Pando: Efficient Byzantine-Tolerant Distributed Sensor Fusion using Forest Ensembles

Hans Walter Behrens, K. Selçuk Candan
Arizona State University
Tempe, Arizona, USA
{hwb, candan}@asu.edu

***Abstract*—Ad hoc communication networks provide a robust and low-power method for sensors to return their observations to an authority. However, ensuring that the returned values accurately represent the environment being sensed poses challenges. In particular, malicious sensors controlled by an adversary may collude to return systematically misleading or falsified results. Previous work examines this challenge for weak adversaries and under strong assumptions, limiting practical applicability. In this work, we propose Pando, a novel approach for mitigating the Byzantine distributed sensor fusion problem. We introduce an approach for the decentralized creation of a forest ensemble, made up of overlapping, hierarchical message passing routes that provide robust and efficient delivery while also mitigating adversarial interference. We then leverage a modified, homomorphic Merkle tree structure to create a novel Byzantine-tolerant message passing protocol. Finally, we propose a classification-informed data fusion algorithm based on these methods. We evaluate the correctness and efficiency of our approach under several attack models, and discuss functionality improvements over the state of the art.**

***Index Terms*—sensor fusion, wireless sensor networks, network security, robustness**

I. Introduction

Many distributed sensing applications require sensor deployments in remote, dangerous, or inhospitable environments. Due to the likelihood of loss, damage, or destruction in these scenarios, such sensors commonly use inexpensive, low-power components and self-contained power sources, and rely on ad hoc communication methods to compensate for absent or inaccessible network infrastructure [1]. Ad hoc approaches also allow for dynamic re-routing as nodes become inaccessible, evolving the network topology [2].

Despite these advantages, ad hoc networks suffer from the disadvantage that, if any nodes intentionally diverge from the established protocol in order to maliciously undermine communication effectiveness, the message will not arrive.

As an example, consider a smart city in which sensors distributed throughout the water supply network monitor municipal water quality. If an adversary wishes to illegally dump industrial runoff, it can mislead these sensors by injecting plausible baseline measurements, allowing the environmental harm from its pollutants to go undetected.

Similarly, a governmental organization surveilling a remote area to reduce the flow of smuggling deploys a number of sensors to notify personnel of unauthorized activity in the area. In response, criminals may compromise a subset of these sensors, creating a blind spot in the network. As a result, movement of illicit goods may proceed unimpeded, without alerting officials to the problem.

In both cases, malicious interference during sensing allows the provision of false data, resulting in worse outcomes than if data were simply absent. Consequently, mitigating adversarial interference forms a critical foundation for sensing, improving security for a broad variety of potential applications. Towards this goal, we make the following contributions:

- We propose a method for distributed creation of an ensemble of spatially-distributed overlay networks using strictly node-local information.
- We propose a novel messaging format utilizing homomorphic Merkle trees to enable self-authenticating payloads.
- Using these structures, we describe a protocol for Byzantine-tolerant distributed sensor fusion.

By detecting message tampering, unexpected absence, and illicit modification, we offer a foundation for distributed sensing platforms that provides a more robust level of service in scenarios with adversarial interference.

II. Background & Related Work

A. Sensor Fusion

Many modern devices and systems contain multiple sensors, either co-located on a single device to collect heterogeneous data, or dispersed across many devices to sense a wider area. Often, due to their low power, low cost, or harsh operating conditions, these sensors return unreliable measurements.

The process of "sensor fusion" refers to combining this ensemble of data into a single, reliable representation of the environment, and serves a crucial role in many domains, such as self-driving cars, UAVs, and other applications with autonomous agents interacting with the environment [3], [4].

To address the most common challenges arising from corrupted or noisy sensors, statistical methods can smooth out natural variability in data [5], but wen dealing with distributed systems, these approaches become difficult to apply due to the distribution of data across potentially many devices.

B. Distributed Sensing

Xiao *et al.* [6] extends statistical smoothing to distributed sensor systems, relying on the concept of message passing to share a weighted average of the observations so far. This approach proves robust against unreliable links and measurements, but does not address malicious interference.

A similar scheme proposed by X. Zhang *et al.* [7] adopts a Bayesian sensor fusion approach using graph information to inform consensus. This distributes responsibility for fusion across the network, but does not address the presence of adversarially-controlled nodes in the topology.

K. Zhang *et al.* [8] examine efficiency improvements offered by a mobile data collection agent, reducing both the total messages exchanged and the sensor density required to achieve effective coverage. However, they again do not consider Byzantine activities.

