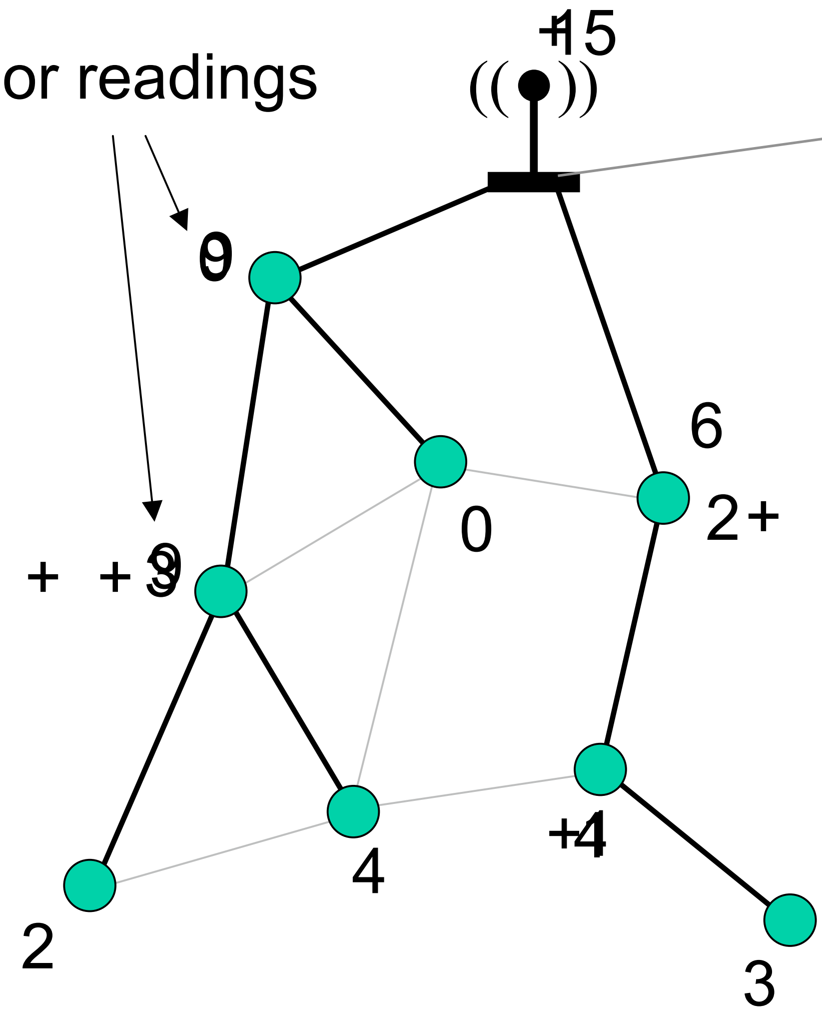
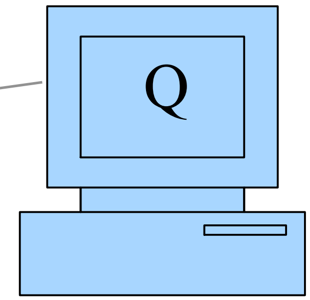
Outline

- The Secure Aggregation Problem
- Algorithm Description
- Algorithm Analysis
  - Proof (sketch) of correctness
  - Proof (sketch) of overhead bound

