University of Glasgow

Securing Industrial Control Systems An E2E Integrity Verification Approach

Sye-Loong Keoh, Ken Wai-Kin Au

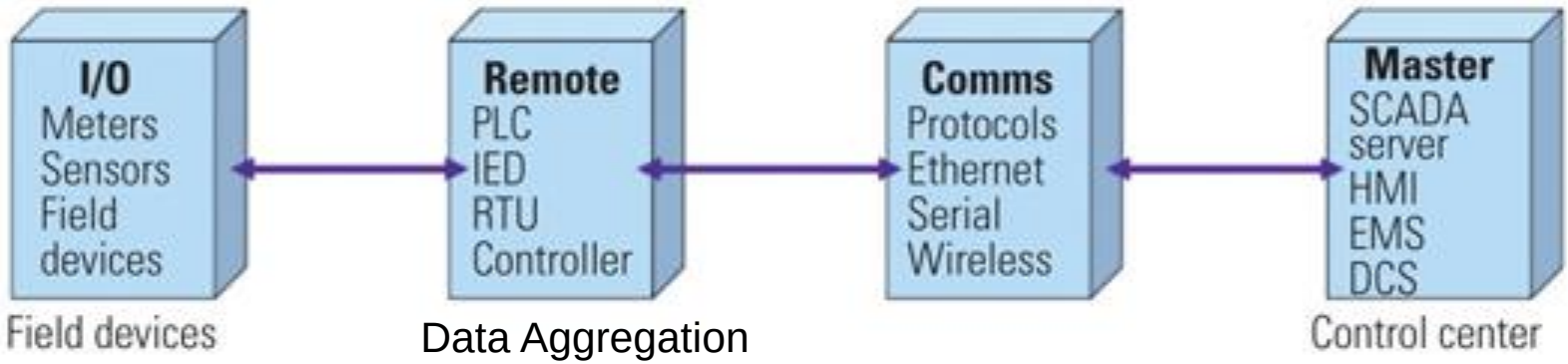
School of Computing Science
University of Glasgow

