Proof Sketches: Verifiable In-Network Aggregation

Minos Garofalakis
Intel Research Berkeley

Joseph M. Hellerstein
UC Berkeley

Petros Maniatis
Intel Research Berkeley

Abstract

Recent work on distributed, in-network aggregation assumes a benign population of participants. Unfortunately, modern distributed systems are plagued by malicious participants. In this paper we present a first step towards verifiable yet efficient distributed, in-network aggregation in adversarial settings. We describe a general framework and threat model for the problem and then present proof sketches, *a compact verification mechanism that combines cryptographic signatures and Flajolet-Martin sketches to guarantee acceptable aggregation error bounds with high probability. We derive proof sketches for count aggregates and extend them for random sampling, which can be used to provide verifiable approximations for a broad class of data-analysis queries, e.g., quantiles and heavy hitters. Finally, we evaluate 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 aggregation 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. A common yet widely applicable motivating scenario comes from corporate network management. It is common practice for most computers owned by a corporation to run locally a host intrusion detection agent (HID) such as SNORT, or other local management agents, e.g., for asset tracking, connection quality monitoring, etc. Such agents generate event streams like “My CPU utilization is 95%” or “I am under a NIMDA attack” and typically communicate over the corporate network with a central console, located at corporate IT headquarters. The querier in this scenario is a network manager at the console, posing aggregate queries on agents to understand a developing performance problem or the outbreak of a worm. A usual query might be “How many Windows XP hosts running patch X have CPU utilization above 95%?”. We call these count queries *predicate polls*, since each host responds with one boolean value (“one machine, one vote”). Besides predicate polls, there are queries like “how many HID log entries indicate a NIMDA exploit?” (a tuple counting query), or even “return the OS versions of k randomly chosen HIDs that identified exploit X” (a random sampling query). In such scenarios, each agent may have zero, one, or many records to contribute to the aggregation.

Data warehousing may be an inappropriate solution to this problem, since there may be many thousands of globally distributed management agents in the corporation, some connected by slow links (WiFi or modem), updating their streams at sub-second intervals. Results to queries may be required within seconds to catch anomalies or other serious problems. Instead, *in-network aggregation*, for instance over a well provisioned distributed infrastructure like Akamai, can cut down on the bandwidth, latency, and processing needs of real-time backhauling, by spreading more of the computation within the network. In addition, for many queries the querier may be interested in detecting only trends or interesting patterns, not precise answers. 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*.

Unfortunately, although in-network aggregation can deliver real-time, efficient results, it must anticipate an untrusted aggregation infrastructure. For instance, the aggregation functionality may be hosted by a third party (e.g., Akamai), or might be shared among multiple organizations that pool their network resources. Even if wholly owned by a single organization, the infrastructure will typically be plagued by viruses and worms. Consequently, queriers posing questions may require extra assurances that the aggregation results reflect accurately what the data sources (the agents) produced. An outstanding and largely overlooked research challenge is to provide trustworthy query results in such environments with mutually distrustful or even adversarial parties. The challenge we consider in this paper is preventing malicious aggregators from undetectably perturbing the results of distributed, in-network aggregation.

Prior Work. Recent contributions have tackled communication faults in this setting [5, 11, 13], but remain vulnerable to malicious misbehavior. Participant misbehavior includes manufacturing spurious subresults that were not derived from authentic data generators, and suppression of true results derived from the data generators; such activities can perturb aggregate results arbitrarily. Related work that addresses misbehavior is typically limited in scale to far smaller settings than the Internet-wide aggregation problem we explore here. The SIA approach [16] prevents a single untrusted aggregator from tampering with aggregation results over a sensor network using spot-checking of aggregates against tamper-evident samples of the input sensor data, but does not address scenarios in which many aggregators must collaborate to compute a result, which are necessary for larger-scale aggregations. Similarly, general secure function evaluation approaches such as Fairplay [12] address in practice only two-party computations and, even then, have prohibitive overheads when considering scales beyond counting a few hundred inputs among a few wide-area parties every few minutes. Wagner's resilient sensor aggregation work [18] uses *robust statistics* to protect from data source misbehavior but does not address the orthogonal problem of misbehaving aggregators. Solving the problem of verifiability of in-network aggregation would complement well much related work on other distributed security problems with databases, such as data source trustworthiness [1] or privacy-preserving query processing [2]).

Our Contributions. In this paper, we propose a family of certificates called *proof sketches*. These allow parties in a distributed aggregate computation to verify that the final result cannot have been perturbed by more than a small error bound with high probability. To our knowledge, this is the first practical work to tackle verifiable, multi-party query processing in the face of adversaries. We believe it represents a significant step towards trustworthy distributed query processing.

We initially 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 relation, distributed across numerous participants. We develop proof sketches of size logarithmic in $|R|$ for a broad class of count aggregates, and prove that they detect tampering beyond a small factor of the true count. We extend our scheme to develop compact proof sketches for *verifiable random sampling*, which can itself be used to give verifiable approximations for a wide variety of data-analysis queries, including quantiles and heavy hitters.

Our basic technique combines Flajolet-Martin (FM) sketches [6] with compact cryptographic signatures we call *authentication manifests*. Authentication manifests ensure that none of the data captured by the sketch were invented by adversarial aggregators. To prevent aggregators from silently omitting valid data from counts or samples, we estimate the overall population count, either directly from the verifiable sample or through an additional counting proof sketch on the *complement* of the query predicate. Assuming the size of the participant population is approximately known – a reasonable assumption in most environments we consider – the querier can check the accuracy of this estimate to detect malicious tampering.

