PG: Byzantine Fault-Tolerant and Privacy-Preserving Sensor Fusion with Guaranteed Output Delivery

Chenglu Jin*[1], Chao Yin*[2,1], Marten van Dijk[1,2], Sisi Duan[3], Fabio Massacci[2], Michael K. Reiter[4], Haibin Zhang[5]

Email: chenglu.jin@cwi.nl

* Shared first-authorship

[1] Centrum Wiskunde & Informatica (CWI) Amsterdam, [2]Vrije University Amsterdam, [3]Tsinghua University, [4]Duke University, [5]Yangtze Delta Region Institute of Tsinghua University, Zhejiang

CWI

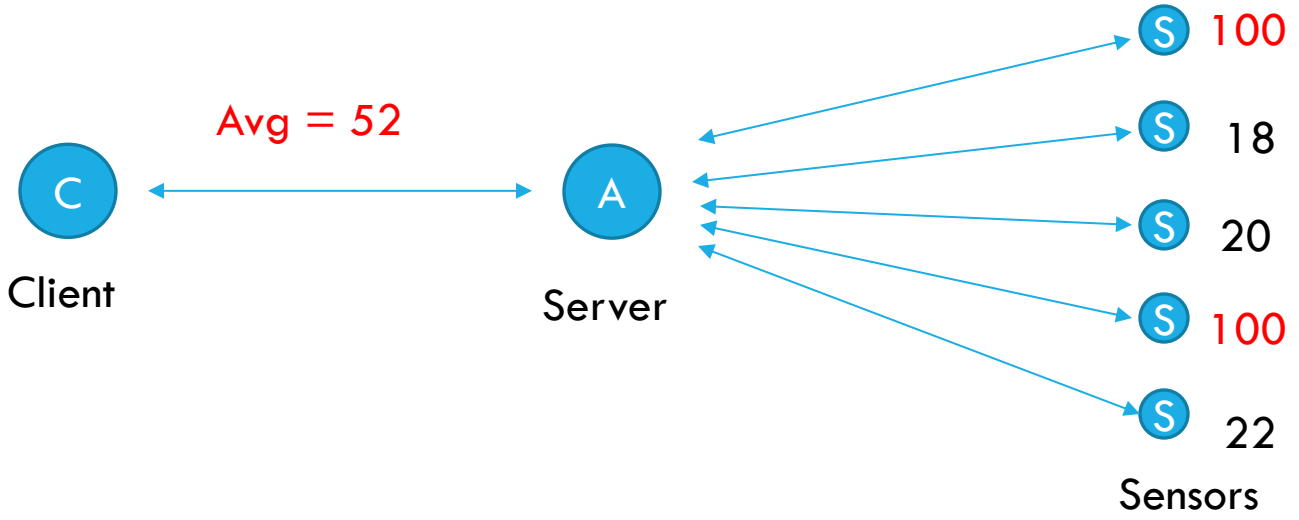
Outline

- Technical overview
- Background
- P0, privacy-preserving and fault-tolerant
- P1, achieving guaranteed output delivery (GOD) in the crash failure model
- P2, achieving GOD in the Byzantine failure model
- P3, realizing privacy against malicious servers
- Experimental evaluation

CWI

Sensor Data Fusion

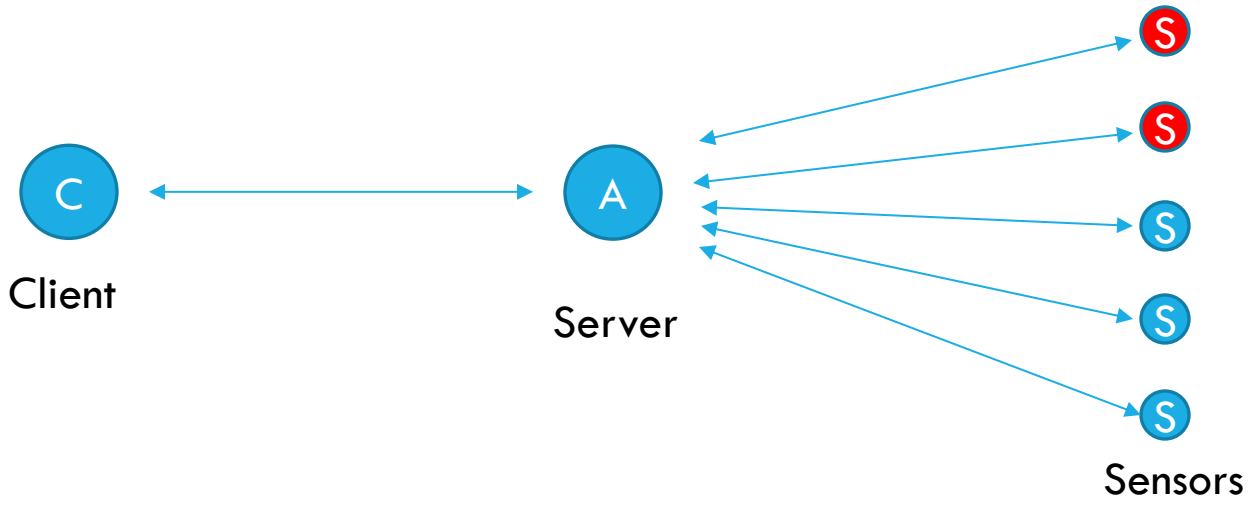
- Combine multiple sensor data to produce more dependable and accurate information. E.g., sensor networks, smart metering.
- In particular, we are focusing on the client-server-sensor model.
- Pollution attack: a small fraction of faulty sensor data can lead to a large error in the aggregated result.



