Security in Sensor Networks

11.1.2006

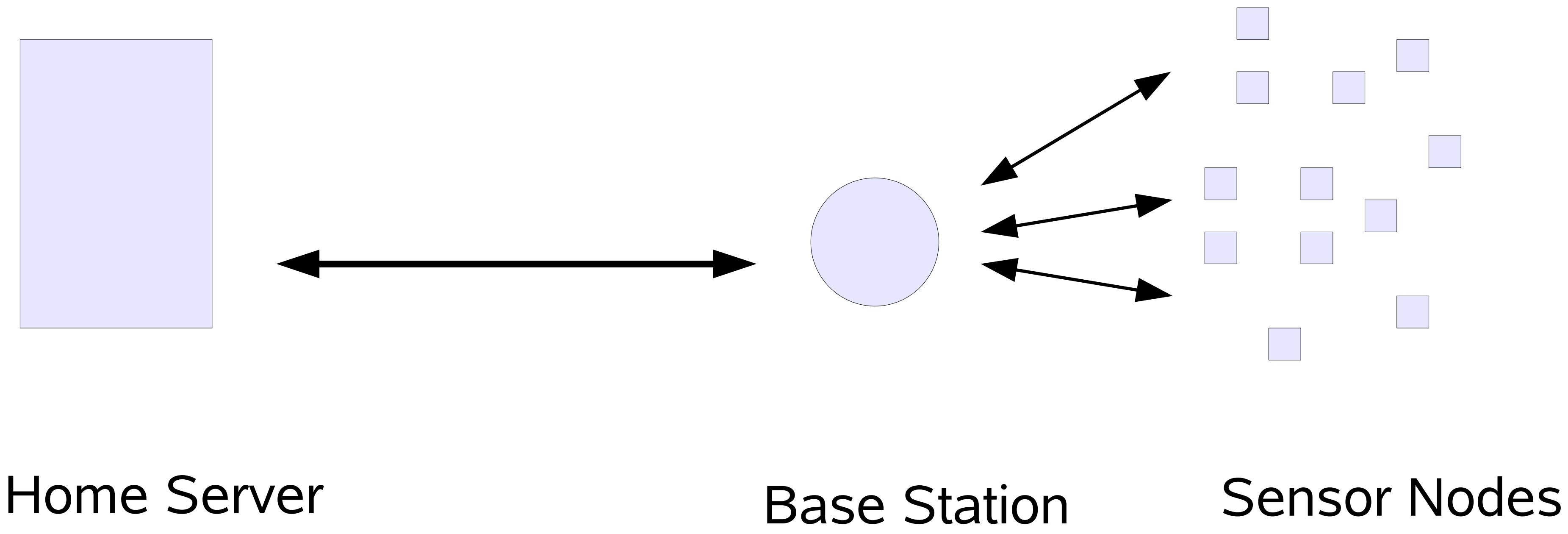
Presentation by Ziroli Plutschow

Outline

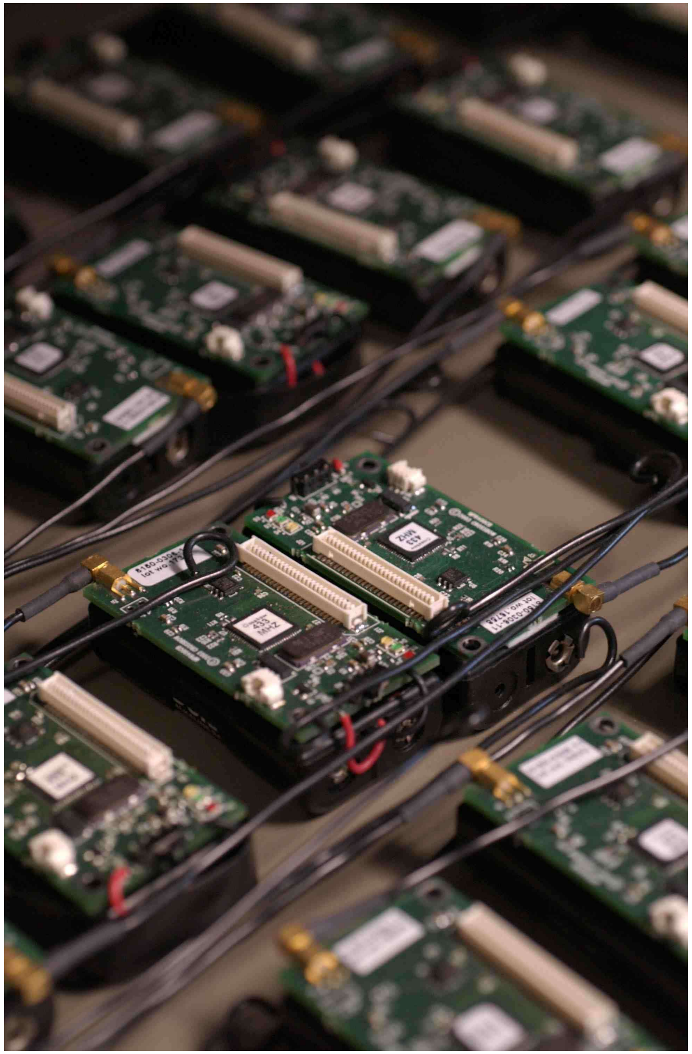
- Introduction into Sensor Networks
  - Security Issues
- Overview: Key Establishment Schemes
- Secure Information Aggregation (SIA)
  - Problem definition
  - Attacker model
  - Excursion: cryptography
  - General Approach
  - Example: Median
  - Hierarchical Aggregation

