Proof Sketches: Verifiable Multi-Party Aggregation

Minos Garofalakis
Joseph M. Hellerstein
Petros Maniatis

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2006-20
http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-20.html

March 1, 2006

Proof Sketches: Verifiable Multi-Party Aggregation

Minos Garofalakis, Joseph M. Hellerstein, Petros Maniatis
Intel Research Berkeley and UC Berkeley

ABSTRACT

Recent work on distributed aggregation has assumed a benign population of participants. In modern distributed systems, it is now necessary to account for adversarial behavior. In this paper we consider the problem of ensuring verifiable yet efficient results to typical aggregation queries in a distributed, multi-party setting. We describe a general framework for the problem, including the threat model for adversaries that we consider. We then present a mechanism called a *proof sketch*, which uses a compact combination of cryptographic signatures and Flajolet-Martin sketches to verify that a query answer is within acceptable error bounds with high probability. When verification fails, we provide efficient mechanisms to identify any participants responsible for the perturbed result. We derive proof sketches for count aggregates, and extend them to proof sketches for verifiable random samples, which, in turn, can be used to provide verifiable approximations for a broad class of data-analysis queries, including quantiles and heavy hitters. In addition to our specific proof sketches developed here, we sketch a general framework for developing new proof sketches. Finally, we examine the practical use of proof sketches, and observe that adversaries can often be reduced to much smaller violations in practice than our worst-case bounds suggest.

1. INTRODUCTION

In recent years, distributed query processing has been a topic of interest in a number of settings, including network and distributed system monitoring, sensor networks, peer-to-peer systems, data integration systems, and web services. Many of these environments depend upon the participation of multiple parties with varying degrees of mutual trust. An outstanding research challenge is to provide trustworthy query results in environments with mutually distrustful or even adversarial parties. Note that even within a single organization like a corporation, viruses and "bot nets" make the presence of adversarial nodes a reality of modern computing that needs to be addressed. The lack of trust in such settings is a significant impediment to the adoption of distributed query technologies.

Trustworthy multiparty query processing is a problem with many facets. In this paper we consider one basic building block: ensuring verifiable but efficient computation of distributed multi-party aggregation queries. We focus particularly on *in-network* aggregation, in which the query processing is pushed down into the network and executed in a distributed fashion by multiple participants. In-network processing is important in settings where centralized data warehousing is undesirable, either due to technical considerations like high data rates (as in network packet monitoring), or to administrative concerns of policy and/or provisioning cost (as in peer-to-peer systems).

The challenge we consider in this paper is to partition the aggregate processing among the participants, without allowing faulty or even malicious parties to undetectably perturb the correct computation and delivery of the result. Recent work has addressed the problem of communication faults in this setting [7, 19, 20, 24], but these techniques remain vulnerable to parties that tamper with the computation. Such malicious misbehavior includes manufacturing spurious data that were not provided by authentic data generators, and suppression of data from the data generators; such activities can perturb aggregate results arbitrarily.

In this paper we propose a family of certificates called *proof sketches* that allow parties in a distributed aggregate computation to prove that the final result could not have been perturbed by more than a small error bound with high probability. A proof sketch can either accompany complete aggregate results as a certificate, or can be used standalone to provide verifiable approximate query results.

We target distributed single-table aggregation queries of the form $\gamma(\sigma_{pred}(R))$, where γ is an aggregate function, σ_{pred} is a selection operator, and R is a set of values. We develop logarithmic-sized proof sketches for a broad class of count aggregates, and prove that they detect tampering beyond a small factor of the true count. We extend that scheme to develop compact proof sketches for verifiably forming random samples, which can themselves be used to give verifiable approximations for a wide variety of data-analysis queries, including, for instance, quantiles and heavy hitters. Both of these proof sketches work by combining the popular Flajolet-Martin (FM) sketch technique [10] with compact signatures we call *authentication manifests*. The authentication manifests ensure that none of the data captured by the sketch was manufactured by adversarial aggregators. To prevent aggregators from silently omitting valid data from counts or samples, we form one additional proof sketch on the *complement* of the query predicate. If the querier knows the aggregate value $\gamma(R)$ for the entire data set, she can check that the combination of the basic proof sketch and the complement proof sketch is close to $\gamma(R)$.

In addition to presenting our FM-based proof sketches, we discuss a number of extensions. We describe techniques to hold aggregators accountable for their computations, and to efficiently identify them when they cheat. We discuss the practical challenge of

maintaining $\gamma(R)$ at the querier, and discuss approaches for relaxing that requirement. We also present general design guidelines for the development of new proof sketches beyond those presented here. Finally, we evaluate our count proof sketch empirically under different adversarial models. Our results show that in practice, adversaries must limit themselves to much smaller-scale tampering to remain undetected.

Organization. In Section 2, we present a practical example scenario that motivated our work, and which we use to explain our ideas. We then present the distributed verifiable aggregation problem in terms of a threat model that includes two basic forms of misbehavior: injection of spurious data in the aggregate, and suppression of authentic data. We introduce proof sketches in Section 3 and we use them to tackle verifiable probabilistic counting; we follow that with a related scheme for verifiable random sampling (Section 4). In Section 5 we address several extensions to the assumptions and techniques we used in the previous sections. Section 6 presents the results of our empirical study of proof sketches. We conclude with related and future work.

2. PROBLEM SETTING

In this section, we introduce a motivating scenario in network security, to place into context the problem of distributed verifiable aggregation. We then describe the aggregation functionality with respect to this scenario, and outline our threat model.

2.1 Motivating Scenario

A simple example scenario in corporate network security was one of the original motivations for our research. It is common practice for most computers owned by a corporation to run a local host intrusion detection (HID) agent such as BlackICE or SNORT. Typically the HIDs communicate over the corporate network with a central management console, located at corporate IT headquarters. HIDs generate events of the sort "I am under a NIMDA attack," or "Host *A* probed my unused TCP port *x*."

The querier in our scenario is a network security engineer working at IT headquarters, who poses distributed aggregate queries on the collection of HIDs, to understand the features of a suspected emerging distributed worm or virus. One typical query might be "how many HIDs identified exploit X?" (a distributed counting query). We refer to this as a *predicate poll* of the HIDs, since it essentially counts the predicate's "yes votes" among the nodes in the system. Another typical query would be "return the OS version of *k* randomly chosen HIDs that identified exploit X" (a distributed sampling query). In both these scenarios, each HID needs to contribute *at most one data record* to the aggregation; this is natural for minimizing bandwidth requirements and latency in these time-sensitive queries[1]. However, one can also imagine queries for which each HID produces multiple matches, e.g., "count all firewall log entries that identify an instance of exploit X."

Data warehousing solutions to this scenario have significant limitations, since there may be thousands of globally distributed HIDs in the corporation, some connected by slow links (WiFi or modem), issuing events at sub-second time intervals. Results to queries may be required within seconds to catch anomalies or other serious problems. Instead of warehousing, *in-network aggregation* may be more appropriate, cutting down on the bandwidth and latency requirements for back-hauling data, by placing more of the computation within the network. In addition, for several of the queries in our example scenario, the querier may only be interested in detecting trends or interesting patterns, and not in answers that are precise to the last decimal. Thus, techniques for fast, *approximate answers* (e.g., approximate predicate polls) are often preferred, especially if they can (a) drastically reduce the burden on the network infrastructure, and (b) provide *approximation-error guarantees*.

Note that while the end systems generating data may belong to one company, the same might not be true of the aggregation infrastructure used. For instance, the aggregation functionality may be hosted at appropriate points on the Internet by a third party – e.g., a well provisioned distributed infrastructure like Akamai – using "edge side includes" [33] or similar technologies. As another example, multiple organizations may want to pool their network resources to compute the aggregates globally. In these cases, the end users posing questions may require extra assurances that the received aggregation results reflect accurately what the trusted data sources (the HIDs) produced.

In the remainder of this paper, we assume that nodes in the system are connected, and that queries from end users can be disseminated to all relevant agents, including data generators and aggregators. The details of how the query dissemination is performed are not the topic of this paper, but are treated extensively in the literature [17, 19, 30, 36]. In some environments, queries can be disseminated along with the HID software (e.g., every HID reports on a predetermined set of predicate polls or other queries periodically). Alternatively, queries can be cryptographicaly signed by the querier to preserve integrity, and disseminated in batch (e.g., "the queries for the next hour are X, Y, and Z") to the relevant agents via a slow but reliable means, such as gossip among HIDs.

2.2 In-Network Aggregation Functionality

In our discussion, we adopt the distributed aggregation terminology of TAG [19] (which does not consider adversarial behavior or approximation). TAG implements simple aggregate functions like SUM, MIN, MAX, or AVERAGE via three functions: an initializer I, a merging function M, and an evaluator function E. In general, M has the form $< z >= M(< x >, < y >)$, where $< x >$ and $< y >$ are multi-valued *partial state records* (PSRs), computed over one or more input data values, and representing the intermediate state over those values that will be required to compute an aggregate. $< z >$ is the PSR resulting from the application of M to $< x >$ and $< y >$. The merging function M is required to be commutative and associative so that the aggregate is well-defined with respect to (unordered) relations. For example, if M is the merging function for AVERAGE, each PSR will consist of a pair of values, SUM and COUNT, and M is specified as follows, given two PSRs $< S_1, C_1 >$ and $< S_2, C_2 >$:

$$M(< S_1, C_1 >, < S_2, C_2 >) = < S_1 + S_2, C_1 + C_2 >$$

The initializer I is needed to specify how to instantiate a PSR for a single input data value; for an AVERAGE over a data value of x, the initializer $I(x)$ returns the tuple $< x, 1 >$. Finally, the evaluator E takes a PSR and computes the actual value of the aggregate. For AVERAGE, the evaluator $E(< S, C >)$ simply returns S/C.

The actual process of in-network aggregation involves multiple entities playing different roles, passing data from one to another. We illustrate the roles in Figure 1 and describe them below:

- The **querier** is the agent that receives the final, fully merged PSR. It runs the verification logic to check the PSR and, if successful, it executes the evaluator function E to generate the final query result (along perhaps with some appropriate quality guarantees).
- **Sensors** are agents that produce raw data values for aggrega-

[1] It is also the common case in sensor query systems that do on-demand data acquisition [6, 19].

