Using Trusted Execution Environments for Secure Stream Processing of Medical Data

DAIS'19 - 18-20 July 2019, Copenhagen (DK) - *Use-Case Paper*

C. Segarra[1] R. Delgado[1] M. Lemay[1] P. Aublin[3] P. Pietzuch[3] V. Schiavoni[2]

[1]CSEM, Neuchâtel, Switzerland, {carlos.segarra,ricard.delgado,mathieu.lemay}@csem.ch

[2]University of Neuchâtel, Switzerland, valerio.schiavoni@unine.ch

[3]Imperial College London, United Kingdom, {p.aublin,prp}@imperial.ac.uk

Thursday, June 13, 2019

Outline

Motivation
Why do we need secure big data processing engines?

We want to:

Process large amounts of **sensitive** information

We want to:

Process large amounts of **sensitive** information

↓ Generally requires...

Outsourcing of data **storage** and **processing**

Motivation

Why do we need secure big data processing engines?

We want to:

Process large amounts of **sensitive** information

↓ Generally requires...

Outsourcing of data **storage** and **processing**

↓ Generally means...

Cloud tenant **can** access **protected** data

Motivation

Why do we need secure big data processing engines?

We want to:

Process large amounts of **sensitive** information

↓ Generally requires...

Outsourcing of data **storage** and **processing**

↓ Generally means...

Cloud tenant **can** access **protected** data

↓ This is...

UNACCEPTABLE FOR INDUSTRIES IN THE MEDICAL DOMAIN

(and many others)

Trusted Execution Environments (TEE)
Formal definition, examples and availability

Trusted Execution Environment

A **TEE** is a secure area of a processor. It guarantees code and data to be protected with respect to **confidentiality** and **integrity**.

Trusted Execution Environments (TEE)

Formal definition, examples and availability

Trusted Execution Environment

A **TEE** is a secure area of a processor. It guarantees code and data to be protected with respect to **confidentiality** and **integrity**.

- Available on a variety of commodity CPUs. For instance:

Trusted Execution Environments (TEE)

Formal definition, examples and availability

Trusted Execution Environment

A **TEE** is a secure area of a processor. It guarantees code and data to be protected with respect to **confidentiality** and **integrity**.

Intel SGX Available on consumer-grade CPUs starting from architecture codename *Skylake*.

Arm Trustzone Available on Cortex-A processors and v8 Cortex-M23 and Cortex-M33 (*e.g.* Raspberry Pi).

- **Intel Software Guard eXtensions (SGX)** are a set of instructions and memory access extensions that enable applications to create **hardware-protected areas** in their address space called **enclaves**.
- Security perimiter includes **only** the internals of the CPU package.
- An **attestation protocol** verifies that code is running in a **genuine enclave** and that it has not been tampered.

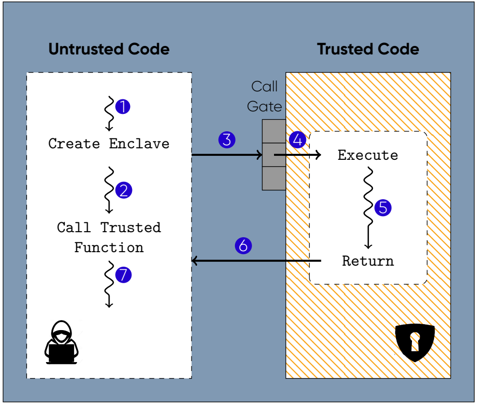


Figure 1: Intel SGX Operating Principles.

Intel Software Guard eXtensions

Definition, Threat Model and Known Vulnerabilities

- **Intel Software Guard eXtensions (SGX)** are a set of instructions and memory access extensions that enable applications to create **hardware-protected areas** in their address space called **enclaves**.
- Security perimiter includes **only** the internals of the CPU package.
- An **attestation protocol** verifies that code is running in a **genuine enclave** and that it has not been tampered.