Sensor Networks

Sensor Network Concept



Applications of Sensor Networks



- Traffic Monitoring
- Wildlife Tracking
- Weather Monitoring
- Military Applications
- Building Security
- Building Automation

Special Security Set-Up

- No Public Key Cryptography
  Use symmetric cryptography
- Attacker has physical access to Sensor Node
  Use independent shared keys for any potential communication channel. (-scalability)
  → Key Establishment Schemes
  Tamper resistant packaging for key (-expensive)

Research Topics

- Key Establishment Schemes
- Secure Routing
- Secure Information Aggregation
- Efficient Cryptographic Primitives

  hash- / one-way - functions, PRG

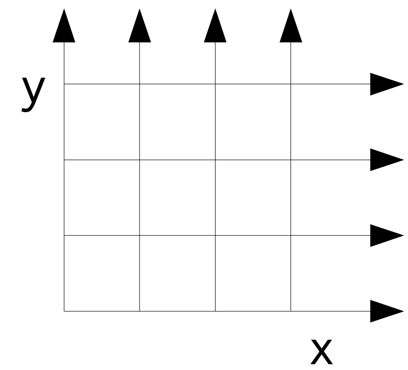
  Public-Key (elliptic curve)

Key Establishment Schemes 1

- Every node shares a key with each other node
  →$O(n^2)$ different keys, memory $O(n)$ per node
- Location Information
  -node shares keys with neighbors
  (maybe base station, home server, aggregator)
  → memory $O(const)$
- Probabilistic
  -node holds a subset of the generated keys
  -node has d neighbors → memory $O(n/d)$

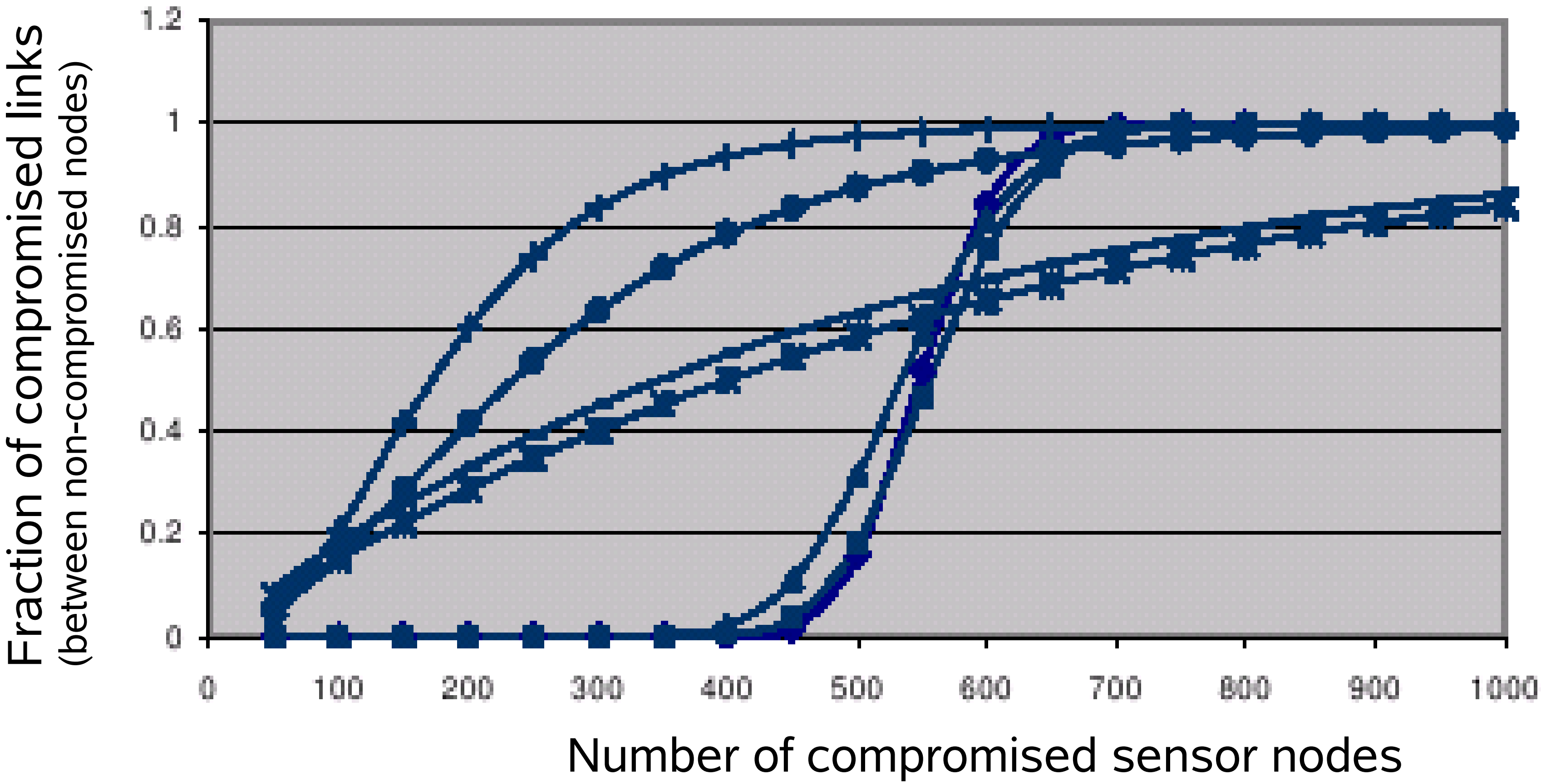
Key Establishment Schemes 2

- Peer intermediary
  - node i has (x_i,y_i)-position
  - $\rightarrow$ memory $O(n^{1/2})$, but trust every node