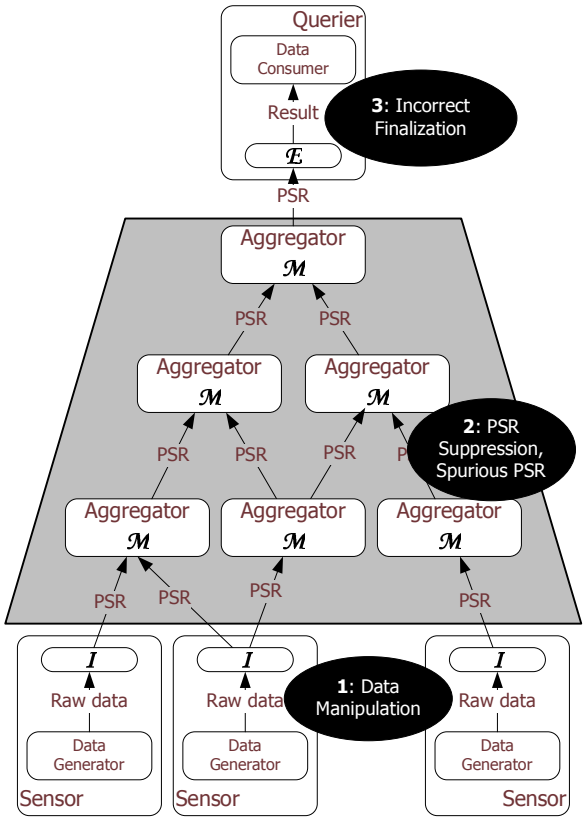

Figure 1: Dataflow among agents in the system, with potential threats by each agent listed in the black ovals. Proof sketches are used to thwart the threats within the gray trapezoid. In each component, we indicate the aggregation functionality (I, M, E) performed.

tion, and invoke initializer functions I to generate PSRs. In our scenario, the HID is a "sensor," and HID measurements or alerts are the data. We require that all the sensors are registered with an organization-wide public key infrastructure (PKI) so that we can authenticate the data they produce.

- **Aggregators** combine multiple PSRs by applying the merging function M. Note that the PSRs handled by an aggregator may distill differing amounts of sensed data from each input; the rather symmetric graph of Figure 1 should not be taken to mean that all aggregation topologies are regular. They can be trees [19], depth-based DAGs [7, 24], random graphs [9] and hybrids of trees and DAGs [20]. In general, we do not make any assumptions about these communication patterns in our work.

2.3 Threat model

We turn now to the threat model we consider, by identifying the tampering opportunities of the dataflow in Figure 1. First, the sensor can suppress data, or insert spurious data into the system. Second, an aggregator can take various actions to perturb the processing of the merging function M; these include suppression of PSRs and introduction of spurious PSRs. Finally, the evaluation function E at the querier may be performed inappropriately, yielding results inconsistent with the input PSRs.

In this paper, we focus on defending vulnerabilities introduced by the aggregators, corresponding to the gray trapezoid in Figure 1. Before discussing those issues, we briefly discuss the other two vulnerabilities. At the bottom of the figure (Vulnerability 1), the actual generation of raw input data by sensors is hard to protect in an application-independent manner. Although heuristics to verify input data may exist in particular environments (e.g., if it is known that a given HID is not powerful enough to register more than 1000 exploits per second), ultimately this task can only be performed by hardware or software attestation of the data generators (see, for instance, work done in the Pioneer project [32]). A wise choice of aggregation function – e.g., a robust statistic like the "trimmed mean" – can mitigate the effects of a small number of spurious input data values [35]. For the purposes of our scenario, we focus on ensuring that the desired aggregates are faithfully computed over whatever raw data the sensors supply to the initialization function.

With regards to evaluation (Vulnerability 3), the querier can still yield the wrong result even if aggregators are correct, by misapplying the evaluator function. Since our scenario assumes that the querier is the entity interested in the result, we do not examine this vulnerability further. If required, the operation of the evaluator can be spot-checked, as with SIA [28].

With regards to aggregation (Vulnerability 2), the primary target of our work, there are two possible attacks. *Partial-state suppression* (or, *deflation*) attacks are mounted by aggregators that omit data from input PSRs during the merging function. The simplest such attack is to suppress an entire PSR, but we will see examples where a complex PSR object (e.g., a set of sampled tuples) may have subparts that are suppressed. *Spurious partial state* (or, *inflation*) attacks introduce data into output PSRs that should not be there. The simplest such attack is to manufacture an entire PSR, but it is also possible to manipulate an existing PSR to reflect manufactured inputs.

In sum, we focus in this paper on threats within the aggregation infrastructure, and present the first results we are aware of for preventing attacks in this context.

3. AM-FM PROOF SKETCHES

To begin our discussion of proof sketches, we consider the special case of "predicate poll" queries discussed in Section 2; we will relax this constraint in Section 5. Our goal is to count the number of nodes that satisfy some boolean predicate. Without loss of generality, let $[U] = \{0, \ldots, U-1\}$ denote the domain of node identifiers, and let *pred* be the specific predicate of interest; also, let C_{pred} ($\leq U$) denote the answer to our predicate poll. First, consider the standard trio of TAG functions for a COUNT aggregate that computes C_{pred}:

$$I(t) = \begin{cases} <1> & \text{if } pred(t) = \text{true} \\ <0> & \text{otherwise} \end{cases} ;$$

$$M(<x>,<y>) = <x+y> \; ; \; E(<x>) = x$$

(where t denotes a sensor's local data record). A rogue aggregator can perturb the count in one of two ways: by inflating the value of a PSR during execution, or by deflating it. We treat these two cases separately; our protection for the inflationary case will serve as the basis for preventing deflation attacks as well.

3.1 Detecting Inflation via Authentication

To start, we need aggregators to prove that they did not inflate the running sum during execution of the merging function. To facilitate exposition, we first present a simple but impractical solution to the problem: aggregating by counting in a unary representation. We modify the PSR for COUNT to be a *bitmap of size* U, with one bit for each of the U sensors being polled. The aggregation logic is modified accordingly ($I(a,t)$ denotes the initializer at node

$a \in [U]$):

$$I(a,t) = \begin{cases} < 2^a > & \text{if } pred(t) = \text{true} \\ < 0 > & \text{otherwise} \end{cases} ;$$

$$M(< p >, < q >) = < p \text{ OR } q > \; ; \; E(< p >) = |p|$$

where $|p|$ is the count of 1-bits in the bitmap p. To prove that the result was not inflated, we require each 1-bit position to be cryptographically signed by its corresponding sensor. We augment the PSR with a set of digital signatures, one per 1-bit; we refer to the set of signatures associated with a bitmap as its *authentication manifest (AM)*. The initializer I at sensor $a \in [U]$ initializes the AM to contain sensor a's signature for bit a, and the merging function unions the AMs in addition to OR-ing the bitmaps. Given a PSR consisting of an AM and a bitmap, the querier can use the sensors' public keys to verify that all the 1-bits in the bitmap are authentically signed, and hence that the count is not too high.

The obvious problem with this technique is that the size of the AM is as large or larger than the collection of U "votes" being counted, and the communication/performance benefits of in-network aggregation are lost. This leaves us with a more focused challenge: how can we form a compact AM for COUNT?

3.1.1 AM-FM: Approximate Inflation Detection

To achieve this, we relax our security requirement: instead of requiring that we detect all inflationary attacks on the count, we can settle for detecting "noticeable" attacks that overcount by more than some small amount. This relaxation suggests the use of space-efficient Flajolet-Martin (FM) sketches for approximately counting the distinct values in a set [10]. By augmenting FM with an authentication manifest, we develop our first proof sketch, which we call AM-FM.

Quick Introduction to FM Sketches. The FM distinct-count estimator [10] is a one-pass (streaming) algorithm that relies on a family of hash functions H for mapping incoming data values from an input domain $[U] = \{0, \ldots, U-1\}$ uniformly and independently over the collection of binary representations of the elements of $[U]$. (The algorithm does not need to know U exactly — an upper bound on the domain size is sufficient.) The basic *FM-sketch* synopsis (for a fixed choice of hash function $h \in H$) is simply a bit vector of size $\Theta(\log U)$[2]. This bit-vector is initialized to all zeros and, for each incoming value i in the input, the bit located at position $\texttt{lsb}(h(i))$ is turned on. It is not difficult to see that, if $h \in H$ and $\texttt{lsb}(s)$ denotes the position of the *least-significant* 1 *bit* in the binary string s, then for any $i \in [U]$, $\texttt{lsb}(h(i)) \in \{0, \ldots, \log U - 1\}$ and $\mathbf{Pr}[\texttt{lsb}(h(i)) = l] = \frac{1}{2^{l+1}}$. To boost accuracy and confidence, the FM algorithm employs averaging over several independent instances (i.e., r independent choices of the mapping hash function $h \in H$ and corresponding FM sketches). A high-level description of the FM estimator is depicted in Figure 2.

Intuitively, due to the randomizing properties of the hash functions in H, we expect a fraction of $\frac{1}{2^{l+1}}$ of the $C \leq U$ distinct values in the stream to map to location l in each sketch; thus, we expect $C/2$ values to map to bit 0, $C/4$ to map to bit 1, and so on. Therefore, the location of the rightmost zero in a bit-vector synopsis is a good indicator of $\log C$. In fact, Flajolet and Martin proved that the estimation procedure depicted in Figure 2 is guaranteed to return an *unbiased* estimate for C (i.e., the expected value of the returned quantity $\hat{C}$ is $E[\hat{C}] = C$).

The analysis of Flajolet and Martin assumes ideal randomizing properties (i.e., full independence) for the hash-function family H [10] that are impossible to guarantee in small space. More recent work [4, 11, 13] has shown that simple variants of the FM-sketch-based estimator can rely on much simpler, limited-independence hash functions (specified through concise, logarithmic-size random seeds). Using only $r = O(\frac{\log(1/\delta)}{\varepsilon^2})$ basic FM sketches, these techniques give (randomized) (ε, δ)-*estimators* for the number of distinct values C; that is, the computed estimate $\hat{C}$ satisfies $\mathbf{Pr}\left[|\hat{C} - C| \leq \varepsilon C\right] \geq 1 - \delta$ [4, 11, 13].

