Secure Data Aggregation Through Proactive Defense

Shuo Chen
VU University Amsterdam
Tsinghua University Beijing
Email: leo.chen.cipher@gmail.com

Guillaume Pierre
VU University Amsterdam
Email: gpierre@cs.vu.nl

Chi-Hung Chi
Tsinghua University Beijing
Email: chichihung@mail.tsinghua.edu.cn

***Abstract*—Gossip based aggregation protocols are a promising approach to monitoring large-scale decentralized IT infrastructures. Compared to traditional approaches they exhibit good properties of scalability, tolerance of churn, and communication overhead. Gossip-based protocols can compute statistical aggregates such as the average, sum or statistical distribution of an attribute across a large system. However, such protocols are extremely vulnerable to malicious attacks, and even a small number of attackers in the system can largely undermine aggregation results. This paper presents a secure protocol for computing attribute averages. In this system, each node autonomously judges whether its neighbors are malicious, and may subsequently stop any interaction with them. A node appearing malicious to its neighbors quickly gets excluded from the system. Instead of defining malicious behavior (and excluding nodes that follow the definition of maliciousness), our system defines correct behavior (and excludes any node that behaves differently). This allows in principle our system to address arbitrary types of attacks. Simulations based on real-world attribute data demonstrate that our system offers good resistance against four different types of attacks.**

I. Introduction

Monitoring large-scale decentralized IT infrastructures such as computing grids and peer-to-peer overlays is difficult. Instead of collecting information from each node individually into some central location, a more scalable alternative relies on gossip-based protocols to efficiently obtain data aggregates – such as the mean of some attribute values across the system, its statistical distribution, or the rank of a node in a given order [1]–[4]. Gossip-based aggregation protocols, like many P2P systems, are very resilient to failures and churn, but very vulnerable against malicious attacks. However, P2P monitoring schemes will not be viable unless they can tolerate the presence of malicious nodes with no major damage.

Past research on secure gossiping has mainly focused on preserving the connectivity of the overlay network and maintaining the message passing integrity [5], [6]. This paper focuses on the data aggregation layer of the classical mean value algorithms [1], [2]. At each periodic gossip cycle each node contacts another node and averages its value with that of the other node. Node values converge exponentially fast towards the global mean.

In such a system, value attacks consist for malicious nodes of distorting the aggregation results by exchanging wrong values at one or more gossip cycle. Value attacks are hard to detect and prevent. At least one relatively weak "baseline" attack cannot be prevented without the help of application-specific knowledge or trusted hardware: malicious nodes can report a wrong attribute value at the beginning of the aggregation but otherwise follow the protocol correctly. We aim at devising secure aggregation protocols such that no other value attack can significantly outperform this baseline attack.

A previous approach to this problem identifies malicious nodes by making nodes sign each interaction cryptographically [7]. By checking signed interaction logs, one can prove that a node is malicious and spread the proof to exclude it from the overlay. This approach however remains vulnerable against colluding attacks where malicious nodes help to cover up the status change of other malicious nodes.

Instead of patching the system to deal with specific attacks, we explore here a different approach where each node is autonomously responsible for deciding that a neighbor behaves in a suspicious manner, and subsequently refusing to interact with it. No coordination between nodes is necessary. Suspicions are based on a strict definition of correct behavior: although no single interaction is sufficient to mark a node as suspicious, a interaction sequence that does not follow the known convergence properties of the aggregation protocol is quickly identified. A malicious node gets gradually blacklisted by all its neighbors, which effectively excludes it from the overlay as a whole. To avoid such exclusion, malicious nodes must weaken their attacks and revert to the baseline attack.

This mechanism may work only under four necessary conditions. First, peers must have persistent identities so that they cannot leave the system and rejoin using a different identity. Second, a malicious node must be disallowed to interact with a new neighbor at each gossip cycle. We enforce repeated interactions between nodes by assigning a static set of neighbors to each node. Third, each node must be able to verify its neighbors' claims autonomously over multiple interactions. We use the known convergence properties of the gossip aggregation protocol to verify each neighbors' claims, then combine past verifications using Dempster-Shafer's law [8]. Finally, our algorithms can have false positives so we must allow excluded nodes to rejoin the overlay. After a given period of time, blacklisted nodes may be accepted again if they behave well.