C. Resilience in the Presence of Malicious Nodes

Previous literature has explored ad hoc networks' sensitivity to adversarial interference, in particular from the perspective of routing. Awerbuch *et al.* [9] proposes using statistical inference of link failures to identify malicious nodes, but does not address modifications to message payloads directly. Hu *et al.* [10] describes a method which protects payloads, but requires expensive per-node key management scaling linearly with network size.

Other work explores the concept of node capture resilience [11], establishing communication channels within the local neighborhood of a node that are robust to post-deployment compromise. However, messages which directly traverse a compromised node may have their payloads changed, an undesirable occurrence in sensing applications.

Liu *et al.* [12] propose the use of a trust model for detecting and excluding adversarial nodes. However, the trust models considered assume a uniform distribution of information importance, with protection amortized across all messages, rather than protecting individual rounds of sensing.

In contrast, the approach proposed by Abrardo *et al.* [13] provides strong guarantees about message fusion in the presence of adversaries, including a theoretically optimal bound. However, this approach is computationally expensive; a followup work by the same authors [14] addresses this challenge through the use of Monte Carlo Markov Chains (MCMC), with both approaches requiring that adversarial nodes do not collude.

In previous work [15], we examined the challenges in interacting with a distributed sensor network under a cooperating adversarial threat model, but did not consider a data fusion context. Additionally, we assumed the existence of a secure out-of-band channel, which may not be possible in practice.

D. Homomorphic Hash Trees

Homomorphic hashing [16], [17], [18], [19] differs from traditional cryptographically secure one-way hash functions by satisfying the following relationship: $H(A) \odot H(B) == H(A \odot B)$ for an operator $\odot$.

In prior work [15], [20], we have shown that we can extend the well-known Merkle tree [21] using homomorphic hashing to allow for imbalanced, mixed-degree trees, which we refer to as *homomorphic Merkle trees* (HMTs). In particular, we proposed these HMTs as a self-authenticating data structure, and applied them in various message authentication contexts [15], [20]. However, these ignore the data collection and fusion problem, focusing exclusively on resilient node addressability.

III. Problem Formulation

A. Setting

We assume that each node distributed in a monitored region contains hardware for sensing and local communication with its nearby neighbors, yet insufficient for long distance communication. Each node can also localize itself, such as through the use of satellite navigation, and localize its neighbors in a spoof-resilient manner using a protocol such as [22].

A fusion coordinator (FC) is tasked with collating sensor information and making a final determination on a fused value, and should be able to collect information from arbitrary nodes within the deployment. Recovery logistics lie outside the scope of this work, but could include physical mobility [8] or long-distance wireless communication [23]. Sensing occurs in synchronous rounds as a response to a query from the FC, with each round assigned a monotonic nonce generated randomly by the FC. Authenticated query delivery is outside the scope of this work, and may use any appropriate method [15], [20].

For simplicity, we assume that sensor observations are drawn from a finite categorical dictionary and, without loss of generality, that the ad hoc network lies in a two dimensional space. Observation series fall beyond the scope of this work.

B. Adversarial Model

A percentage of deployed nodes may be compromised by the adversary, who gains access to all local information stored on the captured node at the time of compromise. This is a reasonable assumption, as given physical access to sensors, extracting the data stored on these devices poses a surmountable problem to a motivated adversary. Furthermore, this information may be shared among the captured nodes to coordinate deception strategies and malicious behavior.

The adversary aims to modify the network behavior such that the system comes to consensus on a value of the adversary's choice (the "incorrect" value), replacing the value which would have been produced in the absence of malicious behavior (the "correct" value) using three possible attacks: Adversarially-controlled sensor nodes

- report incorrect observations (*value injection*);
- generate synthetic observations corresponding to nodes which do not exist (*node injection*); or
- drop incoming observation traffic to give malicious observations more weight (*traffic dropping*).

We consider five deception strategies, where an attacker

- does not drop any observations to better blend in and hide its adversarial nature (*passive*);
- drops all observations except its own, to maximally disrupt sensing functionality (*destructive*);
- drops all observations including its own, to disrupt sensing and hide its presence (*silent*);
- drops observations with a preponderance of correct values, and keeps those with incorrect values to negatively impact the fused result (*selective*); or
- drops observations with a preponderance of incorrect values to mislead Pando's defenses (*sneaky*).

We discuss the impacts of each of these deception strategies in more detail in Section V-C.