[2] All logarithms in this paper denote base-2 logarithms.

procedure FMDistinctEstimator(S, $\{h_1(), \ldots, h_r()\}$)
Input: Stream S of values in the domain $[U] = \{0, \ldots, U-1\}$,
family of randomizing hash functions h_i ($i = 1, \ldots, r$).
Output: Estimate $\hat{C}$ of the number of distinct values in S.
begin
1. **for** $i := 1$ **to** r **do**
2. $\texttt{fmSketch}_i[] := [0, \ldots, 0]$ // bitvector of size $\Theta(\log U)$
3. **for each** $j \in S$ **do**
4. **for** $i := 1$ **to** r **do** $\texttt{fmSketch}_i[\texttt{lsb}(h_i(j))] := 1$
5. **for** $i := 1$ **to** r **do**
6. **for** $m := \log U - 1$ **downto** 0 **do**
7. **if** $\texttt{fmSketch}_i[m] = 0$ **then** rightmostZero := m
8. sum := sum + rightmostZero
9. **endfor**
10. $\hat{C} := 1.2928 \times 2^{\text{sum}/r}$
11. **return**($\hat{C}$)
end

Figure 2: The Flajolet-Martin (FM) distinct-count estimator.

Adding Verifiability: AM-FM Proof Sketches. Assuming an agreed-upon collection of hash functions $\{h_1(), \ldots, h_r()\}$ for FM-sketch construction, it is not difficult to see that FM-sketch summaries are naturally *composable*: simply OR-ing independently built bitmaps (e.g., at nodes a_1, a_2) for the same hash function gives precisely the sketch of the union of the underlying streams (i.e., $a_1 \cup a_2$). This, of course, makes FM sketches ideally suited for in-network computation schemes since they enjoy the properties of set-union: their merging function is commutative, associative, and *"duplicate-insensitive"* (that is, the sketches can be naturally propagated and possibly unioned many times along different paths in the network without affecting the final result) [19, 24]. From the perspective of verifiability, an attractive aspect of FM sketches is that each bit's value is an independent function of the input domain. Thus, assuming a pre-specified collection of hash functions (known to querier and sensor nodes) for building FM sketches, each 1-bit can be authenticated at the querier by a single signed value from the sensor node that turns it on. Hence, we can construct an authentication manifest for a basic FM sketch with $O(\log U)$ (or fewer) signed inputs of the form $< t, a, s_a(t) >$ (one per 1-bit in the sketch), proving that sensor a provided record t. We refer to the resulting sketch structure as an *AM-FM proof sketch*.

Figure 3 outlines the in-network aggregation functions for our AM-FM sketches; the predicate test in I is omitted for brevity. Note that the FM hash function is applied to the $< dataRecord, sensorID >$ pair to ensure uniqueness across all polled data records (of course, the total number of distinct values is still $\leq U$). A_p denotes the authentication manifest for FM bit-vector p: if p's k-th bit is set to 1, the k-th component of A_p has the form $< k, t, a, s_a(t) >$, and is correct (authentic) iff $k = \texttt{lsb}(h(t,a))$ and $s_a(t)$ is a valid signature on t by data source a. The $\sqcup$ operator forms a subset of the union of its inputs, retaining for each distinct 1-bit in position $0 \leq k \leq \log U - 1$ any one input "exemplar" $< t, a, s_a(t) >$ such that $\texttt{lsb}(h(t,a)) = k$[3]. Finally, the evaluator function executes a

[3] The merging function as defined is not strictly commutative, associative or duplicate insensitive, because of the flexibility in the $\sqcup$ operator's choice of exemplar. One can easily define $\sqcup$ more carefully to achieve these properties via a total order on the exemplar domain, but for our purposes we can

$$I(t,a) = < 2^{\mathtt{lsb}(h_i(t,a))},\ \{< \mathtt{lsb}(h_i(t,a)), t, a, s_a(t) >\} > \quad (1)$$

$$M(< p, A_p >, < q, A_q >) = < p \text{ OR } q,\ A_p \sqcup A_q > \quad (2)$$

$$E(\{< p_i, A_{p_i} > : i = 1, \ldots, r\}) = \mathsf{FMEstimate}(\{p_1, \ldots, p_r\}) \quad (3)$$

Figure 3: Definition of (basic) AM-FM proof sketches as an in-network aggregate.

count-estimation procedure over the collection of FM bit-vectors built (e.g., Steps 5–11 in Figure 2).

3.2 Deflation Detection

The authentication manifest in AM-FM sketches prevents inflating the count by turning 0-bits in the FM sketch into 1-bits. The remaining possible attack is to turn 1-bits into 0-bits, and remove the corresponding signatures from the AM. This attack could deflate the count in the FM sketch.

3.2.1 Redundant Communication

One natural approach to preventing this attack is to redundantly route initialized PSRs along multiple aggregation paths between the sensors and the querier. This was previously suggested for FM sketch aggregation in order to tolerate faulty communication links in sensor networks [7,20,24], and relies on the duplicate insensitivity of FM sketches to ensure correct results in the face of redundant merging. That work assumed a benign aggregator population, and was focused on improving answer quality in the face of an unreliable network fabric. In our setting, we have more stringent requirements: we need to provide strong guarantees that our verification procedure bounds the amount of error an adversary can introduce.

To fully protect against a powerful adversary with as many as $k-1$ compromised aggregators, omissions can only be fully prevented by routing each input PSR redundantly along k node-disjoint paths through the aggregators to the querier. A simple approach is to "flood" PSRs through the aggregator network. Then, the verification logic at the querier can test whether the graph is k-connected; this is a special case of finding the min-cut in a graph, which can be done using, e.g., Goldberg and Tarjan's distributed algorithm [14]. Unfortunately, that algorithm requires $O(n^2 \log n)$ time for a network of n nodes[4] and is not itself verifiable; some extension would be needed to detect adversaries that falsified link information.

Even if we could ensure k-connectivity, the verifiability is built on an assumption about the adversary compromising fewer than k nodes. This raises the bar for adversarial behavior, but does not protect against it. Rather than make assumptions about the adversary, we develop a new scheme that instead relies only on some additional information at the querier.

3.2.2 Complementary Proof Sketches

Our next approach to deflation detection makes no assumptions about either the aggregation topology or the adversary. It instead relies on the querier knowing the total count U of the entire data set (the "universe"). This seems like a very strong assumption, but in our predicate poll scenario it is not at all unreasonable: since each sensor produces only one value ("yes" or "no"), U is equal to the number of sensors in the network. Tracking the arrival and departure of each sensor at a central query site is reasonably tractable, and is in fact the common case in corporate HID scenarios like the one in Section 2.1. For the moment then, we assume that the querier can acquire the correct value for U at any time. We discuss the practicality of this assumption in more general settings in Section 5.

The technique we use accompanies each predicate poll *pred* with its *complementary* poll $\neg$*pred*; the intuitive objective being to check that their counts sum up correctly: $C_{pred} + C_{\neg pred} = U$. Since those counts are only approximately known, the technique uses AM-FM proof sketches to estimate the approximate counts $\hat{C}_{pred}$ and $\hat{C}_{\neg pred}$ and prevent probabilistic inflation of either count. By preventing the complement $\hat{C}_{\neg pred}$ from being inflated, we thereby prevent $\hat{C}_{pred}$ from being *deflated*: if the adversary deflates $\hat{C}_{pred}$, she must inflate $\hat{C}_{\neg pred}$ to avoid being detected by the sum check. We focus on this scheme for the remainder of this section, and provide bounds on the undetectable omission error an adversary can introduce with complementary deflation detection.

3.3 Verification and Analysis

The AM-FM proof sketch allows deterministic detection of spurious 1-bits in the FM sketch: the querier first verifies the authenticity of each element of the authentication manifest via the verification of element signatures by registered sensors, and then ensures that the resulting decoded values can be used to reconstruct the FM sketch. Given that the AM and the sketch match up, the remaining question arises from the use of FM approximations: how much "wiggle room" does the inaccuracy in these approximations give an adversary interested in deflating the count?

To answer this specifically, assume FM-based estimators $\hat{C}_{pred}$ and $\hat{C}_{\neg pred}$ that use $O(\frac{\log(2/\delta)}{\varepsilon^2})$ independent AM-FM sketch instantiations to estimate the Yes/No population counts (e.g., as discussed by Ganguly et al. [11]). Our aggregate verification step checks the condition $\hat{C}_{pred} + \hat{C}_{\neg pred} \geq (1-\varepsilon)U$ and flags an adversarial omission attack if the condition is violated. The following theorem and analysis establish the (probabilistic) error guarantees provided by our verifiable AM-FM aggregation scheme.

THEOREM 1. *Using $O(\frac{\log(2/\delta)}{\varepsilon^2})$ AM-FM sketches to estimate $\hat{C}_{pred}$ (and $\hat{C}_{\neg pred}$), and assuming a successful final verification step, the $\hat{C}_{pred}$ estimate is guaranteed to lie in the range $[C_{pred} - \varepsilon(U + C_{\neg pred}),\ C_{pred}(1+\varepsilon)] \subseteq C_{pred} \pm 2\varepsilon U$, with probability $\geq 1-\delta$. For predicate selectivities $\geq \sigma$, this implies an $(\varepsilon(\frac{2}{\sigma} - 1), \delta)$-estimator for C_{pred}.*

Proof: Consider the final estimation step for C_{pred} at the querier. Note that our AM-FM sketches are naturally composable and duplicate-insensitive summaries. We begin by bounding the error on the proof sketch in the absence of any tampering. If this is the case, the summaries arriving at the finalizer are $O(\frac{\log(2/\delta)}{\varepsilon^2})$ independent FM sketches over the underlying data population. Thus, earlier FM-based estimators (e.g., Ganguly et al. [11]) can be used to give $(\varepsilon, \frac{\delta}{2})$ estimates $\hat{C}_{pred}$ and $\hat{C}_{\neg pred}$ for C_{pred} and $C_{\neg pred}$ (respectively); then, a simple application of the union bound implies that $\hat{C}_{pred} + \hat{C}_{\neg pred} \in (1 \pm \varepsilon)U$ with probability $\geq 1-\delta$. Now, even in the case of malicious tampering, the authentication manifest prevents an adversary from inflating the sketch count. So, if verification succeeds, the aggregate count C_{pred} can be no larger than the upper bound of the FM estimation error, $C_{pred}(1+\varepsilon)$. This demonstrates the upper bound of our range for $\hat{C}_{pred}$.

simply gloss over this issue, treating all valid exemplars as being indistinguishable.