- Polynomial based
  - random 2-dim polynomial $p(x,y)$
  - gets $p(x_i,y)$ and $p(x,y_i)$
  - degree t: $\rightarrow$ memory $O(t)$
  - allows t compromised sensor nodes
  - TinyKeyMan for TinyOS

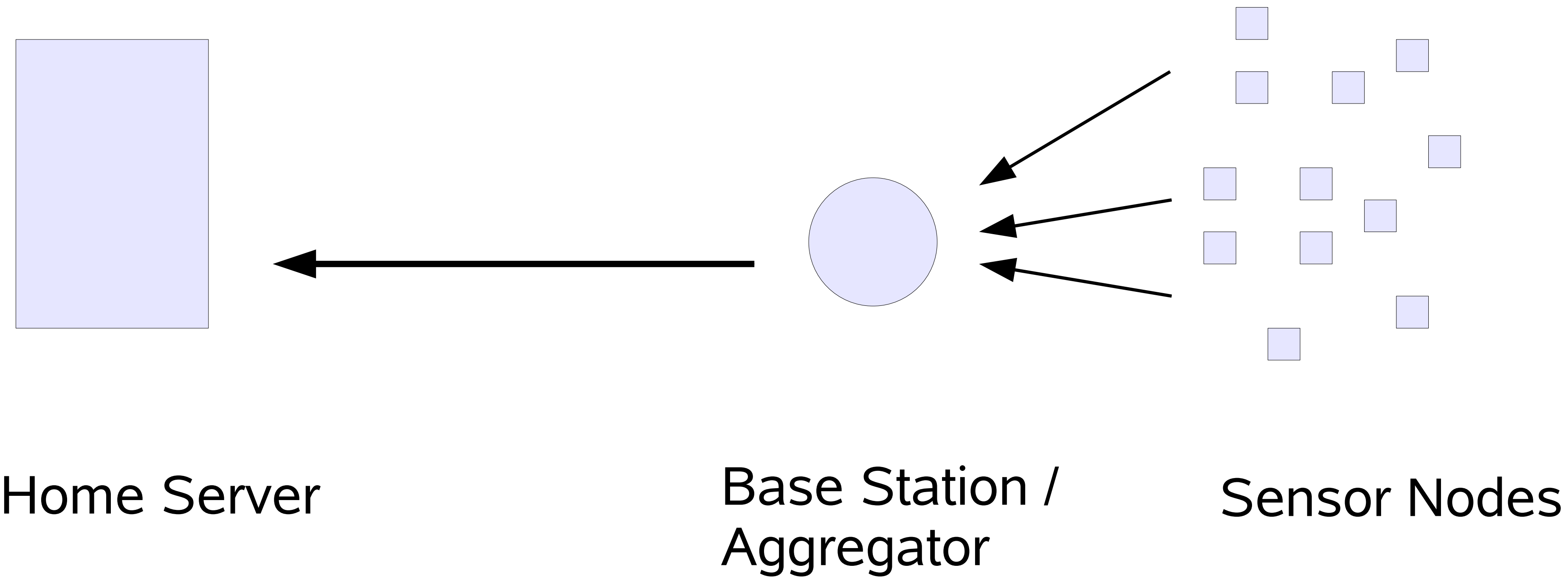
Key Establishment Benchmark



From Paper: *Establishing Pairwise Keys in Distributed Sensor Networks* by D.Liu and P.Ning, NCSU

Secure Information Aggregation

- Problem Setting:



→ **Goal:** Home server accepts only true value

SIA: Attacker Model

- Corrupted / compromised aggregator

  Attacker has full control (stealthy attack)
- Corrupted / compromised sensor nodes

