Secure Distributed Top-k Aggregation

Kristjan V. Jonsson*, Karl Palmskog† and Ymir Vigfusson*
*School of Computer Science, Reykjavik University, Iceland
†School of Computer Science and Communication, KTH Royal Institute of Technology, Sweden

***Abstract*—In-network aggregation is an efficient and scalable distributed approach to global state approximation. However, security remains an open problem in such systems, especially when we consider dynamic network effects, such as mobility, packet loss and churn. In this paper, we consider the resilience of the top-k aggregate to manipulation by active insider adversaries. Unfortunately, this versatile aggregation function is inherently insecure. We propose a simple, low-overhead solution which leverages the principles of trusted systems. The solution we propose is generally applicable, even to the challenging problem of securing distributed aggregation in a dynamic network.**

***Index Terms*—Distributed systems, secure in-network aggregation, top-k aggregation, trusted systems**

I. Introduction

Aggregating local inputs in a distributed system to approximate the global state is an important and powerful technique [1]. One can achieve sub-linear scalability properties when the aggregate is computed in-network by a network of cooperating nodes [1]–[7]. Here, we focus on the TOP-K distributed aggregation function which has numerous applications in sensor networks [8], [9], distributed databases [10] and distributed systems [1], [11], [12]. Our work is motivated by applications that include decentralized web site ranking and meta-data search in information networks [13].

It has been shown that many useful aggregation functions can be trivially manipulated by corrupt insiders [14]; we extend these results to the top-k aggregate. The problem becomes even harder when considering the cooperative processing in distributed aggregation systems, since powerful attacks against the data integrity can be carried out *stealthily* [15] by a small subset of corrupt nodes.

Previous approaches to secure distributed aggregation generally use expensive security protocols or place severe restrictions on the types of networks and services they support [15]–[20]. By contrast, we propose a simple, modular and low-overhead technique, using established principles of trusted systems [21], [22] to perform distributed aggregation in a secure fashion while preserving the scalability properties of the underlying aggregation protocol.

Unlike the previous work, we propose to limit the opportunity for adversaries to influence the aggregate computation *a priori*. To this end, we define an *overlay* composed of *trusted modules* and communications protocols. Trusted modules are *trusted data sources*, such as trusted sensors [23], and *trusted aggregators*, for which trust is established by verification of interfaces and functionality, followed by attestation of correctness by a trusted entity.

The trusted overlay is embedded in a distributed aggregation system of untrusted nodes. All nodes which contribute to a trusted system must host at least a trusted data source, while those which perform in-network aggregation must host a trusted aggregator. The trusted modules operate in a symbiotic relationship with their otherwise untrusted host, providing only those services necessary to enable trustworthy aggregation, while depending on the untrusted hosts for essential services, such as node discovery, networking and inter-process communications.

The mechanism of combining trusted modules and protocols is simple and allows us to claim rather strong security guarantees while at the same time maintaining low overhead. Moreover, our approach is unique in that it applies in the *general case* of distributed aggregation, that is, to arbitrary aggregate data types and functions, as well as to dynamic networked systems.

II. System Model

Network model. We consider a distributed system of *aggregation nodes* that propagate queries and compute aggregate results cooperatively *in-network* over an *aggregation overlay* formed over some arbitrary communications graph. Each node may contribute local input or state. Various overlay topologies and aggregation protocols may be used; we restrict our attention to spanning-tree overlays, and thus a family of tree-based protocols [3]–[5].

Aggregation model. We consider two distinct distributed query models: *one-shot queries* [3] and *continuous aggregation* [5]. One-shot queries are essentially a variant of the broadcast-convergecast protocol [24]: a querier initiates the protocol by broadcasting a query message to its nearest neighbors. The query propagates outward in the network graph until it reaches the leaves, at which point the aggregate computation begins by a convergecast of partial results towards the querier. Each aggregation node v applies a function $f(I_v; m_1, \ldots, m_x)$ over its local input I_v (if any) and update messages received from contributing peers, as shown in Figure 1. Finally, the querier produces the

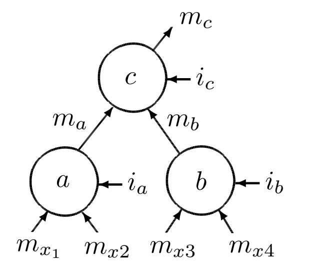

Fig. 1. Aggregation node c receives updates m_a and m_b from children a and b. Local inputs are i_a, i_b and i_c.

aggregate result by combining the received partial aggregates. Continuous aggregation proceeds in a similar manner. The participating nodes autonomously issue updates according to some schedule or in response to an event, such as a threshold crossings [25]. Updates are aggregated in-network as in the one-shot case.