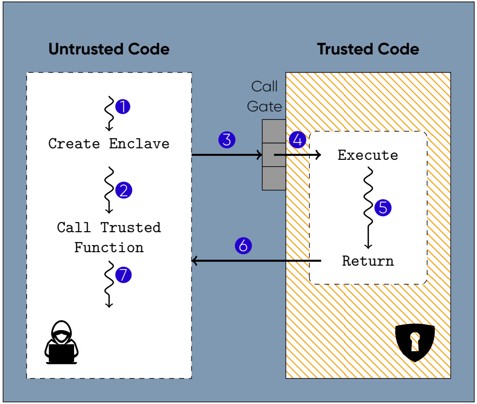


Figure 1: Intel SGX Operating Principles.

Intel Software Guard eXtensions

Definition, Threat Model and Known Vulnerabilities

- **Intel Software Guard eXtensions (SGX)** are a set of instructions and memory access extensions that enable applications to create **hardware-protected areas** in their address space called **enclaves**.
- Security perimiter includes **only** the internals of the CPU package.
- An **attestation protocol** verifies that code is running in a **genuine enclave** and that it has not been tampered.

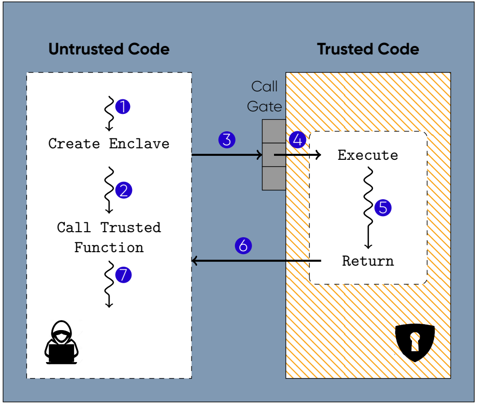


Figure 1: Intel SGX Operating Principles.

SGX-Spark

- SGX-Spark is a framework under-development at the **Imperial College London** to enable seamless deployment of **Spark** jobs inside **enclaves**.
- Protect **confidentiality** and **integrity** of existing Spark jobs **without** modifications to the application code.
- Execute only **sensitive** parts of the application **inside** the enclave. Leave information outside the enclave **encrypted**.

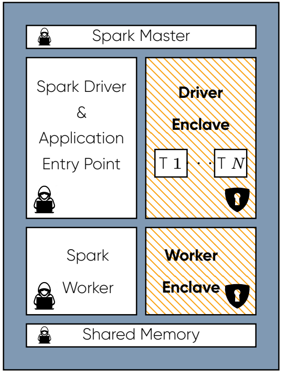


Figure 2: SGX-Spark Architecture.

SGX-Spark

- SGX-Spark is a framework under-development at the **Imperial College London** to enable seamless deployment of **Spark** jobs inside **enclaves**.
- Protect **confidentiality** and **integrity** of existing Spark jobs **without** modifications to the application code.
- Execute only **sensitive** parts of the application **inside** the enclave. Leave information outside the enclave **encrypted**.

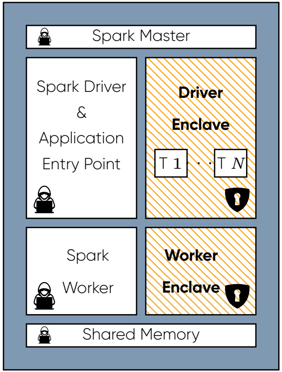


Figure 2: SGX-Spark Architecture.

SGX-Spark

- SGX-Spark is a framework under-development at the **Imperial College London** to enable seamless deployment of **Spark** jobs inside **enclaves**.
- Protect **confidentiality** and **integrity** of existing Spark jobs **without** modifications to the application code.
- Execute only **sensitive** parts of the application **inside** the enclave. Leave information outside the enclave **encrypted**.