  Attacker has full control (stealthy attack)
- No DoS

  Radio based communication → physical
- Routing

  Uncorrupted nodes are connected

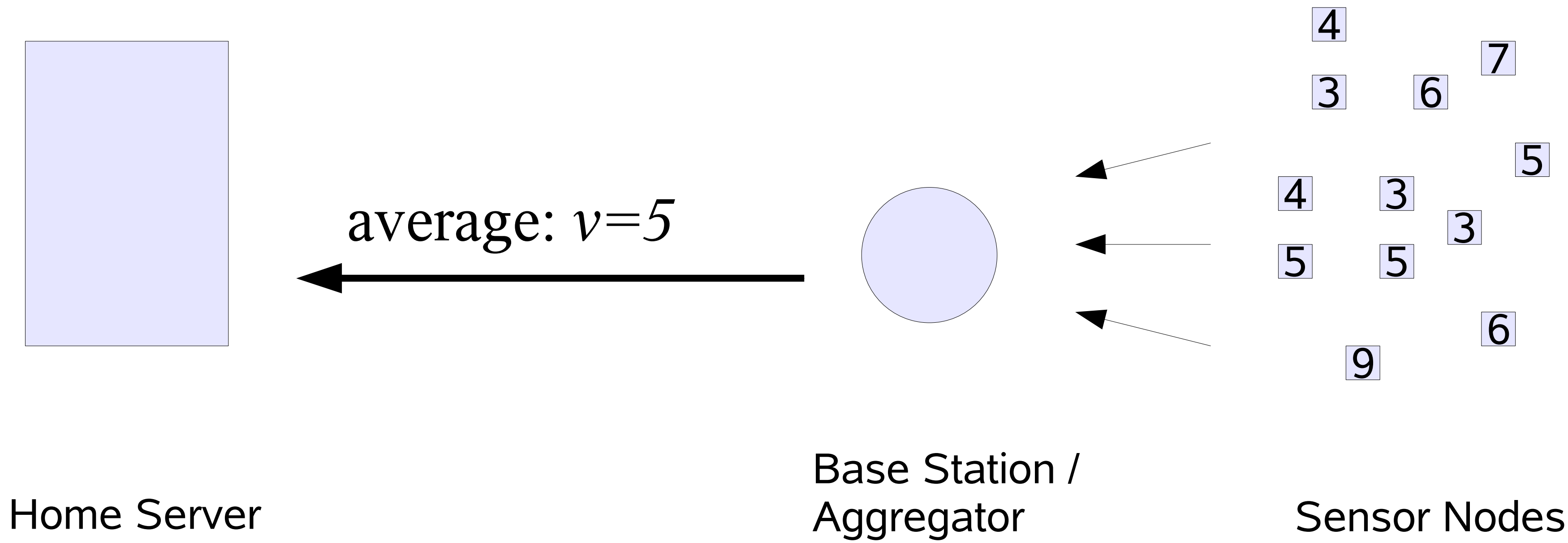
SIA: Key Setup

- Each Sensor Node
  - Unique Id
  - Shares a key with home server and aggregator
    2 keys per node

→ Home server and aggregator are able to authenticate the messages from sensor nodes.