CWI

PG: Privacy-Preserving and Fault-Tolerant Sensor Fusion

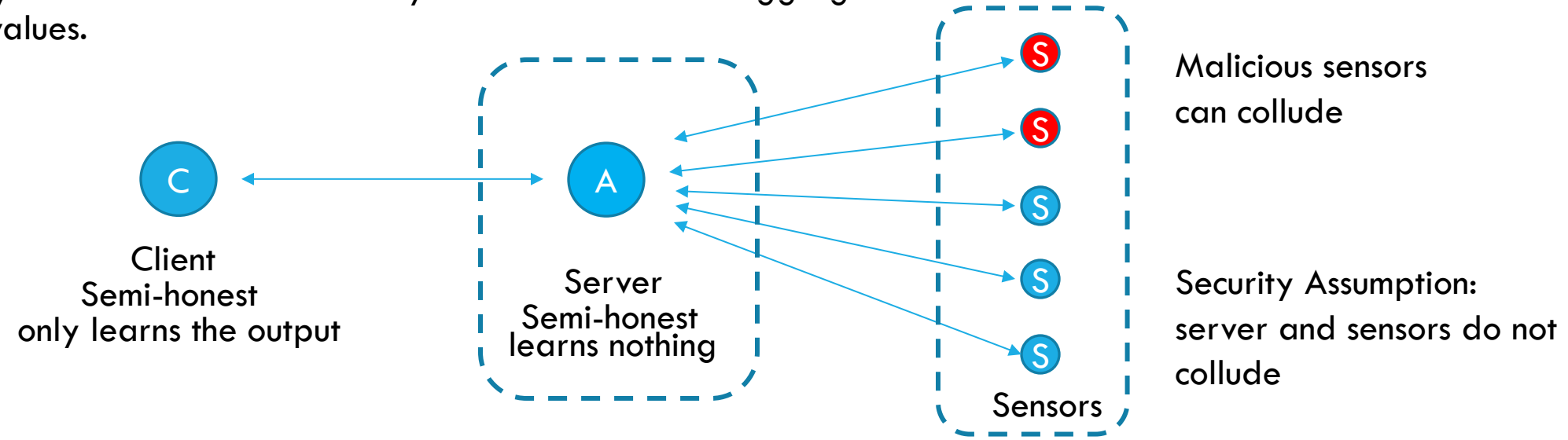
1. Fault tolerant algorithms (FTA).
   - Formally defend against pollution attacks given a bound of the fraction of malicious sensors among all the sensors.
   - E.g. Marzullo's algorithm ensures that the result must contain the correct value if at most g out of 2g+1 sensors are malicious



CWI

PG: Privacy-Preserving and Fault-Tolerant Sensor Fusion

1. Fault tolerant algorithms (FTA).
   - Formally defend against pollution attacks given a bound of the fraction of malicious sensors among all the sensors.
   - E.g. Marzullo's algorithm ensures that the result must contain the correct value if at most g out of 2g+1 sensors are malicious
2. Garbled circuits (GC).
   - Privacy: protect the privacy of individual sensor inputs
   - Authenticity: the server should faithfully return the client the aggregated result rather than some arbitrary values.