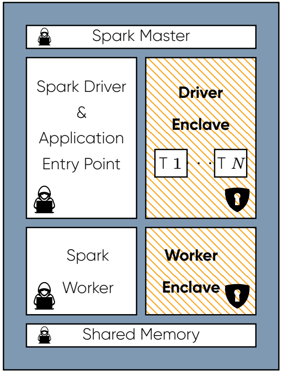
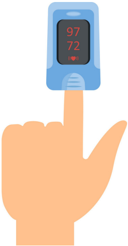


Figure 2: SGX-Spark Architecture.

Medical Use-Case
Cardiac Monitoring, ECG, PPG, and HRV

- Our solution is **independent of the chosen data streams**, hence it could be leveraged in a variety of domains where privacy might be a concern. However,
- The data streams used for this project belong to the **medical** domain, in particular, they are obtained from human **cardiac activity monitoring**. The two most standard procedures are:

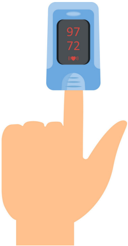
1. **ECG:** measure heart's electrical activity over time. *E.g:* chest straps.
2. **PPG:** measure blood's volume variation over time. *E.g:* smartwatches and pulse oximeters.



Medical Use-Case
Cardiac Monitoring, ECG, PPG, and HRV

- Our solution is **independent of the chosen data streams**, hence it could be leveraged in a variety of domains where privacy might be a concern. However,
- The data streams used for this project belong to the **medical** domain, in particular, they are obtained from human **cardiac activity monitoring**. The two most standard procedures are:

1. **ECG:** measure heart's electrical activity over time. *E.g:* chest straps.
2. **PPG:** measure blood's volume variation over time. *E.g:* smartwatches and pulse oximeters.



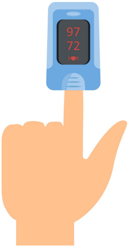
Medical Use-Case

Cardiac Monitoring, ECG, PPG, and HRV

- Our solution is **independent of the chosen data streams**, hence it could be leveraged in a variety of domains where privacy might be a concern. However,
- The data streams used for this project belong to the **medical** domain, in particular, they are obtained from human **cardiac activity monitoring**. The two most standard procedures are:

1. **ECG:** measure heart's electrical activity over time.
   E.g: chest straps.

2. **PPG:** measure blood's volume variation over time.
   E.g: smartwatches and pulse oximeters.



Medical Use-Case
Cardiac Monitoring, ECG, PPG, and HRV

- We are interested in the **inter-beat intervals** from the generated diagram ($23 - 69$ B per second per user).
- From an **ECG** we can obtain obtain these intervals from the time between **R peaks**, abbreviated as RR intervals (Figure 3).
- With their **timestamps** we compute a live analysis of the **Heart Rate Variability (HRV)**.

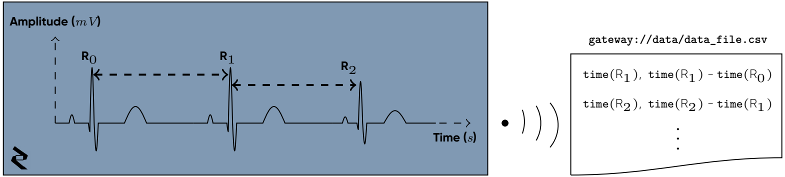


Figure 3: Schematic representation of an ECG signal and the data we gather.

Medical Use-Case
Cardiac Monitoring, ECG, PPG, and HRV

- We are interested in the **inter-beat intervals** from the generated diagram ($23 - 69$ B per second per user).
- From an **ECG** we can obtain obtain these intervals from the time between **R peaks**, abbreviated as RR intervals (Figure 3).
- With their **timestamps** we compute a live analysis of the **Heart Rate Variability (HRV)**.