TABLE I: Symbols and Notation

Operators	Meaning
$A \odot B$	An operation closed under homomorphism
$H()$	A traditional collision-resistant hash function (CRHF)
$\mathcal{H}()$	A CRHF homomorphic under $\odot$
$\emptyset, [], \{\}$	Set, array, or key-value map respectively
$\|A\|$	The cardinality of A
A^{+}	The absolute value of A

IV. Pando: Protocol Description

Pando[1] tackles attacks through the decentralized creation of an ensemble of Trémaux forests. Even though these forests are constructed using only local node information, they serve as secure hierarchical routes for diverse, efficient message passing, mitigating adversarially-inserted network partitions. In this section, we detail this 3-phase process, involving: (Phase 1) ensemble creation; (Phase 2) data collection; (Phase 3) ensemble-based fusion determination.

Prior to deployment, each node ν receives a unique serial (ν_{id}), derives a node signing key ($\nu_{sig} = \mathcal{H}(\nu_{id} \odot \mathcal{S})$) using a shared secret $\mathcal{S}$, and stores ν_{id} and ν_{sig} locally for later use during HMT creation. The FC must also securely store $\mathcal{S}$ for later use validating node authenticity. Although leaking $\mathcal{S}$ would invalidate protocol security, deployed nodes only know a hash of $\mathcal{S}$, and thus post-deployment node capture does not undermine overall network security. During deployment, nodes also determine and store their position as ν_{pos}.

A. Phase 1, Creation of an Ensemble of Forests

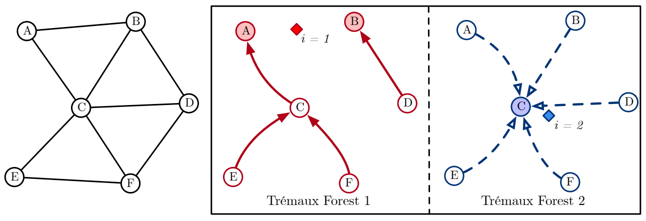
A Trémaux tree describes a hierarchical spanning tree rooted at a particular vertex, such that all pairs of adjacent nodes have a absolute ordering with respect to the root vertex [24]. While ideally one spanning tree would be created, to reduce network costs and other overhead, Pando may instead create a forest of Trémaux trees that collectively span the network. By leveraging their hierarchical nature, these trees allow Pando to route messages efficiently to each root.

To initiate a query, the FC randomly generates a nonce η, and transmits it into the deployment as described in Section III-A. Upon receiving the nonce, each node follows a series of steps to generate a globally-shared ensemble of hierarchical forests. No node knows the entire network topology, and instead produces oblivious routing decisions conducted using local information only. Collectively, this results in useful emergent routing behavior that is both efficient and robust.

1) Trémaux Forest Ensembles: Each node initializes a local, cryptographically secure pseudorandom number generator P with the received nonce, then generates an ensemble coordinate vector, $\vec{e}$, of length m, the pre-determined ensemble size. These coordinates (referred to as "virtual roots") are drawn uniformly at random from the deployment area using a two-dimensional uniform distribution[2]. Note that, as all nodes use the same η, they also produce an identical vector $\vec{e}$.

[1]Pando, meaning "I spread out" in Latin, is the name of a colony of aspen trees determined to be a single living organism, with a shared root system.

[2]Unimodal Gaussian sampling centered on the deployment produces shorter, more homogeneous trees, improving transmission efficiency but reducing resilience; bimodal Gaussian sampling biased toward the periphery produces the reverse behavior. In our evaluations, we consider uniform sampling, which provides a balance between these extremes.



(a) Connectivity (b) Forest ensemble, $m = 2$

Fig. 1: An example forest ensemble, with virtual roots.

Algorithm 1: Generation of a new forest ensemble.

```
GenerateEnsemble ($m, \eta$):
$\vec{e} = []$;
for $i = 0; i < m; i++$; do
    $seed = \eta$;
    for $j = 0; j < i; j++$ do
        $seed = H(seed)$
    $\vec{e}[i] = (x, y) \leftarrow P_{seed} \sim \mathcal{U}$ // Uniform distribution
foreach $\varphi \in \vec{e}$ do
    $\nu_{dist} = L^2(\nu_{pos}, \varphi)$;
    parent, ancestor, descendant = $\emptyset$;
    min-dist = $\infty$;
    foreach neighbor $\in$ neighbors do
        $neighbor_{dist} = L^2(neighbor_{pos}, \varphi)$;
        if $neighbor_{dist}$ < min-dist then
            parent = neighbor;
            min-dist = $neighbor_{dist}$;
        if $neighbor_{dist}$ < min-dist then
            ancestor = ancestor $\cup$ neighbor;
        else
            descendant = descendant $\cup$ neighbor;
```

Subsequently, for each virtual root[3] $\varphi = \vec{e}[i]$ $(1 \leq i \leq m)$, a node computes its routing behavior for the forest rooted at φ. It first computes an absolute ordering of its neighbors (and itself) using their respective Euclidean (L^2) distance from φ. Using this information, it then generates routing rules for a given (η, φ) pair and stores them locally.

