Veriable Order Statistics for Secure Aggregation

Hsu-Chun Hsiao
Chih-Yuan Wang
Joseph M. Hellerstein
Wei-Chung Teng
Chin-Laung Lei

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2009-48
http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-48.html

April 7, 2009

Verifiable Order Statistics for Secure Aggregation

Hsu-Chun Hsiao[†], Chih-Yuan Wang[*], Joseph M. Hellerstein[‡], Wei-Chung Teng[*], Chin-Laung Lei[§]

[†] Carnegie Mellon University

[*] National Taiwan University of Science and Technology

[‡] University of California, Berkeley

[§] National Taiwan University

ABSTRACT

In-network aggregation can save significant bandwidth in a distributed query systems, but is subject to attack by adversaries. Prior work addressed settings where data sources are trusted, but the aggregation infrastructure needs to be secured. We study extensions that also make aggregate queries robust to adversarial data sources, which can inject spurious values into the data stream to be aggregated. Wagner [31] observed that the field of *robust statistics* can provide tools here, since robust estimators (medians, trimmed means, median absolute deviations, etc.) provide formal guarantees on the degree to which perturbed data can have an effect on aggregate results. This raises the challenge of developing verifiable in-network algorithms for robust estimators. Many of the natural robust estimators are built on *order statistics*, so we focus here on verifiable techniques for in-network computation of order statistics. To our knowledge, there is no mechanism guarantees both the efficiency and verifiability of the order statistics computation. In this work, we present the *FM3 Proof Sketch* aggregation protocol, which efficiently and securely computes various approximate order statistics including medians, median absolute deviations, quantiles, ranks, and frequent items). We derive robustness and approximation guarantees for those queries in adversarial environments, and demonstrate empirically that our scheme is practically useful via experiments on real and synthetic data.

Keywords

Secure Aggregation, Order Statistics, Distributed Query Processing

1. INTRODUCTION

In-network aggregation protocols have gained increasing attention in large-scale distributed systems, such as sensor networks [21], distributed databases [18], and peer-to-peer systems [32]. By pushing computation into the network, in-network aggregation eliminates the need to transmit all raw data to a single computing center, and thus significantly reduces overall communication overhead for aggregation queries such as counts, averages, medians, minima, and maxima.

Although in-network aggregation results in significantly lower bandwidth usage than centralized computation, it introduces opportunities for adversaries to perturb the results of aggregation queries by either manipulating the computation of the aggregate function, or by inserting bogus data. A challenge in this context is to obtain robust, verifiable aggregate query results in untrusted environments.

A number of verifiable aggregation protocols for counts and averages have been proposed [4,12,27,33]. They address the threat of misbehaving aggregators, but assume that sources are trustworthy. Unfortunately, aggregate functions like average and standard deviation are sensitive to extreme values (outliers), so attackers who are able to "pollute" the input with even a single bad data item can perturb the aggregation result by an arbitrary amount. Verifiable aggregation does not address this issue of data pollution. On the other hand, a number of techniques address data pollution by misbehaving data sources [19,29–31], but rely on trusted aggregators to detect the outliers. To our knowledge, there is no efficient distributed aggregation mechanism in the literature that guarantees the robustness of results in the face of either misbehaving aggregators, misbehaving data sources, or both.

1.1 Robust Estimators

Many aggregation functions used in query processing are easily perturbed with minimal pollution. For example, consider computing the average value of a set of n readings reported by a network of distributed nodes. An adversary can arbitrarily shift the computed average value via the injection of a single spurious input value. In the terminology of the field of Robust Statistics, the average function has a *breakdown point* of $1/n$: only $1/n$ values need to be perturbed to cause the function's accuracy to "break down" arbitrarily [17].

Wagner [31] observed that aggregation functions from Robust Statistics (so-called *robust estimators*) can be used to mitigate the effects of data pollution in distributed aggregation. A common, intuitive class of robust estimators correspond to *order statistics*, which are based on the rank of an item in order. *Median* is a widely used robust statis-

tic of a distribution's "center" that is often used in place of the easily-perturbed mean, while *median absolute deviation (MAD)* ($median(|X - median(X)|)$ is an estimator of a distribution's "spread" that is often used in place of standard deviation[1]. Both Median and MAD have a breakdown point of 0.5, because an adversary must alter at least 50% of the data points before they can perturb the median and MAD estimators by an arbitrary value. 50% is the highest possible value of a breakdown point [28], so these aggregation functions are in that sense optimally robust. Another popular robust estimator based on order statistics is the k% *trimmed* mean, which drops the lowest and highest k% of values before computing the mean. This aggregate has a k% breakdown point.