[4]Doing this more cheaply via incremental maintenance is difficult; in the face of a fully dynamic network (with edges coming and going) this is an active area of theoretical research even for the simple cases of $k = 1$ or $k = 2$ (see, e.g., [15, 26]. Constructive distributed protocols for *maintaining* k-connectivity are similarly limited; typical results are for $k = 2$ or $k = 3$ [18].

To prove the lower bound of our range, let $\theta \geq 0$ denote the total (additive) underestimation error in the final $\hat{C}_{pred}$ estimate, including both FM estimation error *and* deflation error introduced by the adversary (through, possibly, several aggregator nodes) during the aggregation process. Thus, our verification step estimates U as $\hat{C}_{pred} + \hat{C}_{\neg pred} = C_{pred} - \theta + \hat{C}_{\neg pred}$. Then, if $\theta > \varepsilon(U + C_{\neg pred})$, with probability $\geq 1 - \delta$,

$$\begin{aligned} \hat{C}_{pred} + \hat{C}_{\neg pred} &< C_{pred} - \varepsilon(U + C_{\neg pred}) + \hat{C}_{\neg pred} \\ &\leq C_{pred} - \varepsilon(U + C_{\neg pred}) + (1+\varepsilon)C_{\neg pred} \\ &= (1-\varepsilon)U \end{aligned}$$

which implies that our sanity check will detect the adversarial omission attack (with high probability). The stated (worst-case) additive bounds for the final verified $\hat{C}_{pred}$ estimate follow immediately. For the selectivity-based relative error bound, if $C_{pred} \geq \sigma U$, then substituting $U - C_{pred}$ for $C_{\neg pred}$ in the inequality for θ above gives $\theta \leq \varepsilon(\frac{2}{\sigma} - 1)C_{pred}$. ∎

In other words, with our verifiable AM-FM aggregation scheme, any adversarial deflation attack can cause our final $\hat{C}_{pred}$ estimate to underestimate the true count by at most $\varepsilon(U + C_{\neg pred}) \leq 2\varepsilon U$, or risk being detected with high probability. Thus, the error guarantees for verifiable AM-FM estimate are in terms of εU factors, which are typically sufficient for predicates that represent significant fractions of U. Specifically, for predicates with selectivity $\geq \sigma$, our algorithms can give $\varepsilon(\frac{2}{\sigma} - 1)$-relative error bounds with probability $\geq 1 - \delta$. It is important to note here that the AM portion in our sketches plays a crucial role in our error bounds, by essentially forcing the adversary to inject only *one-sided* (omission) error — otherwise, the adversary could arbitrarily inflate $\hat{C}_{\neg pred}$ while lowering $\hat{C}_{pred}$ (or, vice versa) thus making arbitrarily large errors essentially undetectable.

4. VERIFIABLE RANDOM SAMPLING: THE AM-SAMPLE PROOF SKETCH

We turn to the slightly more involved problem of constructing a *verifiable random sample* of a given size k over the data tuples residing at the leaf nodes (sensors) in our distributed aggregation topology (Figure 1). As in our motivating scenario of Section 2.1, we assume that there is *one data record per node*. We disseminate the request to form a sample to all the nodes, and wish to ensure – as in aggregation – that the message sent by each aggregator is small, and the result at the top can be verified to be an unbiased random sample of the data. Note that such a sample represents a *general-purpose* summary of the sensor contents that can be employed at the querier site to provide (verifiable) approximate aggregate answers for a variety of different aggregation functions and selection predicates (not known beforehand) over the underlying sensors.

A conventional random-sampling summary is simply a pair $(\{t_1, \ldots, t_k\}, N)$ comprising (a) the subset of sampled records, and (b) the total count of the underlying population (sampling rate $= k/N$). Such a sample can be dynamically generated moving up our aggregation architecture, e.g., using a simple adaptation of reservoir random sampling [34]. Unfortunately, it is not difficult to see that such a scheme falls short of our verifiability goals. Specifically, consider an aggregator node in our architecture (Figure 1) receiving two random samples (s_1, N_1) and (s_2, N_2) from its children. Even though it may be possible to authenticate individual tuples in s_1 and s_2 (e.g., through hash signatures), an adversarial aggregator can still introduce arbitrary bias in the sample. Consider, for instance, the above scenario with $|s_1| = |s_2| = k$ (the target sample size) and $N_1 \gg N_2$, and a malicious aggregator that deterministically outputs $(s_2, N_1 + N_2)$ rather than sub-sampling s_1 and s_2 with the appropriate rates; furthermore, even if $N_1 = N_2$, the adversary could bias the sample towards specific data values (e.g., choose the k smallest sensor readings in $s_1 \cup s_2$) resulting in arbitrarily biased approximate query answers.

The key problem with such conventional sampling schemes in our setting is that they essentially offer no means to verify the sampling procedure run at each aggregator (i.e., the validity of each aggregator's random coin flips). Instead, our proposed *AM-Sample proof sketches* collect a random sample by employing hash functions to map $<$ *dataRecord*, *sensorID* $>$ elements to buckets with exponentially-decreasing probabilities, as in FM estimation. The key difference with simple AM-FM sketches is that we now retain authentication manifests for *all* exemplar elements mapping *above* a certain bucket level l (along with their respective level) — these are exactly the elements in our sample. Our sampling scheme is similar in spirit to Gibbons' distinct-sampling technique [12] for approximating `COUNT DISTINCT` queries over data warehouses; essentially, by forcing elements to be distinct (through the addition of the *sensorID* field), we can use similar ideas to collect a *verifiable random sample* over the underlying data tuples.

Formally, given a uniformly randomizing hash function h over $<$ *dataRecord*, *sensorID* $>$ pairs and a target sample size k, an AM-Sample proof sketch comprises a pair $< L, S >$, where

$$S = \{(l_1, < t_1, a_1, s_{a_1}(t_1) >), \ldots, (l_m, < t_m, a_m, s_{a_m}(t_m) >)\}$$

is a subset of $m = \Theta(k)$ tuple authentication manifests $< t_i, a_i, s_{a_i}(t_i) >$ with corresponding bucket levels $l_i = \mathtt{lsb}(h(t_i, a_i))$. The semantics of an AM-Sample sketch (assuming no tampering) is that it stores exactly the authentication manifests (and bucket levels) for $<$ *dataRecord*, *sensorID* $>$ elements at levels greater than or equal to L (which, of course, implies the invariant $l_i \geq L$ for all $i = 1, \ldots, m$). Since each element maps to a bucket level l with probability $1/2^{l+1}$, it is not difficult to see that each element in the AM-Sample sketch is chosen/sampled with probability $\sum_{i \geq L} \frac{1}{2^{i+1}} = 1/2^L$.

A concise TAG-like description of our in-network aggregation scheme for AM-Sample proof sketches is given in Figure 4. In a nutshell, given a target sample size of k, our algorithm starts by computing the authentication manifests (and bucket levels) for individual sensors (Equation (4)). These manifests are then unioned up the aggregation topology, by appropriately sub-sampling elements at higher sampling rates (using the maximum level $\max\{L_1, L_2\}$ to build the output sample); furthermore, to keep the sketch size under control, our aggregation scheme drops the sampling rate by a factor of 2 (setting $L = \max\{L_1, L_2\} + 1$) when the sample size grows beyond $2k(1+\varepsilon)$ (Equation (5)), where $\varepsilon < 1$ denotes an error parameter determined by the target sample size. As we show, for large enough k, this ensures that the size of our AM-Sample sketch never grows beyond a range $[(1-\varepsilon)k, (1+\varepsilon)2k]$ (with high probability) as the summary is propagated towards the querier node.[5]

4.1 Sample Verification and Analysis

Our AM-Sample proof sketches combat adversarial inflation of the collected random sample in two ways. First, through the use of authentication manifests for data tuples, the sketch prevents the adversary from inventing new data, since all tuples come signed by a trusted data source. Second, our technique also prevents the adversary from migrating tuples across bucket levels of the sketch (i.e., biasing random-sampling choices), since the level is contin-

[5] To simplify the exposition, our discussion here assumes full independence for the hash function $h()$, similar to Gibbons [12] — our techniques and results can be extended to the case of limited-independence hash functions using known methods [11, 13].

$$I(t,a) = <0,\ \{(\texttt{lsb}(h(t,a)), <t, a, s_a(t)>\}> \quad (4)$$

$$M(<L_1,S_1>,<L_2,S_2>) = <L,\ S(L,S_1,S_2)> \quad (5)$$

where $S(L,S_1,S_2) = \{(l,<t,a,s_a(t)>) \in S_1 \cup S_2 :\ l \geq L\}$, and

$$L = \begin{cases} \max\{L_1,L_2\} & \text{if } |S(\max\{L_1,L_2\},S_1,S_2)| \leq (1+\varepsilon)2k \\ \max\{L_1,L_2\}+1 & \text{otherwise} \end{cases}$$

$$E(<L,S>) = \{t : (l,<t,a,s_a(t)>) \in S\}, \text{sampling rate} = 2^{-L} \quad (6)$$

Figure 4: Definition of AM-Sample proof sketches as an in-network aggregate.

gent on the signed content of the tuple (through the hash function computation), which cannot be forged. In a sense, by using a hash function $h()$ for sampling, our AM-Sample sketches can verify the coin flips of intermediate aggregators and ensure that no malicious inflation or biasing of the sample has occurred through inappropriate element selections.

As a side-note, observe that our technique – unlike the reservoir sampling scheme alluded to above – is also naturally duplicate-insensitive. Hence it is possible to use this scheme with an aggregation topology that merges PSRs redundantly.