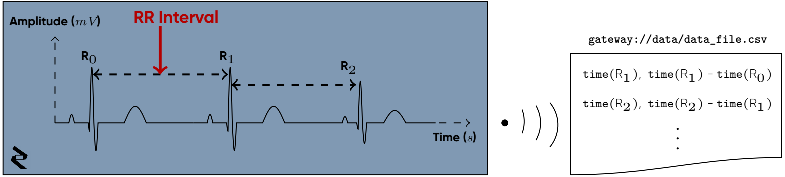


Figure 3: Schematic representation of an ECG signal and the data we gather.

Medical Use-Case
Cardiac Monitoring, ECG, PPG, and HRV

- We are interested in the **inter-beat intervals** from the generated diagram ($23 - 69$ B per second per user).
- From an **ECG** we can obtain obtain these intervals from the time between **R peaks**, abbreviated as RR intervals (Figure 3).
- With their **timestamps** we compute a live analysis of the **Heart Rate Variability (HRV)**.

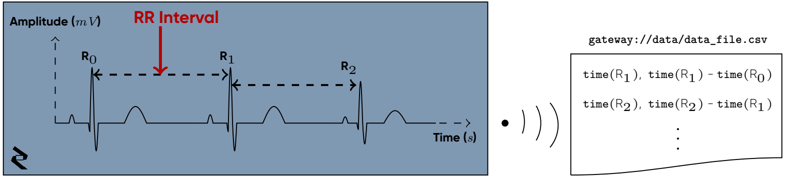


Figure 3: Schematic representation of an ECG signal and the data we gather.

Project Goals

- Use SGX-Spark to implement a **streaming platform** that gathers data from sensors and **securely** outsources computation.
  - Deploy in an existing medical environment.
  - Perform stress tests to assess the system's robustness and reliability.
  - Evaluate and quantify the overhead of providing strong security guarantees.

Secure Stream Processing of Medical Data
Envisioned Scenario

The **chosen scenario** involves:

1. ECG data streamed from a **sensor** to a **gateway**
2. Real-time processing with **HRV** algorithms: SDNN and HRV Bands analysis.
   - **SDNN:** rolling standard deviation of NN (RR) intervals.
   - **HRV Bands:** rolling Discrete Fourier Transform and low/high frequency component.
3. Support for **storage** and result **post-processing**

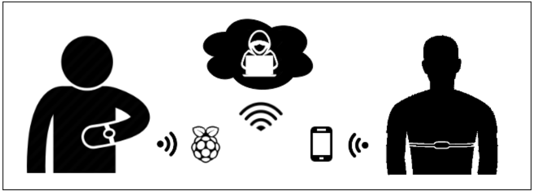


Figure 4: Wearable-enabled ecosystem of our platform.

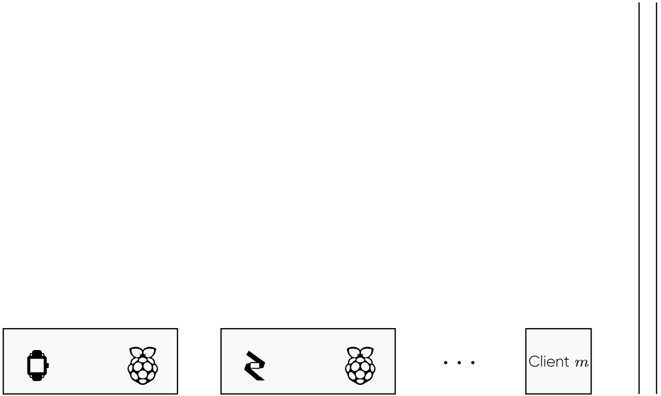


Figure 5: Client-server architecture.