In-Network Data Aggregation

Sensor readings





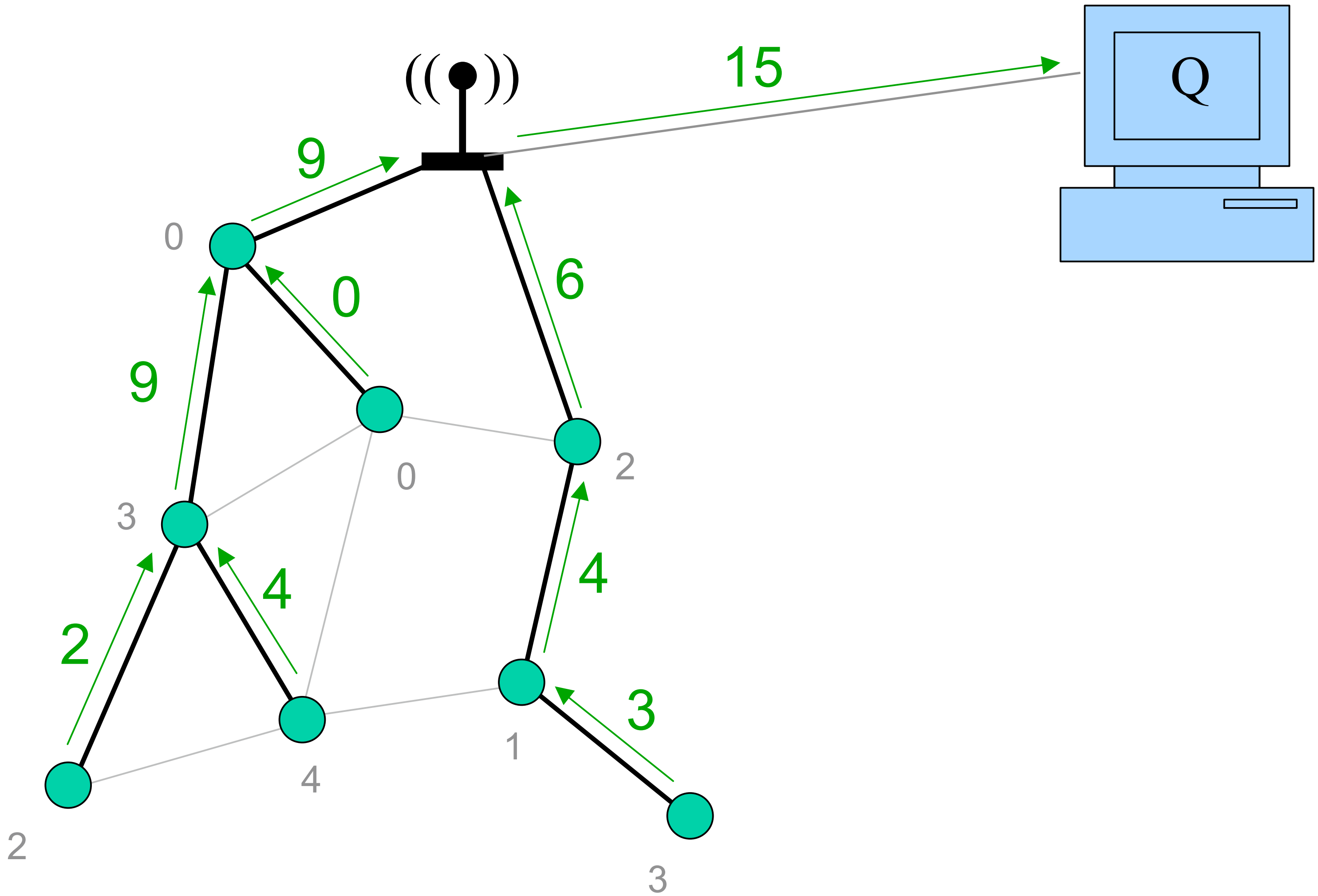
“What is the sum of
all the sensor readings?”

Answer:

Attacker Model