Aggregation function: Our security mechanisms can in principle be applied to any data type and aggregation function. We focus on TOP–K–WEIGHTED, a variant of the TOP–K function [10]. The inputs are vectors $\mathbf{a} = [a_1, \ldots, a_k]$, where $a_j = (j, w, \mathbf{d})$ is a tuple consisting of data item identifier j, weight w and optional meta-data $\mathbf{d}$. The vector $\mathbf{a}$ is ordered by decreasing weight. The local function executing on node v merges local observations $\mathbf{a}_v$ with partial aggregates $\mathbf{a}_1^{|k|}, \ldots, \mathbf{a}_x^{|k|}$ received from contributing peers. A sub-aggregation function, such as SUM, MIN or MAX, is applied to weights in tuples when merging vectors, after which an ordering operation is performed, resulting in a new partial aggregate $\mathbf{a}'$. Finally a message $m_i = \langle \text{TRUNC}_k(\mathbf{a}') \rangle$ is sent to the parent in the aggregation tree.

Adversarial model. All aggregation nodes, except the querier (the root of the aggregation tree), as well as all communications links in the aggregation overlay are considered inherently untrusted. The querier is implicitly trusted because it is both the originator of the query and the consumer of the information produced.

Our focus is on the *integrity* properties of the TOP–K–WEIGHTED aggregation function. We consider an active *stealthy insider adversary* [15], which corrupts some fraction $t \in [0, 1)$ of the node population. Corrupt nodes may attempt to modify either local observations or partial aggregates as well as dropping update messages, but attempt to do so while evading detection. We will not discuss availability attacks (jamming or denial-of-service attacks) because they contradict the objectives of our stealthy adversary. In the same vein, we disregard attacks against node discovery and routing.

III. Applications of TOP–K–WEIGHTED

Secure TOP–K–WEIGHTED computation is motivated by the two following examples.

A. Secure Queries in the Network of Information

The Network of Information (NetInf) [26] is a recent network architecture that differs from current device-centric networks in that each piece of information, called an Information Object (IO), can be retrieved without direct knowledge of its location. The problem of scalable meta-data directed search for IOs has previously been addressed using scalable one-shot and continuous aggregation on tree overlays [13], but without considering security issues. Meta-data directed search can be viewed as a special case of TOP–K–WEIGHTED queries. To implement such searches, one can use the low-churn Dictionary Nodes (DN) which implement the name resolution system and routing infrastructure in NetInf using multiple-level Distributed Hash Tables (DHT). Continuous aggregation can be used as a form of publish/subscribe system since meta-data and content of IOs can be updated over time.

The integrity of IOs – content authenticity and accuracy of meta-data such as creation time – can be guaranteed given a trusted Certificate Authority (CA). However, DNs are expected run in different Autonomous Systems and thus may only have their own CA signing each IO, with DN operators potentially acting maliciously. As adapted to the network model, nodes are DNs in an overlay which is precomputed by a separate protocol. Local inputs are IOs matching the given query that are located at the DN, along with a measure of how close the match was, as provided by the local name resolution system. The root of the tree overlay can then trust the authenticity of the aggregated matches by trusting the devices at each DN that verify the signature for each relevant IO before passing it upwards in the tree. Assuming that the DHT of each DN is not corrupted and that the top-k relevant IOs can be retrieved securely, trusted communication channels ensure that only relevant IOs are propagated, even in the presence of adversaries.

B. Trustworthy Assessment of Web Site Popularity

Web site popularity ranking is an important metric for operators, not least because it attracts advertising revenue. Currently, such statistics are gathered by clicktrackers – a notoriously privacy-invasive mechanism [27].

Let us consider an alternative based on distributed *top-k* aggregation. We build a cooperative network of domains, each of which contributes one aggregation node, equipped with a secure flow monitor (sensor) and an aggregator. Web usage statistics are aggregated per domain by the flow monitor and a global ranking of the top-k sites computed in near-real-time using distributed aggregation.