The contribution of this paper is twofold. First, we introduce a proactive defense mechanism to protect gossip-based aggregation from both known and unknown attacks. Second, we provide simulation evidence to support the effectiveness of our mechanism against four typical types of attack such that no other value attack can significantly outperform the baseline attack.

The rest of this paper is structured as follows. Section 2 describes the context of previous works. Section 3 discusses our proactive defense mechanism model. Section 4 presents the protocol in details, then Section 5 evaluates it again a range of different value attacks. Finally, Section 6 concludes.

II. Related Work

Gossip-based aggregation was first introduced in hierarchical overlays as an efficient and easy way to collect information through a large-scale system [9], [10]. This approach has great scalability. However, maintaining a hierarchy overlay in a dynamic network can be complex and costly and these protocols report aggregates to a single head node only. Gossip-based aggregation over unstructured overlay is a robust yet simple alternative in dynamic network environments [1], [2], [11]. Those aggregation algorithms run in turns. Nodes periodically update their own status, typically a numerical value, based on other nodes' status engaged in a gossip process. These status update mechanisms are carefully designed such that node status converge toward the desired aggregate. Such an approach is robust to failure and churn. It converges exponentially fast to a globally consistent state with a high probability. However, it is also very vulnerable to malicious attacks: a small number of malicious nodes can easily undermine the whole system.

Several approaches have been proposed to secure gossip. These approaches focus on two different levels: connectivity, and application. Some attacks target the peer connectivity by poisoning the peer sampling process. To address this attack, a secure peer sampling service is needed to generate random peer samples of the system even if there exists a significant number of malicious nodes [5], [6], [12]. Our protocol assumes the existence of such protocols and focuses on application-level security. At the application level, approaches focus on specific attacks like free riding in a peer-to-peer streaming service [13]. Our approach focuses on a very different problem but it borrows the pair selection mechanism to defend the aggregation process against frequency attack where malicious node try to initiate as many gossip as possible. Specifically for the averaging protocol, the damage is potentially very important [14].

To our knowledge, there exists only one paper on security for gossip-based aggregation protocols [7]. In this approach, the exchanged status is logged in an undeniable manner using cryptographic methods each time two nodes interact. Any node can verify the history of a suspicious node, and possibly prove it to be malicious. However, besides the relatively high cost of this protocol, it is also vulnerable to a new kind of attack where several malicious nodes collude to provide each other with a "legitimate" history while refuse to provide part of it to normal nodes. On the other hand, we aim to defend this family of protocols against any type of attacks.

III. Model

A gossip-based protocol runs in periodic cycles during which each mode initiates an information exchange with another node. A fix number of cycles constitute an epoch. For each epoch, the protocol restarts itself with potentially new node attributes to aggregate.

Our secure aggregation protocol relies on four essential elements that we discuss next.

A. Persistent Identity Assignment

To build a secure aggregation system, persistent identities are required so that peers cannot leave the system and rejoin using a different identity. Without this, no matter how good a defense is, detected malicious nodes could leave and rejoin under a new identity. Many mechanisms could provide each node with an unforgeable identity, for instance a trusted certificate authority with public-key cryptography [15]. Here we take a simple approach by assuming hosts have a single static IP address, which serves as their identity.

B. Enforced predictable repeated interaction

Frequency attacks, where a malicious node gossips as often as it can, are an effective way for malicious nodes to increase the contamination of a wrong aggregate value. The main difficulty is for a honest node to decide if a received gossip request is legitimate without global knowledge of previous gossips issued by the same initiator node. We address this problem by enforcing predictable repeated interaction between nodes. This mechanism offers two important properties: first, a peer cannot choose its neighbors itself; second, both the initiator and the responder in the gossip process can check if the interaction is valid according to the protocol. We achieve this in two steps.

First we build a static topology on top of a secure peer sampling service such as Brahms [5]. We then construct a similar ring mesh structure as in [16]. For each link i of a node, a secure collision-resistant hash function $hash_{ring}(ip\_address, i)$ maps the node's IP address to an identification number. The identification number for a certain link number is mapped to a ring i. Each node is allowed to gossip only with its predecessor and successor in the ring. Thus for n rings, each peer has exactly $2n$ neighbors. In this way, each node has a fixed number of verifiable neighbors which are pseudo-randomly sampled from all the peers. This overlay structure can be efficiently built using simple gossip protocols like [17].