We refer to the emergent routing behavior generated by these collective rules as a *Trémaux forest* ($\mathcal{F}_\varphi$) for a specific virtual root $\varphi \in \vec{e}$, and to the set of all forests enabled by the vector $\vec{e}$ as a *forest ensemble* ($\mathcal{E}$) for all $\varphi \in \vec{e}$ (Figure 1).

2) Node-Local Process for Forest Ensemble Creation: Alg. 1 presents the node-local computations producing the forest ensemble. For each $\varphi \in \vec{e}$, a node computes its meta-relations, labeling neighbors with the following terminology:

- *Descendants*: Neighbors further from φ.
- *Ancestors*: Neighbors closer to φ.
- *Parent*: The ancestor closest to φ.

We also use *child* to refer to the reciprocal relationship of *parent* – this relationship cannot be easily determined by a node, but can be discovered by the FC by analyzing routing information embedded in the HMTs. The nodes also classify themselves as

- *Roots*: Nodes with no ancestors,
- *Leaves*: Nodes with no descendants

as appropriate.

[3]For simplicity, in the rest of the paper, we will use the notation $\varphi \in \vec{e}$.

Algorithm 2: Transmission of observation message.

```
TRANSMITOBSERVATION$(\eta, \varphi)$:
1  $subtree$ = messages received from descendant children;
2  $obs$ = node-local observation;
3  $\eta_{\mathcal{F}} = H(\eta|\varphi)$;
4  $relation\text{-}sig = \emptyset$;
5  $relation = \emptyset$;
6  foreach $child_i \in$ SORT$(subtree_{keys})$ do
7      $relation\text{-}sig = relation\text{-}sig \odot child_i$;
8      $relation_i = child_i$;
9  $relation\text{-}sig = \mathcal{H}(relation\text{-}sig \odot \nu_{sig})$
10 $obs\text{-}sig = \mathcal{H}(\eta_{\mathcal{F}} \odot obs \odot \nu_{sig})$;
11 $msg = subtree$;
12 $msg_{\nu_{id}} = (obs, obs\text{-}sig, relation, relation\text{-}sig)$;
13 Transmit $msg$ to parent of $\nu$, if one exists for $\varphi$.
```

B. Phase 2, Data Collection within a Trémaux Forest

Once the rules are generated, data observation and collection may begin. This involves (a) data transmission, in parallel, for each forest within the ensemble, and (b) collection of the data from the forest ensemble by the FC.

1) Data Transmission within a Trémaux Forest: Each node makes a local observation o, accepts messages from its descendants, and sends aggregated results to its parent. Nodes maintain sensing timeout windows sized inversely proportionate to the node's distance from φ, waiting to determine if its descendants will return any data. Longer windows provide better sensor coverage, at the expense of higher latency, and should be tuned for usage requirements. This sensing timeout prevents deadlocks caused by observation dependency cycles between descendants across $\mathcal{E}$.

Leaves may sign and transmit their observations immediately. In either case, the node also optionally notifies its non-parent ancestors that it has completed its observation to further reduce latency, at the expense of more transmissions.

We refer to each message received from a descendant as a *subtree*, in reference to their structure as a homomorphic Merkle tree. Nodes may choose to perform validation on received subtrees prior to forwarding, incurring an added computational cost bounded by a maximum validation depth parameter, δ. This process mirrors the validation conducted by the FC detailed in Algorithm 3.

Intuitively, since the FC validates the entire HMT in the next phase to prevent node injection, intermediate validation only reduces falsified message propagation, and δ should therefore be tuned to balance computational and transmission costs. This effectively limits adversarial interference to subtree pruning: although it can make at most one false observation per compromised node, it could ignore potentially many uncompromised descendants by dropping their results.

Once a node receives a message from all descendants, or the sensing window closes, it merges the received subtrees with its own local observation, and sends the combined information to its parent (Algorithm 2). The process repeats until reaching a root, where it awaits further action by the FC.