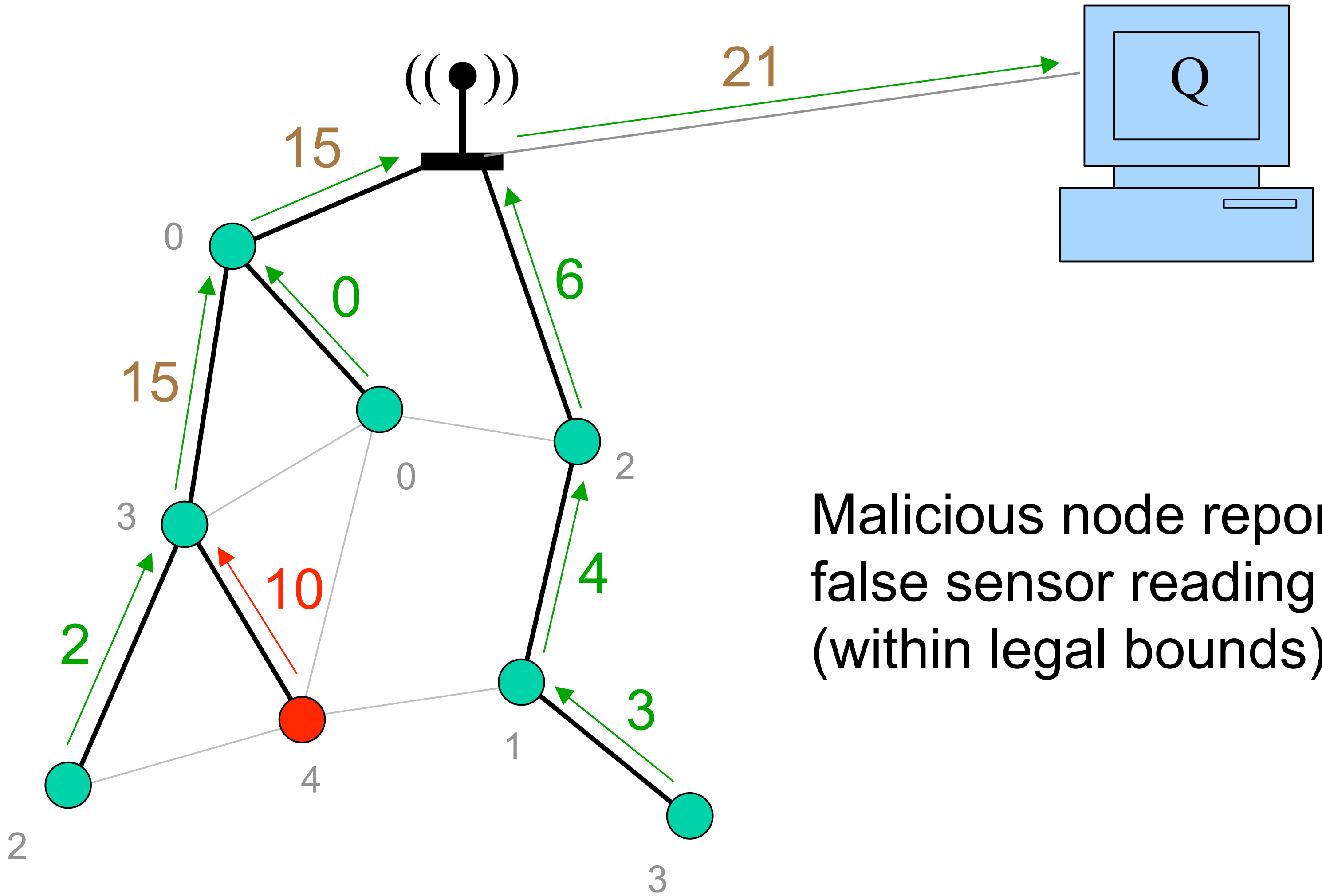
- Unsecured deployment area
- Sensor nodes not tamper-resistant
- Adversary may undetectably take control of sensor nodes or base station