The goals of the cooperating organizations are sometimes in conflict, implying that some parties may have an incentive to manipulate the ranking. To address this issue, we insist that monitoring and aggregation takes place with the assistance of trusted devices. Using a trusted flow monitor to collect Web usage statistics protects user privacy, while trusted aggregation ensures a trustworthy global ranking.

IV. The security of top-k

We have argued that the top-k aggregate has useful applications, but the question of security remains: *can we trust our results?* Let us consider (informally) the integrity properties of the TOP–K–WEIGHTED function as a basis for the subsequent discussion.

We begin by examining manipulation of local inputs in Wagner's single aggregator model [14] in which several corruptible contributors submit their local state to a single aggregator. We decompose the local function into its elementary components:

$$\mathbf{a}' = \mathsf{TRUNC}_k(\mathsf{SORT}(\mathsf{MERGE}(\mathbf{a}_1, \ldots, \mathbf{a}_x)))$$

Let us first consider the MERGE function, assuming a summation operation: for contributions $\mathbf{a}_u$ and $\mathbf{a}_v$, the computing

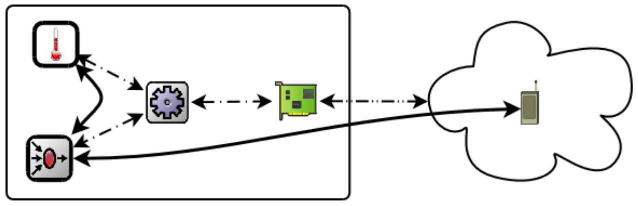

Fig. 2. Aggregation node schematic. Trusted modules and channels are shown with bold outlines. The untrusted local process is indicated by the cogwheel symbol.

node sets weight $w_k = w_{iu} + w_{jv}$, where $\mathrm{ID}_k = \mathrm{ID}_{iu} = \mathrm{ID}_{jv}$. By extension of Wagner's result for SUM [14], we conclude that a merge function based on summation is inherently insecure. The same applies for several other common functions, such as MIN and MAX. The outcome of the outer two functions depends on the computed weights. Hence, their results are trivially influenced via an insecure merge function. We conclude that TOP–K–WEIGHTED must be considered as insecure in Wagner's model as the MERGE function w.r.t. the ordering, and hence top-k ranked data.

Now, we extend the model to the case where the aggregator is untrusted, as is the case in the single aggregator model of Chan *et al.* [15]. It is clear that if the aggregate is vulnerable to local data falsification by the data providers, then it is also vulnerable to manipulation by a corrupt aggregator. Moreover, the influence of a corrupt aggregator is manifold that of a data provider. Considering a distributed system in which one or more aggregators can be corrupt, we must conclude that the distributed aggregation function as a whole is indeed insecure, although the influence of the adversary depends on the number and placement of corrupt nodes.

V. *TSense*: A System for Secure Aggregation

We now present *TSense*, a conceptually simple solution to the problem of securing TOP–K–WEIGHTED, as well as other inherently insecure aggregation functions, based on the principles of trusted systems [21], [22]. TSense enables trustworthy distributed aggregation even if a subset of the node population is corrupt. Further details are presented in earlier work [23], [28].

Our approach is based on limiting the adversaries' opportunity for influencing the aggregate *a priori* by tightening the circle of trust: participants in the trusted aggregation overlay prove the correctness of their actions by deferring to a single root of trust, e.g. a certification authority, via *trusted modules* which perform distributed aggregation in a provably correct manner.

Architecture of a Secured Node. The architecture of a secured aggregation node is shown in Figure 2. Local observations are provided by a *trusted data source*, e.g. a *trusted sensor* [23], while local computations are carried out by a *trusted aggregator*. The properties of the trusted nodes can be defined as follows:

i *Trusted data sources*: A local input is correctly represented and an authenticated data stream produced, but only when in contact with an authenticated trusted peer.

ii *Trusted aggregators*: An aggregation function is correctly computed for a set of verified inputs from partners verified as trusted and an authenticated data stream produced, again only delivered to authenticated trusted partners.