System Architecture
Component Analysis and Execution Workflow

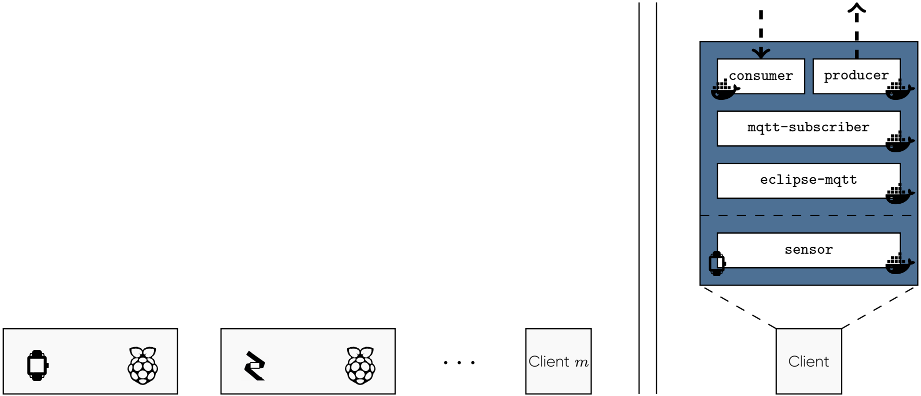


Figure 5: Client-server architecture.

System Architecture
Component Analysis and Execution Workflow

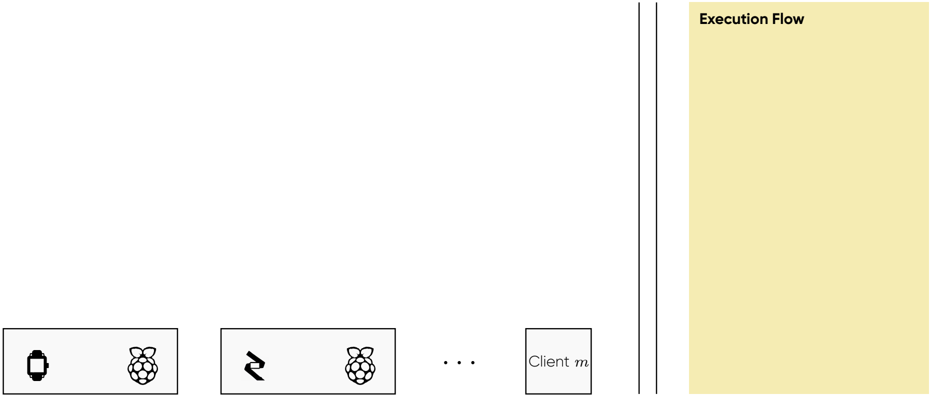


Figure 5: Client-server architecture.

System Architecture
Component Analysis and Execution Workflow

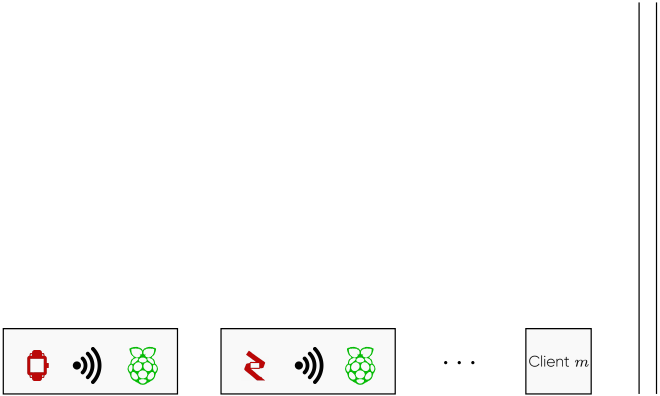


Figure 5: Client-server architecture.