Correct Data Aggregation



haowen chan • cmu

Sensor Reading Falsification

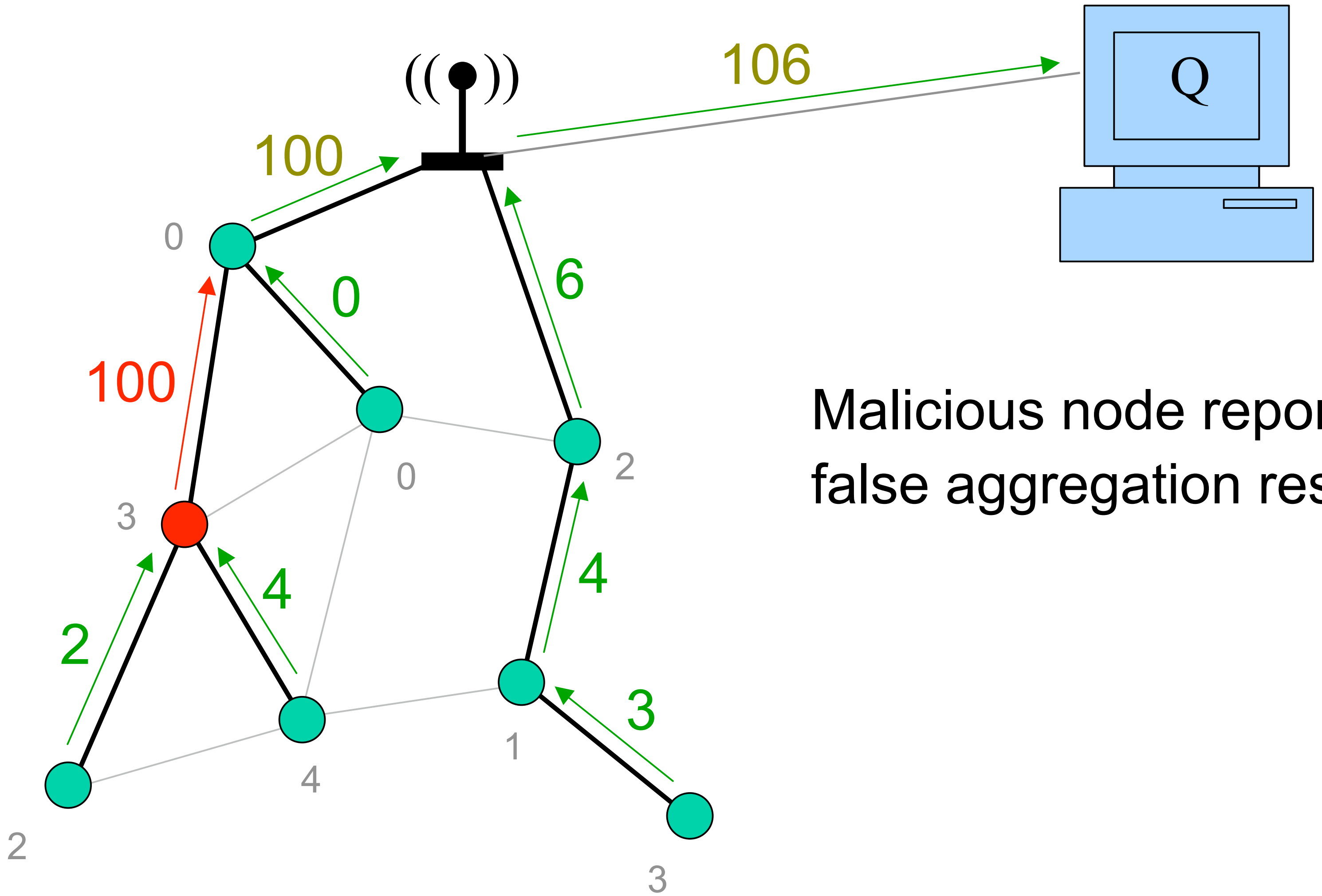


Malicious node reports false sensor reading (within legal bounds)

Sensor Reading Falsification

- General aggregation problem:
  - Assume no application-specific information
- Attacker's data indistinguishable from true data
  - Sensor reading falsification is always possible in any general secure aggregation algorithm
- Attacker's ability limited by how many nodes compromised

Aggregation Result Falsification



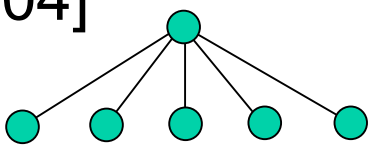
Malicious node reports false aggregation result

Aggregation Result Falsification

- Single malicious node may cause unbounded deviation in query result
- Secure aggregation problem:
  - Can we restrict the attacker's ability to falsify aggregation results?
- Tightest possible restriction without application knowledge:
  - Attacker can only perform sensor reading falsification attacks or equivalent

Prior Related Work

- Either probabilistic detection or only for special cases
- Single malicious node
  - L. Hu and D. Evans [2003]
  - P. Jadia and A. Mathuria [2004]
- Flat aggregator topology



  - B. Przydatek, A. Perrig, D. Song [2003]
  - W. Du, J. Deng, Y. Han, P.K. Varshney [2003]
- Probabilistic Detection
  - B. Przydatek, A. Perrig, D. Song [2003]
  - Y. Yang, X. Wang, S. Zhu, G. Cao [2006]

Our Algorithm

- General hierarchical (tree-based) aggregation topologies
- Multiple (unbounded) number of compromised nodes
- Achieves tightest possible bound on adversary ability to change aggregation result
- Low communication overhead
  - $O(\log^2 n)$ edge-congestion

Outline

- The Secure Aggregation Problem
- **Algorithm Description**
- Algorithm Analysis
  - Proof (sketch) of correctness
  - Proof (sketch) of overhead bound

Preventing SUM Result Deflation

- Consider only the SUM aggregate
  - Straightforward reductions from COUNT, AVG, MEDIAN to SUM
- Adversary only wishes to *reduce* the aggregate result
- Sensor readings are nonnegative: in $[0, m]$
- Let the sum of reported sensor readings

  of all legitimate nodes be S.

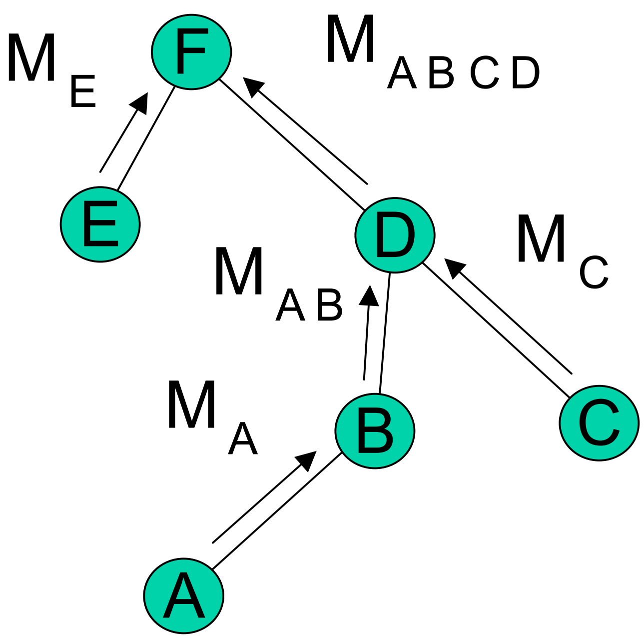
  If adversary reports any $S' < S$ then we detect its presence.
- Adversary gains no additional benefit from aggregation result falsification vs. sensor reading falsification