Of course, as with AM-FM sketches, it is possible for the adversary to deflate the random sample (and the corresponding sampling-based approximate answers at the querier) by removing element signatures from the AM-Sample. This can be done by an aggregator node either artificially increasing the bucket level (to discard all lower-level elements), or by removing specific elements at the current bucket level. We now quantify the verifiable deflation-error guarantees provided by our AM-Sample proof sketches. We start by asserting that, given a large enough target sample size, and assuming no malicious tampering, the final sample size at the querier must be within a small factor of k; this, in turn, implies that the adversary cannot hope to deflate the sample by a large factor without being detected (with high probability).

THEOREM 2. *For a target sample size of at least $k = O(\frac{\log(2/\delta)}{\varepsilon^2})$, and assuming no malicious aggregator deflations, the final sample size $|S|$ at the querier is at least $(1-\varepsilon)k$ with probability $\geq 1-\delta$.*

Proof: Let l denote the (unique) bucket level such that $k < U/2^l \leq 2k$, and let X denote the (random) number of elements mapped to buckets at levels $\geq l$ through our $\texttt{lsb}(h())$ hash mappings. Since we hash a total of U distinct elements, by Chernoff bounds [23]

$$\begin{aligned} \mathbf{Pr}[X \in [(1-\varepsilon)k,(1+\varepsilon)2k]] &< \mathbf{Pr}\left[X \in (1\pm\varepsilon)\frac{U}{2^l}\right] \\ &< 2e^{-\varepsilon^2 U/2^{l+1}} < 2e^{-\varepsilon^2 k/2} \leq \delta, \end{aligned}$$

since $k = O(\frac{\log(2/\delta)}{\varepsilon^2})$. Based on our aggregation scheme for AM-Sample sketches (Figure 4), and assuming no malicious tampering, all elements mapping to levels $\geq l$ should "survive" all the way to the querier node; thus, the final sample size $|S|$ satisfies $|S| \geq X \geq (1-\varepsilon)k$ with probability $\geq 1-\delta$. ∎

We now consider the error guarantees provided by a collected AM-Sample proof sketch for predicate-poll queries at the querier node. Let σ denote (a lower bound on) the selectivity of predicate polls run over the final sample. Given a target sample size of (at least) $k = O(\frac{\log(6/\delta)}{\sigma(1-\varepsilon)\varepsilon^2})$ and the total number of sensors U, the querier can limit the potential impact of adversarial deflation on the collected AM-Sample $<L,S>$ by performing two simple verification steps: (1) checking that the final sample size $|S|$ satisfies the condition $|S| \geq (1-\varepsilon)k$ (Theorem 2); and, (2) ensuring that $2^L \cdot |S| \geq (1-\varepsilon\sqrt{\sigma})U$. If either of the above conditions is violated, the querier flags an adversarial deflation attack with high probability ($\geq 1-\delta$). The following theorem establishes the verifiable deflation-error guarantees for poll queries over our AM-Sample proof sketches.

THEOREM 3. *Assume an AM-Sample proof sketch synopsis collected with a target sample size of $k = O(\frac{\log(6/\delta)}{\sigma(1-\varepsilon)\varepsilon^2})$, and that both final verification steps at the querier are successful. Then, for the cardinality C_{pred} of any given predicate poll pred with selectivity $\geq \sigma$ over the nodes, the estimate $\hat{C}_{pred}$ obtained from the AM-Sample is guaranteed to lie in the range $[C_{pred}(1-\varepsilon(\frac{2}{\sqrt{\sigma}}+1)), C_{pred}(1+\varepsilon)] \subseteq C_{pred}(1\pm\varepsilon(\frac{2}{\sqrt{\sigma}}+1))$ with probability $\geq 1-\delta$ (i.e., to give an $(\varepsilon(\frac{2}{\sqrt{\sigma}}+1),\delta)$-estimator for C_{pred}).*

Proof: A simple application of Theorem 2 shows that, assuming no malicious deflations, the final sample size $|S|$ at the querier node is $|S| \geq (1-\varepsilon)k = O(\frac{\log(6/\delta)}{\sigma\varepsilon^2})$ with probability $\geq 1-\frac{\delta}{3}$. Given such a (non-deflated) AM-Sample sketch $<L,S>$ with $|S| \geq O(\frac{\log(6/\delta)}{\sigma\varepsilon^2})$, it is not difficult to see that $\hat{U} = 2^L|S|$ is an $(\varepsilon\sqrt{\sigma},\frac{\delta}{3})$-estimate for U (see, for instance, Gibbons [12]); this implies that, $\hat{U} = 2^L|S| \geq (1-\varepsilon\sqrt{\sigma})U$ with probability $\geq 1-\frac{\delta}{3}$. Thus, if either of our two verification checks fails, then we can declare that, with high probability ($>1-\delta$), one (or more) adversaries in our aggregator topology have deflated our collected AM-Sample.

Now, assuming that both verification steps are successful, the querier estimates the predicate-poll cardinality C_{pred} as $\hat{C}_{pred} = 2^L|S_{pred}|$, where S_{pred} is the subset of elements in our AM-Sample that satisfy the *pred* predicate. Given a sample size $\geq O(\frac{\log(6/\delta)}{\sigma\varepsilon^2})$ where $\sigma \leq$ selectivity(*pred*), and assuming no deflation bias from adversarial aggregator(s), we have $\hat{C}_{pred} \in (1\pm\varepsilon)C_{pred}$ with probability $\geq 1-\frac{\delta}{3}$. But, from our bounds on the $\hat{U}$ estimate (verification step (2)), we know that any adversarial omissions in our aggregator topology cannot deflate the sample by more than $2\varepsilon\sqrt{\sigma}U/2^L$ elements (in the worst case) without being detected at the querier (with high probability). Of course, in the worst case for our $\hat{C}_{pred}$ estimate, all the omitted elements will come from S_{pred}. Thus, summing up the failure probabilities for our two verification steps and good estimation of C_{pred} through the union bound, we have that, with probability $\geq 1-\delta$

$$\hat{C}_{pred} \in [C_{pred}(1-\varepsilon) - 2\varepsilon\sqrt{\sigma}U, (1+\varepsilon)C_{pred}] \subseteq C_{pred}(1\pm\varepsilon(\frac{2}{\sqrt{\sigma}}+1))$$

(since $C_{pred} \geq \sigma U$). This completes the proof. ∎

4.2 Leveraging a Verifiable Random Sample

Based on the bounds in Theorems 1 and 3, it is easy to see that, for a *given* predicate *pred*, AM-FM sketches offer a better space/accuracy tradeoff than AM-Sample sketches (which, of course, implies less communication for a given error guarantee for the $\hat{C}_{pred}$ estimate). On the other hand, the final AM-Sample is a *general-purpose* summary of the data content in the sensor population, that can be leveraged to provide approximate answers for different classes of data-analysis queries at the querier node.

For instance, a (verified) AM-Sample (of appropriate size) can be used in a straightforward manner to construct *approximate quantile summaries* [22] over different attributes in the sampled tuples, or to

discover *heavy-hitters/frequent items* [8, 21] in the attribute-value space. A key property of such data-analysis queries is that their error requirements are typically expressed in terms of $\pm\varepsilon U$ factors, since they look for either value ranges (quantile intervals) or individual values (heavy-hitters) that represent a significant fraction of the overall population U. This implies that our deflation error bounds for AM-Sample sketches (Theorem 3) can be naturally translated into strong, verifiable error guarantees for approximate quantiles and heavy-hitters at the querier node. We defer the detailed development to the full paper.

In addition, note that, by Theorem 2, our techniques guarantee that the final collection of AM-Sample data records at the querier is a random sample of the underlying sensor population from which at most a small fraction of elements have been dropped (possibly in an adversarial manner). Besides predicate poll queries, such a guaranteed "low-omission" sample can have other potential uses, e.g., for estimating different data-value statistics or for approximate data visualization.

5. EXTENSIONS

Our discussion up to this point focused on verification in the simple scenario of Section 2.1. In this section, we take a broader view. First, we consider the universal knowledge requirements for complementary deflation detection, and relax those requirements. Then, we move beyond verification and discuss techniques for pinpointing faulty aggregators. Finally, we provide a basic template for proof sketches that can be used as a guideline for new settings.

5.1 On Universal Knowledge

Our complementary deflation protection scheme requires additional knowledge to be maintained somehow at the querier. For example, to protect AM-FM sketches for predicate polls, our protection requires that the querier know U, the total size of the "universe" of votes (Section 3.2.2). In this section, we argue that knowing the size of the universe is reasonable in many practical settings. We also explore scenarios in which the requirement may be relaxed while retaining the benefits of complementary deflation protection.

Consider first the running scenario, in which there is a single tuple per sensor. The problem of knowing the size of the universe breaks down into two parts: the size of the *possible* universe and the subset of that currently *operational*. The former can be trivially obtained from the sensor registry, which holds, among other things, the sensor public keys required for verifying signatures. The latter amounts to tracking the status ("up" or connected, vs. "down" or inoperative) of each sensor. Thankfully, tracking the status of this population accurately is already a high-priority task in our scenario. HIDs run on a company's firewalls and servers, which are closely managed, and highly available devices; and they run on company assets such as employee laptops, which are typically tracked via a Virtual Private Network (VPN) gateway connection. As such, devices operating sensors in our scheme have either well understood availability properties (e.g., number of "nines" or half-life period), or are closely – yet incidentally – tracked for connectivity purposes. In any case, a querier can leverage this facility of the trusted infrastructure to obtain an accurate live-sensor count in a tractable fashion, which it can then use to safeguard the query results that cannot be tractably obtained via the same infrastructure.

Given a generally knowable live sensor population size, going from single-tuple-per-sensor to multiple-tuples-per-sensor is an easy step, as long as the universe of tuples has a well known mapping to the universe of sensors. For example, predicate polls involving exactly the last 5 measurements from each sensor are trivial. The situation is slightly more complicated when there is an arbitrary, unpredictable number of tuples per sensor. A rewrite of the predicate in question to one that falls back on the single-tuple-per-sensor setting may be the easiest solution. For example, instead of the query "count the attack log entries across all sensors" one might ask instead "count all sensors registering more than 5 attacks."

