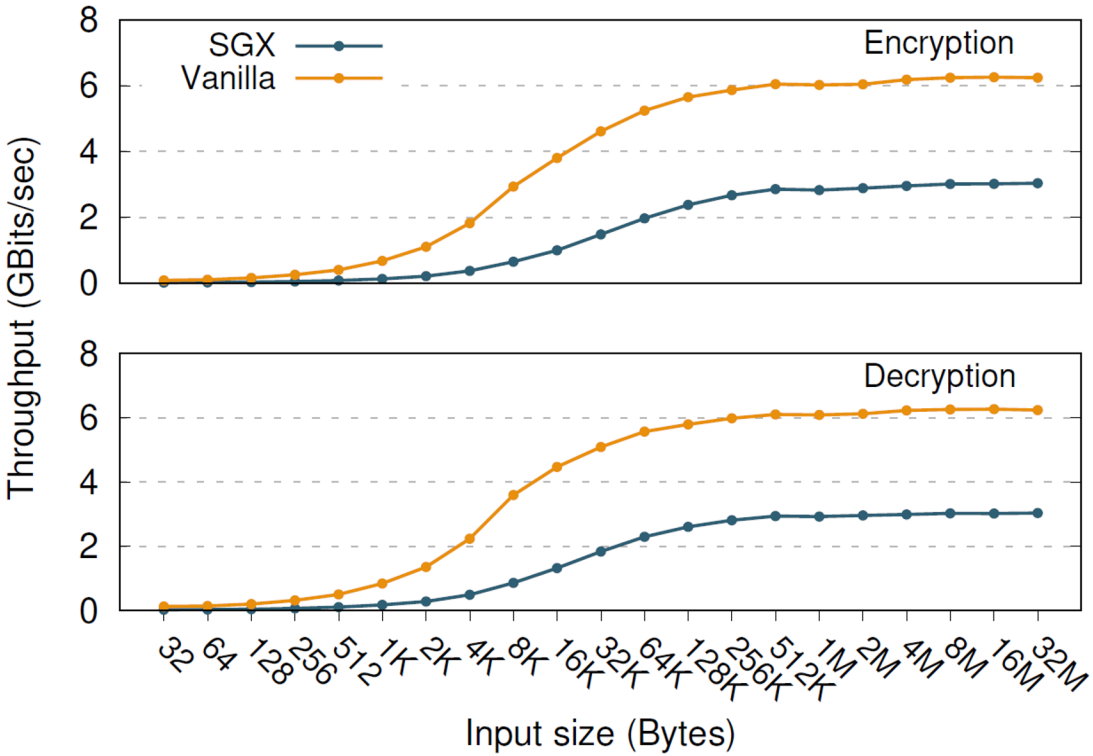
- Java API for the SGX-enabled secure data vault
- Multiple vault instances
- Data storing and retrieval
- Sealing and unsealing operations for secure persistent storage
- Data integrity checks after device reboots

Evaluation: AES-128-CTR (Keystore internal)



- **Vanilla**: Native C implementation of the vanilla Keystore
- **SGX**: SGX-enabled version

Evaluation: AES-128-CTR (Keystore internal)