Second, to eliminate the possibility of frequency attack, a node must not be able to gossip with any of its neighbors at any time. At turn l in a gossip epoch, a peer is authorized to initiate a gossip only with its successor in ring $hash_{gossip}(ip\_address, l)\%n$. With this enforcement, both parties are able to check autonomously that the gossip is legitimate.

This method for constructing 'static' overlays has two advantages. First, a node's list of neighbors changes only if one of these neighbors leaves the system. Should a node leave the system and rejoin anew, it would end up with the set of neighbors. On the other hand, this static overlay topology is naturally resilient to churn as a departed node gets quickly replaced by one of the departed node's own former neighbors.

Second, this overlay can be built in a secure manner: each node can judge independently if a new node it finds belongs to its list of neighbors or not. Since the overlay relies on a secure peer sampling service, attackers cannot prevent honest nodes from discovering each other. The n rings structure provide each node with $2n$ neighbors so that even the malicious nodes play dead to part of honest nodes to break the overlay also would not work. So this process could not be compromised by malicious behaviors.

C. Verifying one neighbor's claims

Each node must be able to verify its neighbors' claims autonomously over multiple interactions. We rely for this on known convergence properties of the aggregation algorithm [1]. The variance value across different peers reduces each turn:

$$\frac{\textit{Variance}_{i-1}}{\textit{Variance}_i} = \mathbb{E}(2^{-\phi}) = \frac{1}{2\sqrt{e}} \tag{1}$$

where ϕ is the number of times for a peer engaged in the gossip process in turn i, $\mathbb{E}$ is the mathematical expectation function and e is the natural log base. In a gossip cycle, if every peer selects its own gossip target randomly, the distribution of the number of gossips per peer can be approximated by a Poisson distribution with mean equal to 2. The estimate system wide variance therefore reduces by a factor of $1/2$ each time it gossips with another peer.

The aggregation protocol can also be viewed as a value diffusion process [2]. After a few initial gossip turns, the value distribution comes close to a binomial distribution, which can be further simplified as a normal distribution. Thus the difference between two values held by different peers should also follow a normal distribution.

Using these two properties, each node can judge statistically whether another node's value is suspicious or not. The probability that a value contributed by node b is not drawn from the legitimate distribution can be estimated by node a as follows:

$$\textit{Suspicion}_{a\to b} = \textit{erf}(\frac{|V_a - V_b|}{\sqrt{2V_{ea}}}) \tag{2}$$

where V_{ea} is the expected variance from peer a, V_a, V_b are the contributed values for peer a and b and $\textit{erf}()$ is the Gauss error function [18].

At the beginning of each epoch, each node exchanges initial values with each of its neighbors to get a first estimate of the variance of values across the whole system. This estimate is subsequently updated according to the expected convergence properties of the aggregation algorithm.

In our system, a single suspicious exchange is not sufficient to blacklist a node; but subsequent suspicion values can quickly combine to build high confidence in a decision to blacklist a node.

We use Dempster-Shafer law to combine multiple suspicion values to arrive at a degree of belief that takes into account all the available evidence [8]. Using Dempster-Shafer's law, the combination of multiple suspicion values provides a single probability estimation called belief. In our system, if a belief value is larger than a threshold θ, the peer considers its neighbor as malicious and thus ceases any gossip interaction with it. Such a decision to blacklist a suspicious node is an entirely autonomous decision from the adjacent node. If the blacklisted node is malicious, then we expect all its neighbors to eventually blacklist it.

Dempster-Shafer combination assumes that different suspicion values are independent from each other. However, an attacker can somewhat contaminate subsequent iterations (e.g., by behaving well for λ cycles, then exploiting the gained trust to issue large damages). We address this in two steps. First, each node maintains a window of the n most recent interactions with its neighbors. The choice of n depends on a tradeoff between the recovery system readiness and the necessary number of aggregated values to build sufficient confidence in the system. Second, instead of allowing suspicion values to use the full range of $[0, 1]$, we restrict each suspicion value into a smaller range $[\alpha, 1-\alpha]$. Otherwise, a single interaction where two nodes expose the same value with suspicions value 0 would overrides any other (potentially malicious) interaction. This restricted suspicion value range should also be symmetric around 0.5 such that a honest aggregation process does not generate an unbalanced suspicions value distribution and subsequently break down the overlay connections.