Generating Commitments

- Require nodes to cryptographically commit to a single version of the aggregation process
- Any aggregation result falsification cause in an inconsistency in some position in the commitment structure
  - Verification process can discover inconsistency

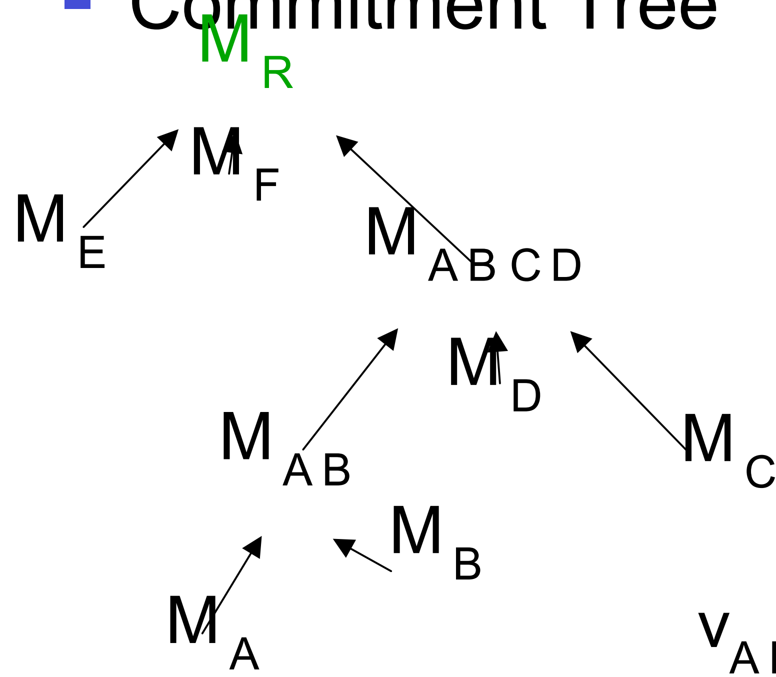
Commitment Tree

- Aggregation Tree



$$M_A = A; v_A$$
$$M_B = B; v_B$$
$$\vdots$$

- Commitment Tree



$$M_{AB} = h(M_A || M_B); \underbrace{v_A + v_B}_{v_{AB}}$$
$$M_{ABCD} = h(M_{AB} || M_D || M_C); v_{AB} + v_D + v_C$$
$$\vdots$$

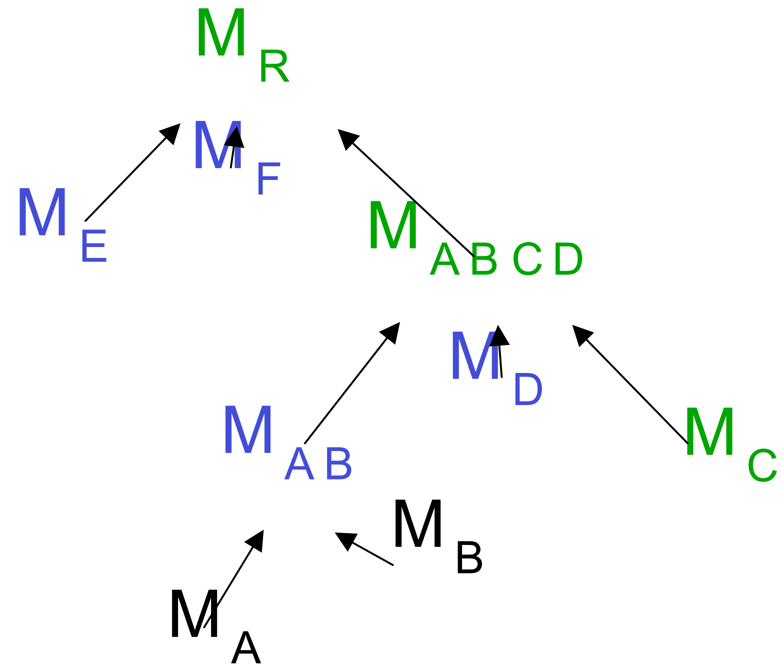
Main Idea

- Commitment structure is probed to verify aggregation correctness
- Prior work: Querier performs probing
  - Cannot probe every node
  - Too much congestion near base station
- New idea: Distribute the verification process to the sensor nodes
- **Every** sensor node checks that its sensor reading was included in the aggregate

Self-verification

- Querier disseminates commitment tree root M_R using authenticated broadcast
  - E.g. µTESLA [Perrig et al. '01]
- Node A verifies its own contribution:
  - Node A receives commitment tree root M_R
  - Node A requests all off-path vertices for M_A
  - Verify that the inputs to each aggregation step are non-negative
  - Verify that the correct M_R can be recomputed