Trusted modules operate in a close symbiotic relationship with the otherwise untrusted hosting nodes. Specifically, we assume the existence of an inherently untrusted process executing on the nodes whose assistance is required to execute the aggregation protocol. Services accessed by this process include inherently untrusted platform services such as peer node discovery, inter-process communication and networking facilities.

Trusted aggregation overlay. A *trusted overlay* is formed by connecting trusted modules with secure virtual channels, either using the internal IPC mechanisms of the hosting node or generic network links. The trusted channels are established by strong mutual authentication and cryptographic protocols. We leave the choice of protocols open for this work, remarking that secure channels can be readily constructed using protocols such as EC-STS [29]. Figure 2 shows trusted intra-node channels established between the trusted sensor and aggregator, while trusted inter-node channels connect the aggregation node (or rather its trusted aggregator) to a peer in the aggregation overlay.

Device Trust Establishment. Establishing trust between modules, and hence channel setup, requires us to establish a per-device basis of trust. We must accept as fact that a given device is trustworthy. This requires (a) an unique and verifiable trusted module identity and (b) a trusted process which verifies the operation of the device against a well-defined set of operational and security specifications.

We assume the module identity is in the form of a public globally unique identity and a set of asymmetric keys for signing and verifying signatures. The identity must be certified by a digital signature traceable to a trusted third party – that is, a party which the querier trusts – and permanently bound to the trusted module. Hence, an adversary must not be able to alter or replace device identities of "captured" devices.

The signature must also imply that the trusted party has verified the device as correct according to the set of specifications. The strictness of the verification procedures depends on the level of trust that the consumer of the data places in the data. A thorough code review may be sufficient for some consumers, while others may require more absolute guarantees, the most strict being a complete formal verification process. However, such methods are still in their infancy and as yet not practical except for fairly restricted systems.

In any case, verification implies minimality: In order to verification to be tractable, the device should only implement the bare minimum of functionality necessary to meet the security requirements. This motivates our choice of a symbiotic relationship of small trusted modules and otherwise untrusted generic hosts.

Tamper resistance. Having formally verified a trusted module as correct, we want to ensure that it cannot be tampered with. We have previously described a solution using tamper-proof hardware devices, verified as correct and signed by a trusted entity [23], [28]. One could also envision an alternative approach in which trusted software modules are delivered by trustworthy means and instantiated in a trusted execution environment, such as a trusted hypervisor.

VI. A Brief Analysis of System Properties

The simple solution of using trusted devices to enable secure aggregation has a number of strong properties in terms of our adversarial model. We summarize the key properties below, and refer the reader to previous work [23], [28].

First of all, we have established that given strong pairwise trust, traceable to a single mutually trusted entity, we can claim that all contributing nodes must follow the protocol: each node executes code verified to comply with a security specification and in a manner which cannot be tampered with. Hence, we can claim that all contributions to the aggregate computation are *correct*, as defined by Narasimha and Tsudik [30] in the context of outsourced databases. The other side of the coin, *completeness* [30] in the dynamic case is more problematic, since we must account for natural effects in dynamic systems such as churn, mobility and node failures, in addition to the adversarial situation in which corrupt nodes drop or corrupt messages according to some strategy. We discuss the impact of completeness further in Section VII, although measures to ensure completeness in the adversarial case are not addressed in this work.

Second, we claim that the system imposes a low overhead. Let us ignore for the time being local per-node overhead, such as that due to economic costs, processing, memory or power requirements. These factors can be accounted for in the design and provisioning of trusted modules and hosting platforms. Instead, we concentrate on the communications complexity which poses a challenge from a scalability standpoint. The primary requirement for our solution is that it does not negatively impact the inherent scalability properties of the underlying distributed aggregation algorithm. The communications complexity on top of the underlying protocol is due to (a) authentication and (b) data transfer. Assuming that authentication is a rare event in a relatively stable network, we can state that the majority of traffic is due to data transfer, which adds constant overhead on top of the underlying aggregation protocol stack.

Third, we claim that the solution effectively and transparently secures dynamic aggregation protocols, such as GAP [5]. We substantiate this claim by the fact that the trusted modules construct a secure distributed execution environment for the essential sub-protocols. Given that the aggregation protocol in question is decomposed into its secure and generic components, we can claim that it retains its original scalability properties (modulo authentication) as well as completeness guarantees and accuracy objectives [31]. Note that although we focus on tree-based aggregation protocols in this work, the trusted devices approach is generally applicable to distributed aggregation protocols, e.g. gossip-based aggregation protocols [6].