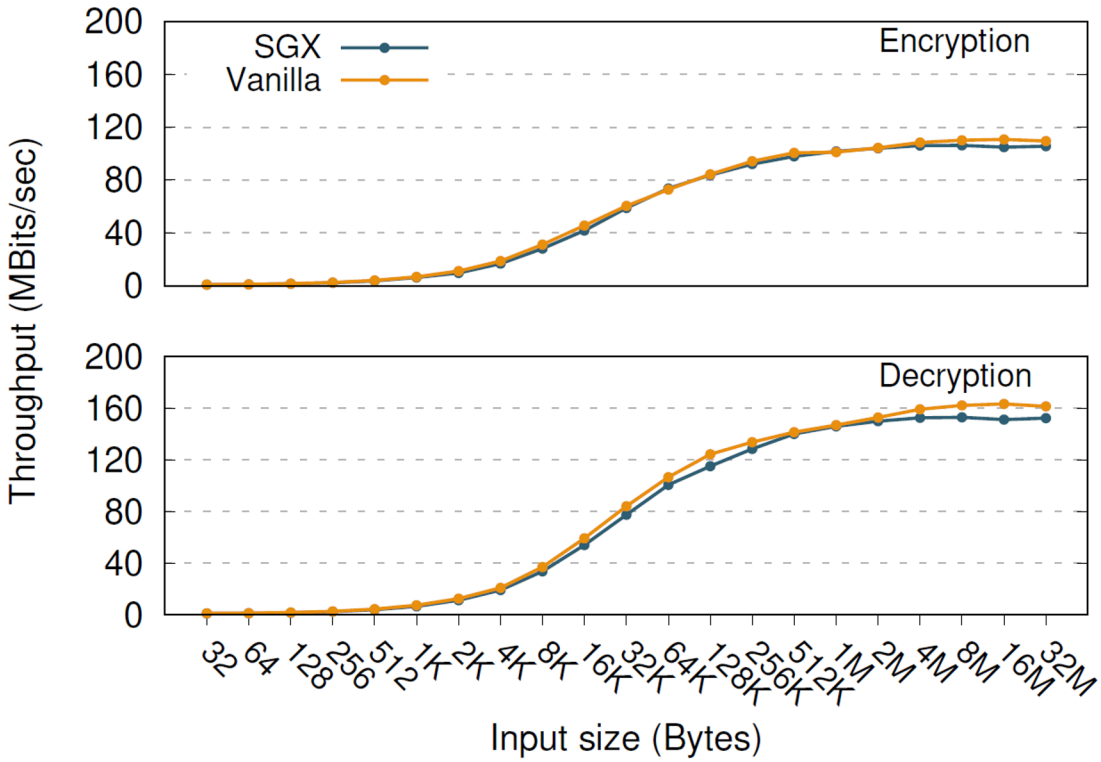


- **Vanilla**: Native C implementation of the vanilla Keystore
- **SGX**: SGX-enabled version

- Expensive enclave I/O
- Minimal processing

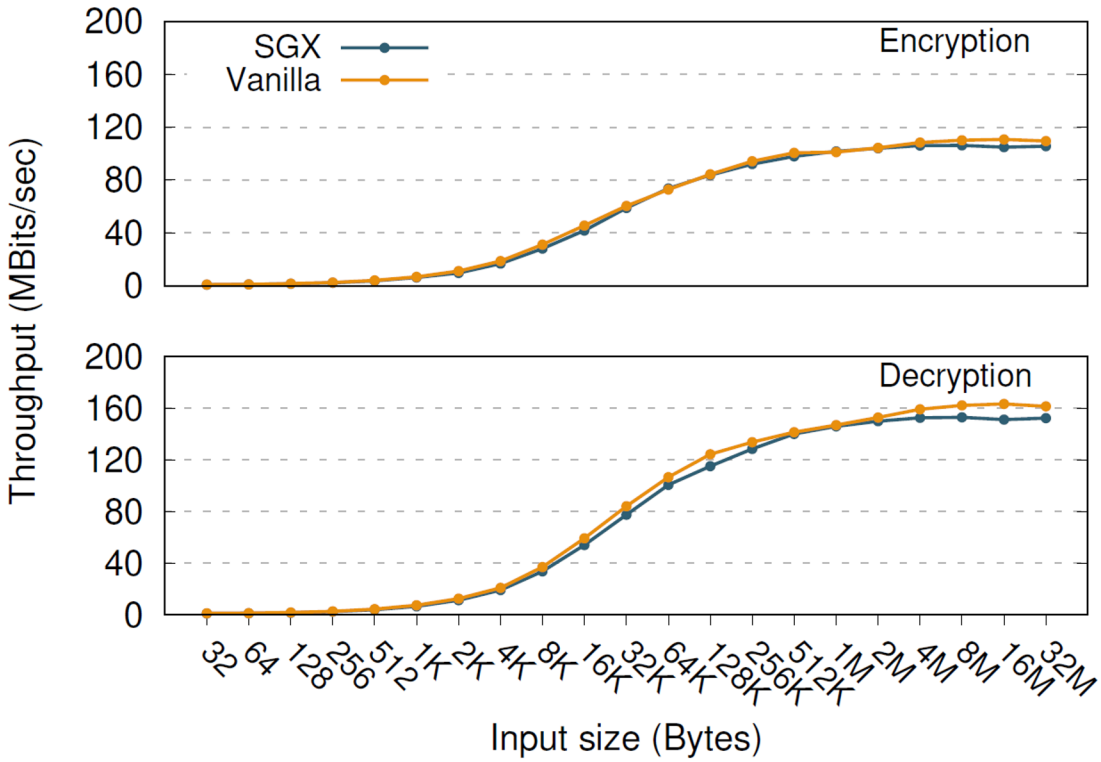
- Encryption overhead: 51% - 84%
- Decryption overhead: 51% - 78%