When rewrites are not an option, or there is no clear mapping from the tuple universe to the more knowable sensor population, or when the sensor population size itself is hard to know (e.g., for less well managed sensors), approximate knowledge of the tuple universe size may be a reasonable solution. For instance, the querier can maintain deterministic bounds on the count using the deterministic approximate-counting schemes provided by Olston and Widom [25]. Since we require this task itself to be verifiable, it may have to run over the sensor population itself, or in a centralized but infrequent fashion. Given the deterministic bounds of such techniques (e.g., $U - \varepsilon_U^- \leq \hat{U} \leq U + \varepsilon_U^+$), our bounds of Sections 3.3 and 4.1 are modified by simply replacing U by $U - \varepsilon_U^-$ in the lower bounds, and $U + \varepsilon_U^+$ in the upper bounds. Trusted *probabilistic* $(\varepsilon_U, \delta_U)$-estimates for U at the querier (e.g., obtained through simple FM counting over trusted nodes) similarly introduce $\pm\varepsilon_U U$ additive error in the bounds of Theorems 1–3, but also increase the failure probability to $\delta + \delta_U$ by the union bound (this is not a serious issue, since all our proof sketch space bounds have weak logarithmic dependence on $1/\delta$).

In practice, the ability to provide a useful degree of verifiability will depend on a number of variables, including the communication budget for maintaining $\hat{U}$, the rate of change at the data sources, and the application-specific tolerances for εU and δ. A thorough study of all these variables is beyond the scope of this paper; we focus on the predicate poll case in our experiments of Section 6.

5.2 Verification Failure and Accountability

Our verification tests for AM-FM sketches raise alarms when either the authentication manifest does not match the sketch, or the complementary deflation detection check fails. Taken alone, such alarms are useful in flagging a result as untrustworthy. However, it may be desirable to go a step further, to trace misbehavior where it happens – this is useful not only for identifying malicious aggregators, but also for identifying false alarms. We consider some alternatives below.

Pinpointing inflation attacks is relatively simple. When the authentication manifest does not match the sketch, we know that one or more aggregators incorrectly performed the merge logic, e.g., turned on an FM bit without a corresponding signed data value. To trace an unsupported FM bit, one can interactively follow the aggregate computation backwards through the topology to the aggregator that introduced the unsupported bit. Unfortunately, malicious aggregators are likely to shed responsibility for their incorrect output, for instance, by blaming it on one of their topology children. To combat such misbehavior, we can require that aggregators sign their messages to each other, each verifying the signatures on its inputs (as correct, and coming from a fellow aggregator) before computing and transmitting its output PSR. If each aggregator retains the signed inputs to its local computation for a short while, it can conclusively respond to traceback requests when asked by a querier. To trace back a faulty bit in an authentication manifest, a querier checks which input the faulty bit came from and then proceeds to the aggregator that issued that input PSR, asking it for its inputs. The process will terminate either when the querier reaches a faulty sensor (which we have assumed correct), an aggregator who received correct inputs but produced an incorrect output signifying misbehavior, or an aggregator that fails to respond. In any of these cases, assuming a tree-like topology, a misbehavior can be detected

and dealt with, in $O(\log n)$ steps through the n aggregators.

When the correctness condition fails, on the other hand, pointing to the possibility of a deflation attack, the situation is slightly trickier. The reason is that, unlike an invalid authentication manifest, a deflated PSR is not directly recognizable on its own; additional information is required to identify it as deflated. One approach to tracing back PSR deflations is to require more information about the aggregation topology; specifically, extending the requirement for the size U of the universe, we can require that for aggregator i the size U_i of the tuple universe *covered* by that aggregator be known. For example, in a tree topology, for aggregator i, U_i would be the number of leaves in the subtree rooted at i. Given these "sub-universe" sizes, the deflation criterion of Theorem 1 can be applied to every aggregator individually. Consequently, the querier may trace back the deflation by starting at the final aggregation node, following it to its immediate children, applying the deflation criterion, and following recursively that child aggregator (or aggregators) on which the criterion fails, while verifying the correctness of the merging function at every step (i.e., that no 1-bits were suppressed by the aggregator). As with inflation trace back, deflation trace back in this scenario can directly walk tree-like topologies to malicious aggregators in roughly $O(\log n)$ steps. Note, however, that there is a low probability that the deflation criterion may fail even in the absence of adversarial tampering; in such *false positive* cases, the trace back will follow failing deflation criteria down to aggregators that do not fail the criterion or to the input sensors.

Enforcing accountability on individual aggregators can be helpful, but may be best reserved as an option for important queries (e.g., those on which more complex views rely). Alternatives to tracing back misbehavior that burden the querier less may be boosting the query accuracy (by requiring more sketches) or boosting the survivability of data (by submitting each sensor PSR to multiple aggregators).

5.3 A Generalized Template for Proof Sketches

While both of the proof sketches we develop in this paper are based on FM sketches, this is not a requirement of the approach, and we expect that a variety of proof sketches could be constructed beyond our initial ideas here. In general, the basic guidelines for constructing a proof sketch for an aggregation function γ are as follows:

Compact Manifest: A key challenge in designing a proof sketch is to develop a compact authentication manifest. Compact manifests are trivial to construct for sketches like FM where each bit is independent of the others, and can be authenticated by a single exemplar value. Otherwise, one must reason about the m-to-n relationship between input values and sketch bits: a subset (or possibly multiple alternative subsets) of the input values may constitute a "support" for an output bit, and each input value may influence multiple output bits. In general, this leads to a combinatorial optimization problem of choosing (from the input data) a minimal manifest for a given output sketch. It would seem desirable to choose sketches that not only have compact manifests, but avoid combinatorial complexity in evaluating them.

Deflation Bounds: For aggregate γ, our complementary deflation protection requires that the querier maintain $\gamma(U)$, and that one can derive tight bounds on the probability that $\gamma(U) - (\gamma(\hat{C}_{pred}) + \gamma(\hat{C}_{\neg pred}))$ exceeds some threshold. Note that, in some cases, it is sufficient to bound the approximation using a simpler aggregate than the one being computed; for example, the deflation detection bounds for our AM-Sample sketch depend on testing the count. As of yet, we have no general rule for identifying such scenarios or characterizing the "minimality" of the aggregate used for complementary deflation detection. This is an open question of both theoretical and practical interest.

Clearly these are neither formal characterizations of the requirements for "proof-sketchability," nor are they turnkey guidelines for developing new proof sketches. However, we believe they are of use in developing new verifiability techniques.

As an example of using these criteria in a fairly different setting, we briefly consider Bloom Filters [5]; for brevity we assume familiarity with their general construction and use. A Bloom Filter can be formed via in-network aggregation in a straightforward way. Like FM sketches, the bits in a Bloom Filter are independent functions of the input domain. Hence a simple authentication manifest for a Bloom Filter can maintain one exemplar per bit. Actually, one can perhaps do better than this, since each input value maps to multiple bits of the Bloom Filter. Minimizing the Bloom Filter manifest size is an instance of the combinatorics we warn against above, but is perhaps an unnecessary optimization of the simple technique. With respect to deflation bounds, there is a scenario with an appropriate analogy for set-membership tests. Assume that the data set in the network contains tuples of the form (*id*, *value*). We wish to form a Bloom filter for the set of *id*s of tuples that satisfy a selection predicate $\sigma_{pred(value)}$ (e.g., so that we can perform a semi-join with another table at the querier). In this case we can prevent deflation attacks by also requiring a Bloom Filter to be formed on $\sigma_{\neg pred(value)}$. Assuming the querier knows the universe of *id*s currently stored in the network, each membership test it performs should succeed on at least one of the two Bloom Filters (again, within the error guarantees of the filter(s)); if not, a deflation attack occurred.

6. EXPERIMENTAL EVALUATION

In this section, we experimentally evaluate AM-FM to understand its behavior during an attack as well as in the absence of an attack. Furthermore, we explore some practical adversarial suppression strategies to demonstrate that, in the average case, the adversary has a low probability of getting away with deflating a predicate count significantly.

6.1 Adversary Strategies

Though our worst-case analysis (Section 3.2) provides probabilistic guarantees about the effects of adversarial behavior during aggregation, in practice not many adversaries will be able to achieve worst-case tampering, for at least two reasons. First, unless the adversary succeeds in compromising the "root" aggregator in the network, she will not have access to the last PSR given the querier. Instead she will be able to affect only PSRs further away from the querier in the aggregation topology. This means that she has only partial information on which to determine her suppression strategy; for example, some sketch bits that she suppresses will be re-set by other, well behaved aggregators. Second, the adversary does not control the hash functions used by the estimators. Consequently, she cannot always achieve worst-case suppression undetected if, for instance, $\hat{C}_{\neg pred}$ is underestimated by FM in a particular query – this tightens the width of the range from Theorem 1.

We examine below adversary strategies that approximate different adversarial goals:

Targeted Strategy Given target count $C_{malicious}$, suppress as many sketch bits as will bring $\hat{C}_{pred}$ close to that target count.

Safe Strategy Suppress as many sketch bits as will deflate $\hat{C}_{pred}$ without violating the verification condition (see Theorem 1).

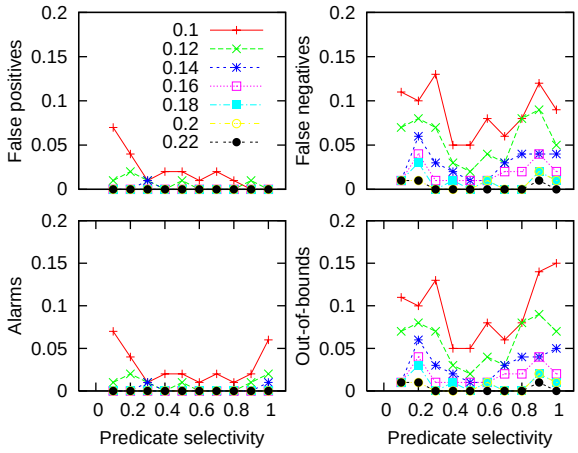

Figure 5: Frequencies of alarms, out-of-bounds estimates, alarms for in-bounds estimates (false positives) and undetected out-of-bounds estimates (false negatives) for different selectivities (x axis) and different ε parameters (one per curve).

With each strategy we vary a *coverage* parameter $0 < G < 1$ that reflects the fraction of the universe aggregated via malicious aggregators. For instance, in a tree aggregation topology, an adversarial aggregator *covers* all data values ingested into the topology via the subtree of which it is the root. Clearly, the greater the coverage, the more likely it is that sketch bits suppressed by the adversary will not be set again by another aggregator.