SIA: Example, compute average

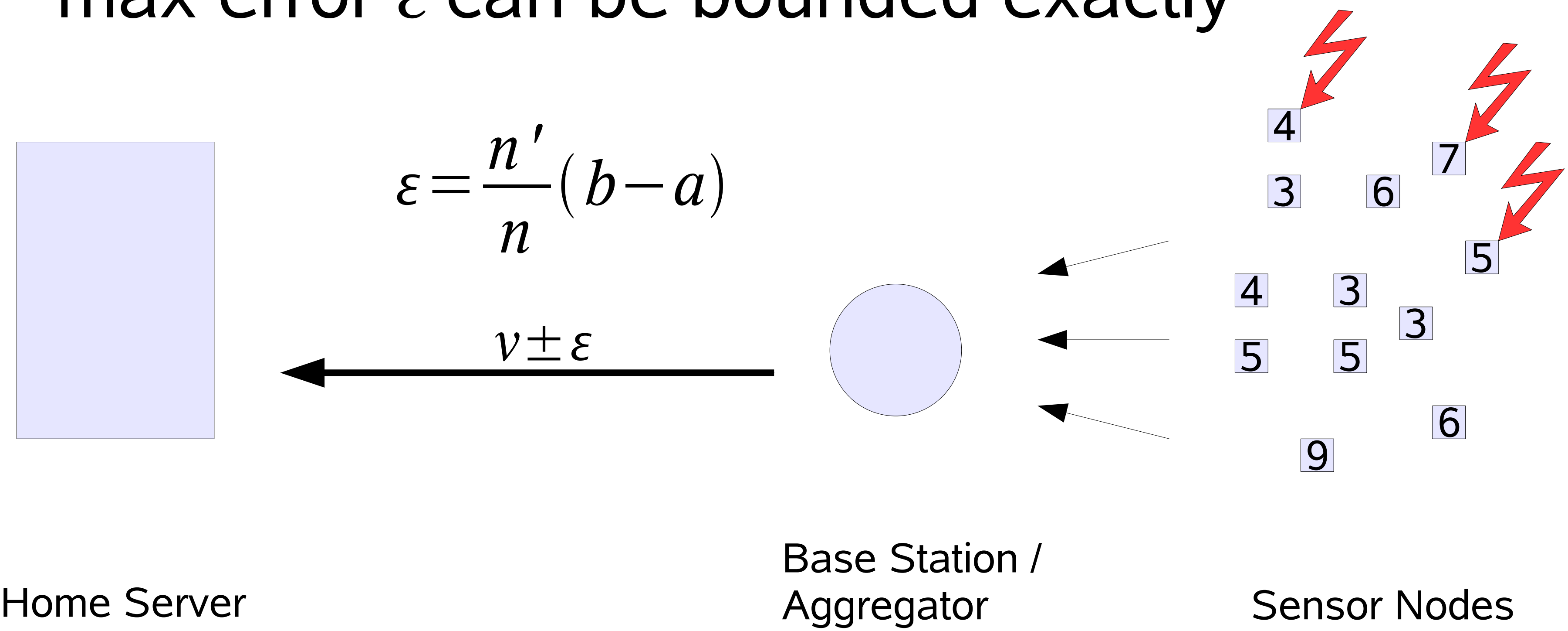
- 12 sensors, range 1…9, honest



SIA: Example, compute average

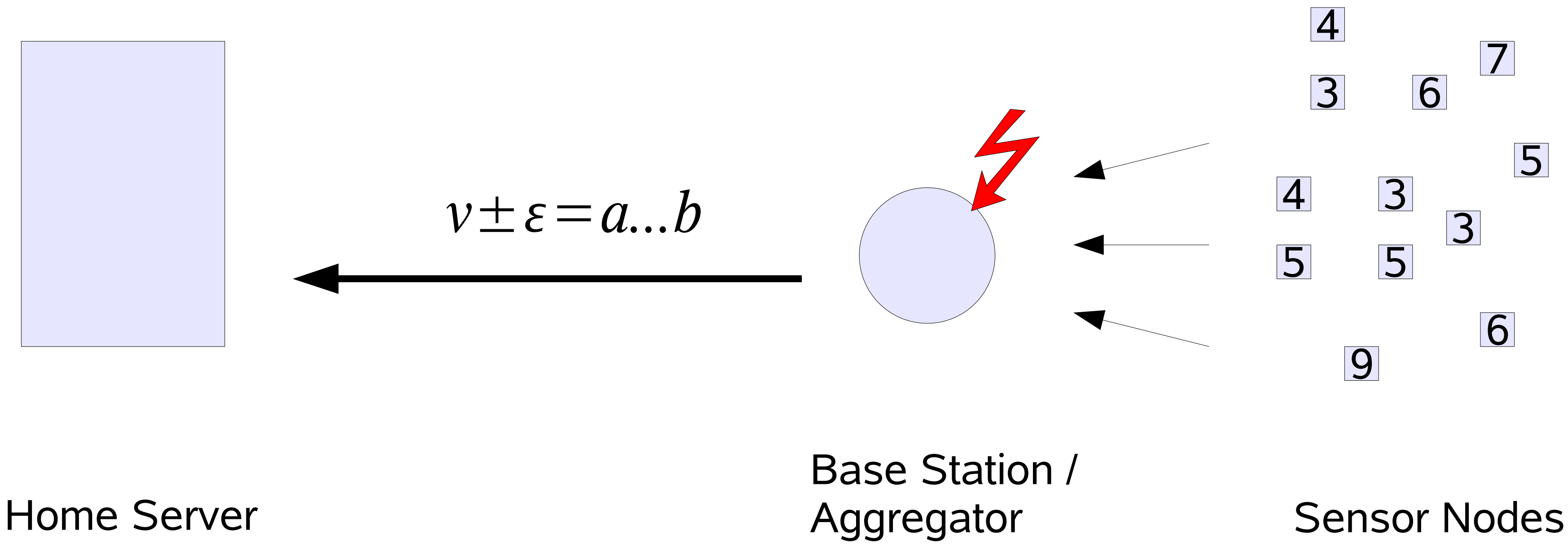
- n sensors, range $a...b$, n' corrupted sensors
- max error ε can be bounded exactly

$$\varepsilon = \frac{n'}{n}(b-a)$$



SIA: Example, compute average

- n sensors, range $a \ldots b$, corrupted aggregator
- max error: $\varepsilon = b - a$



→ SIA can help

Minimize ε (corrupted aggregator)

- Aggregator sends all signed sensor values to home server.
  - very inefficient
- SIA: Agg. proves that he aggregated correct
  Cryptographic techniques
  - commitment scheme
  - interactive proof

Cryptographic Hash Function

- Hash $y=h(x)$: $\{0,1\}^* \rightarrow \{0,1\}^n$

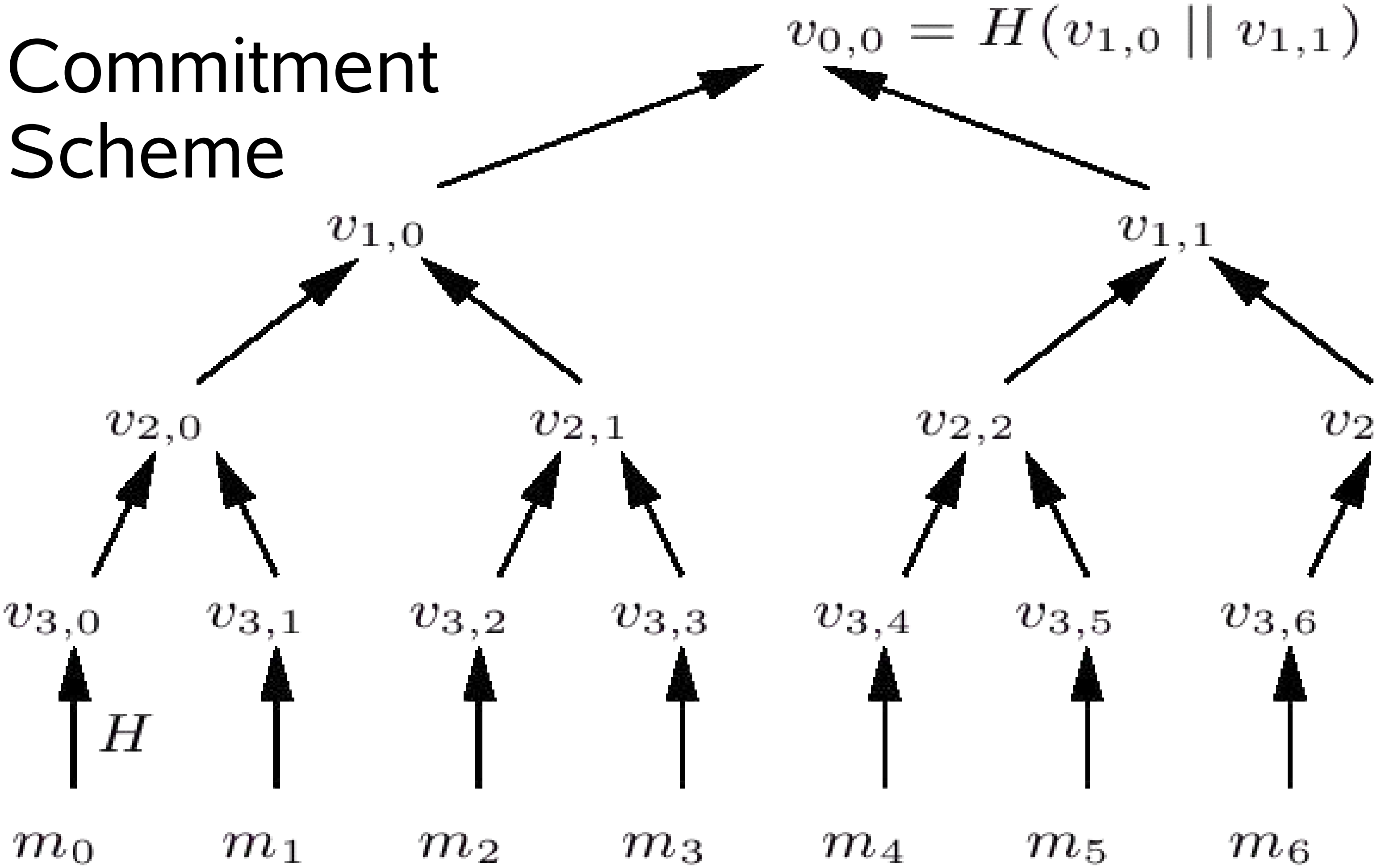
  one-way:
    given y, you can not calculate x

