GOSSIPING ALGORITHM AND ATTACK MODEL

Algorithm 1: Randomized consensus protocol

Input: the number of iterations $T$. Sufficiently large for $t = 1 : T$ do

1. Uniformly wake up a random node $i \in V$.
2. Node $i$ selects node $j$ from its neighborhood with the probability
   
   $P_{ij}$, where $j \in N_i$ and $N_i = \{j : (i, j) \in E\}$.

3. Nodes $i$ and $j$ update their states as follows
   
   $x_i(t + 1) = x_i(t) + x_j(t) / 2$.

Other nodes keep their original states, i.e.,

$x_v(t + 1) = x_v(t)$ for all $v \neq i, j$.

“good nodes”

Fig. 1. Example application: state estimation in the grid.

Fig. 2. Network convergence with one attacker present. Normal nodes (dashed) converge towards the value of the attacker (black) instead of the true mean (green). The attacker follows the expected convergence (black).

Algorithm to find the system average of the nodes’ state $x^k$.

1. $j$ ignores (1) and chooses its next state:
   
   $x^k_j(t) = \alpha^k + m^k_j(t)$.

2. The attacker steers the network towards $\alpha^k$, away from the true value (which is $x^k_0 = \frac{1}{N} \sum x^k_i(0)$).

3. The disguise follows the expected convergence of the network.

Fig. 3. Simulation topology; node 1 is attacking.

GOSSIPING ALGORITHM AND ATTACK MODEL

MOTIVATION

- Gossiping algorithms are attractive:
  - No central point of failure.
  - Built-in fault tolerance to node failures.
  - Nodes can reorganize themselves.
  - Protected against outsider attackers with authentication and encryption.
  - Vulnerable against insider attackers.

- Our method provides:
  - Decentralized attack detection.
  - Decentralized attacker identification.
  - Self-Healing Network.
  - No communication overhead.

- If both $m$ and $j$ are not attackers, then $E[x^m_i(t) - x_j^m(t)] = 0$.
- If $m$ or $j$ is an attacker, then $E[x^m_i(t) - x_j^m(t)] \neq 0$.

Fig. 4. Scatterplots of normal node vs. attacker, & two normal nodes. $x_1$ not correlated, while $x_1$ and $x_2$ are correlated.

- We perform an anomaly test & detect whether an attacker is present and which node is malicious (see paper for details).

RESULTS

- If both $m$ and $j$ are not attackers, then $E[x^m_i(t) - x_j^m(t)] = 0$.
- If $m$ or $j$ is an attacker, then $E[x^m_i(t) - x_j^m(t)] \neq 0$.

Fig. 5. Detection and localization of an attacker (solid lines = Monte Carlo simulation; dashed lines = theoretical bound)

Observations:

- The more we average over $K$ instances, the better.
- Gaussian approximation is very good for localization, but only a bound for detection.
- Also works when the detecting node is not directly next to an attacker.

- Our method can also be applied to cases with more than one attacker.
  - Detection performance degrades as the number of attackers increases or if more attackers are surrounding a node.
  - As the number of attacking neighbors increases, localization performance degrades faster than detection.

ALTERNATIVE STRATEGY: TEMPORAL METHOD

- Node $j$ converges towards the attacker’s state at time $\infty$, $E[x^k_j(\infty)] = \bar{\alpha} = E[x^k_0(\infty)]$.
- At the beginning of the algorithm, each node has a different state.
  
  $E[x^k_i(0)] = \bar{\alpha} \neq \bar{\gamma} = E[x^k_j(0)]$.

- Therefore, the quantity $|x^k_i(0) - x^k_j(0)|$ is (in expectation) close to zero if node $i$ is an attacker; it is large if $i$ is a normal node and an attacker is present.

- See paper for detailed description and results.

FUTURE EFFORTS

- Research more complicated topologies.
- Apply these results to our work in decentralized networks.
- Combine temporal and spatial method together for even better performance.