Evaluation: AES-128-CTR (Keystore API)



- **Vanilla**: Vanilla Keystore throughput observed from the APK level
- **SGX**: SGX-enabled version

Evaluation: AES-128-CTR (Keystore API)

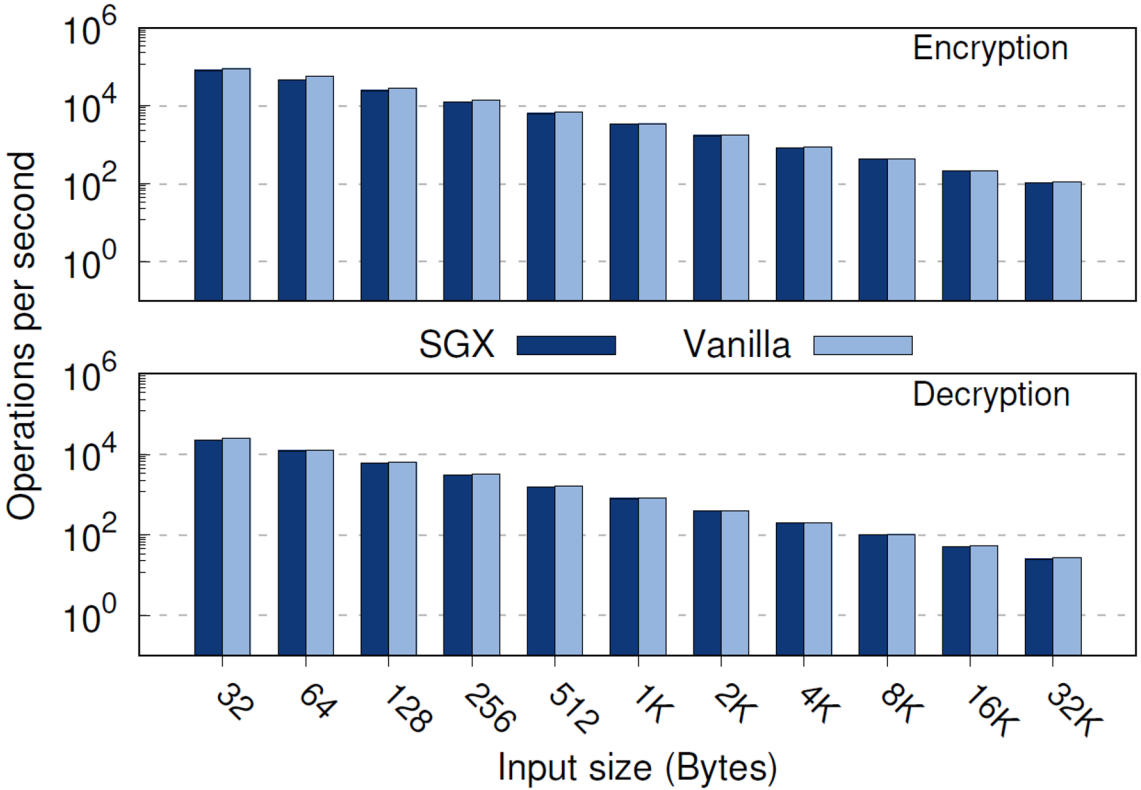


- **Vanilla**: Vanilla Keystore throughput observed from the APK level
- **SGX**: SGX-enabled version