Zhaohui Tang

School of Infocomm
Republic Polytechnic, Singapore

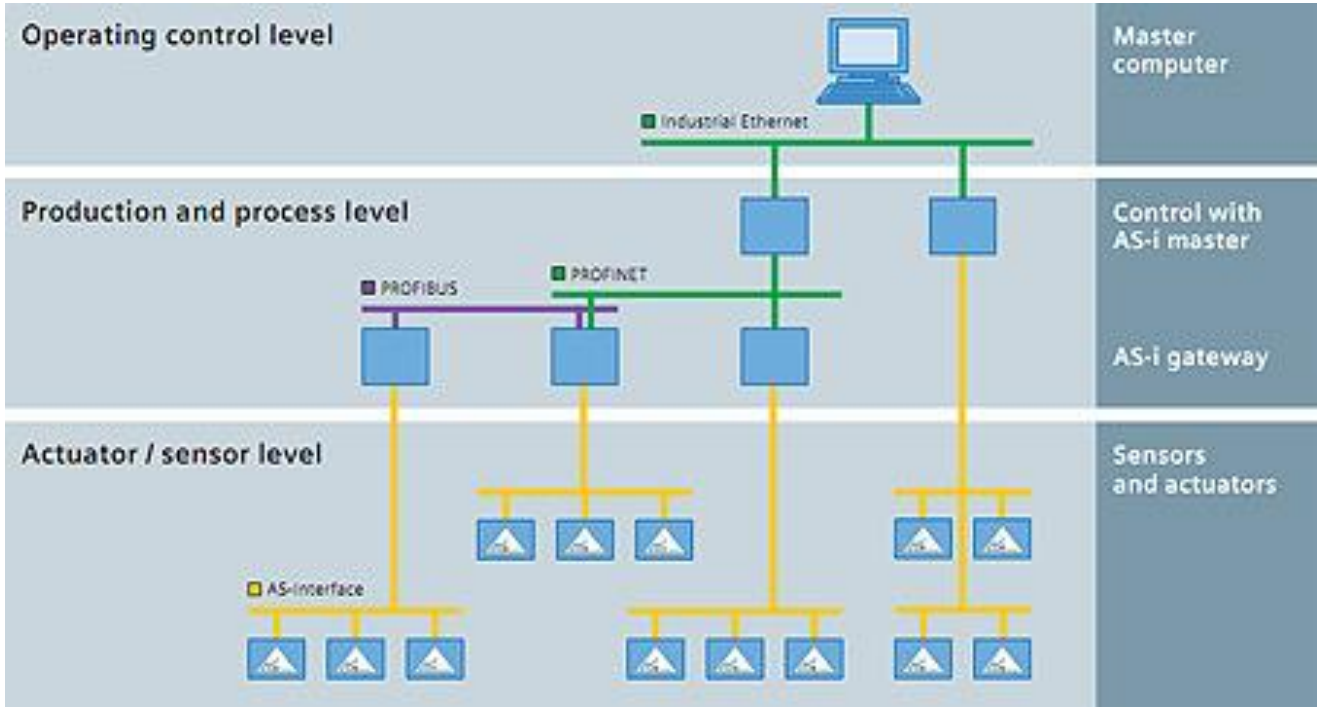
Introduction

- Industrial Control Systems (ICS) are used to monitor and control industrial facilities and processes:
  - **Power Grid:** generation, distribution, load balancing and billing
  - **Chemical and Nuclear Plant:** control of safety critical processes.
  - **Gas and Water Facilities:** collect measurements from PLC/sensors and issue commands to actuators.



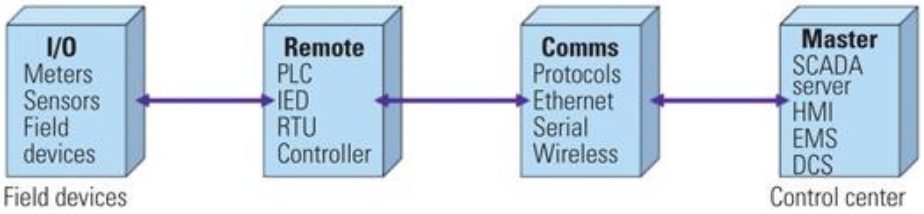
An Example ICS Architecture

- Master ensures data exchange with the slaves (field controller) by means of cyclic polling.
- Data collected at the field controller can be aggregated.



[Siemens]

Integrity of Sensor Data



Vulnerabilities
- fraud
- selectively reporting
- single point of failure

Central controller

$(m'_{s1}, m'_{s2}, m'_{s3})$

field controllers

$\{m_{s1}, m_{s2}, m_{s3}\}$ $\{m_{s4}, m_{s5}, m_{s6}\}$ …

field devices

m_{s1} m_{s2} m_{s3} m_{s4} m_{s5} m_{s6}

fd_1 fd_2 fd_3 fd_4 fd_5 fd_6 …

Security Requirements

- **Data Integrity** – the measurements on the field devices must reflect the current state of the instruments in the plant.

  ✗ modification and tampering.

- **Data Origin Authentication** – important to ensure that measurements are taken using the designated field devices.

  ✗ spoofing

- **Secure Data Aggregation** – though data are aggregated to save bandwidth, the central controller (Back End Master) must have the ability to check the integrity and data origin.

  ✓ integrity ✓ data origin

University of Glasgow

Background: Chameleon Hashing

- Chameleon Hashing
  - Hash function with a trapdoor for finding collusion.
  - Associated with a pair of public-private key.
  - Private-key serves as the trapdoor.

- Properties
  - Chameleon Hash Value [CHV] = CHA_HASH(*y*, m, r).
  - given trapdoor *x*, find a collision [**m', r'**] where m' ≠ m and r' ≠ r.
  - Hence [CHV] = CHA_HASH(*y*, m', r').

- Chameleon Signature
  - Apply traditional signature, e.g., DSA, RSA, ECC to Chameleon Hash.

System Setup

Field Devices	Field Controllers	Back-end
Trapdoor Hash Key ***(x)***	Chameleon Hash Key ***(y)***	Chameleon Hash Key ***(y)***
Trapdoor Chameleon Hash Function	Chameleon Hash Function	Chameleon Hash Function
Device ID ***(Id_{fd})***		

Secure Channel

Secure Channel

Key Generation

- Krawczyk and Rabin's discrete logarithm construction

  - Two primes p and q are randomly generated such that $p = kq+1$ where q is a large prime factor.