Self-Verification of Node C



Request o®-path vertices for M_C

Check that $v_{AB}; v_D; v_E; v_F$ are all non-negative

Recompute $M_{ABCD}; M_R$

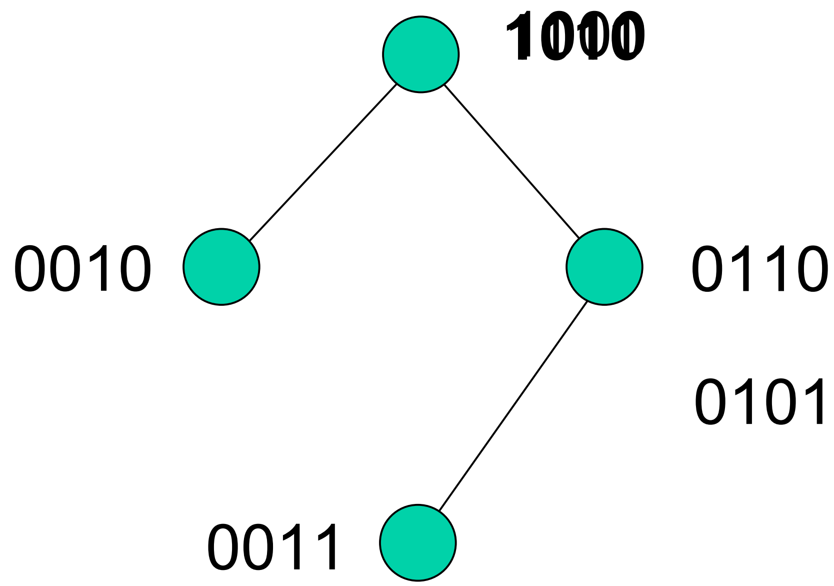
Aggregating Verification Results

- Each node shares a secret key with querier
- Node A's "OK" bit phrase for query k:

  $$MAC_{K_A}(\text{Query } k \text{ verified OK by node } A)$$

- OK bit phrases are aggregated using XOR on the way to the querier
- Querier verifies that received aggregate bitphrase is XOR of all bit phrases
  - If any node does not respond with OK, this test will fail: aggregation result rejected.

Aggregating with XOR



Outline

- The Secure Aggregation Problem
- Algorithm Description
- **Algorithm Analysis**
    - Proof (sketch) of correctness
    - Proof (sketch) of overhead bound

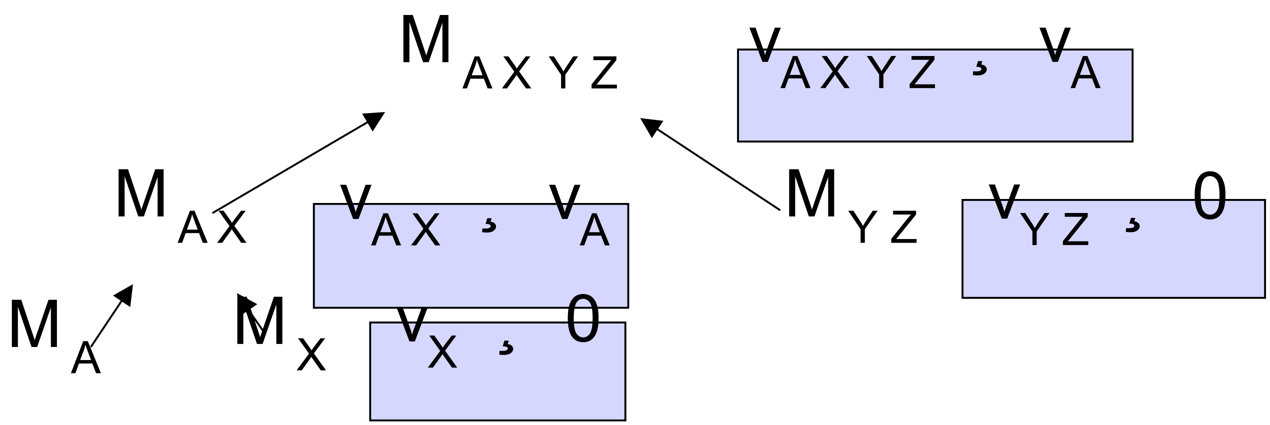
Motivating Observations

- Correctness:
  - Self-verification is cumulative
  - Net result of all nodes performing independent self-verification is equivalent to having a central querier verify every node
- Efficiency:
  - Standard metric: congestion
    - maximum communication load on any single edge
  - Self-verification incurs low congestion
  - Even if **every** node performs self-verification