- High overhead introduced by the Keystore stack (JNI, IPC, etc.)
- Throughput decreases by an order of magnitude
- Actual overhead perceived by APKs

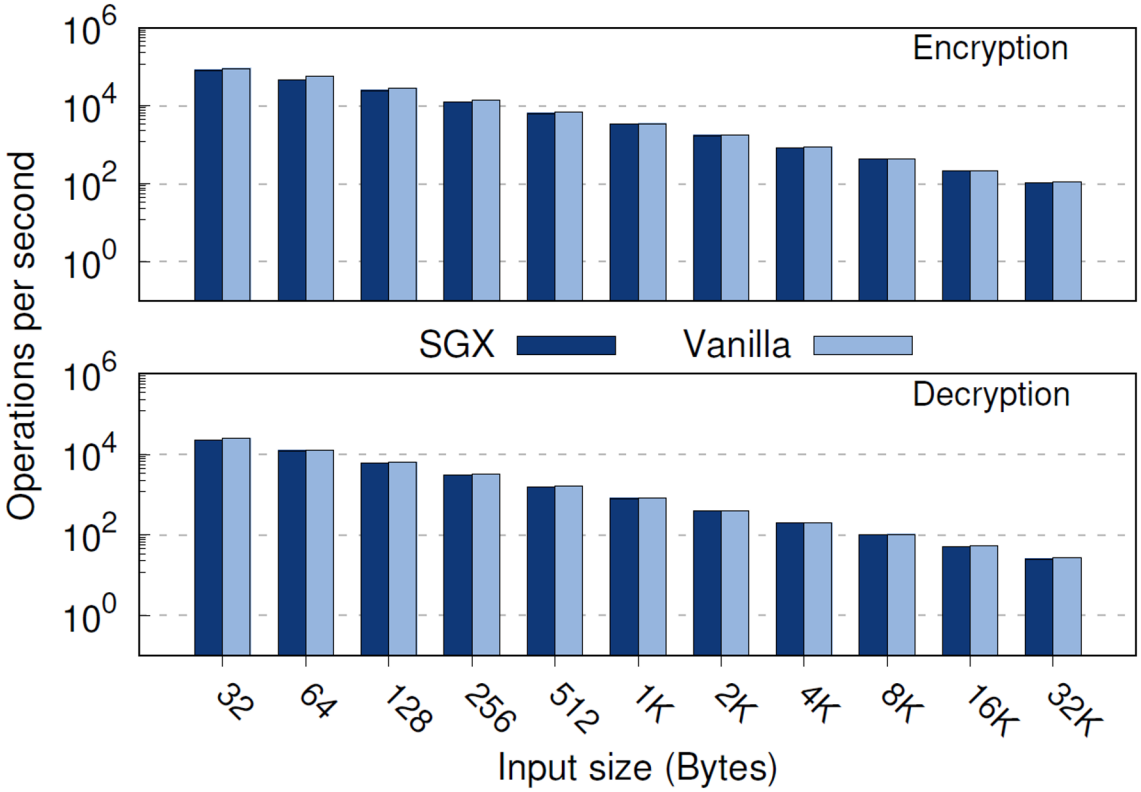
- Encryption overhead: 0.6% - 13%
- Decryption overhead: 0.6% - 11%

Evaluation: RSA-1024 (Keystore Internal)



- **Vanilla**: Native C implementation of the vanilla Keystore
- **SGX**: SGX-enabled version

Evaluation: RSA-1024 (Keystore Internal)