- An element g of order q in $\mathbb{Z}_p^*$ is chosen so that the private key, $x \in \mathbb{Z}_p^*$. The public-key, y is generated as

$$y = g^x \bmod p$$

Generation of Chameleon Hash

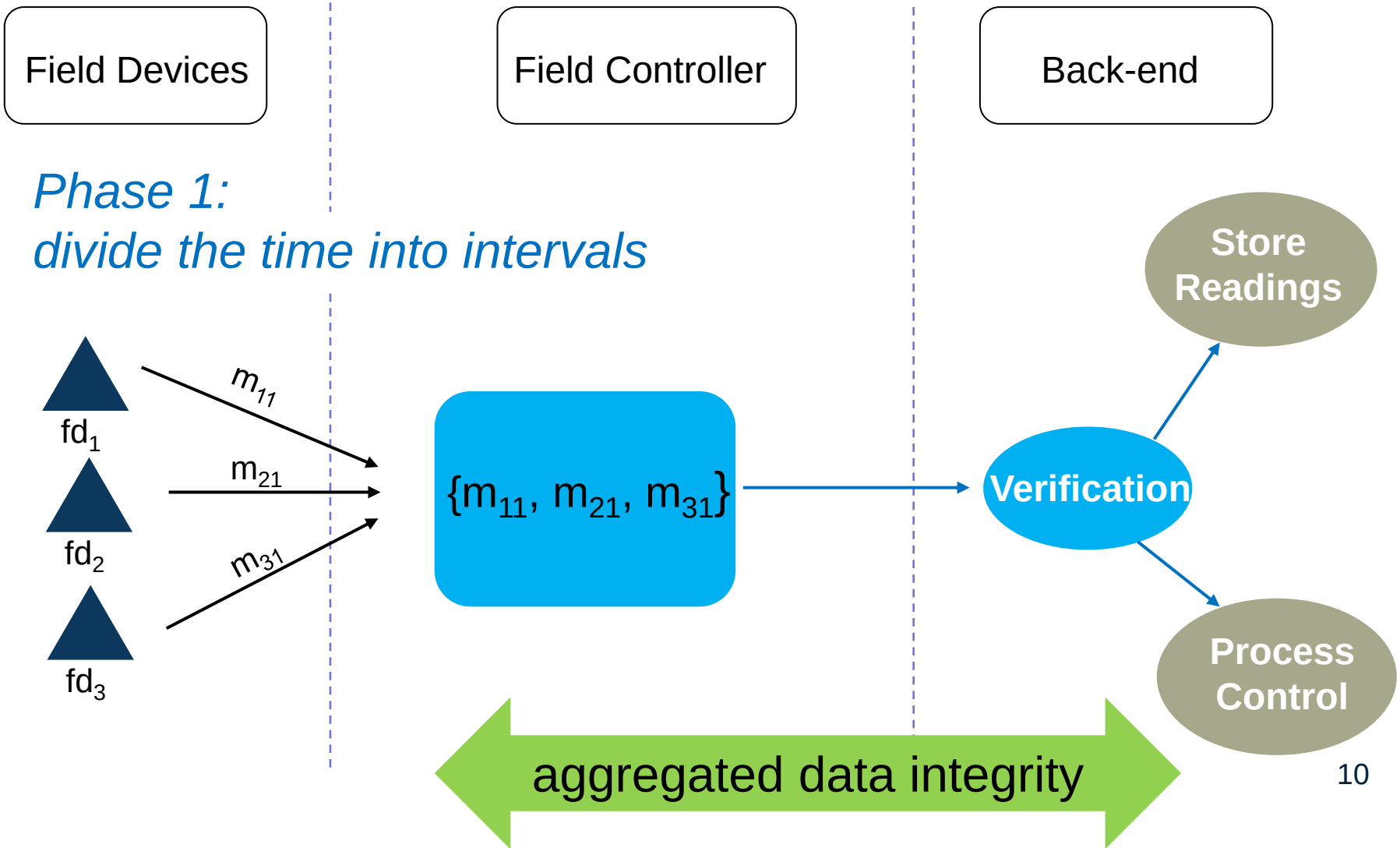
- Given a message $m \in \mathbb{Z}_p^*$, choose a random value $r \in \mathbb{Z}_p^*$, the Chameleon Hash denoted as *CHV* can be computed as:

$$\text{CHA\_Hash}(m,r) = g^m y^r \bmod p$$

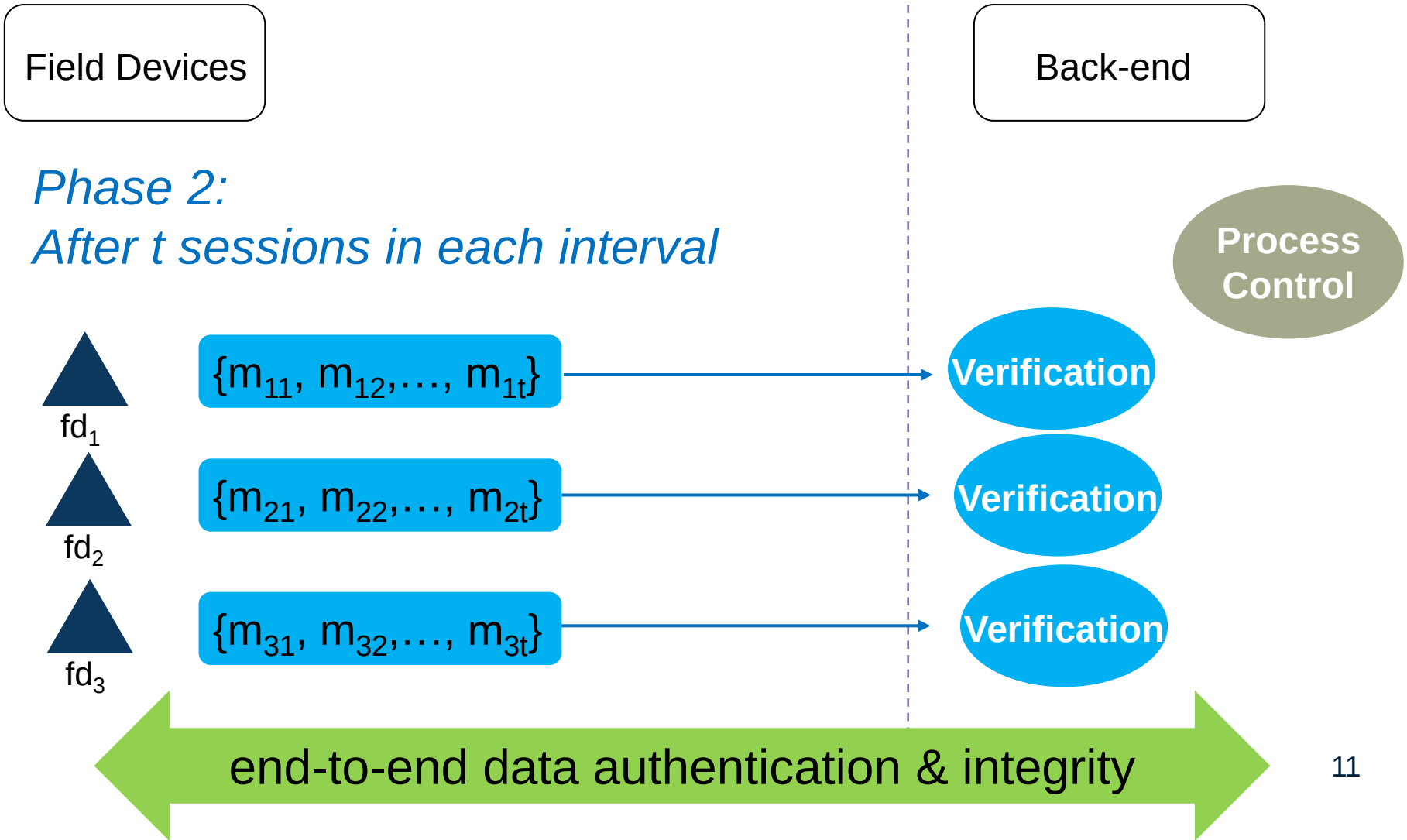
- Only the field devices have the ability to produce the same Chameleon Hash using a different message, *m'* such that CHA_Hash(*m,r*) = CHA_HASH(*m',r'*) by solving *r'*