We briefly discuss a number of extensions to our core proof-sketching ideas, including general design guidelines for the development of new proof sketches and accountability mechanisms. Our empirical evaluation of verifiable approximate count via proof sketches under different adversarial strategies shows that, to remain undetected in practice, adversaries must limit themselves to much smaller-scale tampering than the worst-case analysis suggests[1].

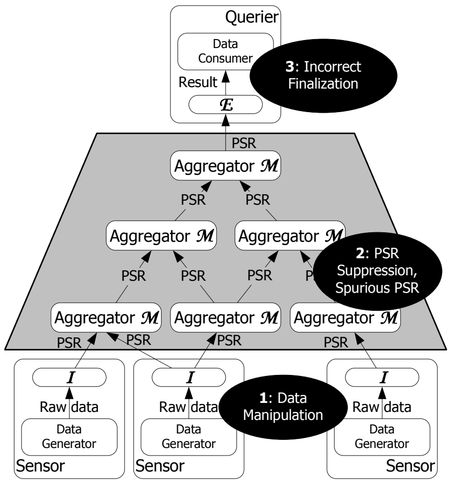

Figure 1. Dataflow among agents in the system, with potential threats listed in the black ovals. Proof sketches are used to thwart the threats within the gray trapezoid. In each component, we indicate the aggregation functionality ($\mathcal{I}$, $\mathcal{M}$, $\mathcal{E}$) performed.

2. Preliminaries

In-Network Aggregation Functionality. In our discussion, we adopt the distributed aggregation terminology of TAG [11]. TAG implements simple aggregates like COUNT, SUM, or AVERAGE via three functions: an initializer $\mathcal{I}$, a merging function $\mathcal{M}$, and an evaluator function $\mathcal{E}$. $\mathcal{M}$ has the form $\langle z\rangle = \mathcal{M}(\langle x\rangle, \langle y\rangle)$, where $\langle x\rangle$ and $\langle y\rangle$ 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. $\langle z\rangle$ is the PSR resulting from the application of $\mathcal{M}$ to $\langle x\rangle$ and $\langle y\rangle$. For example, if $\mathcal{M}$ is the merging function for AVERAGE, each PSR will consist of a pair of values, sum S and count C, and, given two PSRs $\langle S_1, C_1\rangle$ and $\langle S_2, C_2\rangle$, $\mathcal{M}(\langle S_1, C_1\rangle, \langle S_2, C_2\rangle) = \langle S_1 + S_2, C_1 + C_2\rangle$. The initializer $\mathcal{I}$ specifies how to instantiate a PSR for a single input data value; for AVERAGE, $\mathcal{I}(x) = \langle x, 1\rangle$. Finally, the evaluator $\mathcal{E}$ maps a PSR to an aggregate. For AVERAGE, $\mathcal{E}(\langle S, C\rangle) = S/C$.

In-network aggregation entities play different roles (Figure 1). *Sensors* are agents that produce raw data values for aggregation, and invoke initializer functions $\mathcal{I}$ to generate PSRs. In our scenario, the management agent is a "sensor,"

[1]Due to space constraints, we defer proof arguments and several details to the full paper.

and agent measurements or HID alerts are the data. *Aggregators* combine multiple PSRs by applying the merging function $\mathcal{M}$ on their inputs and forwarding a merged PSR to their outputs, according to an aggregation topology such as trees [11], depth-based DAGs [5], and hybrids of trees and DAGs [13]. The *querier* is the agent that receives the final PSR, validates it and, if successful, executes the evaluator $\mathcal{E}$ to generate the result.

We assume that nodes in the system are connected. Queries from end users have unique IDs. They can be disseminated to all relevant agents, including sensors and aggregators [10, 11]. In some environments, queries can be disseminated along with the agent software (e.g., every HID reports on a predetermined set of predicate polls or other queries periodically). Alternatively, queries can be 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 agents.

Threat Model. The adversary may control any and all aggregators in the infrastructure to implement a holistic strategy of her choosing. Two types of aggregator misbehavior are possible (gray trapezoid in Figure 1). In *partial-state suppression* (or *deflation*), aggregators omit data from input PSRs during the merging function. The simplest such attack is to suppress an entire PSR, but others are possible (e.g., suppressing a subset of a sample). In *spurious partial state* (or *inflation*), aggregators introduce data into output PSRs that should not be there. The simplest such attack is to manufacture an entire PSR from scratch, but it is also possible to manipulate an existing PSR to reflect manufactured inputs, or inputs from elsewhere in the aggregation topology.

The adversary is bound to common limits of computational tractability: she cannot forge signatures for keys she does not possess. We require all sensors to be registered with an organization-wide public key infrastructure (PKI) so that we can authenticate the data they produce.

3. AM-FM Proof Sketches

To begin our discussion of proof sketches, we consider the special case of "predicate poll" queries; we expand to arbitrary counts 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 predicate of interest; also, let C_{pred} ($\leq U$) denote the answer to our predicate poll. Consider the predicate COUNT aggregate:

$$I(t) = \begin{cases} \langle 1 \rangle & \text{if } pred(t) = \text{true} \\ \langle 0 \rangle & \text{otherwise} \end{cases} \;;$$
$$\mathcal{M}(\langle x \rangle, \langle y \rangle) = \langle x + y \rangle \;;\; \mathcal{E}(\langle x \rangle) = x$$

where t denotes a sensor's local data record. We treat malicious inflation and deflation of a PSR's value separately.