D. Overlay Recovery

In our system, it is impossible to guarantee that a honest node never gets blacklisted by some of its neighbors. For such a node, losing a few links is not a major issue but this can eventually slow down the aggregation. We must therefore allow blacklisted nodes to be eventually "forgiven" and to rejoin the overlay. However, if malicious nodes can determine the moment when they are rejoin the overlay they can start distorting the system again. Instead, we decided that nodes would not be notified of being blacklisted: when a node blacklists another, it continues interacting with it normally but simply does not update its computed value listed on they interactions.

We use a simple approach to gradually make excluded nodes rejoin overlay. When a node blacklists another, it continues interacting with it normally, but ceases to update its own value according to interactions with the blacklisted node. It however keeps on updating its suspicion value towards the other node with suspicion value 0.5, which means "full uncertainty". After each turn one (presumably low) suspicion value gets removed from the window of suspicion values, and replaced with a neutral one. After a number of "punishment" rounds, the suspicious neighbor gets reintegrated in the overlay.

Fig. 1. Active process

```
var ← initVariance()
while value not converged do
  Wait for δ time
  q ← GetNeighbor()
  send S_p, Time_p to q
  S_q, Time_q ← receive(q)
  if believe(q, S_p, S_q) then
    updateBelief(CalculateBelief(S_p, S_q, var))
    S_p ← updateStates(S_p, S_q)
    var ← updateVar()
  else
    updateBelief(0.5)
  end if
end while
```

Fig. 2. Passive process

```
while value not converged do
  S_q, Time_q ← receive()
  if not checkNeighbor(q, Time_q) then
    continue
  end if
  send S_p, Time_p to q
  if believe(q, S_q, S_p) then
    updateBelief(CalculateBelief(S_p, S_q, var))
    S_p ← updateStates(S_p, S_q)
    var ← updateVar()
  else
    updateBelief(0.5)
  end if
end while
```

This recovery mechanism has an interesting feature: a blacklisted node does not get informed of being blacklisted. All excluded nodes (malicious and non-malicious ones) are eventually forgiven, but malicious nodes cannot know when they should act maliciously.

E. Result post-processing

Besides distorting final aggregation results, value attacks often also result in disturbing the convergence of all nodes to the same final aggregation value. In the worst case scenarios the system does not converge at all.

We use a simple mechanism to reduce the divergence between final aggregation results at different nodes. After the end of the aggregation process, each honest node sample the final aggregation values of its immediate neighbors and retains the median of these values.

IV. Protocol

Our protocol is based on a classical gossip-based aggregation scheme where each node has two different processes as shown in Figure 1 and 2. The active process periodically initiates a state exchange operation with one of the node's neighbors. The passive process waits for incoming connections. The functions in the protocol work as follows:

$updateStates()$ updates the current state of the node by computing $\frac{S_p+S_q}{2}$. $GetNeighbor()$ selects a neighbor from the neighbor list based on the mechanism discussed in Section III-B. $checkNeighbor()$ checks if the other party can legitimately issue a gossip to the concerned node: it first checks if the gossip was initiated by an authorized neighbor at an authorized time; second, it checks based on the other party's suspicion history if the other party is a suspicious node. $updateBelief()$ updates the suspicion value towards a neighbor node based on the current states. We use formula (2) to calculate the suspicion value in function $CalculateBelief()$ of node a towards node b for current gossip cycle. This suspicion value is then stored locally. As described in Section III-C, only the last n suspicion values are stored for the belief combination. The current belief value used in $believe()$ function is calculated from stored suspicion values using Dempster-Shafer's law as follows:

$$Conf_{a\oplus b} = \frac{Conf_a \times Conf_b}{1 - Conf_a - Conf_b + Conf_a \times Conf_b} \quad (3)$$

$updateVar()$ updates this node's estimated global variance based on the analysis of the convergence speed described in section III-C. The estimated global variance is divided by 2 each time the node gossips. $initVariance()$ estimates the global variance from its neighbor's value at the beginning of each epoch as described in Section III-C.