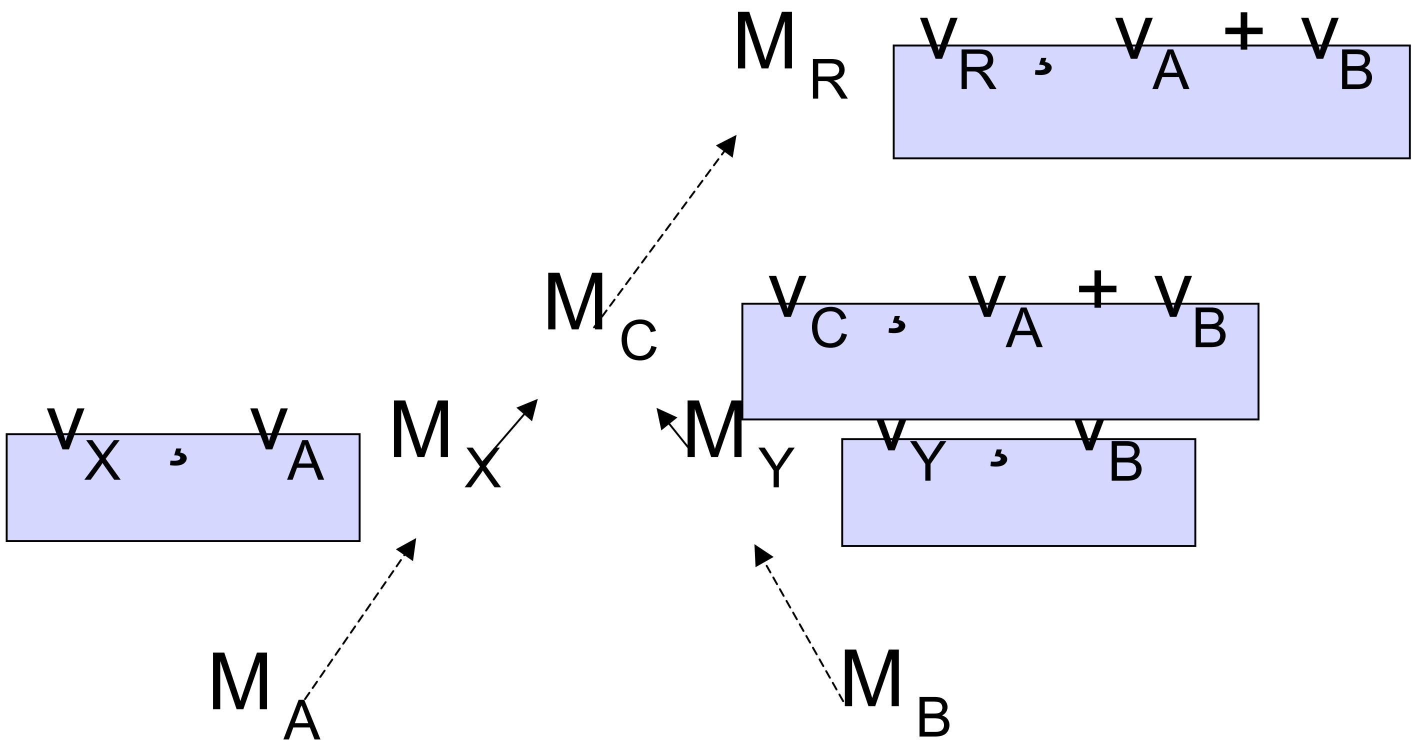
Correctness

- Lemma: If two legitimate nodes A and B both pass their verifications, then the SUM aggregate has value at least $v_A + v_B$

Observation: Intermediate sums are non-decreasing.



Correctness



M_C: LCA of M_A and M_B

M_X and M_Y are distinct since h is collision-resistant

Correctness

- Corollary: If all legitimate nodes pass their verifications, then the final aggregation result is at least

$$S = \sum_{i \in \text{legit}} v_i$$

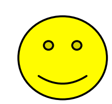
- Lower bound: Adversary cannot report result less than sum of legitimate sensor readings. 
- Upper bound?

Upper Bound

- Reduce upper bound problem to lower bound
- Compute simultaneously the *complement sum* aggregate $\underline{S}$ (recall that $v_i \in [0, m]$)

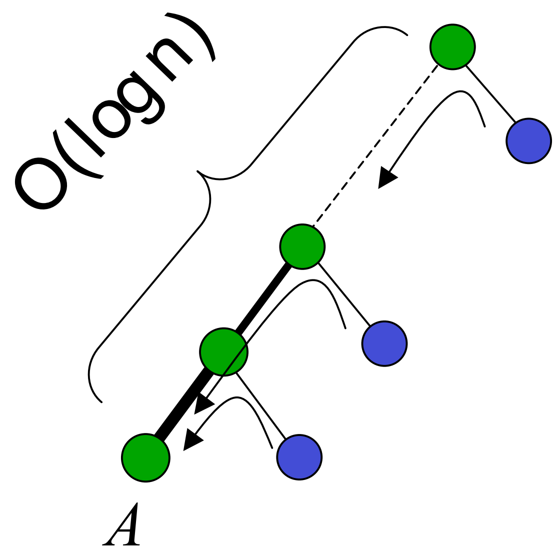
$$S = \sum_{i=1}^{n} v_i \qquad \underline{S} = \sum_{i=1}^{n} (m - v_i)$$

- Querier checks: $\underline{S} = nm - S$
- Adversary: to increase S, must decrease $\underline{S}$.
  - But neither S nor $\underline{S}$ can be decreased below contribution of legitimate nodes.