Finally, let us examine what an adversary can accomplish by a strategy of ignoring trusted devices. In fact, an adversary accomplishes little except reducing his own importance: an adversary that ignores trusted devices, either on its own host or neighboring machines, is effectively excluding nodes under his control, which is precisely the goal of protocols for detection and elimination of corrupt nodes, such as reputation mechanisms [32]. Since a device which cannot produce a trusted data stream cannot be a member of the trusted aggregation overlay, a node carrying trusted devices can elect to participate fully and truthfully in the protocol or to not do so at all.

VII. A Simulation Study

Let us now study the Web site popularity ranking example from Section III by means of a small simulation study.

In the scenario shown, N nodes, representing Web portals, are initially created and assigned randomly generated web site usage statistics over some $m = \mathcal{N}(\mu, \sigma)$ visits. We assume that hits to Web sites follow a power-law distribution, ranking sites in expected descending order according to the probability density function $p(i, \alpha) \propto i^{-\alpha}$, where $i \in \mathbb{N}$ is both the website ID and its expected rank and α is a parameter of the distribution. Web site popularity rankings, as well as a plethora of other network phenomena, have been shown to follow a power-law distribution [33]. We show simulation runs for a small n-ary tree ($\Delta = 3$, $h = 3$) and a single one-shot query with $k = 10$ issued by the root. We average 50 independent simulation runs to produce the result. The local state of each node is initialized at startup of each run using $\lceil \mathcal{N}(10000, 2500) \rceil$ web requests, the target of each being randomly selected according to a power-law distribution with $\alpha = 1.0$.

Unrestricted adversaries. We begin by considering the case in which the adversary is unrestricted, meaning that corrupt nodes may modify local observations or partial aggregate computation in an arbitrary manner. The objective of the adversary in this scenario is to falsely promote the low ranking sites 22, 24 and 26. An expected $t = 15\%$ of nodes, excluding the root, are corrupted at the start of the simulation and execute a local corruption function $w_i = f_C(\mathbf{a}) = \mathcal{N}(\gamma \cdot w_1, \varrho \cdot w_1)$: the weight of a falsely promoted site i is drawn from a normal distribution centered around the weight of the currently highest ranked site. Figure 3(a) shows the results for $\gamma = 1$ and $\varrho = 1/3$. In the first experiment, we let the corrupt nodes modify only local observations, while modifying the aggregate computation itself in the second. The unmodified aggregate is shown as a baseline reference and follows a power-law over several orders of magnitude. Observe that the attack on the aggregate computation is the more powerful of the two, as expected. The results support our earlier findings in Section IV that the TOP-K-WEIGHTED function is insecure: arbitrary bias can be introduced by even a single node. Note that the querier has no way of disputing the aggregate result. Even statistical comparisons with previous runs must be considered inconclusive as transient effects, such as flash crowds [34], might well introduce legitimate bias in current or past results.

Limiting the adversary. We now restrict the capability of corrupt to dropping packets, simulating the introduction of trusted devices. Two such cases are shown in Figure 3(b). Note that completeness is not addressed in this scenario, allowing the adversary the opportunity to drop updates at will. In the first case considered, the adversary directs the corrupt nodes to discard all aggregate updates. In the second case, we assume that the corrupt nodes can view, but not modify, partial aggregates. This enables the corrupt nodes to mount a more intelligent attack. We consider an attack in which the adversary drops all packets other than those containing a measurement for site 24. Neither attack is particularly effective in influencing the ranking, although the intelligent dropper manages to introduce a small bias, as can be seen in the enlarged view.

VIII. Background and Related Work

Our work benefits from a rich collection of prior work; space constraints force us to review only a few select papers from the literature.

Distributed aggregation protocols [1]–[7] are inherently vulnerable to stealthy data modification [14], [15] attacks, severely reducing their utility for critical applications. Extensive work has been carried out in the field of secure aggregation to assure the integrity of aggregates [15]–[20]. However, the work cited imposes severe restrictions on the aggregation system, such as assuming a limited set of aggregation functions, specific data types, a small number of possible adversaries, static networks and imposing expensive integrity assurance protocols that may undermine the scaling properties of distributed aggregation protocols. In contrast, TSense supports arbitrarily complex data types and aggregation functions, and does so in dynamic networks without any restriction on the number of compromised nodes and while preserving the scalability properties of the underlying protocol stack.