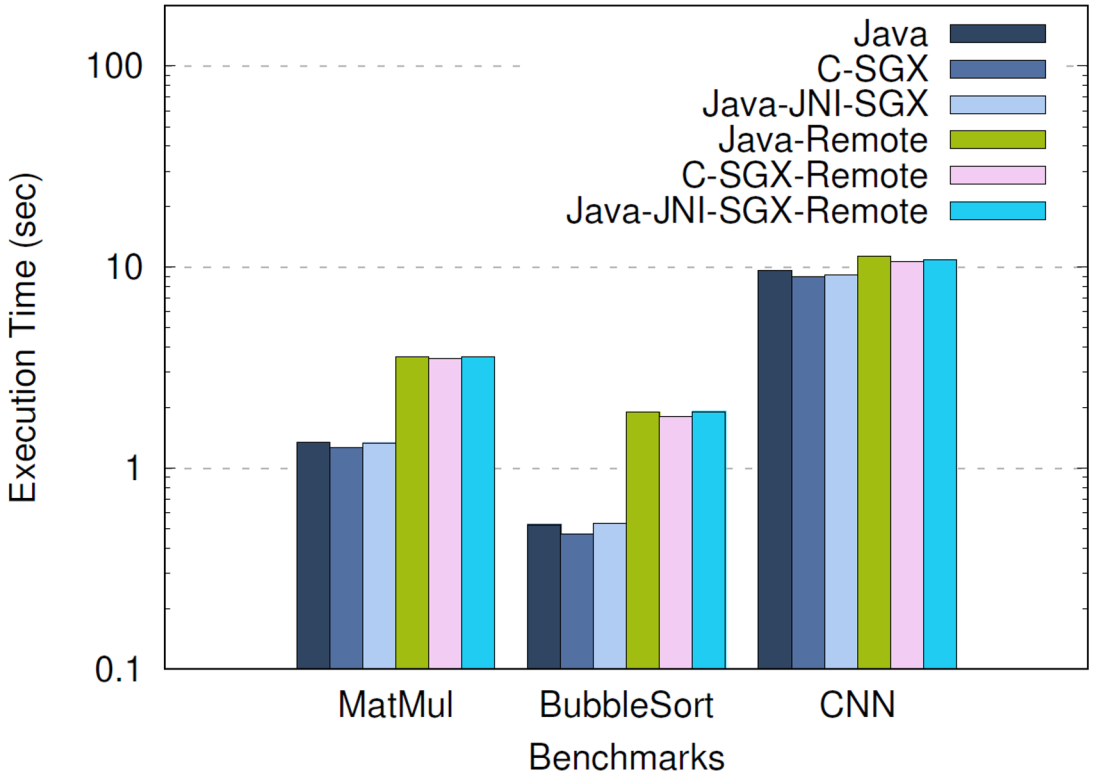


- **Vanilla**: Native C implementation of the vanilla Keystore
- **SGX**: SGX-enabled version

- I/O overhead minimised due to the processing complexity

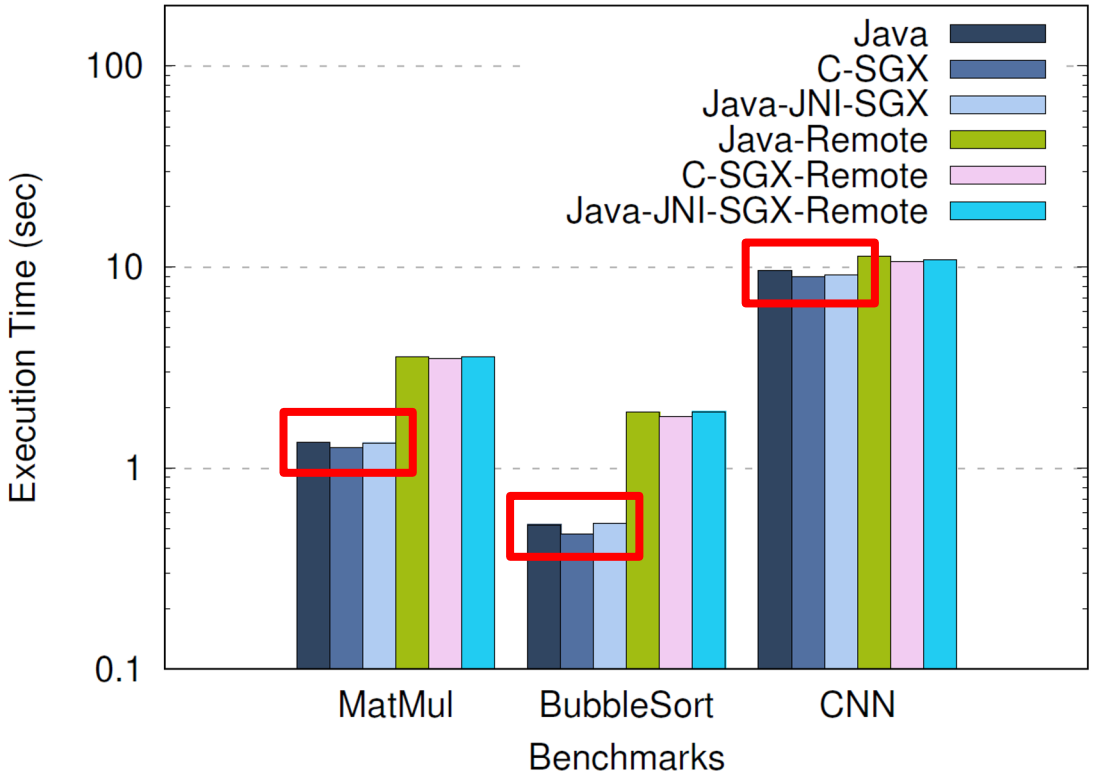
- Encryption overhead: 2.3% - 16%
- Decryption overhead: 0.9% - 12.6%