3.1. Detecting Inflation

To offer intuition, we start with a simple but impractical way to ensure that aggregators do not inflate the running sum: counting in a unary representation, where the PSR is a *bitmap of size* U. The aggregation logic becomes:

$$I(t) = \begin{cases} \langle 2^a \rangle & \text{if } pred(t) = \text{true at node } a \\ \langle 0 \rangle & \text{otherwise} \end{cases} \;;$$
$$\mathcal{M}(\langle p \rangle, \langle q \rangle) = \langle p \text{ OR } q \rangle \;;\; \mathcal{E}(\langle p \rangle) = |p|$$

where $|p|$ counts 1-bits in bitmap p. We protect from inflation by requiring that every sensor voting "yes" also cryptographically sign the unique identifier of the predicate poll, and we collect all such signatures into a signature set called the *authentication manifest (AM)* for a PSR. The initializer I at sensor $a \in [U]$ initializes the AM to contain sensor a's signature if a satisfies the predicate. The merging function $\mathcal{M}$ unions input AMs in addition to OR-ing input bitmaps. To validate the final PSR, the querier verifies that, for each 1-bit in the bitmap at position a, the AM contains a valid signature of the query identifier by sensor a's public key.

Though impractical, since the size of a thus defined PSR is as large as the number of "yes" votes counted (roughly $O(U)$), this basic idea motivates our actual approach, which produces a compact AM. To that goal, we relax our security requirement: instead of detecting all count inflations, 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 [6]. 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 [6] is a one-pass (streaming) algorithm that relies on a family of hash functions $\mathcal{H}$ 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 basic *FM-sketch* synopsis (for a fixed choice of hash function $h \in \mathcal{H}$) is 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, where $\texttt{lsb}(s)$ denotes the position of the *least-significant* 1 *bit* in the binary string s. It is not difficult to see that for any $i \in [U]$, $\texttt{lsb}(h(i)) \in \{0, \ldots, \log U - 1\}$ and $\mathbf{Pr}[\texttt{lsb}(h(i)) = l] = \frac{1}{2^{l+1}}$. After C distinct values from $[U]$ have been encountered in the stream, the location of the rightmost zero in the bit-vector synopsis is an indicator of $\log C$. To boost accuracy and confidence, the FM algorithm employs averaging over several independent instances (i.e., r independent choices of the mapping hash function $h_j \in \mathcal{H}$ for $j \in [1, r]$ and corresponding FM sketches). Recent work [3, 7, 9] has shown that, using only $r = O(\frac{\log(1/\delta)}{\varepsilon^2})$ FM bit vectors built with simple, limited-independence hash functions, one can provide (randomized) (ε, δ)-*estimators* for the number of distinct values C; that is, the computed estimate $\hat{C}$ satisfies $\mathbf{Pr}\left[|\hat{C} - C| \leq \varepsilon C\right] \geq 1 - \delta$.

FM-sketch summaries are naturally *composable*: simply

[2]Logarithms in this paper have base 2 unless otherwise noted.

$$I(t) = \langle 2^{\mathtt{lsb}(h_j(t\|a))},\ \{\langle \mathtt{lsb}(h_j(t\|a)), t, a, s_a(t)\rangle\}\rangle$$
$$\mathcal{M}(\langle p, \mathcal{A}_p\rangle, \langle q, \mathcal{A}_q\rangle) = \langle p \text{ OR } q,\ \mathcal{A}_p \sqcup \mathcal{A}_q\rangle$$
$$\mathcal{E}(\{\langle p_j, \mathcal{A}_{p_j}\rangle : j = 1, \ldots, r\}) = \mathsf{FMEstimate}(\{p_1, \ldots, p_r\})$$

Figure 2. Definition of (basic) AM-FM. The predicate test in I is omitted for brevity. $\|$ denotes concatenation.

OR-ing independently built bitmaps (e.g., over data sets a_1 and a_2) for the same hash function gives precisely the sketch of the union of the underlying sets (i.e., $a_1 \cup a_2$). This makes FM sketches ideally suited for in-network computation [5].

Inflation-free FM. 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 for building FM sketches, each 1-bit can be authenticated at the querier by a single signed value from one of the sensor nodes that turn 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 $\langle k, t, a, s_a(t)\rangle$, where k is the bit position, t the tuple, a the sensor identifier, and $s_a(t)$ the sensor's signature for the tuple. We refer to the resulting sketch structure as an *AM-FM proof sketch*.

Figure 2 outlines the definition of our AM-FM sketches. Note that the FM hash function is applied to the $\langle$*dataRecord*, *sensorID*$\rangle$ pair to ensure uniqueness across all input data records. $\mathcal{A}_p$ denotes the authentication manifest for FM bit-vector p: if p's k-th bit is set to 1, the k-th component of $\mathcal{A}_p$ is $\langle k, t, a, s_a(t)\rangle$, and is valid iff $k = \mathtt{lsb}(h(t\|a))$ and $s_a(t)$ is a valid signature on t by sensor a. The $\sqcup$ operator forms a subset of the union of its inputs, retaining one input "exemplar" $\langle k, t, a, s_a(t)\rangle$ for each $k \in [0, \log U - 1]$ (e.g., chosen randomly). Finally, the evaluator function executes a count-estimation procedure over the collection of FM bit-vectors built [7, 9].