2) Data Collection from Roots: Finally, once the results have propagated to the roots of the forests, the FC collects the data. Using the nonce η, the FC generates the ensemble vector $\vec{e}$, using the geographic information to guide data collection; this could include physical mobility [8], long-distance wireless communication [23], secure node-to-node sensor network protocols [25], or other mechanisms.

Algorithm 3: Validation of observation message.

```
VALIDATE$(message, curr\text{-}depth, \delta)$:
1  if $\mathcal{H}(\eta_{\mathcal{F}} \odot obs \odot obs\text{-}id \odot \nu_{sig}) \neq \mathcal{H}(obs\text{-}sig \odot \nu_{id})$ then
2      return False;
3  if $curr\text{-}depth < \delta$ then
4      if
       $child\text{-}id_1 \odot \ldots \odot child\text{-}id_n \odot parent\text{-}id \odot \nu_{sig} \neq relation\text{-}sig \odot \nu_{id}$
       then
5          return False;
6      foreach $child \in \nu_{children}$ do
7          if $\neg$VALIDATE$(message, curr\text{-}depth + 1, \delta)$ then
8              return False;
9  return True;
```

Algorithm 4: Classification-based weight generation.

```
CLASSIFICATION$(\mathcal{E})$:
1  $was\text{-}parent, was\text{-}child = \{\}$;
2  $\mathbb{N} = \{\}$;
3  $valid\text{-}subtree = \emptyset$;
4  foreach $\mathcal{F} \in \mathcal{E}$ do
5      $valid$ = VALIDATE$(\mathcal{F}, 0, \infty)$;
6      if $valid$ then
7          $valid\text{-}subtree = valid\text{-}subtree \cup \mathcal{F}$;
8          foreach $\nu \in \mathcal{F}$ do
9              $was\text{-}parent[\nu] = was\text{-}parent[\nu] + 1$;
10             foreach $child \in children(\nu, \mathcal{F})$ do
11                 $was\text{-}child[child] = was\text{-}child[child] + 1$;
12                 $\mathbb{N}[child] = \mathbb{N}[child] \cup \nu$;
13                 $\mathbb{N}[\nu_{id}] = \mathbb{N}[\nu] \cup child$;
14 $\omega = \{\}$;
15 foreach $\nu \in \mathbb{N}$ do
16     $\omega[\nu] = 0.5$;
17     $observed = (was\text{-}parent[\nu] > 0 \vee was\text{-}child[\nu] > 0)$;
18     $imbalance = abs(was\text{-}parent[\nu] - was\text{-}child[\nu])$;
19     if $imbalance > |\mathbb{N}[\nu]|$ then
20         $\omega[\nu] = 0$;
21     else if $observed$ then
22         $\omega[\nu] = 1$;
23 return $valid\text{-}subtree$, $\omega$;
```

C. Phase 3, Ensemble-based Fusion Determination

Prior to interpreting the data, the FC must validate the composition of each collected subtree within the ensemble (via Algorithm 3), discarding results from roots whose signatures do not authenticate. This filters out all node injections from the data, although the remaining data may still contain value injections (see Sec. III-B).

1) Normalized Observational Keyscore: After evaluating the forest ensemble, the FC computes a *keyscore*, $\kappa(o)$, for each observation, o, according to the following equation:

$$\kappa(o) = \sum_{\mathcal{F} \in \mathcal{E}} \frac{\sum_{\nu \in \mathcal{F}} obs(\nu, o, \mathcal{F})}{|\mathcal{F}|}. \tag{1}$$

Here, $\mathcal{E}$ denotes the ensemble, $\mathcal{F}$ denotes a forest, $|\mathcal{F}|$ denotes the number of nodes in the forest, and $obs(\nu, o, \mathcal{F}) = 1$ if the node ν reported observation o for forest $\mathcal{F}$ (and 0 otherwise). Intuitively, the keyscore quantifies the prevalence of an observation within and across forests in the ensemble.

2) Deception Classification: Since under certain attack models, the adversary may drop observations, Pando attempts to classify node behavior. Specifically, the FC assesses structural differences between forests in $\mathcal{E}$ to gather information about routing behavior of participating nodes (Alg. 4).

To do so, the FC evaluates how many times a node served as a parent, how often it was a child, and counts the number of distinct neighbors with which the node interacted. Using this information, Pando computes a heuristic-based behavioral classification score for each node.