The top-k aggregation problem is relevant in many fields: sensor networks [8], [9], distributed databases [10], distributed systems [1], [11], [12], cloud computing [35] and meta-data directed search in the network of information [13] to name a few. Security-wise, there is extensive literature on privacy-preserving top-k aggregation [36], [37] which we have also considered in previous work [38]. However, there is surprisingly little work on the integrity vulnerabilities of top-k queries in distributed aggregation.

Secure devices and trusted systems principles [22] have been considered extensively before, for instance as a solution to the fair exchange problem [39]. Secure storage is considered by Maheshwari *et al.* [40] and simple non-equivocation mechanisms by Levin *et al.* [41]. Secure sensors have been proposed previously [23], [42]–[44]. To the best of our knowledge, we are the first to consider a trusted systems approach to the wider problem of integrity preservation in distributed aggregation. Our solution is in some aspects comparable to the outsourced aggregation model, which Nath *et al.* address by the application of one-way hash chains [45]. However, our solution is conceptually simpler and continues to apply in more general distributed aggregation settings.

IX. Conclusions and Future Work

We present TSense, a system for simple and efficient trusted distributed aggregation based on trusted systems principles. At a basic level, we construct a system for trusted distributed function evaluation by using provably correct trusted modules, which are traceable to a single root of trust. This approach is generally applicable to the entire family of distributed aggregation protocols, including those intended for dynamic networked systems.

We focus our attention on the security of the top-k aggregate, motivated by application examples. We extend previous results to show that aggregate is inherently insecure – more precisely, as insecure as the sub-aggregate used to compute the ranking weights. Using a small simulation, we demonstrate that the trusted systems principles do succeed in limiting the adversarial capabilities sufficiently to consider distributed top-k aggregation secure with respect to the integrity of the global estimate, modulo a small bias due to malicious drops.

Our results are a stepping stone towards the larger goal of developing TSense into a comprehensive framework for secure sensing, information gathering and distributed aggregation. Future work includes constructing a system for distributing attested software modules, which can be securely instantiated on participating platforms in such a manner that they can prove the correctness of executing modules.

Acknowledgements

The authors thank the anonymous reviewers for their valuable input on preparing this final manuscript. Kristjan is supported in part by by Graduate Studies Grant #080520008 by Rannís, the Icelandic Centre for Research, the Icelandic Center of Excellence in Theoretical Computer Science (ICE-TCS) and Rannís grant #090032021, as well as funding by Reykjavik University.

References

[1] S. Keshav, "Efficient and decentralized computation of approximate global state," *SIGCOMM CCR*, vol. 36, no. 1, pp. 69–74, 2006.

[2] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *MobiCom*, 2000, pp. 56–67.

[3] K.-S. Lim and R. Stadler, "A navigation pattern for scalable internet management," in *IFIP/IEEE INM*, Seattle, USA, 2001, pp. 405–420.

[4] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "TAG: A Tiny AGgregation service for ad-hoc sensor networks," in *OSDI*, 2002, pp. 131–146.

[5] M. Dam and R. Stadler, "A generic protocol for network state aggregation," in *RVK 05*, Linköping, Sweden, Jun. 2005.

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

[7] R. Stadler, M. Dam, A. Gonzalez, and F. Wuhib, "Decentralized real-time monitoring of network-wide aggregates," in *LADIS*. New York, NY, USA: ACM, 2008, pp. 1–6.

[8] B. Chen and G. Min, "Robust top-k query evaluation in wireless sensor networks," in *IEEE CIT*, July 2010, pp. 660–667.