3.2. Deflation Detection

The authentication manifest in AM-FM prevents malicious aggregators from turning 0-bits in an FM PSR into 1-bits. The remaining possible attack is to turn 1-bits into 0-bits, removing the corresponding signatures from the AM. This attack could deflate the count in the FM sketch.

One natural approach to preventing this attack is to route initialized PSRs along multiple redundant aggregation paths between the sensors and the querier. Such redundant routing schemes exploit the duplicate-insensitive nature of FM sketches, and have been proposed for adding fault tolerance for benign aggregator populations in the face of an unreliable network fabric [13]. Our environment introduces more stringent requirements: we need to provide strong guarantees that our verification procedure bounds the amount of error an adversary can introduce. Extending redundant communication schemes to our setting requires strong assumptions on the adversary and the aggregation topology, and introduces difficult algorithmic and practical challenges, which AM-FM does not face.

Complementary Proof Sketches. Our proposed 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 sensor population (the "universe"). This seems like a very strong assumption, but in many scenarios it is not at all unreasonable: 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 management-agent scenarios. For the moment then, we assume that the querier can acquire the correct value for U at any time.

The technique we use accompanies each predicate poll *pred* with its *complementary* poll $\neg$*pred*; the intuitive objective is 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 to prevent significant inflation of either count. By preventing the complement $\hat{C}_{\neg pred}$ from being inflated, we thereby prevent $\hat{C}_{pred}$ from being undetectably *deflated*: to deflate $\hat{C}_{pred}$, the adversary would have to inflate $\hat{C}_{\neg pred}$ to avoid detection by the sum check. We focus on this scheme for the remainder of this section, and provide bounds on the undetectable deflation 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 as per the validation described in Section 3.1. Given a valid AM for an FM sketch, 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?

We 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 [7]. Our deflation verification step flags an attack when the condition $\hat{C}_{pred} + \hat{C}_{\neg pred} \geq (1 - \varepsilon)U$ is violated. The following theorem establishes 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 verification, 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}.*

Thus with AM-FM any 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. The offered error guarantees are in terms of εU factors, which are typically sufficient for predicates that represent significant fractions of U. For low-selectivity predicates, as with regular sketch-based approximation, it can be preferable to compute an exact result rather than a high-relative-error approximation. In the AM-FM setting, this means that when $\hat{C}_{\neg pred}$ is found to be close to U, the querier can request that the few sensors with "Yes" votes contact it directly. This request should be disseminated directly from the querier to the sensors, without any untrusted intermediaries, to prevent adversarial aggregators from suppressing the request. Direct communication with the sensor population can take more time than dissemination over the aggregation population. Hence AM-FM is best suited to high-selectivity predicate polls while ensuring verifiable results for low selectivities at a higher cost.

4. Verifiable Random Sampling

We turn to the more involved problem of constructing a *verifiable random sample* of given size k over the sensors' data tuples. For clarity, we limit sensors to a single tuple each, but relax this limitation in Section 5. As in aggregation, we wish to ensure that messages sent by aggregators are small, and the result is verifiably an unbiased random sample of the data. Such a sample is a *general-purpose* summary of the sensor contents that the querier can use to approximate verifiably a variety of different aggregation functions and selection predicates, not known beforehand.

A conventional random-sampling summary is 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 $= k/N$). A simple adaptation of reservoir sampling [17], for instance, can generate such a sample dynamically moving up the aggregation topology, but falls short of our verifiability goals. Consider an aggregator receiving two random samples (s_1, N_1) and (s_2, N_2) from its children. Despite individual sampled tuple authentication (e.g., via signatures), the aggregator can still introduce arbitrary bias in the sample: for example, instead of sub-sampling s_1 and s_2 with the appropriate rates, it can favor tuples from s_1 over tuples from s_2, or even deterministically choose the smallest values, arbitrarily poisoning approximate results computed over the resulting sample. The key problem in our threat model is making the sampling procedure run by each aggregator – e.g., the random coin flips – verifiable.

Our solution, *AM-Sample proof sketches*, collects a random sample by mapping $\langle \mathit{dataRecord}, \mathit{sensorID} \rangle$ elements to buckets with exponentially decreasing probabilities, using hash functions as in FM. The key difference from AM-FM 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. Using sensor IDs to make tuples distinct, we essentially use a method similar in spirit to Gibbons' distinct-sampling technique [8] for approximating `COUNT DISTINCT` queries over data warehouses.

$$I(t) = \langle 0,\ \{\langle \texttt{lsb}(h(t \| a)), t, a, s_a(t) \rangle\} \rangle \qquad (1)$$

$$\mathcal{M}(\langle L_1, \mathcal{S}_1 \rangle, \langle L_2, \mathcal{S}_2 \rangle) = \langle L,\ \mathcal{S}(L, \mathcal{S}_1, \mathcal{S}_2) \rangle \qquad (2)$$

where $\mathcal{S}(L, \mathcal{S}_1, \mathcal{S}_2) = \{\langle l, t, a, s_a(t) \rangle \in \mathcal{S}_1 \cup \mathcal{S}_2 :\ l \geq L\}$, and

$$L = \begin{cases} \max(L_1, L_2) & \text{if } |\mathcal{S}(\max(L_1, L_2), \mathcal{S}_1, \mathcal{S}_2)| \leq (1+\varepsilon)2k \\ \max(L_1, L_2) + 1 & \text{otherwise} \end{cases}$$

$$\mathcal{E}(\langle L, \mathcal{S} \rangle) = \{t : \langle l, t, a, s_a(t) \rangle \in \mathcal{S}\}, \text{sampling rate} = 2^{-L} \qquad (3)$$

Figure 3. Definition of AM-Sample.