Let $\mathit{children}(\nu, \mathcal{F})$ denote the set of children of the node ν on the forest $\mathcal{F}$, and let $\mathit{parent}(\nu, \mathcal{F})$ denote the set of nodes which claimed ν as a descendant.

Also let $pf(\nu)$ denote the number of times node ν reported having a child:

$$pf(\nu) = \sum_{\mathcal{F} \in \mathcal{E}} [|\mathit{children}(\nu, \mathcal{F})| > 0], \tag{2}$$

and let $cf(\nu)$ denote the count of occurrences where ν was recorded as a child of another node in the ensemble:

$$cf(\nu) = \sum_{\mathcal{F} \in \mathcal{E}} \sum_{u \in \mathcal{F}} [\nu \in \mathit{children}(u, \mathcal{F})]. \tag{3}$$

Intuitively, as the ensemble size increases, we expect $pf \sim cf$, and a deviation from this could highlight attempted deception. We thus define node trust $\tau(\nu)$ as:

$$\tau(\nu) = \begin{cases} 1, & (cf(\nu) = 0) \text{ or } (\frac{pf(\nu)}{cf(\nu)} > 1 - \sigma) \\ 0, & \text{otherwise} \end{cases}. \tag{4}$$

Here, σ represents risk tolerance, with larger values corresponding to a greater willingness to trust nodes with imbalanced roles, and smaller values representing a preference for strict balance. Too high, and adversarial nodes will be improperly trusted; too low, and it may reject non-malicious nodes solely due to stochastic fluctuations in tree formation.

3) Classification-Informed Keyscore Boosting: We then leverage $\tau(\nu)$ to perform *classification-informed keyscore boosting* (CIKB) as follows:

$$\kappa_b(o) = \sum_{\mathcal{F} \in \mathcal{E}} \frac{\sum_{\nu \in \mathcal{F}} \mathit{obs}(\nu, o, \mathcal{F}) * \tau(\nu)}{\sum_{\nu \in \mathcal{F}} \tau(\nu)}. \tag{5}$$

Note that, since the $\tau(\nu)$ term is considered for both numerator and the denominator of the observation counting in forests, the cumulative sum of keyscores across all entries in the observation set remains equal to 1.0.

4) Fusion Determination: Once the FC computes $k_b(o)$ for all distinct, valid observations, it makes the final fusion determination by ordering the observations by descending keyscore and selecting the observation with the highest keyscore as the fused observation.

D. Messaging Costs

Under tree-based routing, each node will send a message only once per ensemble; thus, the number of transmissions during the data gathering phase of Pando is $O(nm)$, where n represents the number of nodes and m represents the size of the ensemble. This aligns with efficiency bounds reported by other message-passing approaches [14], although Pando operates under a stronger “colluding” adversarial model. FC data collection costs are proportional to the number of roots in the forest ensemble, and depend on the collection mechanism.

V. Evaluation

A. Experimental Setup

To evaluate Pando, we performed network simulations to trace message flows, including adversarial behavior, under different scenarios. All reported results represent averages across 100 executions. Network topologies were generated randomly using a uniform distribution, and naturally-partitioned networks were re-generated; although not required for Pando functionality, this simplifies evaluation without reducing generality. We tested against $n = 1000$ nodes with average degree ~ 14, for ensemble sizes of $m = \{1, 2, 5, 10, 20\}$ and $\sigma = 0.25$ (appropriate for uniformly distributed networks). Observation cardinality was two, the binary case.

Compromised nodes were chosen at random from the base topology, with adversarial population sizes of $\{0, 10, 20, 30, 40, 50\}\%$. Since Pando is able to detect and filter out all node injections, in the experiments we consider the *worst case* where the compromised nodes perform value injection but not easily-detected node injection. For the traffic dropping behavior, we considered all five deception strategies reported in Section III-B).

B. Effectiveness Measure

We report observation correctness of a sensing round as the effectiveness measure. More specifically, since we evaluate the binary case, with true, $\top$, and false, $\bot$, observations from the network, and since confidence-boosted keyscores across all observations sum to 1, a keyscore for the true observation $\kappa_b(\top)$ less than 0.5 indicates fusion error (failure).