1.2 Design Goals

Robust estimators make it difficult for adversaries to control the outcomes of aggregation via poisoning. But this is not sufficient: adversaries can still compromise the nodes involved in performing steps in the in-network aggregation protocol. To provide a complete solution, we need a verifiable aggregation protocol for robust estimators. In addition to verifiability, an effective solution should be based in the following three design goals:

Logarithmic Communication. To ensure scalability, we want to match the logarithmic communication complexity offered by standard, insecure in-network aggregation [21], rather than the linear-cost complexity that results from backhauling all the data to a centralized data warehouse.

Approximate Results with Guaranteed Error Bounds. One can compute order statistic queries exactly via a logarithmic number of sequential count queries [4]. However, in an environment where sources may contain polluted data, exact results over that polluted data are more trouble than they are worth; it is reasonable to tolerate a bounded degree of aggregation error in order to gain performance efficiency.

One Pass Protocol. Some of the prior techniques for verifiable aggregation are multi-pass solutions, requiring multiple traversals of a spanning tree of the network. In particular, the aggregate-commit-proof model [4] requires at least three passes to complete aggregation and verification procedures. These solutions are undesirable in large scale systems because of their communication overhead, but more importantly because network dynamics may make it difficult or impossible to maintain a fixed spanning tree communication structure across the required number of passes.

1.3 Contribution

We present a novel aggregation protocol for robust estimators based in order statistics, including the median, MAD, and trimmed mean mentioned above, as well as quantile, rank, frequent value, and other order-statistics-based aggregation functions. Our technique is based on the notion of a *Proof Sketch* [12]: a small summary structure that can be computed in-network, providing both approximate answers and verifiability guarantees at the query site. The protocol provides two security guarantees: verifiability against misbehaving aggregators and robustness to misbehaving data sources. Furthermore, the protocol satisfies the three design goals mentioned in Section 1.2.

Our solution consists of three components:

- We introduce a new sketch technique, the *FM3 Proof Sketch*, which summarizes a data set into a sublinear structure based on the Flajolet-Martin (FM) sketch [10] augmented with (authenticated) Minimum and Maximum (MM) "witness" values for each bit.
- Based on the FM3, we present a suite of *approximate query algorithms* for order statistics (quantiles, rank, and frequent items) with their error bound analysis. We also introduce the *MADAM* estimator – the *Median Absolute Deviation from an Approximate Median* – and provide bounds on its computation, and on its estimation of the true MAD.
- Using the verifiable order statistic algorithms and the FM3, we design a *secure aggregation protocol for order statistics*.

1.4 Outline

In Section 2, we define our network model and threat model. We then describe the preliminary problem of verifiable counting. Section 3 introduces our FM3 Proof Sketch and its methods. Section 4 uses the abstract FM3 for building a verifiable multiparty aggregation protocol. In section 5, we evaluate experimentally the security properties and performance of FM3. Section 6 contains discussion and related work.

2. PRELIMINARIES

We proceed to describe the network model and the threat model. We also give a brief introduction of a verifiable counting algorithm, which serves as a basis for our approach to order statistics.

2.1 Network Model

There are three different roles in the aggregation network. The *data generators (or data sources)* generate raw data that will be aggregated by the *aggregators*. The aggregators can form multiple tiers, where the top one is the *querier*. The querier can be different in different query sessions, but in one session we only consider one querier. We support multi-path aggregation, where a data generator can have multiple routes to the querier. Figure 1 illustrates the data flow of the distributed aggregation. We assume that a network-wide public key infrastructure exists. Source i can sign its value v_i with its private key sk_i. The signature then can be verified by other parties using the public key pk_i. The querier does not require knowledge of the network topology, but for secure verification, we do assume the querier knows the approximate network population size [12].

Since we target order statistics, the data should be from an ordered domain. It is possible to have ordered text strings as the raw data, but without loss of generality, we assume

[1]In practice, a threshold of x standard deviations is typically replaced by $1.4826x$ MADs, also known as the Hampel X84 measure [16].