Formally, given a uniformly randomizing hash function h over $\langle \mathit{dataRecord}, \mathit{sensorID} \rangle$ pairs and a sample size k, an AM-Sample proof sketch is a pair $\langle L, \mathcal{S} \rangle$, where $\mathcal{S} = \{\langle l_1, t_1, a_1, s_{a_1}(t_1) \rangle, \ldots, \langle l_m, t_m, a_m, s_{a_m}(t_m) \rangle\}$ is a subset of $m = \Theta(k)$ authentication manifests $\langle l_i, t_i, a_i, s_{a_i}(t_i) \rangle$ with corresponding bucket levels $l_i = \texttt{lsb}(h(t_i \| a_i))$. A well-formed AM-Sample sketch stores exactly the authentication manifests for $\langle \mathit{dataRecord}, \mathit{sensorID} \rangle$ elements at levels greater than or equal to L, which 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 description of our in-network aggregation scheme for AM-Sample proof sketches is given in Figure 3. Briefly, our algorithm starts by computing the authentication manifests and bucket levels for individual sensors (Equation 1). These manifests are then unioned up the aggregation topology, only keeping elements at the maximum level $\max(L_1, L_2)$ with every PSR merge, effectively sub-sampling elements at the lowest sampling rate among input PSRs. To keep the sketch size under control, the sampling rate drops by a factor of 2 (setting $L = \max(L_1, L_2) + 1$) when the sample size grows beyond $2k(1+\varepsilon)$ (Equation 2); $\varepsilon < 1$ denotes an error parameter determined by the target sample size. As we show, for large enough k, the size of our AM-Sample sketch never grows beyond a range $[(1-\varepsilon)k, (1+\varepsilon)2k]$ (with high probability) during aggregation.

4.1. Verification and Analysis

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 aggregators from inventing new data, since all tuples are signed by a sensor. Second, AM signatures also prevent aggregators from migrating tuples across bucket levels (thereby biasing random-sampling choices) since the level is determined through hashing by the signed tuple and sensor identifier. In a sense, the hash function can dictate the coin flips of intermediate aggregators, ensuring that no inappropriate element selections bias the sample. Note that our technique – unlike reservoir sampling – is also naturally duplicate-insensitive and can, therefore, be used in an aggregation topology that merges PSRs redundantly.

Besides inflation, the adversary may also deflate samples, as with AM-FM sketches, either by artificially increasing the bucket level (to discard all lower-level elements), or by removing specific elements at the current bucket level. To quantify the deflation-error guarantees provided by our AM-Sample proof sketches, we assert 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* $|\mathcal{S}|$ *at the querier is at least* $(1-\varepsilon)k$ *with probability* $\geq 1-\delta$.

Given this result, a querier can raise a deflation alarm iff the collected AM-Sample proof sketch sample size $|\mathcal{S}|$ fails the condition $|\mathcal{S}| \geq (1-\varepsilon)k$.

We now consider the error guarantees provided for given polls by such a sample. Let σ denote a known lower bound on the selectivity of a predicate poll that the querier wishes to 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 population size U, the querier can limit the potential impact of adversarial deflation on the collected AM-Sample $\langle L, \mathcal{S}\rangle$ by additionally ensuring that $2^L \cdot |\mathcal{S}| \geq (1-\varepsilon\sqrt{\sigma})U$. Again, if this condition is violated, the querier flags a deflation attack with high probability ($\geq 1-\delta$). The following theorem establishes these verifiable deflation-error guarantees for poll queries over AM-Sample proof sketches.

Theorem 3 *Assume an AM-Sample proof sketch 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}*).*

Leveraging verifiable samples. The bounds in Theorems 1 and 3 show that, for a poll on a *given* predicate, AM-FM sketches offer a better space/accuracy trade-off 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, and can be leveraged to provide approximate answers for different classes of data-analysis queries at the querier.

For instance, a (verified) AM-Sample can be used directly to construct *approximate quantile summaries* [15] over different attributes in the sampled tuples, or to discover *heavy-hitters/frequent items* [14] 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) representing 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.

5. Extensions

Multi-Tuple Sensors. We now consider general distributed query processing scenarios, where each sensor in the underlying "universe" $[U]$ can store multiple data tuples. Specifically, let m_i denote the total number of tuples at sensor i and let M denote the total tuple population, i.e., $M = \sum_{i\in[U]} m_i$. Our verifiable-aggregation protocols and analysis for AM-FM proof sketches and AM-Sample proof sketches are immediately applicable in this more general setting by simply replacing U with M, *assuming* that the querier has accurate knowledge of M. Of course, in any realistic scenario, M is not a static quantity but varies over time as sensors update their local data streams. We present a communication-efficient *verifiable approximate counting algorithm* to estimate a tuple population M in the system, which builds on our predicate polling mechanism while still requiring only logarithmic-size messages. With a selection predicate, the algorithm performs a "tuple predicate poll" whereas, without a selection predicate, it can estimate the size of the total tuple population, which can be used with the verification steps of subsequent tuple predicate polls as described above.