V. Evaluation

We evaluate our proactive defense mechanism using simulations where a given fraction of nodes is malicious and tries to distort the results as much as possible. We evaluate our algorithm in the presence of four attack models of various severity. We do not use synthetic value attributes (whose distribution would be known in advance) but real world node attributes from the SETI@home project [19]. Our evaluation results demonstrate that no attack (including very powerful ones) could do significant larger damage compared to the (unavoidable) baseline attack.

A. Methodology

Several parameters regulate the static overlay maintenance process and the proactive defense aggregation protocol. In our simulations, we assign 20 neighbors to each node in the static random overlay. The suspicion range parameter α is set to 0.05. The window size for the past suspicion value is set to 5. The malicious threshold θ is set to 0.9.

We evaluate our algorithm against four attacks of different severity:

Baseline Attack Each malicious node initially selects a wrong value, then follows the protocol correctly. Theoretically, no secure mechanism can detect such behavior without application-specific knowledge.

Blind Attack Each malicious node acts independently and injects wrong values into the system each round. Such values

TABLE I
VALUE DRAG UNDER DIFFERENT ATTACKS WITH NO RESULT POST-PROCESSING

# of malicious nodes	5%	10%	15%	20%	30%
Baseline attack (wos)[a]	4.62%	9.24%	13.86%	18.48%	27.72%
Baseline attack (ws)[b]	0.64%	3.2%	7.3%	12.4%	23.8%
Blind attack	0.78%	3.67%	8.75%	14.9%	26.9%
Colluding attack	1.50%	5.16%	9.72%	14.8%	23.7%
Supernatural attack	2.17%	6.21%	11.6%	16.7%	27.77%

[a]Baseline attack in a system without security mechanisms
[b]Baseline attack in a system with security mechanisms

TABLE II
VALUE DRAG UNDER DIFFERENT ATTACKS WITH RESULT POST-PROCESSING

# of malicious nodes	5%	10%	15%	20%	30%
Baseline attack (wos)[a]	4.62%	9.24%	13.86%	18.48%	27.72%
Baseline attack (ws)[b]	0.76%	3.2%	7.3%	12.7%	26.8%
Blind attack	0.63%	3.36%	8.97%	15.74%	30.5%
Colluding attack	1.39%	5.06%	9.76%	15.2%	26.6%
Supernatural attack	1.65%	5.79%	11.44%	17.26%	32.12%

[a]Baseline attack in a system without security mechanisms
[b]Baseline attack in a system with security mechanisms

derive from the attacker's estimation of the range of the values that its victim would accept. Malicious nodes do not collude so this estimated acceptable range is constructed only from past interactions experienced by the attacker with its own neighbors. Although malicious nodes do not collude, they all try to distort the result in the same direction (i.e., they do not cancel each other's effects).

Colluding Attack This attack is similar to the blind attack except that all malicious nodes in the system collude to improve their estimation of their victims' acceptable ranges.

Supernatural Attack This attack is a worst-case scenario in which the attackers "magically" know the exact entire internal state of their victims.

Our simulations measure the disturbance that malicious nodes can apply to the aggregation result:

$$ValueDrag = \frac{|V_{result\_avg} - V_{original\_avg}|}{V_{max} - V_{min}} \quad (4)$$

This metric allows to measure the influence of malicious nodes, irrespectively from the actual aggregated result of the system. Also, it allows to compare the secure approach with that of a completely unprotected one. A value drag of 0% represents the ideal case where malicious nodes have no influence on the aggregation result.

B. SETI@home data aggregation

We simulate a system with 10,000 nodes, and give each node a value drawn from public log files of the SETI@home project. We aggregate values that represent the memory size of SETI@home nodes. These values range from 0 to 9×10^9. The average value is 6.84×10^8. Each aggregation lasts for 15 cycles. We execute 50 aggregation epochs (aggregating newly drawn values each time) to give nodes enough time to build confidence values about their neighbors.

A system with no malicious node at all observes a value drag of 0.61% because some honest nodes with outlier values get excluded by their neighbors. This value is however comparable to that of an unsecured system under a simple Baseline attack.

Table I shows the value drag of our system for different fractions of malicious nodes in the system and different types of attacks. Clearly, a large number of malicious nodes has more effect than a small one. For the baseline attack (that our mechanisms are not specifically designed to address), the security measures however allow to mitigate the value drag by eliminating the most extreme contributed values. For example, 10% of malicious nodes can only drag the aggregation value by 3.2%.