  2nd pre-image resistance:
    given x and y,
    you can not calculate a x' with h(x')=y

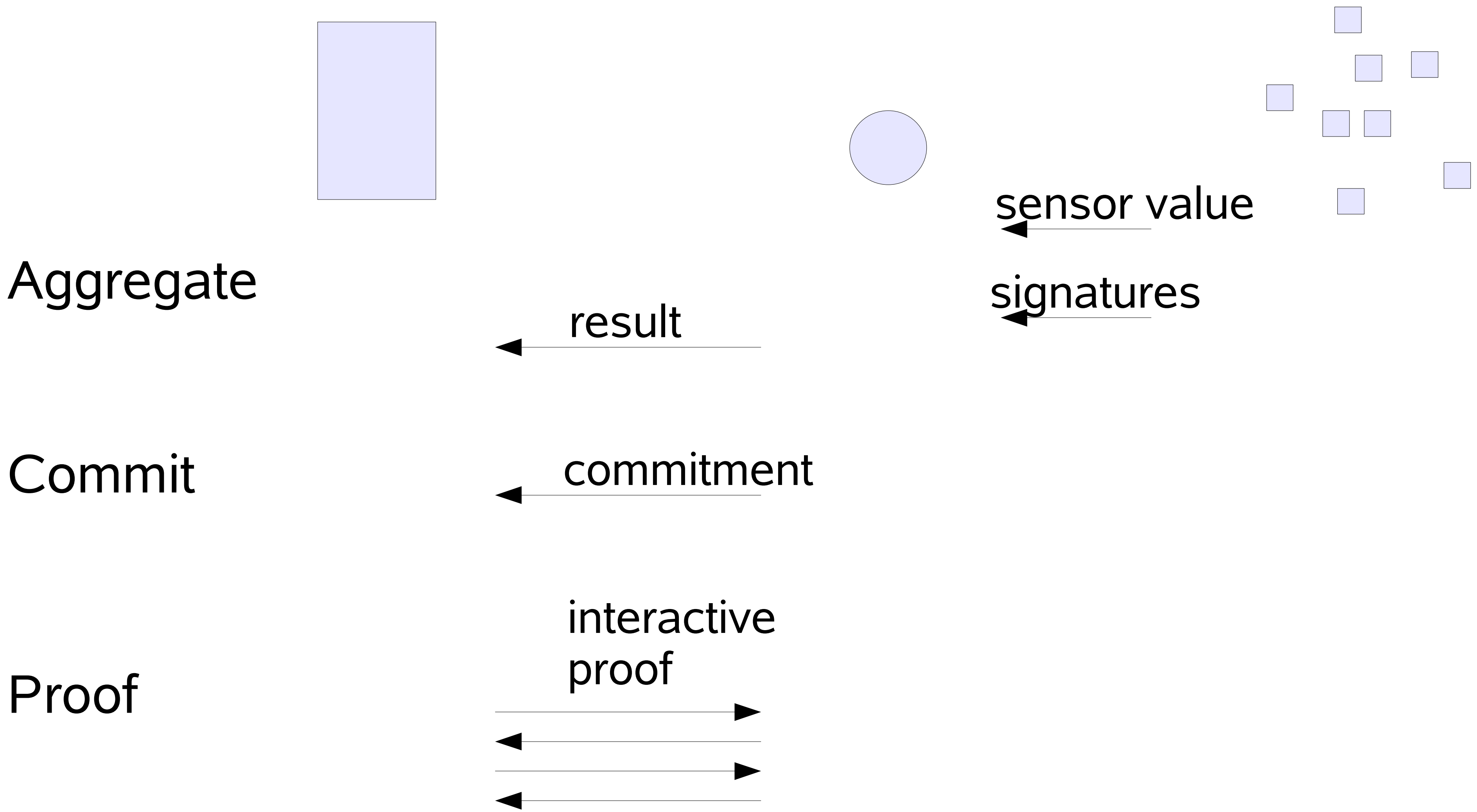
  collision resistance:
    you can not find x≠x' where h(x)=h(x')