C. Results & Discussion

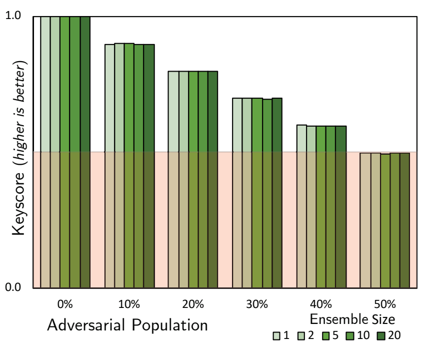
Figure 2(a) shows that for a passive adversary, which does not perform pruning, but injects false observations, the final keyscore reflects its share of the population.

In contrast, for destructive or silent adversaries who aggressively prune the network to reduce the amount of data reaching the FC, Figures 2(b) and 2(c) show that these approaches are easily detectable by the classifier, even for small ensemble sizes and when the adversary controls many nodes.

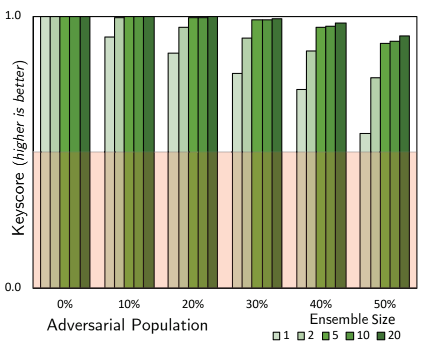
In the sneaky model (Figure 2(d)), the attacker attempts to mislead the FC into misclassifying good nodes by actively (self-)pruning malicious nodes. However, since the heuristic relies on detecting the pruning behavior itself, this actually hurts the adversary more than it helps and correct values dominate the propagation.

When considering the most-challenging selective case, as in Figure 2(e), the classification heuristic provides a strong countermeasure against the attack model by significantly boosting keyscore – even an ensemble size of two provides protection against an adversary controlling 50% of the network.

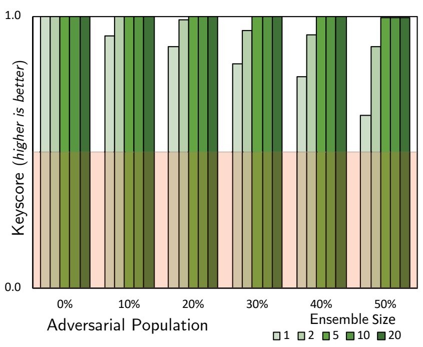
To illustrate the contrast, we also examine the selective model’s performance under a scenario where the classification



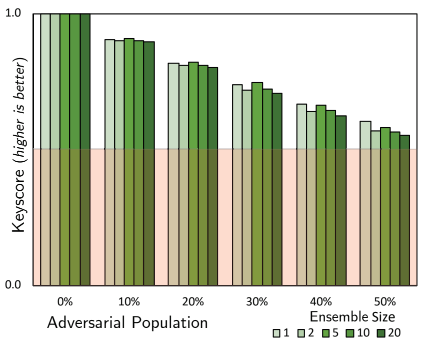
(a) Passive (CIKB)



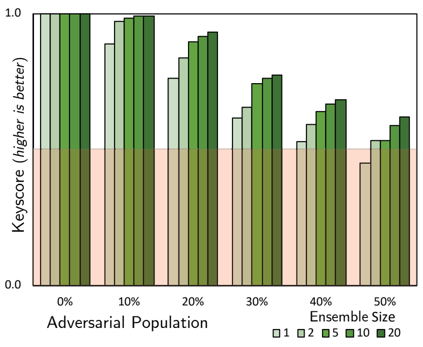
(b) Destructive (CIKB)



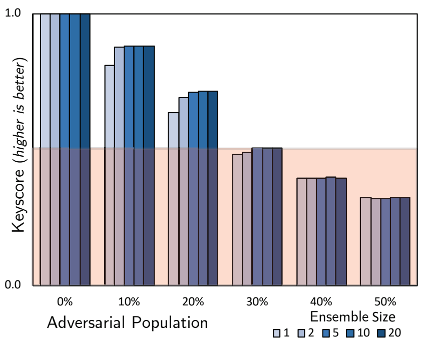
(c) Silent (CIKB)



(d) Sneaky (CIKB)



(e) Selective (CIKB)



(f) Selective (without CIKB)

Fig. 2: Results by deception strategy, adversarial population, and ensemble size; shaded region indicates fusion error

heuristic is artificially disabled (Figure 2(f)). As we see here, when CIKB is not used, the adversary would be able to pass the fusion error threshold when controlling only ∼30% of the network, even for larger ensembles.

Interestingly, we conclude that the adversary's best choice for maximizing fusion error lies in adopting a passive attack model. While this appears counter-intuitive, it aligns well with previous work assessing game theoretic optimality for message-passing fusion protocols [13].