The colluding and supernatural attacks are significantly more effective than the baseline attack. This is caused by the fact that shared information between colluding malicious nodes results in a more precise estimation of the range of values that 'victim' nodes would accept as plausible. Building a colluding attack however requires powerful attackers capable of deploying and coordinating large numbers of nodes.

The supernatural attack is the strongest of all considered attacks, but its implementation in real life is probably impossible. It is worth mentioning that even such powerful attack makes less damage against our security mechanism than the baseline attack with no security measure. This is in line with our initial goal that no attack can be more effective than the baseline. Note that colluding attacks were not considered in [7].

Finally, by comparing Table I and Table II we can see that the result post-processing mechanism discussed in Section III-E does not significantly affect the value drag of the system.

One possible limitation of this work is that malicious nodes have more opportunity to disturb the aggregation if the initial distribution of value extends to the full range of acceptable values. In such a case, it is impossible for the system to distinguish a honest outlier from a malicious node. We however note that the experiments shown in this section are based on real-world machine properties (their amount of installed memory) which has a standard deviation in the same order of magnitude as the average value (mean=6.84×10^8, stddev=4.70×10^8). We can therefore consider that our system performs reasonably well, even in a realistic but difficult scenario.

C. Convergence Properties

In a system with no attacker, all nodes from the overlay end up with a final aggregation value very close from each other. However, the presence of a value attack can also compromise the convergence properties of the aggregation algorithm.

We measure the statistical dispersion of aggregation values across honest nodes using the interdecile range metric [20]: the interdecile range is defined as the relative difference between the 90th percentile and the 10th percentile of aggregation

TABLE III
INTERDECILE RANGE UNDER DIFFERENT ATTACKS WITH NO RESULT POST-PROCESSING

# of malicious nodes	5%	10%	15%	20%	30%
Baseline	0.69%	1.58%	3.17%	4.94%	5.98%
Blind	0.87%	3.05%	5.89%	8.91%	10.06%
Colluding	1.46%	3.41%	4.92%	6.54%	9.63%
Supernatural	2.96%	5.75%	8.08%	10.3%	16.0%

TABLE IV
INTERDECILE RANGE UNDER DIFFERENT ATTACKS WITH RESULT POST-PROCESSING

# of malicious nodes	5%	10%	15%	20%	30%
Baseline	0.21%	0.53%	1.02%	1.54%	3.33%
Blind	0.24%	0.80%	1.86%	3.08%	6.94%
Colluding	0.41%	1.02%	1.62%	2.55%	7.12%
Supernatural	0.77%	1.66%	2.46%	3.71%	13.8%

results across the overlay. The lower the interdecile range, the more concentrated the aggregation results are.

$$InterdecileRange = \frac{V_{90th} - V_{10th}}{V_{max} - V_{min}} \quad (5)$$

Table III shows the interdecile ratio before the final result post-processing for different attacks and proportions of malicious nodes in the overlay. Using our system, most nodes converge to values relatively close to each other, while only a few 'victim' nodes located next to malicious nodes end up with very different values (and consequently end up being expelled from the overlay).

Table IV shows the effectiveness of the result post-processing mechanism: the interdecile range values are consistently much lower than in Table III, showing that the final aggregation values are very close from each other among honest nodes.

VI. CONCLUSION

We have presented a proactive defense mechanism for gossip-based data aggregation. In our mechanism, each node is responsible for autonomously judging its neighbors' behaviors and blacklisting them if they appear malicious. This judgment is based on known convergence properties, so any behavior that does not follow these properties is considered as suspicious. This enables our mechanism to defend against know and unknown attacks.

Our system does not provide a perfect protection against value attacks. However, to significantly disturb aggregation results, attackers would need to use and coordinate large numbers of malicious colluding nodes. Such attacks were not considered in [7]. Our mechanism is fully distributed and only adds minor communication overheads to traditional aggregation protocols.

We have focused only on securing the protocol to compute averages. More sophisticated aggregation protocols are based on this basic one [1], [3] so one may consider securing them in a similar fashion.

REFERENCES

[1] M. Jelasity, A. Montresor, and O. Babaoglu, "Gossip-based aggregation in large dynamic networks," *ACM Trans. Comput. Syst.*, vol. 23, no. 3, pp. 219–252, 2005.