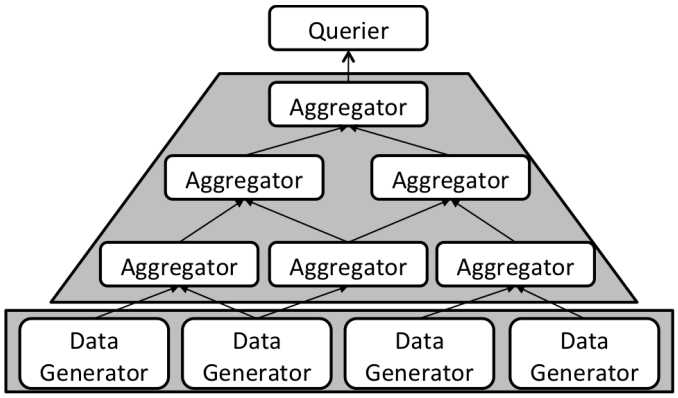

Figure 1: Data flow in distributed aggregation network

the sources generate numerical data, which can be integers or floating point values.

Note that the high breakdown point of median means that we need not impose constraints on the range of values that may be injected. By contrast, the aggregation protocols for non-robust statistics (e.g, averages) usually enforce range checking to confine the range of values in order to obtain a nonzero breakdown point. Such constraints not only limit the usability of the system, but are hard to enforce — untrusted aggregators are given the responsibility of performing range checks.

2.2 Threat Model

We consider three types of misbehavior: deflation, inflation, and pollution. The first two types are data manipulation during aggregation. The aggregators can omit messages, insert bogus messages, or modify the content of messages for arbitrary data forgery. These manipulation results in value *deflation* or *inflation* in the final aggregate. The third type of misbehavior is data *pollution* (sometime called a direct data injection attack [4]) performed by the compromised data sources. The two grey zones in Figure 1 indicate where the misbehavior may occur.

The attacker can also be a coalition of several aggregators and sources, who share private information and attempt to alter the final result collaboratively (*collusion attack*). The attacker's goal is to perform stealthy attacks [27], perturbing the aggregation result without being detected.

We consider computationally-bounded attackers, who are unable to break cryptographic functions such as forging signatures or finding collisions for hash functions in any reasonable timeframe.

2.3 Verifiable Counting using AM-FM Sketches

Before we discuss the problem of verifiable order statistics, we look at the preliminary problem of verifiable counting, which serves as a basis for our technique.

Garofalakis et al. [12] presented AM-FM Proof Sketches for verifiable counting in adversarial aggregator settings. The AM-FM sketch combines an FM sketch [10] with an Authentication Manifest (a set of cryptographically signed values) to secure the counting result. The protocol produces a public verifiable result of size $O(\ln n)$ without the need for trusted third parties. If x out of n sources satisfy a predicate (i.e., answer "YES" to a question), the protocol guarantees a bounded output x', $Pr[|x' - x| < \epsilon n] > 1 - \delta$ with space requirement $O(\frac{\log(2/\delta)}{\epsilon^2})$. Ideally, the protocol works for any ϵ and δ. In practice, extreme parameters result in an unreasonably large number of sketches, thus offering advantage to backhauling, which forwards all the source data to the queries. Based on our experiment (see Section 5), reasonable communication performance can be achieved with $\epsilon = 0.16$ and $\delta = 0.05$.

The FM algorithm digests a data set into a bitvector representation (FM sketch) as an estimator of the number of distinct elements in the set. The FM sketch is initialized to zeros, and a zero bit is turned on when an element is mapped to the zero bit. A deterministic mapping from elements to bit positions is constructed, such that elements are mapped to bth-bit with probability $1/2^b$. Let $lsb_0(\cdot)$ be the function returning the lowest-order 0-bit position of a bit vector (the LSB bit is indexed with 0). For example, $lsb_0(01010111) = 3$. Given a random variable X over $\{0,1\}^*$, $Pr[lsb_0(X) = b] = 1/2^b$. Therefore, the random variable can provide the desired probability distribution. In practice, the randomness can be modeled by cryptographic hash functions, such as MD5 or SHA-1.

It is shown in [10] that the FM sketch provides a good estimate of counting distinct elements by a single scan and logarithmic storage. Suppose we are interested in the size of a set C, a set contains the sources satisfying a predicate. $|C| \approx \frac{2^{lsb_0(\mathrm{S})}}{0.77351}$.

However, a single sketch provides an approximate count with high standard deviation. Fortunately, using multiple independent sketches in parallel decreases the deviation by logarithmic order. If K independent sketches are used and each element is mapped randomly into a sketch, the equation can be refined as:

$$|C| = K \frac{2^{\sum lsb_0(\mathrm{S}_k)/K}}{0.77351} \quad (1)$$

In the AM-FM scheme, the cryptographic signatures protect the FM sketch from one type of bit modification, i.e. flipping zero to one (inflation attack), because each 1-bit in the AM-FM sketch is accompanied by a "witness" value cryptographically signed by the party who flipped the bit to one. The witness value contains the bit position and/or other fields based on the application scenario. Since each signature is unforgeable and tractable, only signatures linked to correct bit positions and generated by privileged users can pass the verification.