Execution Flow

1. Sensors generate samples and streams them to the gateway over `mqtt`

System Architecture
Component Analysis and Execution Workflow

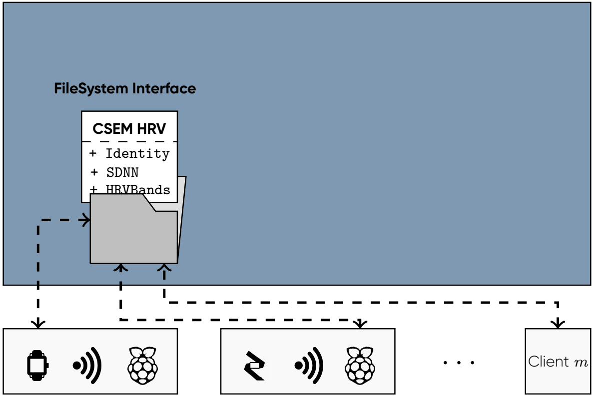
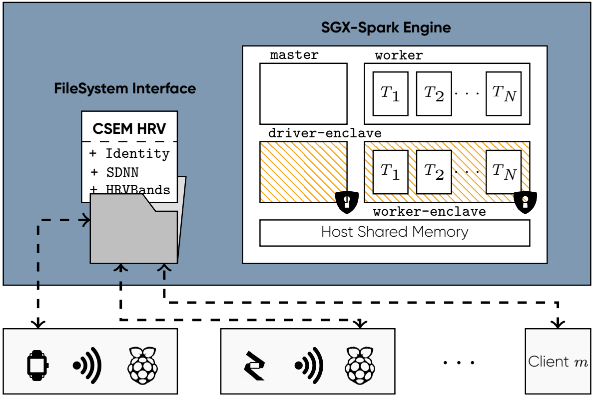


Figure 5: Client-server architecture.

Execution Flow

1. Sensors generate samples and streams them to the gateway over `mqtt`
2. Gateways aggregate data, encrypts it and sends it to the server's filesystem interface over SFTP

System Architecture
Component Analysis and Execution Workflow



Execution Flow

1. Sensors generate samples and streams them to the gateway over `mqtt`
2. Gateways aggregate data, encrypts it and sends it to the server's filesystem interface over SFTP
3. `SGX-Spark` streaming job monitors the FS, batch-processes data, and saves encrypted results back in the FS

Figure 5: Client-server architecture.

System Architecture
Component Analysis and Execution Workflow

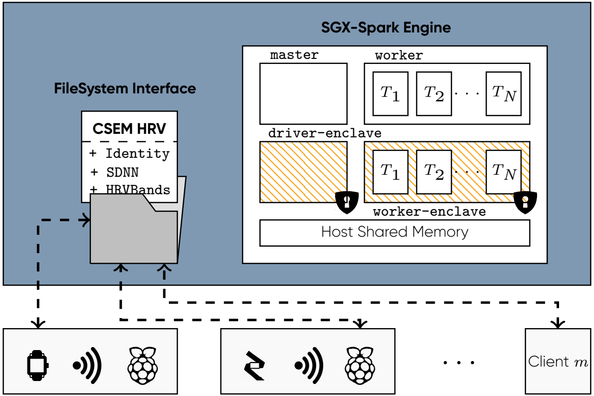


Figure 5: Client-server architecture.

Execution Flow

1. Sensors generate samples and streams them to the gateway over `mqtt`
2. Gateways aggregate data, encrypts it and sends it to the server's filesystem interface over SFTP
3. `SGX-Spark` streaming job monitors the FS, batch-processes data, and saves encrypted results back in the FS
4. Gateways fetch result files

Evaluation & Results
Experimental Setup & Metrics

Three Execution Modes:

1. Vanilla Spark
2. SGX-Spark w/o Enclaves
3. **SGX-Spark w/ Enclaves**

Two Algorithms:

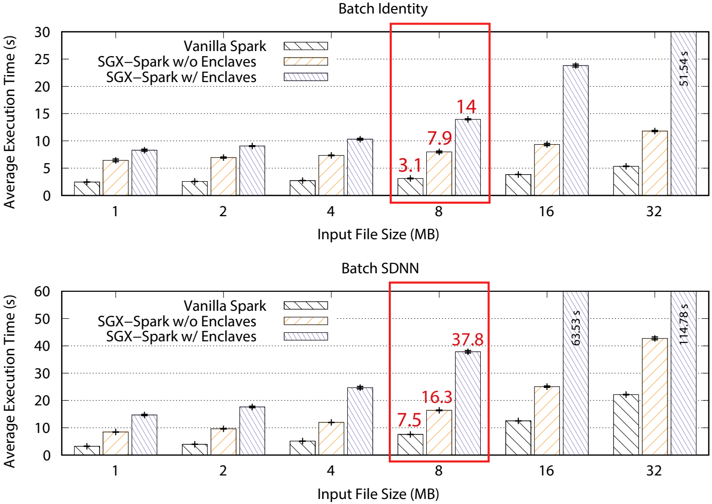
1. `Identity` (Batch & Stream)
2. `SDNN` (Batch & Stream)

Two Metrics:

1. Elapsed Time
2. Avg. Batch Processing Time

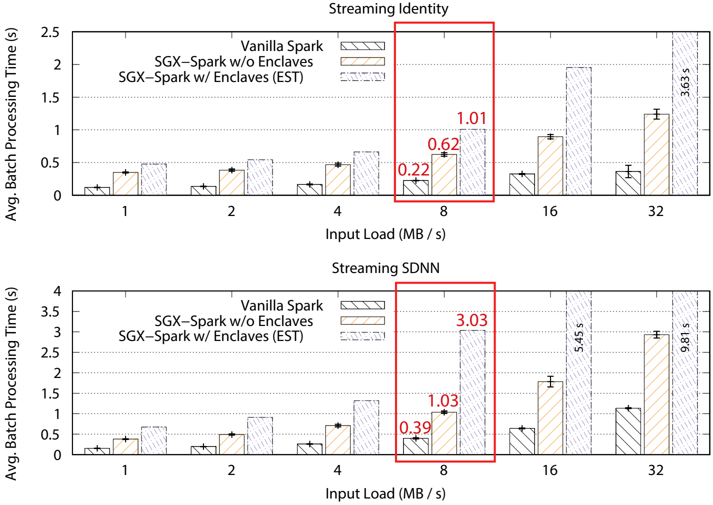
Workload	`s_rate` (samples / sec)	Input Load
Batch - Small	$\{44, 89, 178, 356, 712, 1424\}$	$\{1, 2, 4, 8, 16, 32\}$ kB
Stream - Small	$\{44, 89, 178, 356, 712, 1424\}$	$\{1, 2, 4, 8, 16, 32\}$ kB / sec
Batch - Big	$\{44, 89, 178, 356, 712, 1424\} * 1024$	$\{1, 2, 4, 8, 16, 32\}$ MB
Stream - Big	$\{44, 89, 178, 356, 712, 1424\} * 1024$	$\{1, 2, 4, 8, 16, 32\}$ MB / sec

Evaluation & Results
Results: Batch Execution - Big Load



Evaluation & Results

Results: Stream Execution - Big Load



Conclusion

- Introduced a PoC of a **privacy-preserving streaming platform**.
    - Introduces $4\times - 5\times$ slowdown vs. **vanilla Spark Streaming** (load < 4 MB per second).
    - Requires **no changes** to the application source code.
- Further lines of research:
    - Perform an economical evaluation of the cost of deploying our system to the cloud: **how expensive is privacy?**
    - **Reduce the TCB** on the client side leveraging TEEs for low-power devices (*e.g.* ARM TrustZone).

Conclusion

- Introduced a PoC of a **privacy-preserving streaming platform**.
  - Introduces $4\times - 5\times$ slowdown vs. **vanilla Spark Streaming** (load < 4 MB per second).
  - Requires **no changes** to the application source code.
- Further lines of research:
  - Perform an economical evaluation of the cost of deploying our system to the cloud: **how expensive is privacy?**
  - **Reduce the TCB** on the client side leveraging TEEs for low-power devices (*e.g.* ARM TrustZone).

Thank you very much for your attention! Questions?