6.2 Results

The results presented below use 128 sketches — qualitatively similar numbers were obtained for other summary sizes. Based on the worst-case bound of $O(\frac{\log(1/\delta)}{\varepsilon^2})$ for our FM estimator, this implies that a worst-case error guarantee of $\varepsilon = 0.15$ can be achieved with a probability of about 73% (ignoring constants in FM space bound). For each datapoint, 50 independent runs were computed. We fix the size of the universe to $100,000$ and vary the cardinality of the polled predicate between $10,000$ and $100,000$ at $10,000$ increments. We vary adversary coverage between $1/64$-th of the universe to 100%.

We raise an alarm when the verification condition fails (for given ε parameter), and we claim a measurement/estimate $\hat{C}_{pred}$ *"in-bounds"* when it satisfies the error bounds of Theorem 1. Alarms for which the $\hat{C}_{pred}$ estimate was actually within the error bounds are conservatively termed *"false positives"* below, whereas estimates that are not "in-bounds" but fail to raise an alarm are termed *"false negatives"*.

6.2.1 Benign Behavior

We begin with the behavior of AM-FM in the absence of adversaries. We wish to understand how frequently the verification condition triggers and when it does, whether it is justified by the occasional outlying estimate. Figure 5 plots the frequency of alarms and out-of-bounds estimates, as well as the frequency of false positives (alarms raised when the estimate is within bounds) and false negatives (no alarm raised when the estimate is out-of-bounds). Given the number of sketches (128), the out-of-bounds occurrences are below 10% for $\varepsilon \geq 0.12$, which is in practice much better than the worst-case error and confidence probability described above for 128 sketches. False negatives are also well below 10% for all ε parameters but the smallest value (0.1). Therefore, our implemented estimators perform well within the worst-case bounds of our method. Note that even without the adversary around, the verification condition does catch some outlier estimates, especially for high count cardinalities. Those appear as alarms that are not false positives.

6.2.2 Targeted Strategy

The targeted strategy approximates an adversary who cares about a particular count suppression at any cost, even if she is detected. She can suppress the sketch bits at her disposal to achieve the target estimated count $C_{malicious} < \hat{C}_{pred}$. To that effect, she suppresses as many sketch bits as will cause the estimate (as she can compute it locally) to reach $C_{malicious}$. When she has limited coverage, her attempts are thwarted as the remaining, non-malicious aggregators merge their PSRs with those she has concocted.

Figure 6 explores this strategy for predicate polls of cardinalities 0.2, 0.5, and 0.7. Estimates of the selectivity are affected much more dramatically when the adversary has high coverage ($1/4$-th or more) and otherwise remain fairly close to the actual selectivity. However, the dramatic suppressions incurred by the high coverage adversary necessarily trigger alarms at the querier. When the querier applies a tight bound on the verification condition ($\varepsilon = 0.15$ in the second column), alarm frequency is greater as the adversary strives for larger estimate deflation. Lower cardinality predicates (higher rows) suffer less from those alarms, since there's less wiggle room offered by our εU verification condition.

In terms of false positives, the two rightmost columns feature a sharp drop in the high coverage curves; the x-axis point of the sharp drop off is exactly the lower-bound of Theorem 1. For instance, in the bottom right plot, the full coverage false positives curve (magenta) drops sharply at target selectivity $C_{pred} - \varepsilon(U + C_{\neg pred}) = 0.7 - 0.25(1 + 0.3) = 0.375$. The top row (predicate of selectivity 0.2) does not feature this sharp drop off because the lower bound of our theorem falls below 0.

Alarm frequencies are lower for lower coverage, since lower coverage results in much less dramatic estimate deflation (compare the $1/4$-th coverage blue curve to the full coverage magenta curve). Note however that the $1/4$-th coverage curve does not feature any sharp drops. This is an artefact's of our conservative definition of "false positives." Since the adversary (with lower coverage) cannot suppress the estimate significantly, she can suppress it enough to raise an alarm (for the 0.5 and 0.7 selectivities) but not enough to violate a tight error bound (for $\varepsilon = 0.15$), thereby registering as a false positive. For a more appropriate ε given our number of sketches, this is not the case.

6.2.3 Safe Strategy

The safe strategy represents an adversary with varying levels of coverage. This strategy's primary goal is *not to raise an alarm* while deflating the count as much as possible. To do so, the adversary suppresses sketch bits that do not violate the verification condition (evaluated as best as possible given the adversary's coverage). We conservatively assume for this strategy that the adversary knows her own coverage exactly, as well as the size of the universe U.

Figure 7 plots the average deflation bias introduced by this adversary, under two ε parameters, one very conservative ($\varepsilon = 0.1$) given the number of sketches, and one more permissive that assumes a relaxed querier ($\varepsilon = 0.25$). (The $X = Y$ line represents the perfect (i.e., zero-error) estimate.) At the timid setting, the adversary does not succeed in biasing the estimate by more than εU, except under very high coverage, a quarter of the universe and above. At the

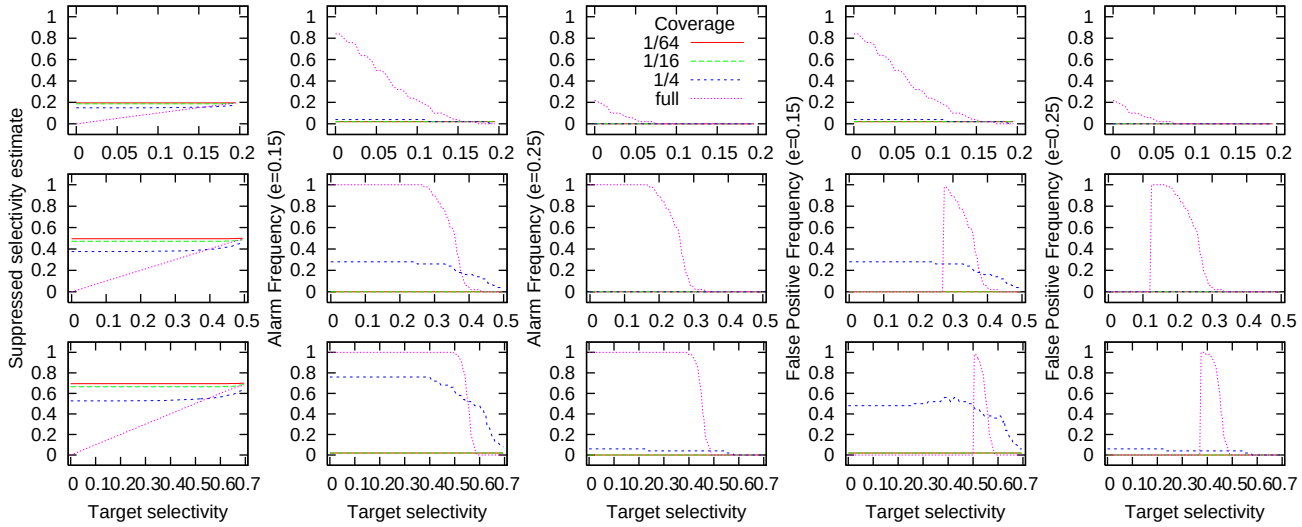

Figure 6: The effects of the targeted strategy on three predicate polls of cardinalities 0.2 (top), 0.5 (middle), and 0.7 (bottom). All x axes are the target selectivity of the adversary, from the predicate cardinality down to 0. On the left, we show the average suppressed count estimate. The next two columns show the alarm frequency (with an aggressive $\varepsilon = 0.15$) and with a more conservative $\varepsilon = 0.25$. The final two columns show the false positive frequency with the same two ε values. In each graph, we show different curves for four levels of adversarial coverage.

more relaxed ε setting, the adversary hovers around εU for full universe coverage, but still remains fairly close to the correct count for lower coverage values.

Note that, especially when the aggregation topology is beyond the adversary's control, the ability of the adversary to place herself at high coverage positions can be kept low. As a result, the *expected* deflation bias in this strategy is strongly weighted towards the low coverage values. For instance, in a binary tree topology, half of the aggregators cover only their own PSRs and the fraction of the value universe corresponding to a single aggregator.

6.3 Discussion

At a high level, our experimental study demonstrates that our techniques are quite robust: to get near our worst-case bounds undetected, an adversary needs both to compromise aggregators near the root of the topology, and to get even luckier than our analysis might suggest. The former issue can be mitigated by design; for instance, by implementing multiple redundant aggregation trees for the same query. Furthermore, to remain undetected, the adversary need limit herself to much lower deflations than those tolerated in the worst-case by our result. At the same time, our implementation raises interesting issues with respect to fine-tuning our verifiable sketching mechanisms in a real-life setting — we are currently exploring different techniques in that context.

7. RELATED WORK

Our work here was inspired by efforts in the SIA project for verifiable (centralized) aggregation in the context of sensor networks [28]. In SIA, a single untrusted aggregator acts as an intermediary between (largely) trusted battery-operated sensors and a querier. To verify aggregation results, the querier can spot-check the requisite computations, by requiring samples of input data that justify particular results, while ensuring that the sampling mechanism remains untampered. We target a broader problem than SIA, by involving multiple aggregators, making the computations we seek to verify more complex. However, our work is intended for a traditional computing environment, rather than the power-constrained battery-operated sensors in SIA. Hence we can take advantage of significant computational power at all the agents, including the sensors.

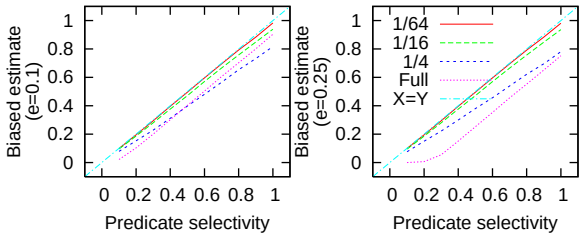

Figure 7: Average deflated estimate for an adversary of varying coverage (different curves) using the safe interior strategy. On the left, the strategy uses a timid $\varepsilon = 0.1$, while on the right the strategy uses a more permissive $\varepsilon = 0.25$.