Fix a small $\theta > 0$. Let S_k denote the subset of sensors in the system with at least $(1+\theta)^k$ data tuples; that is, $S_k = \{i \in U : m_i \geq (1+\theta)^k\}$, and let $u_k = |S_k|$. Also, let K denote the maximum index k for which $u_k > 0$ – obviously, $K \leq \log_{(1+\theta)} M \approx \frac{\log M}{\theta}$. Now, define an approximate (quantized) tuple count $M_a = \sum_{k=0}^{K}(u_k - u_{k+1})(1+\theta)^k$ (i.e., rounding down each m_i to the closest power of $(1+\theta)$). It is not difficult to see that M_a is within a $(1+\theta)$ multiplicative factor of the true tuple count M, i.e., $(1+\theta)^{-1}M \leq M_a \leq (1+\theta)M$.

The algorithm uses a logarithmic number of AM-FM predicate polls to produce accurate estimates $\hat{u}_k$ for each u_k, and then employs these estimates to approximate M_a. It is crucial to avoid error bounds that depend on the size of the overall sensor population U, especially for large values of k (i.e., error factors of $\varepsilon U(1+\theta)^k$), which can have a significant effect on the estimate. Instead, our estimator works in an "incremental" manner, taking advantage of the inclusion relationship $S_k \subseteq S_{k-1}$. Briefly, our algorithm runs its k^{th} predicate poll over the S_{k-1} subset of sensors, and uses the $\hat{u}_{k-1}$ estimate in the verification step for $\hat{u}_k$, for each $k = 0, \ldots, K+1$. For convenience, we define $u_{-1} = U$.

More formally, for each $k = 0, \ldots, K+1$, our algorithm uses $O(\frac{\log(\log M/(\theta\delta))}{\varepsilon^2})$ AM-FM sketches to estimate u_k as the approximate predicate-poll count $\hat{C}_{p(k,S_{k-1})}$ (i.e., $\hat{u}_k = \hat{C}_{p(k,S_{k-1})}$), where $p(k,S_{k-1})$ and its negation are defined as:

$$p(k,S_{k-1}): \quad \text{“ } (1+\theta)^k \le m_i ? \text{ ”} \quad \text{and}$$
$$\neg p(k,S_{k-1}): \quad \text{“ } (1+\theta)^{k-1} \le m_i < (1+\theta)^k ? \text{ ”}$$

for each sensor i; for step k, the querier flags an adversarial omission attack iff the condition $\hat{C}_{p(k,S_{k-1})} + \hat{C}_{\neg p(k,S_{k-1})} \ge (1-\varepsilon)\hat{u}_{k-1}$ is violated. Once the index $K' \le K \le \frac{\log M}{\theta}$ such that $\hat{u}_{K'+1} = 0$ is reached, our algorithm returns the count estimate $\hat{M}_a = \sum_{k=0}^{K'} (\hat{u}_k - \hat{u}_{k+1})(1+\theta)^k$ for the total tuple count M. Note that all steps can run in parallel. The error guarantees for our verifiable approximate counting scheme are summarized in the following:

Theorem 4 *Using a total of $O(\frac{\log M \log(\log M/(\theta\delta))}{\theta\varepsilon^2})$ AM-FM sketches to estimate the approximate tuple count $\hat{M}_a$, and assuming successful final verification steps for all $k = 0, \ldots, K+1$, our counting algorithm guarantees that, with probability $\ge 1-\delta$,*

$$\hat{M}_a \in \left[\frac{(1-2\varepsilon(1+\varepsilon)(1+\theta))M}{1+\theta} - \frac{2\varepsilon U}{1+\theta} \,,\; (1+\varepsilon)(1+\theta)M \right].$$

This mechanism can be directly used to also compute `SUM`s over sensor values instead of YES counts over multiple tuples per sensor. Due to space constraints, we defer to the full paper the details of extensions to verifiable aggregates such as `SUM`s and `AVERAGE`s, as well as optimizations for our basic verifiable approximate counting scheme.

A Generalized Template for Proof Sketches. While 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 are as follows:

–*Compact Manifest:* A key challenge 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. This leads to a combinatorial optimization problem of choosing 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:* Complementary deflation protection hinges on the querier's ability to detect deflations given only the aggregate results on a subset of the sensor population and on its complement. In the proof sketches we have described, if an aggregate value can be approximately *reduced* to the sensor population size that verifiably justifies that value, deflation verification checks can be devised. Fortunately, building reductions by combining simpler, already known reductions – as with the geometric superimposition of AM-FM proof sketches earlier in this section – covers typical SQL aggregates. Going beyond to arbitrary aggregate functions is an open problem of both theoretical and practical importance.

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

Using these criteria in a fairly different setting, we briefly consider the example of Bloom Filters [4]; for brevity we assume familiarity with their 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 $\sigma_{pred(value)}$. 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.

As an example, verifiable Bloom Filters can be useful in approximating the size of an intersection: first we run a verifiable Bloom Filter aggregation on the values of one set, and then we compute the size of the intersection as an AM-FM count over the other set, where the selection predicate is a match with the Bloom Filter. The count of any semi-join-like operation can be similarly computed (e.g., key/foreign-key queries without referential integrity), and an analogous approach can be used to compute the size of anti-joins (e.g., set differences via SQL's EXCEPT and NOT IN clauses).

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. Alone, such alarms are useful in flagging a result as suspect. However, a subsequent forensic analysis tracing the cause of the alarm may be invaluable, not only for identifying malicious aggregators, but also for identifying false alarms. In the full paper, we will discuss different options for pinpointing inflation and deflation attacks through efficient trace-back mechanisms over the aggregation topology.