VI. Conclusion

In this work, we introduced Pando, a method for addressing the Byzantine distributed sensor fusion problem. We proposed a method for the creation of decentralized forest ensembles using minimal node-local storage and shared information. Using these ensembles, we described a message passing and validation framework using homomorphic Merkle trees (HMTs) for efficient and secure communication and data aggregation. Finally, we detailed a classification-informed keyscore metric, which leverages topological information embedded in HMTs for robust and reliable sensor fusion.

References

[1] I. F. Akyildiz *et al.*, "Wireless sensor networks: A survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, Mar. 2002.
[2] R. Bruno *et al.*, "Mesh networks: Commodity multihop ad hoc networks," *IEEE Communications Magazine*, Mar. 2005.
[3] J. Kelly and G. S. Sukhatme, "Visual-Inertial Sensor Fusion: Localization, Mapping and Sensor-to-Sensor Self-calibration," *IJRR*, 2011.
[4] W. Fang *et al.*, "Real-time motion tracking for mobile augmented/virtual reality using adaptive visual-inertial fusion," *Sensors*, 2017.
[5] F. Gustafsson, *Statistical Sensor Fusion*. Studentlitteratur AB, 2010.
[6] L. Xiao *et al.*, "A scheme for robust distributed sensor fusion based on average consensus," in *Info. Proc. in Sensor Networks (IPSN)*, 2005.
[7] X. Zhang *et al.*, "Consensus-based distributed sensor fusion over a network," in *Conf. on Control Tech. and Applications (CCTA)*, 2017.
[8] K. Zhang *et al.*, "Distributed Hierarchical Information Acquisition Systems Based on AUV Enabled Sensor Networks," in *ICC*, 2019.
[9] B. Awerbuch *et al.*, "An On-demand Secure Routing Protocol Resilient to Byzantine Failures," in *ACM Workshop on Wireless Security*, 2002.
[10] Y.-C. Hu *et al.*, "Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks," *Wireless Networks; New York*, 2005.
[11] O. Delgado-Mohatar *et al.*, "A light-weight authentication scheme for wireless sensor networks," *Ad Hoc Networks*, 2011.
[12] X. Liu and J. S. Baras, "Using trust in distributed consensus with adversaries in sensor and other networks," in *FUSION*, 2014.
[13] A. Abrardo *et al.*, "A Game-Theoretic Framework for Optimum Decision Fusion in the Presence of Byzantines," *IEEE TIFS*, 2016.
[14] ——, "A message passing approach for decision fusion in adversarial multi-sensor networks," *Information Fusion*, 2018.
[15] H. W. Behrens and K. S. Candan, "Adversarially-Resistant On-Demand Topic Channels for Wireless Sensor Networks," in *SRDS*, 2018.
[16] M. Bellare *et al.*, "Incremental Cryptography: The Case of Hashing and Signing," in *Advances in Cryptology — CRYPTO '94*, 1994.
[17] M. N. Krohn *et al.*, "On-the-fly verification of rateless erasure codes for efficient content distribution," in *Security and Privacy (S&P)*, 2004.
[18] J. Maitin-Shepard *et al.*, "Elliptic Curve Multiset Hash," *The Computer Journal*, vol. 60, no. 4, pp. 476–490, Mar. 2017.
[19] K. Lewi *et al.*, "Securing Update Propagation with Homomorphic Hashing," Facebook, Inc., Tech. Rep. 227, 2019.
[20] H. W. Behrens and K. S. Candan, "Windrose: Adversarially-resistant oblivious routing with masked geographic targeting," in *IEEE Conference on Computer Communications (INFOCOM)*, 2020, in review.
[21] R. C. Merkle, "A Certified Digital Signature," in *Advances in Cryptology (CRYPTO) Proceedings*, 1990.
[22] S. Gil *et al.*, "Guaranteeing spoof-resilient multi-robot networks," *Autonomous Robots*, vol. 41, no. 6, pp. 1383–1400, Aug. 2017.
[23] R. Flickenger *et al.*, "Very Long Distance Wi-fi Networks," in *ACM SIGCOMM Networked Systems for Developing Regions (NSDR)*, 2008.
[24] S. Even, *Graph Algorithms*. Cambridge University Press, 2011.
[25] K. T. Nguyen *et al.*, "Survey on secure communication protocols for the Internet of Things," *Ad Hoc Networks*, vol. 32, pp. 17–31, Sep. 2015.