$$m + xr = m' + xr' \bmod p$$

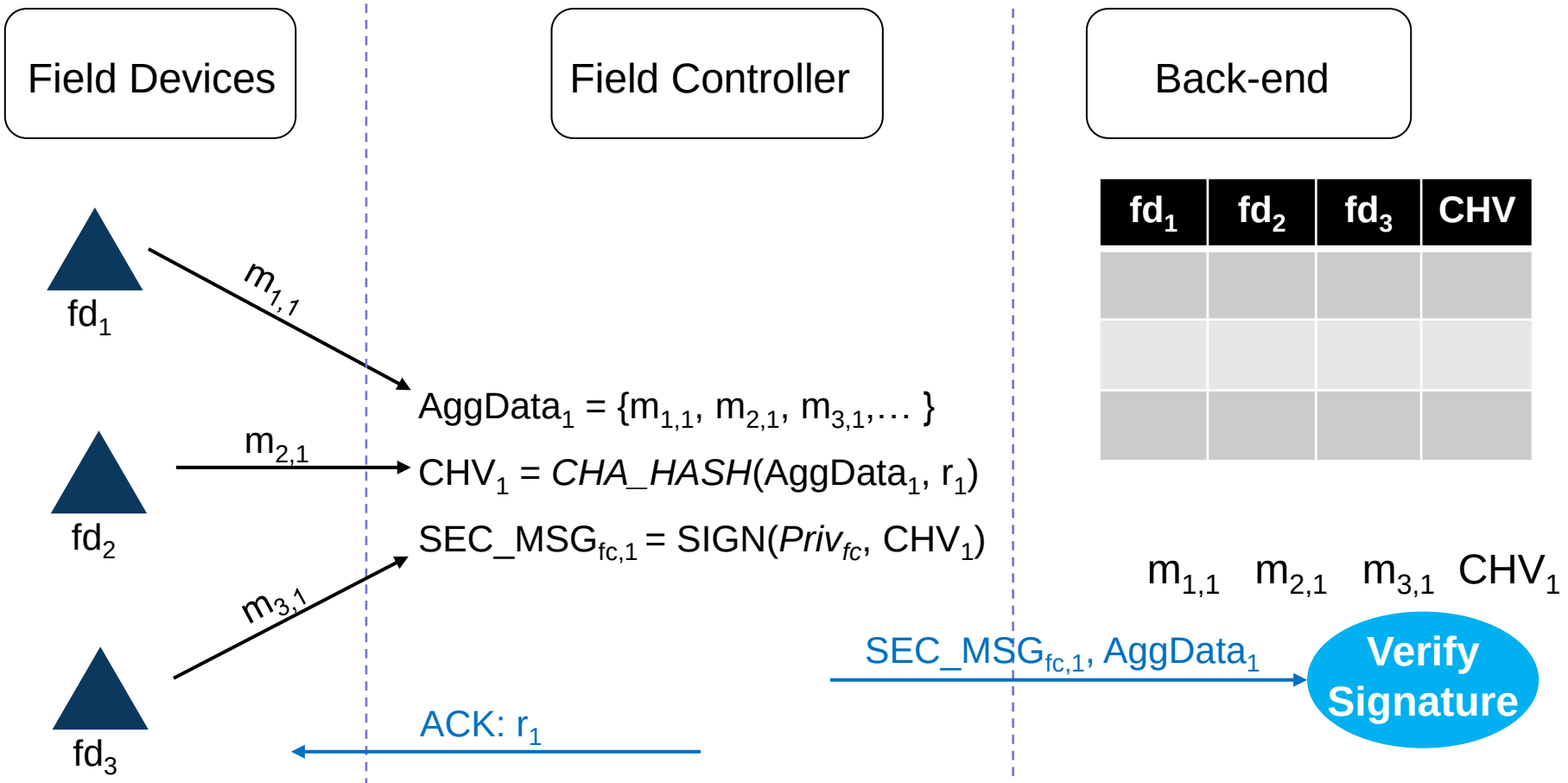
Protocol Overview



Protocol Overview

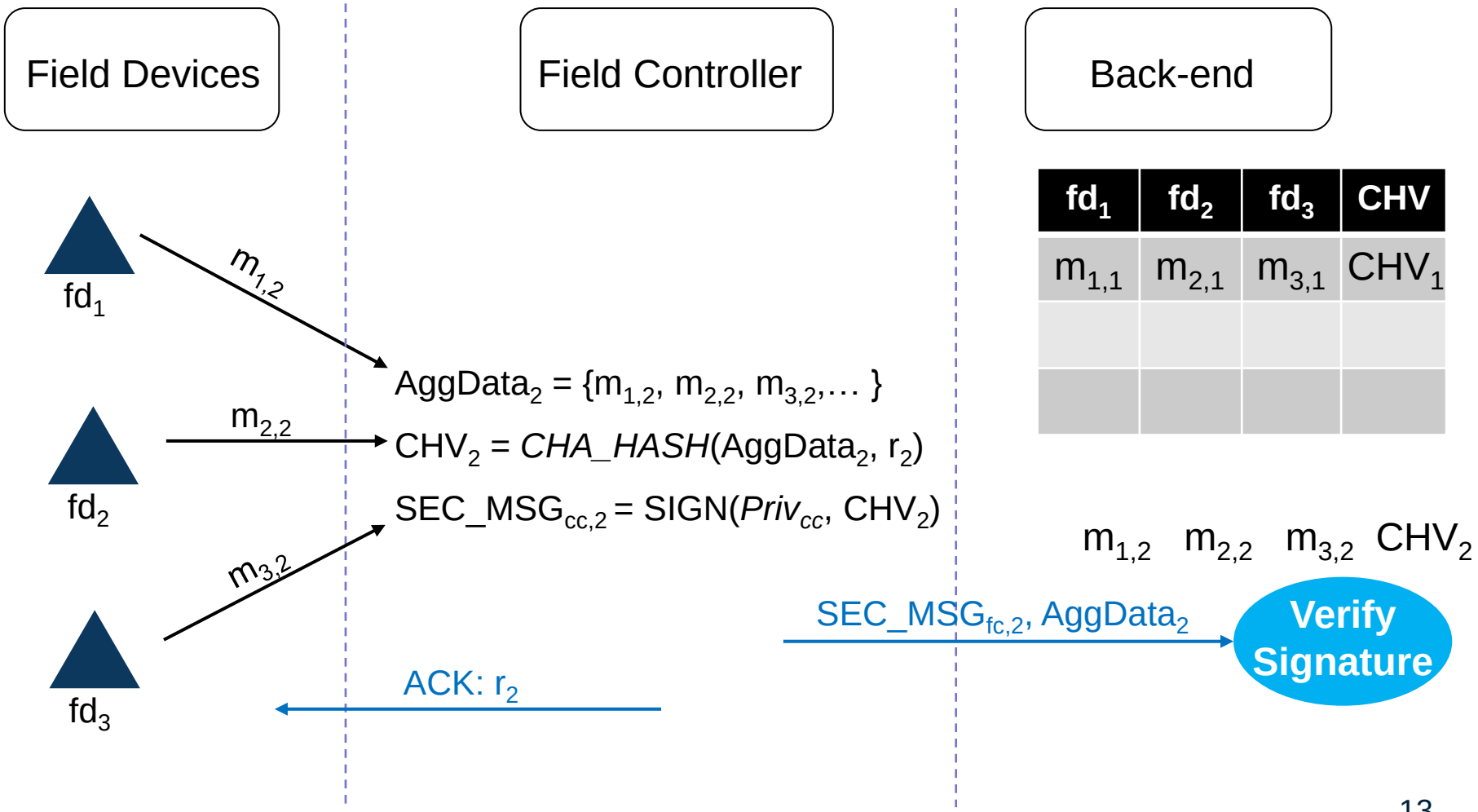


Secure End-to-End Data Aggregation



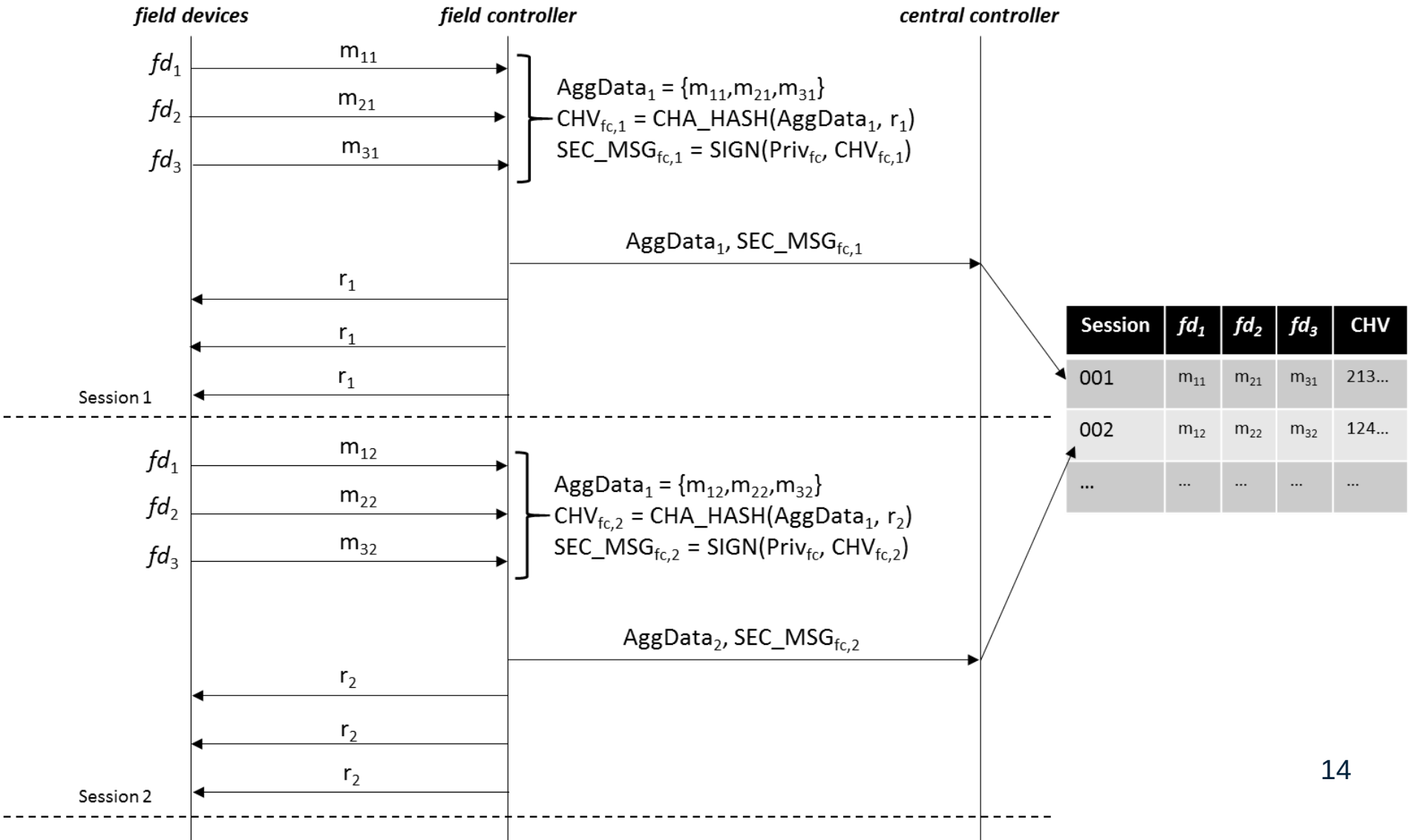
Phase 1: interval 1:Session 1

Secure End-to-End Data Aggregation



Phase 1: interval 1: Session 2

Phase 1: Protocol Summary



Session	fd_1	fd_2	fd_3	CHV
001	m_{11}	m_{21}	m_{31}	213…
002	m_{12}	m_{22}	m_{32}	124…
…	…	…	…	…

Phase 2: E2E Integrity Verification

Transmission of Evidence

- Time is divided into intervals, where each interval consists of *t* sessions.