SIA: Merkle hash tree

Commitment Scheme



SIA: General Approach



SIA: Two proofs

- Correct values as input for hash tree

  $(a_1,a_2,\ldots,a_n) = (m_1,m_2,\ldots m_n)$

  → check signature of randomly chosen values

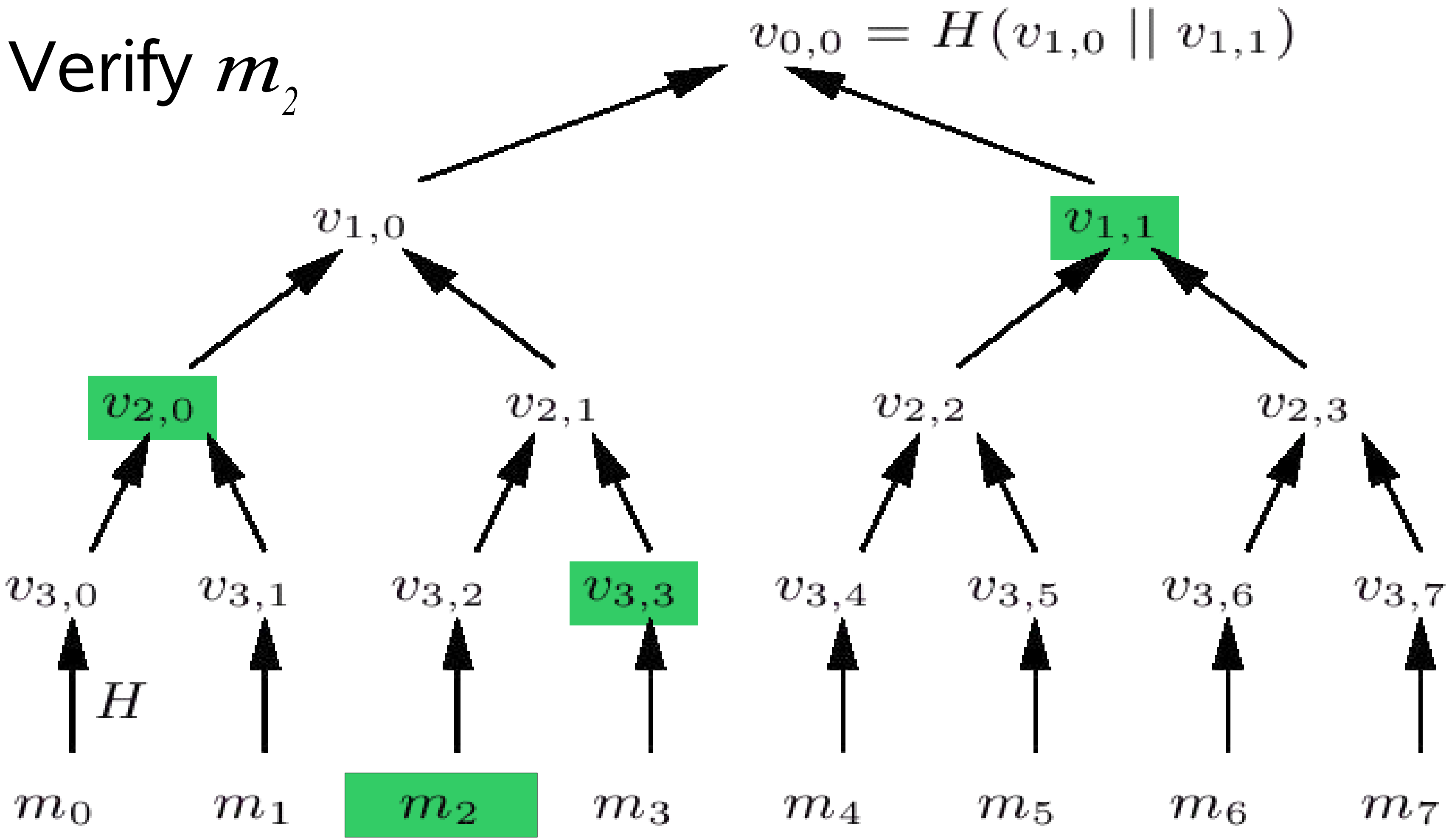
- Correct calculation of aggregation function

  result = $f(m_1,m_2,\ldots m_n)$

  → approximate with the randomly chosen values

SIA: Merkle hash tree 2

Verify m_2



SIA: General Solution

- Allows to verify if the aggregator is honest

  If he cheats the result is rejected.

- Works for any aggregation function $f(a_1,a_2,\ldots,a_n)$, that can be approximated by a random subset of the input.

  - for concrete f, we can find better approx

  - example: median…

Median (General Approach)

- n sensors with distinct values
  - if not distinct, use pair (value, sensor-Id)
  - sorted sequence $(a_1, a_2, \ldots, a_n)$, median$=a_{n/2}$
- n' corrupted sensors
  - can cause a result n' positions away form true median → focus on corrupted aggregator
- General Approach: test m values
  - Accept, if median of chosen set is close to the reported median.

Median (General Approach) 2

- Analyze the General Approach

  n values, sorted sequence $(a_1,a_2,\ldots,a_n)=A$

  uniform sample S of m values from A

  allowed approximation fault ε:
  median(S) is in A between positions $n/2 \pm \varepsilon n$

  $\delta =$ Pr[detect violating approx. fault]

  $\rightarrow \delta \geq 1-(2/e^{2m\varepsilon^2})$

- For ε-approximation with constant probability δ

  Choose size of sample S: $m = O(1/\varepsilon^2)$

Median (Specialized)

- Trick: aggregator commits *sorted* sequence A
- Check m elements (if seq. is sorted + signature)
- Analysis

  Cheat-result is out of range $n/2 \pm \varepsilon n$ → at least εn elements are in wrong half of sequence.