CWI

Garbled Circuits

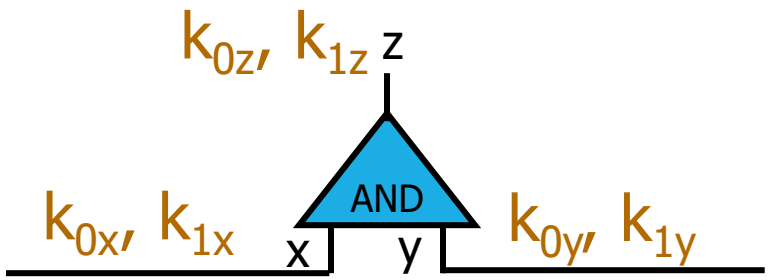
Bit values on each wire

0,1

Randomly Chosen Labels

k_{0z}, k_{1z}

Boolean gate



Encrypted Truth Table

$E_{k0x}(E_{k0y}(k_{0z}))$

$E_{k1x}(E_{k1y}(k_{1z}))$

$E_{k1x}(E_{k0y}(k_{0z}))$

$E_{k0x}(E_{k1y}(k_{0z}))$

Only one ciphertext will be decrypted

- Reveals nothing more than the output, because of the randomly chosen labels

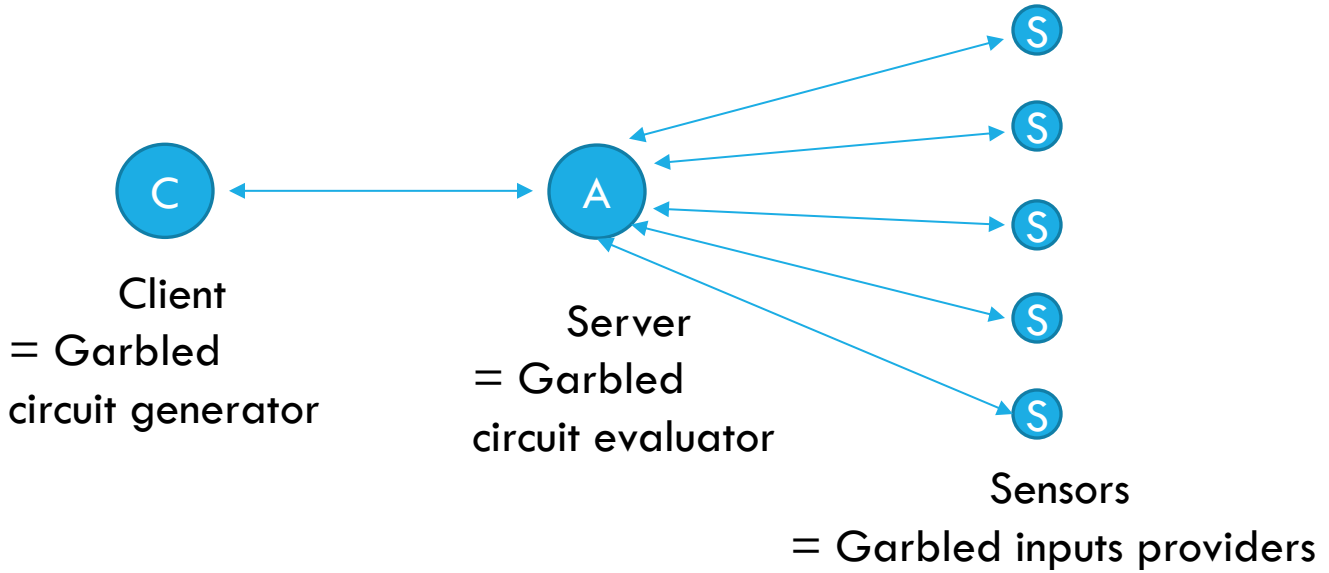
CWI

P0: Apply GC and FTA to Our System

- We use a pre-shared secret key between the client and each sensor to derive the same randomness needed to garble the circuit or the inputs
- This key should not be exposed to the server.

1. The client garbles a fault-tolerant algorithm *f()* that performs the sensor fusion and sends the garbled circuit *Gb(f)* to the server.
2. Server fetches garbled inputs $En(X_i)$ from the sensors
3. Server evaluates the garbled circuit
4. Garbled output *Y* is sent to Client
5. Client decodes De(Y) to get f(X)