Local and remote execution



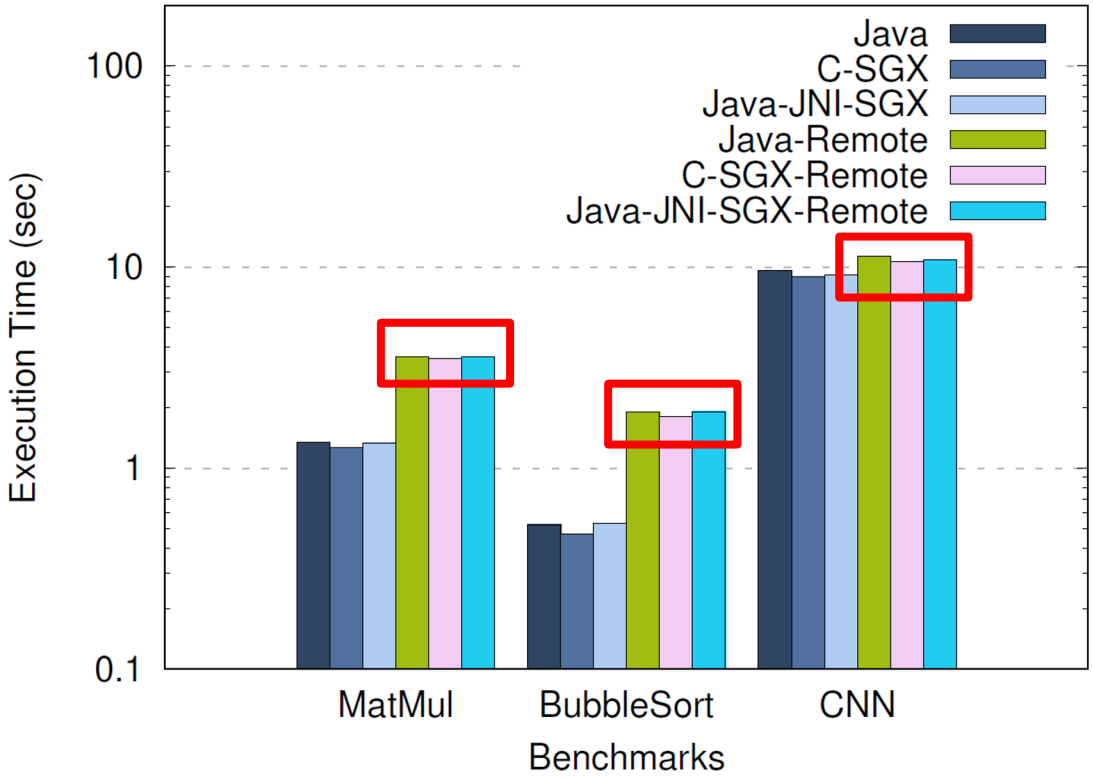
- **Java**: Pure Java implementation
- **C-SGX**: SGX-enabled native C implementation
- **Java-JNI-SGX**: Java implementation, critical parts implemented with SGX-enabled native C
- ***-Remote**: Execution requested from external device, performed on Andromeda