Our other source of inspiration is the recent work on using duplicate-insensitive FM-based sketches and redundant communication to tolerate failures in sensor network aggregation [7, 20, 24]. Unfortunately, as described in Section 3.2.1, it is unclear whether such techniques can be made robust to adversarial suppression attacks.

Trustworthy multi-party aggregation is a problem with many facets, ranging from robustness to external attack, to malicious data sources, and to privacy leaks. We sample all three categories. Sanli et al. [31] describe machinery for secure key distribution and aggregation-aware encryption in sensor networks. They set encryption parameters so as to provide greater security (higher-quality encryption) the closer an aggregator lies to the root of the aggregation topology and, therefore, the greater fraction of the aggregated sensed values that aggregator summarizes. Unlike proof sketches, this work does not consider the problem of aggregators acting maliciously.

Wagner discusses the idea of "resilient" aggregation in sensor networks [35], but his architectural model does not include in-network

aggregation. Instead, he ensures that an adversarial sensor node maliciously generating false data cannot perturb the aggregate by more than a small amount. In Wagner's threat model, the count aggregate is "resilient," so he offers no specific defenses for it. He proposes the use of *robust statistics* [16] (e.g., the trimmed mean rather than the true mean) to bound this impact. Our verifiable sampling technique here can be used in conjunction with robust statistics like the trimmed mean, which are usually designed explicitly to work over samples. More generally, work on evaluating the trustworthiness of data sources for query processing focuses on techniques like watermarking [1] and reputation systems [27], which again can in principle be carried through samples. Deeper integration of any of these technologies with verifiable aggregates other than sampling is an interesting open problem.

Moving further afield, we distinguish our efforts here from the growing body of work on preserving the privacy of information in databases. This literature includes query processing over distributed data, including aggregation queries (e.g., Agrawal et al. [2] and the references therein). However, we are not aware of work on privacy preservation for in-network aggregation. The combination of verifiability and privacy preservation for distributed aggregation appears to be a rich challenge for future work.

8. CONCLUSIONS AND FUTURE WORK

The work we present here on proof sketches represents an effort to marry two historically disjoint technologies: cryptographic authentication and approximate query processing. While this sounds complex, our FM-based proof sketches provide a remarkably simple defense against the introduction of spurious data during aggregation. Our complement technique for detecting suppression attacks is also simple, but less generally applicable: it requires the querier to track an aggregate function on the entire distributed data set. While this is quite realistic in a number of important practical settings, there are scenarios for which alternative suppression defenses would be welcome. We believe this is an important area for future research.

We are optimistic about a number of potential extensions to this work. One direction is to combine our techniques with various ideas in privacy-preserving query processing, in the hopes of developing a more multifaceted notion of trustworthy querying than has been considered to date. Another interesting challenge is to move beyond aggregation queries to provide verifiable "enumerative," non-aggregated queries – perhaps aiming to verify approximate [3] or "partial" [29] results. Finally, it would be interesting to consider developing proof sketches for a wider variety of aggregates without requiring the construction of a sample. To this end, we hope to develop more formal algebraic characterizations of the "proof sketchability" of various functions.

Acknowledgments

We are grateful to Aydan Yumerefendi for exploring many alternatives to our approach at an early stage, to David Wagner and Dahlia Malkhi for their helpful, insightful comments on earlier drafts of this work, and to Satish Rao for his perspective on *k*-connectivity.

9. REFERENCES

[1] R. Agrawal, P. J. Haas, and J. Kiernan. A system for watermarking relational databases. In *ACM SIGMOD International Conference on Management of Data*, 2003.

[2] R. Agrawal, R. Srikant, and D. Thomas. Privacy preserving OLAP. In *ACM SIGMOD International Conference on Management of Data*, 2005.

[3] B. Babcock, S. Chaudhuri, and G. Das. Dynamic sample selection for approximate query processing. In *Proceedings of the ACM SIGMOD international conference on Management of data (SIGMOD)*, pages 539–550, San Diego, CA, USA, June 2003.

[4] Z. Bar-Yossef, T. Jayram, R. Kumar, D. Sivakumar, and L. Trevisan. Counting distinct elements in a data stream. In *Proceedings of RANDOM*, Cambridge, MA, USA, Sept. 2002.

[5] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[6] P. Bonnet, J. Gehrke, and P. Seshadri. Towards sensor database systems. In *Proceedings of the Second International Conference on Mobile Data Management (MDM)*, pages 3–14, Hong Kong, China, Jan. 2001.

[7] J. Considine, F. Li, G. Kollios, and J. Byers. Approximate Aggregation Techniques for Sensor Databases. In *Proc. of the International Conference on Data Engineering (ICDE)*, 2004.

[8] G. Cormode and S. Muthukrishnan. "What's Hot and What's Not: Tracking Most Frequent Items Dynamically". In *Proceedings of ACM PODS*, pages 296–306, San Diego, California, June 2003.

[9] A. Dobra, M. Garofalakis, J. Gehrke, and R. Rastogi. Processing complex aggregate queries over data streams. In *Proceedings of the ACM SIGMOD international conference on Management of data (SIGMOD)*, pages 61–72, Madison, WI, USA, 2002.

[10] P. Flajolet and G. N. Martin. Probabilistic Counting Algorithms for Data Base Applications. *J. Comput. Syst. Sci.*, 31(2):182–209, 1985.

[11] S. Ganguly, M. Garofalakis, and R. Rastogi. Processing set expressions over continuous update streams. In *Proceedings of the ACM SIGMOD international conference on Management of data (SIGMOD)*, pages 265–276, San Diego, CA, USA, June 2003.

[12] P. B. Gibbons. Distinct sampling for highly-accurate answers to distinct values queries and event reports. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB)*, pages 541–550, Rome, Italy, Sept. 2001.

[13] P. B. Gibbons and S. Tirthapura. Estimating simple functions on the union of data streams. In *Proceedings of the thirteenth annual ACM symposium on Parallel algorithms and architectures (SPAA)*, pages 281–291, Crete, Greece, July 2001.

[14] A. V. Goldberg and R. E. Tarjan. A new approach to the maximum-flow problem. *J. ACM*, 35(4):921–940, 1988.

[15] J. Holm, K. de Lichtenberg, and M. Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing (STOC)*, pages 79–89, Dallas, TX, United States, 1998.

[16] P. J. Huber. *Robust Statistics*. Wiley-Interscience, 2003.

[17] R. Huebsch, B. N. Chun, J. M. Hellerstein, B. T. Loo, P. Maniatis, T. Roscoe, S. Shenker, I. Stoica, and A. R. Yumerefendi. The architecture of PIER: an Internet-scale query processor. In *CIDR*, pages 28–43, 2005.

[18] W. Liang, R. P. Brent, and H. Shen. Fully dynamic maintenance of *k*-connectivity in parallel. *IEEE Trans. Parallel Distrib. Syst.*, 12(8):846–864, 2001.

[19] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: A Tiny AGgregation service for ad-hoc sensor networks. In *Fifth Symposium on Operating Systems Design and Implementation (OSDI)*, pages 131–146, Boston, MA, USA, Dec. 2002.

[20] A. Manjhi, S. Nath, and P. B. Gibbons. Tributaries and deltas: efficient and robust aggregation in sensor network streams. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 287–298, Baltimore, MD, USA, 2005.

[21] G. S. Manku and R. Motwani. "Approximate Frequency Counts over Data Streams". In *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB)*, pages 346–357, Hong Kong, China, Aug. 2002.

[22] G. S. Manku, S. Rajagopalan, and B. G. Lindsay. "Random sampling techniques for space efficient online computation of order statistics of large datasets". In *Proceedings of ACM SIGMOD*, June 1999.

[23] R. Motwani and P. Raghavan. *Randomized algorithms*. Cambridge University Press, New York, NY, USA, 1995.

[24] S. Nath, P. Gibbons, Z. Anderson, and S. Seshan. Synopsis Diffusion for Robust Aggregation in Sensor Networks. In *Proceedings of ACM SenSys*, Baltimore, MD, USA, Nov. 2004.

[25] C. Olston and J. Widom. Offering a precision-performance tradeoff for aggregation queries over replicated data. In *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB)*, pages 144–155, Cairo, Egypt, 2000.

[26] M. Patrascu and E. D. Demaine. Lower bounds for dynamic connectivity. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing (STOC)*, pages 546–553, Chicago, IL, USA, 2004.

[27] F. Perich, J. L. Undercoffer, L. Kagal, A. Joshi, T. Finin, and Y. Yesha. In Reputation We Believe: Query Processing in Mobile Ad-Hoc Networks. In *International Conference on Mobile and Ubiquitous Systems: Networking and Services*, Boston, MA, August 2004.

[28] B. Przydatek, D. Song, and A. Perrig. SIA: Secure Information Aggregation in Sensor Networks. In *Proceedings of the 2nd ACM Conference on Embedded Network Sensor Systems (SenSys)*, Los Angeles, CA, USA, Nov. 2004.

[29] V. Raman and J. M. Hellerstein. Partial results for online query processing. In *Proceedings of the ACM SIGMOD international conference on Management of data (SIGMOD)*, pages 275–286, Madison, WI, USA, June 2002.

[30] R. V. 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.*, 21(2):164–206, 2003.

[31] H. O. Sanli, S. Ozdemir, and C. Hasan ˙SRDA: secure reference-based data aggregation protocol for wireless sensor networks. In *Proceedings of the Vehicular Technology Conference*, pages 4650–4654. IEEE, Sept. 2004.

[32] A. Seshadri, M. Luk, E. Shi, A. Perrig, L. van Doorn, and P. Khosla. Pioneer: verifying code integrity and enforcing untampered code execution on legacy systems. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*, pages 1–16, Brighton, UK, Oct. 2005.

[33] M. Tsimelzon, B. Weihl, and L. Jacobs. ESI language specification. Technical Report 1.0, Akamai Technologies and Oracle Corporation, 2001. http://www.esi.org/language_spec_1-0.html.

[34] J. S. Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, 11(1):37–57, 1985.

[35] D. Wagner. Resilient aggregation in sensor networks. In *ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, Oct. 2004.

[36] P. Yalagandula and M. Dahlin. A scalable distributed information management system. In *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM)*, pages 379–390, Portland, OR, USA, Sept. 2004.