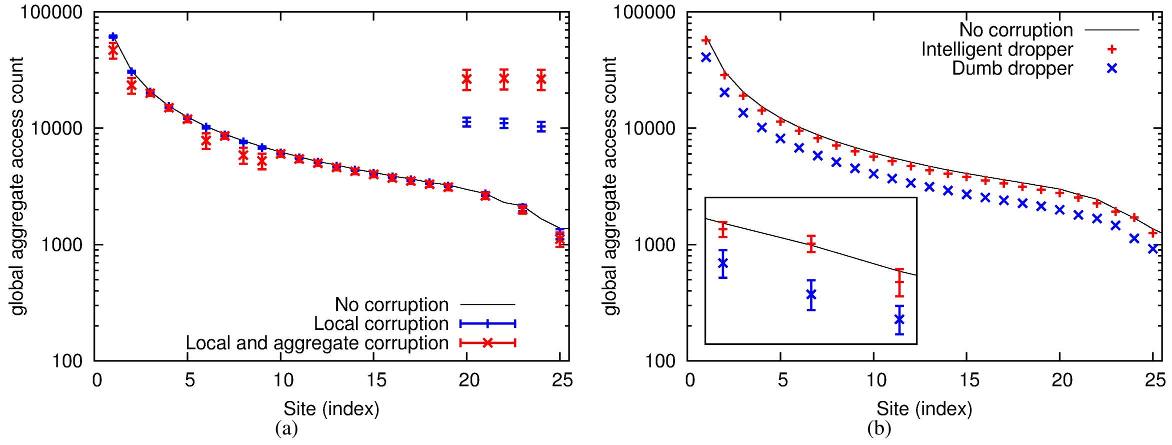

Fig. 3. Web site popularity assessed by distributed in-network aggregation over an n-ary tree with $\Delta = 3$, $h = 3$. Expected $t = 15\%$ of nodes corrupted at the beginning of the simulation. (a) Unrestricted adversary. (b) Adversary restricted to drop attacks. Error bars omitted for clarity. Insert shows expanded view centered around the measurement for site 24.

[9] B. Malhotra, M. A. Nascimento, and I. Nikolaidis, "Exact top-k queries in wireless sensor networks," *IEEE Trans. Knowledge and Data Eng.*, vol. 23, pp. 1513–1525, 2011.

[10] A. Marian, N. Bruno, and L. Gravano, "Evaluating top-k queries over web-accessible databases," *ACM Trans. Database Syst.*, vol. 29, pp. 319–362, June 2004.

[11] B. Babcock and C. Olston, "Distributed top-k monitoring," in *ACM SIGMOD*. New York, NY, USA: ACM, 2003, pp. 28–39.

[12] W.-T. Balke, W. Nejdl, W. Siberski, and U. Thaden, "Progressive distributed top-k retrieval in peer-to-peer networks," in *ICDE*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 174–185.

[13] K. Palmskog, A. Gonzalez Prieto, C. Meirosu, R. Stadler, and M. Dam, "Scalable metadata-directed search in a network of information," in *Future Network & Mobile Summit*, 2010.

[14] D. Wagner, "Resilient aggregation in sensor networks," in *SASN*. New York, NY, USA: ACM, 2004, pp. 78–87.

[15] H. Chan, A. Perrig, B. Przydatek, and D. Song, "SIA: Secure information aggregation in sensor networks," *Journ. of Computer Sec.*, vol. 15, no. 1, pp. 69–102, 2007.

[16] L. Hu and D. Evans, "Secure aggregation for wireless networks," *SAINT*, pp. 384–391, Jan. 2003.

[17] H. Chan, A. Perrig, and D. Song, "Secure hierarchical in-network aggregation in sensor networks," in *CCS*. New York, NY, USA: ACM, 2006, pp. 278–287.

[18] M. Garofalakis, J. Hellerstein, and P. Maniatis, "Proof sketches: Verifiable in-network aggregation," *IEEE ICDE*, pp. 996–1005, April 2007.

[19] S. Nath, P. B. Gibbons, S. Seshan, and Z. Anderson, "Synopsis diffusion for robust aggregation in sensor networks," *ACM Trans. Sen. Netw.*, vol. 4, no. 2, pp. 1–40, 2008.

[20] H. Yu, "Secure and highly-available aggregation queries in large-scale sensor networks via set sampling," in *IEEE IPSN*, Washington, DC, USA, 2009, pp. 1–12.

[21] J. P. Anderson, "Computer security technology planning study," U.S. Airforce Electronic Systems Division, Deputy for Command and Management Systems, HQ Electronic Systems Division, Tech. Rep. ESD-TR-73-51, October 1972.

[22] U.S. Department of Defense, "Trusted computer system evaluation criteria (orange book)," December 1985.