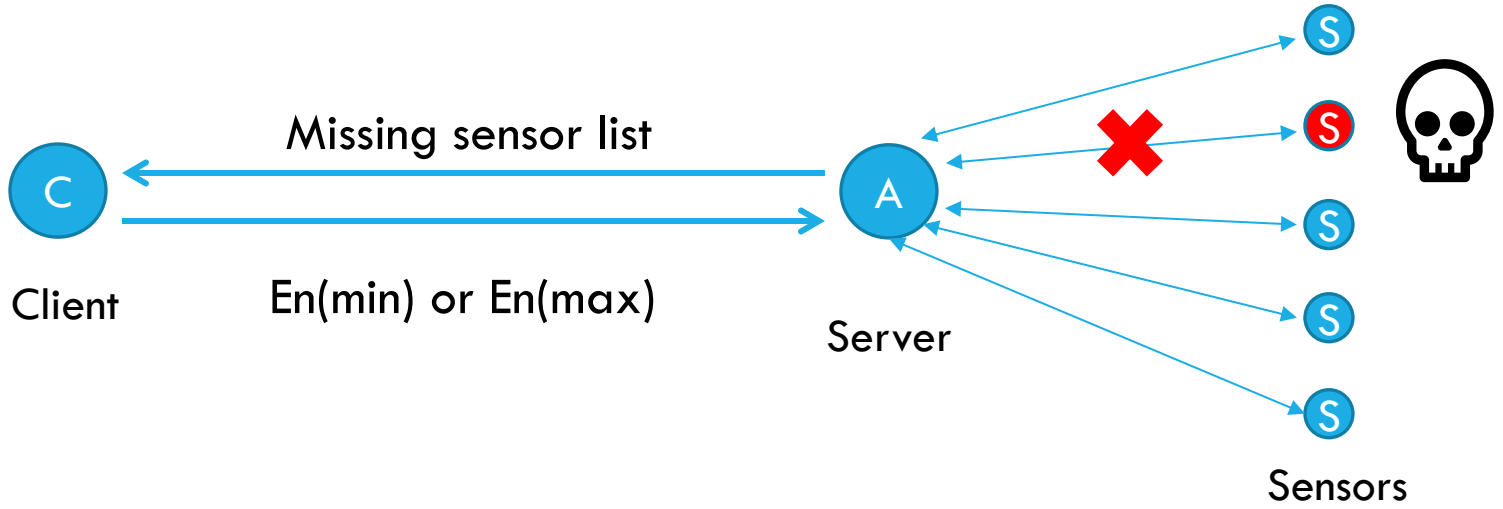
- Input Privacy
- Tolerate incorrect sensor inputs



CWI

P1: Achieving GOD in the Crash Failure Model

- The completion of P0 protocol requires all the sensors to provide an input.
- Easy DoS attack by compromising just one sensor and not sending anything.
- One more round of interaction: if the server does not receive all the garbled inputs before a timer expires, it requests from the client for the missing sensor inputs that encode the minimum or maximum.
- The missing inputs become faulty inputs that will be tolerated by FTA
- GOD is achievable because our protocol is fault-tolerant.