To detect deflation on C, a complementary sketch is applied to count the complementary set $\bar{C}$. Since inflation on $\bar{C}$ is also impossible for a computationally bounded attacker, the deflation attack will decrease the approximate population, i.e. the summation of the estimating counts for C and $\bar{C}$. Therefore, assume a known population n, we can detect both

Table 1: Symbols and basic functions

Notation	Explanation
n	the size of the data set
ID_i	a unique identity of source i, $1 \leq i \leq n$
(pk_i, sk_i)	the public/private key pair of ID_i
$\mathbb{S} = \{S_k\}_{1 \leq k \leq K}$	a set of K sketches
$S_k[b]$	the b-th slot in the sketch S_k
$H(\cdot)$	a cryptographic hash function
$\text{sign}(H(msg), sk)$	sign msg by the private key sk
$\text{verify}(msg, sig, pk)$	verify sig on msg by the public key pk
$\text{lsb}_0(\{0,1\}^*)$	the lowest 0-bit position of a bit string
$\text{element}(\mathbb{S})$	the set of elements represented by $\mathbb{S}$
$\text{fm\_est}(bits_k, K)$	count distinct estimation by Eq. 1

inflation and deflation attacks on the aggregation protocol.

The verifiable counting algorithm provides a direct construction of a verifiable order statistic algorithm. We can construct a (fixed-bin-width) histogram structure where each bin is secured by the verifiable counting technique. However, this approach imposes $O(a \ln n)$ communication overhead, where a is the number of bins in the histogram.

2.4 Proof Sketches

Garofalakis et al. outlined a generalized template for a proof sketch: a proof sketch for an aggregation function should support a compact authentication manifest and deflation bounds [12]. The AM-FM sketch is a proof sketch specialized for count-based aggregation functions.

In the following sections, we give an abstract presentation of our (FM3 Proof Sketch) data structure, and its methods for construction. FM3 is a new example of a proof sketch, focusing on – but not limited to – the order statistic functions. In Section 4, we use the FM3 object for developing $O(\ln a \ln n)$ verifiable aggregation protocols.

3. THE FM3 PROOF SKETCH

Having explained the verifiable counting of YES/NO votes, we now focus our attention on order statistics over an ordered domain.

We introduce the FM3 Proof Sketch, a new synopsis data structure with sublinear space requirements and built-in functionality for manipulation detection. We show that the FM3 allows various approximate queries (quantiles, MADs, frequent items) with guaranteed error bounds. In the following sections, based on the FM3 and the approximate order statistic queries, we design and evaluate a secure in-network aggregation scheme for handling both misbehaving aggregators and data sources.

3.1 FM3 Overview

The FM3 Proof Sketch is based on the AM-FM sketch [12], with a small change that we will leverage extensively in our query algorithms. Rather than keeping a single witness for each bit turned on in an FM sketch, the FM3 Proof Sketch keeps the maximum and minimum values that have been mapped to the position. These extreme values per bit, accompanied with their signatures, are calculated during construction of the sketch. As in all FM sketch variants, this compares favorably with naive hash-based counting: the cost is reduced from $O(a)$ to $O(\ln a)$, where a is the number of distinct elements.

Our efficient solution comes from an observation on the nature of FM sketches. Consider two sets of values A and B, and their corresponding FM sketches S_A and S_B respectively. If $A \subseteq B$, then $(S_A \& S_B) = S_A$ – i.e., the 1-bits of S_A are a subset of the 1-bits of S_B. Therefore, instead of keeping just the FM bit array for each subset, the FM3 sketch stores two values, the maximum and minimum, corresponding to each bit. After computing the FM3 over all the data, it can be used to answer predicates of the form "how many values were larger than (smaller than) x" for any value x. This can be done by simply setting to 0 those bits in the FM sketch where the maximum (minimum) is not greater than (less than) x.

The FM3 sketch provides three types of methods: basic methods, verification methods, and query methods.