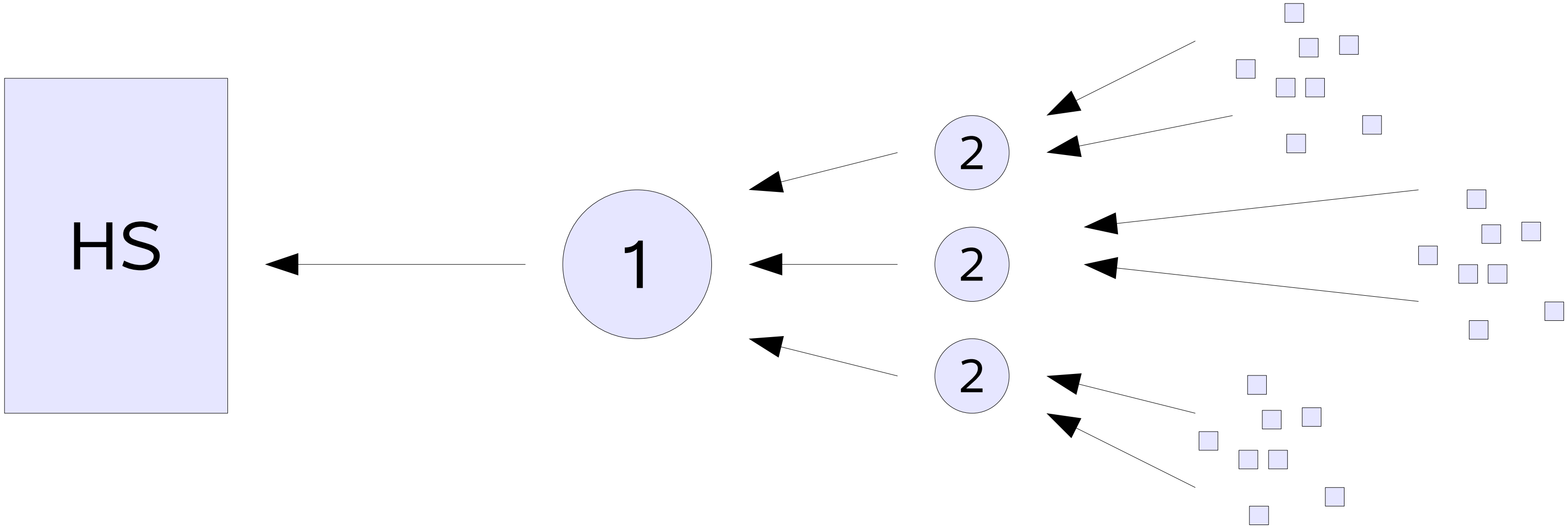
  $\rightarrow \delta =$ Pr[detect cheating] $\geq 1\text{-}(1\text{-}\varepsilon)^m$
- For constant $\delta > 0.5$, we choose $m=O(1/\varepsilon)$

SIA: Outlook, Remarks

- Median method can be used for any position k of a sequence, not only median at pos. $n/2$.
- The paper proposes specialized methods for
  - median
  - average
  - min/max
  - counting distinct elements (counting network size)

Secure Hierarchical Aggregation

- i) 1 verifies 2, ii) HS verifies 1



- (Not-) hierarchical aggregatabel functions

  min/max, average, count vs. median

  → compute median of medians

Forward Secure Authentication

- Querying past data

  became interesting later / no connection

  sensor stored (data, sig(k,data))

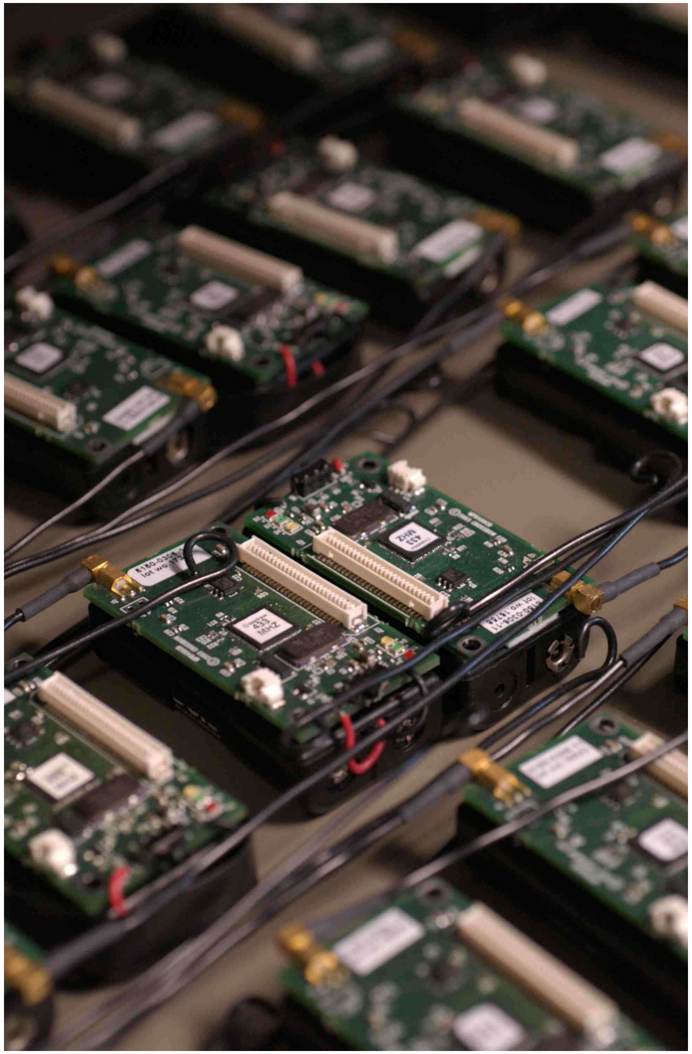
  sensor could be compromised since that time

- Update k with one-way function $k_{new} = OW(k_{old})$

  Define time interval

→ Attacker must answer correct, or keep silent.

Thank you for your attention!



Questions?