CWI

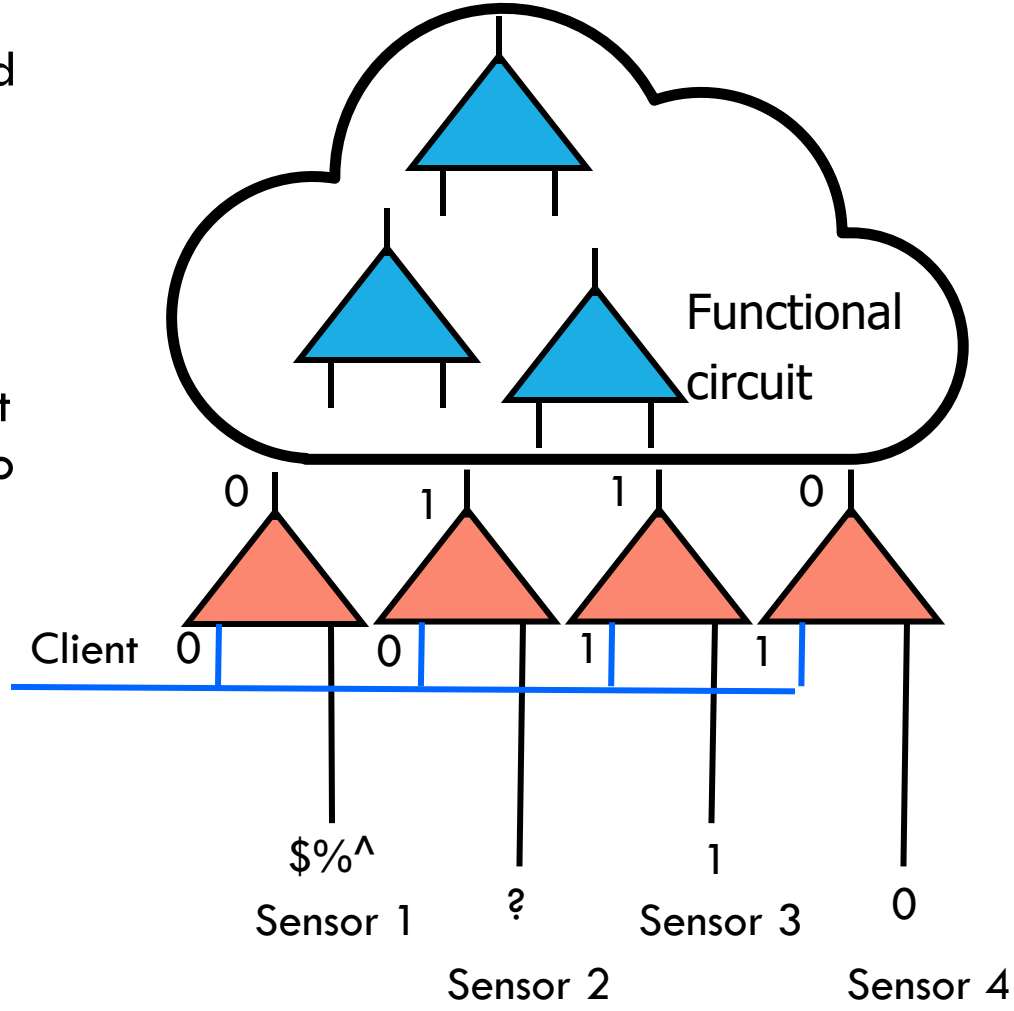
P2: Achieving GOD in the Byzantine Failure Model

- Byzantine failure model means the malicious sensors can do anything they want
- Sensors may send ill-formed inputs (not correctly garbled inputs)
- Easy DoS attack by compromising one sensor and always sending ill-formed inputs. The aggregated result is not decodable by the client.
- Adding checking gates (encrypted truth tables) besides the functional circuit to detect ill-formed inputs:
  - All-zero-strings, instead of output labels, are encrypted by valid input labels pairs
  - If all inputs from a sensor pass the check, use them in the functional circuit; otherwise request valid labels that encode minimum/maximum from the client.
  - For N-bit inputs, we need to add additional N/2 checking gates
- Cover all possible behaviors of the malicious sensors: not sending anything, sending ill-formed values, sending valid but incorrect values

CWI

P3: Secure against a Malicious Server

- A malicious server may trick the client and ask for a valid input of an honest sensor, such that it gets two valid labels of the same wire in the circuit.
  - Violation of privacy and authenticity.
- Filter gate (encrypted truth table): if the server "**claims**" some sensor input labels are missing/ill-formed, the client sends labels to help decrypt one row of the truth table to get valid output labels that encodes the min/max; otherwise the client sends labels to forward the sematic values of sensor inputs to the output labels.
- Implicitly transfer the max/min values of the functional circuit inputs via the filter gate truth tables
- Guarantee: at most one label per wire can be revealed to the server
- Still need to assume that the malicious server and the malicious sensors do not collude