[2] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, Washington, DC, USA, 2003, pp. 482+.

[3] J. Sacha, J. Napper, C. Stratan, and G. Pierre, "Adam2: Reliable distribution estimation in decentralised environments," in *Proceedings of the 30th IEEE International Conference on Distributed Computing Systems (ICDCS)*, Jun. 2010.

[4] M. Jelasity and A.-M. Kermarrec, "Ordered slicing of very large-scale overlay networks," in *Proceedings of the 6th International Conference on Peer-to-Peer Computing*, 2006.

[5] E. Bortnikov, M. Gurevich, I. Keidar, G. Kliot, and A. Shraer, "Brahms: byzantine resilient random membership sampling," in *Proceedings of the twenty-seventh ACM symposium on Principles of distributed computing (PODC)*. New York, NY, USA: ACM, 2008, pp. 145–154.

[6] G. P. Jesi, D. Hales, and M. van Steen, "Identifying malicious peers before it's too late: A decentralized secure peer sampling service," in *SASO '07: Proceedings of the First International Conference on Self-Adaptive and Self-Organizing Systems*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 237–246.

[7] M. Jelasity, A. Montresor, and O. Babaoglu, "Towards secure epidemics: Detection and removal of malicious peers in epidemic-style protocols," 2003.

[8] G. Shafer, "Perspectives on the theory and practice of belief functions," *International Journal of Approximate Reasoning*, vol. 4, no. 5-6, pp. 323–362+, 1990.

[9] I. Gupta, R. v. Renesse, and K. P. Birman, "Scalable fault-tolerant aggregation in large process groups," in *DSN '01: Proceedings of the 2001 International Conference on Dependable Systems and Networks (formerly: FTCS)*. Washington, DC, USA: IEEE Computer Society, 2001, pp. 433–442.

[10] R. Van Renesse, K. P. Birman, and W. Vogels, "Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining," *ACM Trans. Comput. Syst.*, vol. 21, no. 2, pp. 164–206, 2003.

[11] A. Montresor, M. Jelasity, and O. Babaoglu, "Decentralized ranking in large-scale overlay networks," in *Self-Adaptive and Self-Organizing Systems Workshops, 2008. SASOW 2008. Second IEEE International Conference on*, 20-24 Oct. 2008, pp. 208 – 213.

[12] G. P. Jesi, E. Mollona, S. K. Nair, and M. van Steen, "Prestige-based peer sampling service: interdisciplinary approach to secure gossip," in *SAC '09: Proceedings of the 2009 ACM symposium on Applied Computing*. New York, NY, USA: ACM, 2009, pp. 1209–1213.

[13] H. C. Li, A. Clement, E. L. Wong, J. Napper, I. Roy, L. Alvisi, and M. Dahlin, "Bar gossip," in *OSDI '06: Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation*. Berkeley, CA, USA: USENIX Association, 2006, pp. 14–14.

[14] Y. Bachrach, A. Parnes, A. D. Procaccia, and J. S. Rosenschein, "Gossip-based aggregation of trust in decentralized reputation systems," *Autonomous Agents and Multi-Agent Systems*, vol. 19, no. 2, pp. 153–172, October 2009.

[15] J. R. Douceur, "The sybil attack," in *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*. London, UK: Springer-Verlag, 2002, pp. 251–260.

[16] H. Johansen, A. Allavena, and R. van Renesse, "Fireflies: scalable support for intrusion-tolerant network overlays," in *EuroSys '06: Proceedings of the 1st ACM SIGOPS/EuroSys European Conference on Computer Systems 2006*. New York, NY, USA: ACM, 2006, pp. 3–13.

[17] S. Voulgaris, M. Jelasity, and M. van Steen, "A robust and scalable peer-to-peer gossiping protocol," in *Agents and Peer-to-Peer Computing*, ser. Lecture Notes in Computer Science, vol. 2872. Springer Berlin / Heidelberg, 2005, pp. 47–58.

[18] Wikipedia, "Error function." [Online]. Available: http://en.wikipedia.org/wiki/Error_function

[19] "SETI@home statistics," http://setiathome.berkeley.edu/stats/.

[20] Wikipedia, "Interdecile range," http://en.wikipedia.org/wiki/Interdecile_range.