Local and remote execution



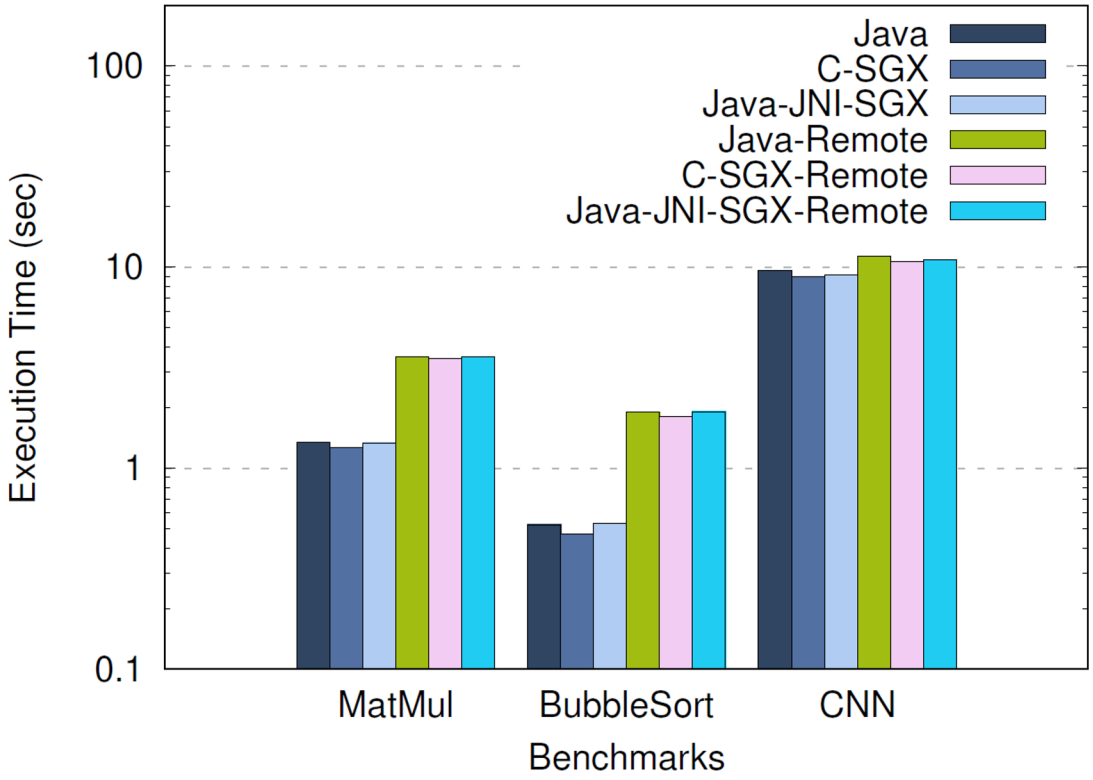
- The Java version is 5.4% - 11.2% slower, locally, compared to the C and JNI SGX-enabled versions

Local and remote execution



- The Java version is 5.4% - 11.2% slower, locally, compared to the C and JNI SGX-enabled versions
- The Java version is 5.13% - 7.5% slower compared to the C and JNI SGX-enabled versions when executed remotely

Local and remote execution



- The Java version is 5.4% - 11.2% slower, locally, compared to the C and JNI SGX-enabled versions
- The Java version is 5.13% - 7.5% slower compared to the C and JNI SGX-enabled versions when executed remotely
- The speedup gained by the native execution overshadows the minimal I/O overhead and outperforms the Java implementation

Conclusions

- Andromeda is the first (to our knowledge) port of the SGX framework to Android including:
  - SGX kernel driver
  - Libraries and background services
  - Custom cross-compiler for native app development
  - JNI bindings for APKs

- Andromeda offers popular Android services enhanced with SGX capabilities
  - SGX-enabled Android Keystore
  - Trusted Device Pairing
  - Secure data storage

- Andromeda enables external devices to securely offload data storage and request secure computations without necessarily being equipped with TEE-enabled CPUs