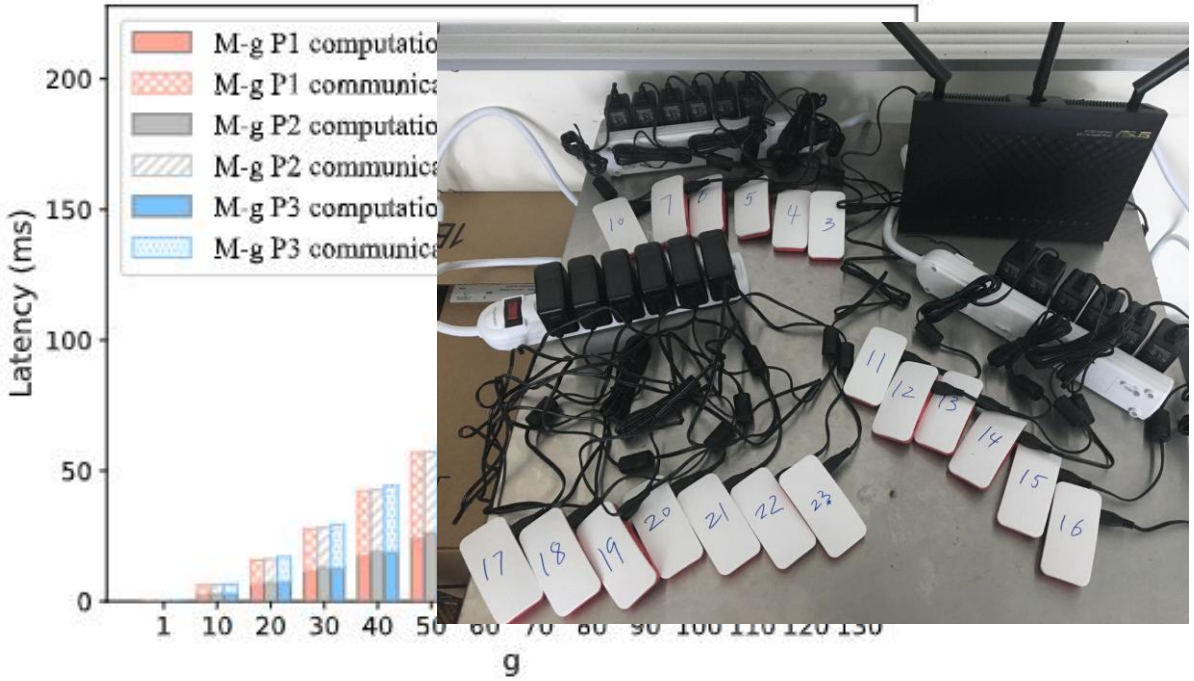
CWI

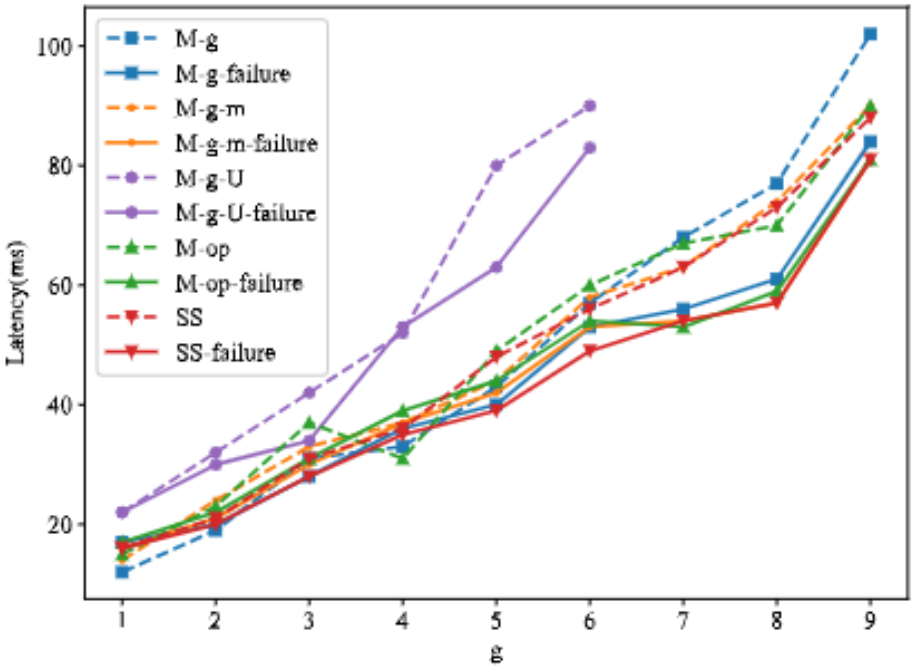
Performance Evaluation

Check out our paper for details

- Five fault tolerant sensor fusion algorithms, together with their optimized circuit designs.
- Cloud-based evaluation: Each sensor/server/client is one AWS node. Used up to 261 sensor nodes
- CPS-based evaluation: up to 19 sensors (Raspberry Pi Zero W) with Wifi connection

algorithms
M-g-U
M-g
M-g-m
M-op
SS





CWI

Summary

- Design an efficient and scalable framework for privacy-preserving and Byzantine fault-tolerant sensor fusion. It fits the resource constrained sensors.
- Develop new techniques to achieve guaranteed output delivery in GC when a fraction of sensors are Byzantine malicious.
- Extend our system to be secure against a malicious server.
- Optimize the circuits designs realizing the fault-tolerant sensor fusion algorithms.
- Evaluate the performance of our system on AWS cloud with up to 261 sensors and a cyber-physical system with up to 19 sensors.
- Source code: https://figshare.com/articles/software/PG_source_code/25669026

Thank you! Questions?

Email: chenglu.jin@cwi.nl