Universal Population Knowledge. Our complementary deflation protection requires some "ground truth" knowledge, namely U (the total size of the universe of sensors), to be maintained somehow at the querier. Knowing this size is reasonable in many practical settings, such as in an enterprise where nodes must register with a VPN. The full paper will further support this argument in more detail and explore scenarios in which the requirement may be relaxed (e.g., having only *approximate* knowledge of U) while retaining the benefits of complementary deflation protection.

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 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 final PSR given to the querier. Instead she will be able to affect only PSRs further down the aggregation topology. This means that she has only partial information on which to determine her suppression strategy; for example, some suppressed sketch bits 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 adversary strategies that approximate two different goals. In the **Targeted Strategy**, given target count $C_{malicious}$, the adversary suppresses as many sketch bits as will bring $\hat{C}_{pred}$ close to that target count. In the **Safe Strategy**, the adversary suppresses as many sketch bits as will deflate $\hat{C}_{pred}$ without violating the deflation verification condition (see Theorem 1). 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. The greater the coverage, the more likely it is that sketch bits suppressed by the adversary will not be set again by another aggregator.

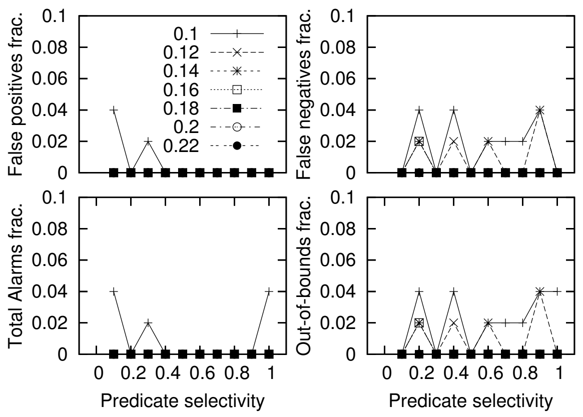

Figure 4. Fraction of alarms, out-of-bounds estimates, alarms for in-bounds estimates (false positives) and undetected out-of-bounds estimates (false negatives) over all runs for different selectivities (x axis) and different ε parameters (one per curve).

6.2. Results

The results presented below use 256 FM sketches—qualitatively similar numbers were obtained for other summary sizes and estimators (e.g., those from Ganguly et al. [7]). Even though we present results using the FM estimator, we heuristically select $\varepsilon = 0.15$ to match the Ganguly et al. worst-case bound of $O(\frac{\log(1/\delta)}{\varepsilon^2})$ with a probability of about 80% ignoring constants. 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 $100,000$. We vary adversary coverage from $1/64$-th of the universe to 100%.

We raise an alarm when the deflation verification condition fails (for given ε parameter), and we claim an 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 in bounds are conservatively termed *false positives* below, whereas estimates that are not in bounds but fail to raise an alarm are termed *false negatives*.

In this small scenario, our approach delivers to the querier no more than $256 \times \lceil \log 10^5 \rceil = 4252$ signed tuples, versus $100,000$ signed tuples via backhauling; backhauling is preferable only up to a population size of $2,950$. The ratio of signed-tuple volumes delivered to the querier grows with $O(U/\log U)$, and more than justifies the $\varepsilon = 0.15$ verifiable error bound for our scenario.

Benign Behavior. We begin with the behavior of AM-FM in the absence of adversaries. We wish to understand how frequently the deflation verification condition triggers and when it does, whether it is justified by the occasional outlying estimate. Figure 4 plots the frequency of alarms and out-of-bounds estimates, as well as the frequency of false positives and false negatives.