[23] K. V. Jónsson and Ý. Vigfússon, "Bootstrapping trust in networked measurement systems with secure sensors," in *IEEE Sensor Applications Symposium (SAS)*, Brescia, Italy, Feb. 2012.

[24] D. Peleg, *Distributed Computing: A Locality-Sensitive Approach (SIAM Monographs on Discrete Mathematics and Applications 5)*. Society for Industrial and Applied Mathematics SIAM, 2000.

[25] F. Wuhib, M. Dam, and R. Stadler, "Decentralized detection of global threshold crossings using aggregation trees," *Computer Networks*, vol. 52, no. 9, pp. 1745–1761, Feb. 2008.

[26] M. D'Ambrosio, M. Marchisio, and V. Vercellone, "Second NetInf architecture description," 4WARD project, Tech. Rep., 2010, fP7-ICT-2007-1-216041-4WARD/D-6.2.

[27] B. Krishnamurthy and C. E. Wills, "Generating a privacy footprint on the internet," in *ACM SIGCOMM Conf. Internet Measurement*. New York, NY, USA: ACM, 2006, pp. 65–70.

[28] K. V. Jónsson and Ý. Vigfússon, "Securing distributed aggregation with trusted devices," in *NordSec*, Tallinn, Estonia, Oct. 2011.

[29] D. Hankerson, A. Menezes, and S. Vanslone, *Guide to Elliptic Curve Cryptography*. Springer, 2004.

[30] M. Narasimha and G. Tsudik, "Authentication of outsourced databases using signature aggregation and chaining," in *DASFAA*, 2006, pp. 420–436.

[31] A. Gonzalez Prieto and R. Stadler, "A-GAP: An adaptive protocol for continuous network monitoring with accuracy objectives," *IEEE Trans. on Network and Service Management*, 2007.

[32] S. Buchegger and J.-Y. Le Boudec, "A robust reputation system for mobile ad-hoc networks," EPFL IC, Tech. Rep. IC/2003/50, 2003.

[33] L. A. Adamic and B. A. Huberman, "Zipfs law and the internet," *Glottometrics*, vol. 3, pp. 143–150, 2002.

[34] P. Barford, J. Kline, D. Plonka, and A. Ron, "A signal analysis of network traffic anomalies," in *ACM SIGCOMM Internet Measurement Workshop*, 2002.

[35] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure ranked keyword search over encrypted cloud data," *International Conf. on Distributed Comput. Sys.*, vol. 0, pp. 253–262, 2010.

[36] J. Vaidya and C. Clifton, "Privacy - preserving top-k queries," in *ICDE*, april 2005, pp. 545 – 546.

[37] M. Burkhart and X. Dimitropoulos, "Fast privacy-preserving top-k queries using secret sharing," in *ICCCN*, Zurich, Switzerland, Aug. 2010.

[38] K. V. Jónsson, G. Kreitz, and M. Uddin, "Secure multi-party sorting and applications," in *ACNS*, 2011.

[39] G. Avoine, F. Gärtner, R. Guerraoui, and M. Vukolić, "Gracefully degrading fair exchange with security modules," in *EDCC*, 2005.

[40] U. Maheshwari, R. Vingralek, and W. Shapiro, "How to build a trusted database system on untrusted storage," in *OSDI*. Berkeley, CA, USA: USENIX, 2000, pp. 10–10.

[41] D. Levin, J. R. Douceur, J. R. Lorch, and T. Moscibroda, "TrInc: small trusted hardware for large distributed systems," in *NSDI*. Berkeley, CA, USA: USENIX, 2009, pp. 1–14.

[42] A. Dua, N. Bulusu, W.-C. Feng, and W. Hu, "Towards trustworthy participatory sensing," in *USENIX conf. on Hot topics in security*, Berkeley, CA, USA, 2009, pp. 8–8.

[43] S. Saroiu and A. Wolman, "I am a sensor, and I approve this message," in *HotMobile*. New York, NY, USA: ACM, 2010, pp. 37–42.

[44] T. Winkler and B. Rinner, "Securing embedded smart cameras with trusted computing," *EURASIP Journ. Wireless Comm. and Networking*, 2011.

[45] S. Nath, H. Yu, and H. Chan, "Secure outsourced aggregation via one-way chains," in *SIGMOD*. New York, NY, USA: ACM, 2009, pp. 31–44.