Efficiency

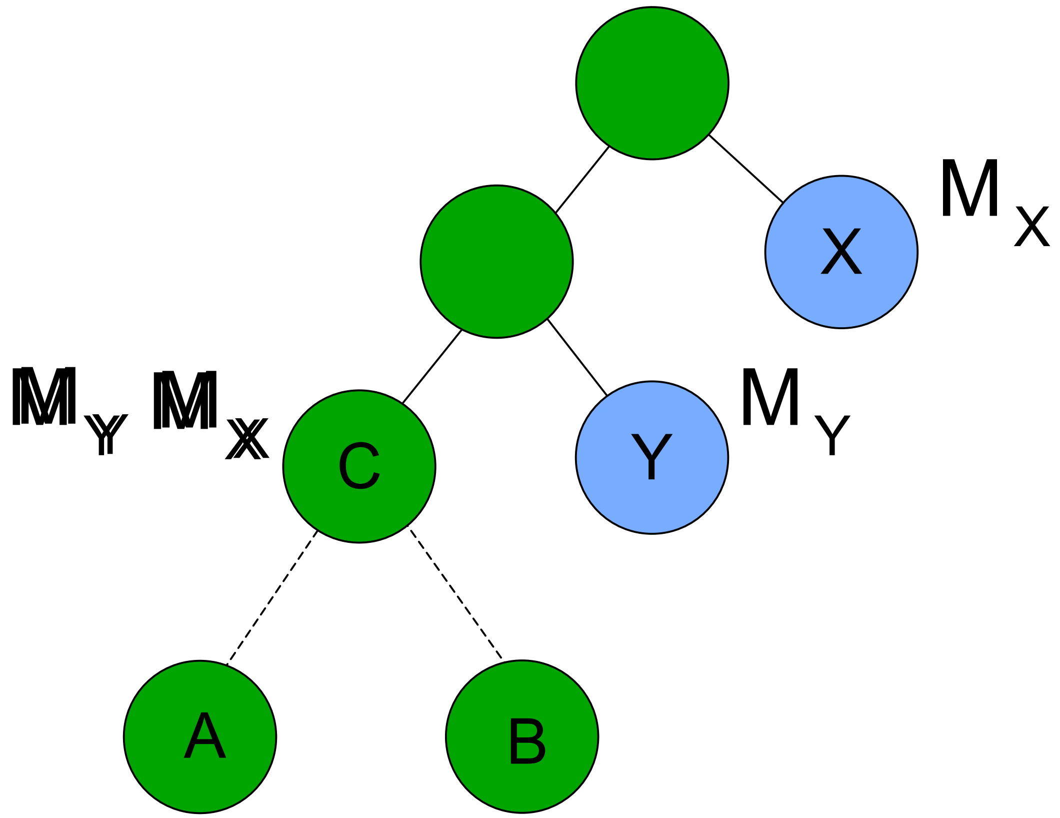
- Suppose aggregation tree is *balanced*
- When node A self-verifies, it receives all off-path vertices in the commitment tree



- Maximum congestion: leaf edge
  - O(log n) messages

Efficiency

- Self-verification of other nodes (e.g. node B) does not increase communication load on any edge of the path between node A and the root



Efficiency

- Edge congestion in balanced aggregation trees: $O(\log n)$
- For arbitrary unbalanced aggregation topology:
  - Define a balanced *logical* aggregation overlay over the physical topology (details in paper)
  - Incurs multiplicative $\log n$ factor
- Edge congestion for general aggregation trees: $O(\log^2 n)$

Conclusion

- Secure data aggregation algorithm
  - Suitable for general tree-based aggregation topologies
  - Resilient vs multiple malicious nodes
  - Tightest possible guarantees on adversary detection (without assuming application knowledge)
  - Low $O(\log^2 n)$ edge congestion
  - Limitation: need to know the set of responding nodes
- Future Work:
  - Secure versions of more sophisticated aggregation functions
  - Defences vs sensor reading falsification

Secure Hierarchical In-network Data Aggregation for Sensor Networks

Haowen Chan

Adrian Perrig and Dawn Song

Carnegie Mellon University