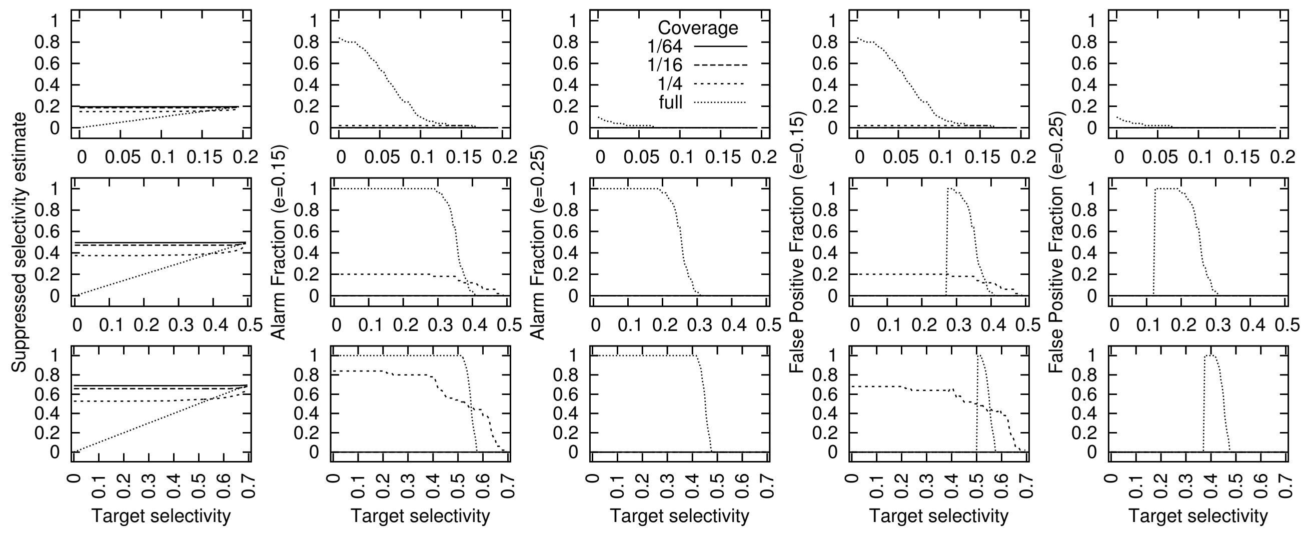

Figure 5. The effects of the targeted strategy on three predicate polls of selectivities 0.2 (top), 0.5 (middle), and 0.7 (bottom). All x axes are the target selectivity of the adversary, from the predicate selectivity down to 0. On the left, we show the average suppressed selectivity. The next two columns show the fraction of alarms over all runs with an aggressive $\varepsilon = 0.15$ and with a more conservative $\varepsilon = 0.25$. The final two columns show the fraction of false positives over all runs with the same two ε values. In each graph, we show different curves for four levels of adversarial coverage.

Given the number of sketches (256), the out-of-bounds occurrences are below 5% for $\varepsilon \geq 0.1$, which is in practice much better than the worst-case error and confidence probability described above for 256 sketches. False negatives are also well below 5% for all ε parameters. 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 selectivities. Those appear as alarms that are not false positives.

Targeted Strategy. The targeted strategy approximates an adversary who cares about a particular count suppression, event at the cost of being detected. She suppresses as many sketch bits as will cause the estimate (which she computes locally) to reach the target count $C_{malicious} < \hat{C}_{pred}$. When she has limited coverage, her attempts are thwarted as the remaining, non-malicious aggregators merge their PSRs with those she has concocted.

Figure 5 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 column 2), alarm frequency is greater as the adversary strives for larger estimate deflation. Lower-selectivity predicates (higher rows) suffer less from those alarms, since there is less wiggle room from 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 drops sharply at target selectivity $C_{pred} - \varepsilon(U + C_{\neg pred}) = 0.7 - 0.25(1 + 0.3) = 0.375$. The top row (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 less dramatic estimate deflation; compare the $1/4$-th coverage curve to the full coverage curve. Note however that the $1/4$-th coverage curve does not feature any sharp drops. This is an artifact of our conservative definition of "false positives." Since the low-coverage adversary 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 $\varepsilon = 0.15$), causing a false positive. For the more conservative $\varepsilon = 0.25$ this is not the case.

Safe Strategy. The safe strategy represents an adversary whose 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 6 plots the average deflation bias introduced by

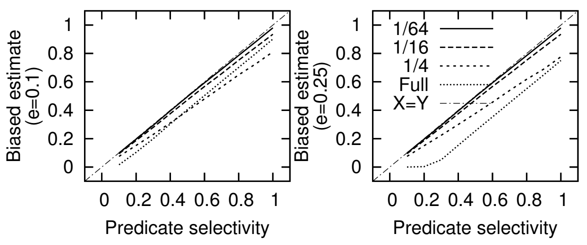

Figure 6. Average deflated selectivity estimate for an adversary of varying coverage (different curves) using the safe interior strategy. On the left, the strategy uses a timid $\varepsilon = 0.1$. On the right the strategy uses a more permissive $\varepsilon = 0.25$.

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, zero-error, estimate.) At the tight 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 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.

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 for a real-life setting—we are currently exploring different techniques in that context.

7. Conclusions and Future Work

This work on proof sketches represents a first step in an agenda towards general-purpose verifiable distributed query processing. Our approach marries 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 suppressions is also simple, though it does require the querier to track the size of the sensor population. 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.

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 the anonymous reviewers for the feedback.

References

[1] R. Agrawal, P. J. Haas, and J. Kiernan. A system for watermarking relational databases. In *SIGMOD*, 2003.

[2] R. Agrawal, R. Srikant, and D. Thomas. Privacy preserving OLAP. In *SIGMOD*, 2005.

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

[4] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Comm. of the ACM*, 13(7), 1970.

[5] J. Considine, F. Li, G. Kollios, and J. Byers. Approximate Aggregation Techniques for Sensor Databases. In *ICDE*, 2004.

[6] P. Flajolet and G. N. Martin. Probabilistic Counting Algorithms for Data Base Applications. *JCSS*, 31(2), 1985.

[7] S. Ganguly, M. Garofalakis, and R. Rastogi. Processing set expressions over continuous update streams. In *SIGMOD*, 2003.

[8] P. B. Gibbons. Distinct sampling for highly-accurate answers to distinct values queries and event reports. In *VLDB*, 2001.

[9] P. B. Gibbons and S. Tirthapura. Estimating simple functions on the union of data streams. In *SPAA*, 2001.

[10] 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*, 2005.

[11] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: A Tiny AGgregation service for ad-hoc sensor networks. In *OSDI*, 2002.

[12] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay—A Secure Two-Party Computation System. In *USENIX Security*, 2004.

[13] A. Manjhi, S. Nath, and P. B. Gibbons. Tributaries and deltas: efficient and robust aggregation in sensor network streams. In *SIGMOD*, 2005.

[14] G. S. Manku and R. Motwani. Approximate Frequency Counts over Data Streams. In *VLDB*, 2002.

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

[16] B. Przydatek, D. Song, and A. Perrig. SIA: Secure Information Aggregation in Sensor Networks. In *SenSys*, 2004.

[17] J. S. Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, 11(1), 1985.

[18] D. Wagner. Resilient aggregation in sensor networks. In *ACM SASN*, 2004.