- At the end of each interval, field devices choose an r_v where $1 \leq v \leq t$, so that

$$\text{CHA\_HASH}(m'_i, r'_i) = \text{CHA\_HASH}(AggData_v, r_v)$$

- *m'* denotes all the readings recorded by the field device *i* in the interval $\{Id_{fd,i}, m_{i,1}, m_{i,2}, \ldots, m_{i,t}\}$

Transmission of Evidence

- To verify this, we need to solve r'_i

$$r'_i \bmod p = (AggData_v + xr_v - m')x^{-1} \bmod p$$

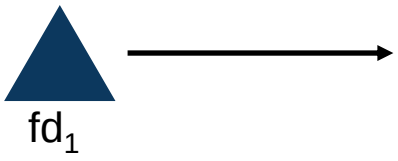
- However, field devices do not know $AggData_v$ (sent by the field controller). Instead they can compute a ***commitment*** that allows the back-end to verify integrity and authenticity.

$$\left\{ y^{-x} \bmod p \right\}, \left\{ \frac{y^{xr_V(-x)}}{y^{m'(-x)}} \bmod p \right\}$$

Delayed-Integrity-Verification

Field Devices

Back-end



fd_1

$m' = \{ID_{fd,i}, m_{1,1}, m_{1,2}, m_{1,3}, \ldots\}$

Find a collision (m', r')

m' = $\{ID_{fd,1}, m_{1,1}, m_{1,2}, m_{1,3}\ldots\}$

commitment: $y^{-x} \bmod p$

$$\frac{y^{xr_1(-x)}}{y^{m'(-x)}} \bmod p$$

fd_2

Any

e.g., using r_1

fd_3

r_1	r_2	r_3	…

fd_1	fd_2	fd_3	CHV
$m_{1,1}$	$m_{2,1}$	$m_{3,1}$	CHV_1 √
$m_{1,2}$	$m_{2,2}$	$m_{3,2}$	CHV_2
$m_{1,3}$	$m_{2,3}$	$m_{3,3}$	CHV_3

$ID_{fd,1}$

CHV_1

fd_1,commitment

Verify Hash

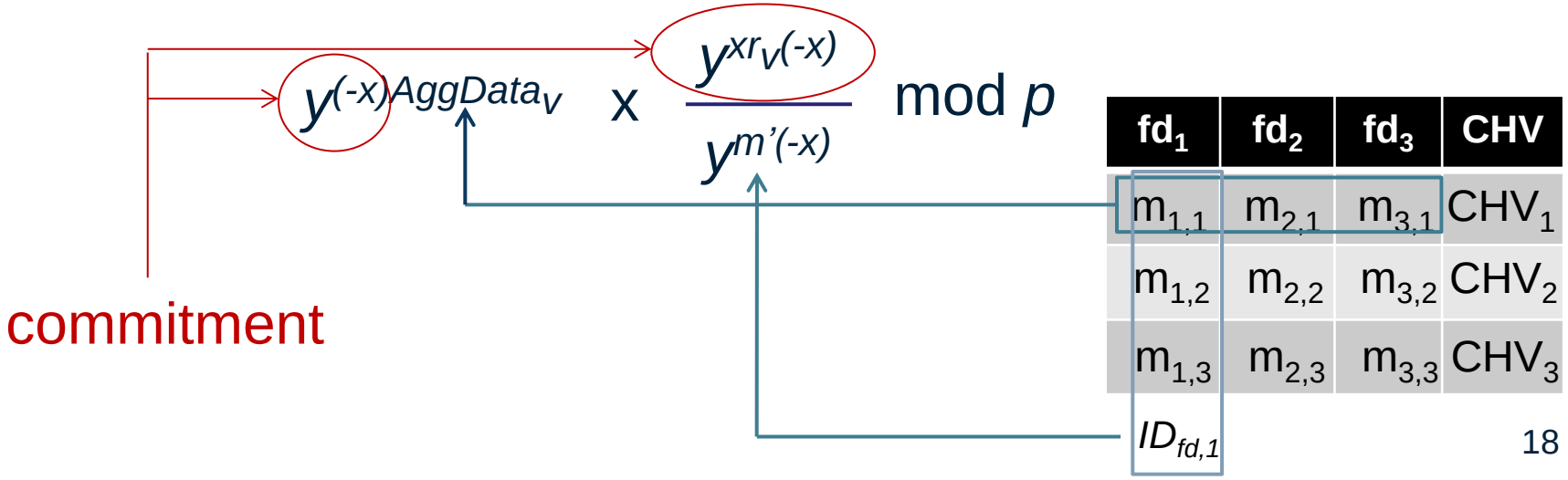
Phase 2

Integrity Verification

- We need to solve this:

$$r'_i \bmod p = (AggData_v + xr_v - m')x^{-1} \bmod p$$

- But, essentially we want to compute CHA_HASH(*m'*,*r'*), so we need $y^{r'_i} \bmod p$, which is



fd_1	fd_2	fd_3	CHV
$m_{1,1}$	$m_{2,1}$	$m_{3,1}$	CHV_1
$m_{1,2}$	$m_{2,2}$	$m_{3,2}$	CHV_2
$m_{1,3}$	$m_{2,3}$	$m_{3,3}$	CHV_3

University of Glasgow

Prototype Implementation

- Prototype was implemented using Java, and deployed on Raspberry Pi Model B+
  - CPU: 700 MHz Low Power ARM processor
  - Memory: 512 MB

- Preliminary performance results

Device	Operation	Time (ms)
Controller	Chameleon Hashing	0.955955 (PC)
Field Device	**Generation of Commitment**	**111.6 (Pi)**
Back End	Integrity Verification	2.288591 (PC)
Field Device	**Signature generation**	**5830 (Pi)**

University of Glasgow

Conclusions

- Our scheme provides:
  - Data Integrity
  - Data Origin Authentication
  - Secure Data Aggregation
- Novel use of Chameleon Hashing and Signature other than its traditional usage, to detect **misbehaviour of controllers or aggregators in ICS/SCADA**.
- Future work:
  - Implement the protocol on real hardware or ICS platform.
  - Protocol can be generalized to be used in AMI, body sensor network, or any network with a hierarchical structure.

Thank You

Sye-Loong Keoh
School of Computing Science
University of Glasgow
SyeLoong.Keoh@glasgow.ac.uk