- The basic methods are used for sketch construction, such as initialization, adding a new element, and combining multiple sketches.
- The verification methods verify whether the witnesses have genuine signatures and are placed in correct positions. The verification functionality also checks if the sum of "larger than or equal to x" and "smaller than x" approximately equals the population size n. (Compared with the two complementary sketches used in AM-FM, here we have $<$ and $\geq$ complementing each other in a single sketch.)
- The query methods answer queries using the final sketches to compute approximate answers. They utilize the verification methods to ensure error bounds on the approximate answers. The details of the query algorithms will be discussed in the next section.

Formally, let $\mathbb{S}$ represents a set of K sketches, $S_k, 1 \leq k \leq K$. Each sketch has B bit positions, and each position contains at most two signed values corresponding to $S_k[b].max$ and $S_k[b].min$. $S_k[b]$ represents the b-th bit of S_k. Recall that $lsb_0(\cdot)$ is the function returning the lowest 0-bit position of a bitvector indexed from zero.

Table 1 summarizes the symbols and basic functions used in the following sections.

3.2 Basic Methods

There are four methods for FM3 construction and basic operation. *Init, Add, Aggregate, and Estimate.* We describe each of them by its algorithmic construction.

Init The Init method initializes a set of K sketches S_k, $1 \leq k \leq K$. Each sketch has B bit positions.

Add The Add method adds a new element v to the FM3. The ID_i is used in computing b and k to ensure the unique-

Method 1 Init

Require: input K and B
Ensure: output $\mathbb{S} = \{S_k\}_{1 \le k \le K}$. element($\mathbb{S}$) = $\emptyset$
  for $k = 1$ to K **do**
    for $b = 1$ to B **do**
      $S_k[b].max \leftarrow \emptyset$
      $S_k[b].min \leftarrow \emptyset$
    end for
  end for

Method 2 Add

Require: input $\mathbb{S} = \{S_k\}_{1 \le k \le K}$, v, pk, sk
Ensure: element($\mathbb{S}^{out}$) =element($\mathbb{S}$) $\cup \{v\}$.
  $k \leftarrow H(nonce||pk)$ mod K
  $b \leftarrow lsb_0(H(nonce||k||pk))$
  if !CheckSig($S_k[b].max.sig$)$\vee$!CheckSig($S_k[b].max.sig$) **then**
    delete failed signatures
  end if
  if $(S_k[b].max = \emptyset) \vee (v > S_k[b].max.v)$ **then**
    $S_k[b].max.v \leftarrow v$
    $S_k[b].max.id \leftarrow pk$
    $S_k[b].max.sig \leftarrow sign(H(nonce||k||pk||v), sk)$
  end if
  if $(S_k[b].min = \emptyset) \vee (v < S_k[b].min.v)$ **then**
    $S_k[b].min.v \leftarrow v$
    $S_k[b].min.id \leftarrow pk$
    $S_k[b].min.sig \leftarrow sign(H(nonce||k||pk||v), sk)$
  end if

Method 3 Aggregate

Require: input L sketch sets, $\mathbb{S}^1$,..., $\mathbb{S}^L$
Ensure: element($\mathbb{S}^{out}$) = $\bigcup_l$ element($\mathbb{S}^l$)
  for all b and k **do**
    $i = \arg\max_l S_k^l[b].max.v$
    $j = \arg\min_l S_k^l[b].min.v$
    $S_k^{out}[b].max \leftarrow S_k^i[b].max$
    $S_k^{out}[b].min \leftarrow S_k^j[b].min$
  end for

ness of the hash input. Without loss of generality, here we use pk_i as the unique ID_i. Note that the value v_i is only used when storing signed minima and maxima; the FM bits are set to 1 based on node identities, not data values.

Aggregate Suppose the incoming sets are $\mathbb{S}^l$, $1 \le l \le L$. The Aggregate method maintains the invariant that the min/max values associated with the b-th bit position are the smallest/largest witnesses respectively seen so far.

Estimate Given a value v, the Estimate method answers the number of elements smaller than v, and number of elements larger or equal to v.

3.3 Verification Methods

There are two methods for verifying the sketch content. *CheckSig* validates a signature and its position in the sketch. *CheckSum* detects inflation and deflation attacks, where the

Method 4 Estimate

Require: input v, a sketch set $\mathbb{S}$
Ensure: two estimate counts $c^>$ and $c^\le$
  $\forall k, b$, $bits_k^> \leftarrow \{0\}^B$ and $bits^\le \leftarrow \{0\}^B$
  for all k, b **do**
    if $v > S_k[b].min.v$ **then**
      $array_k^>[b] \leftarrow 1$
    end if
    if $v \le S_k[b].max.v$ **then**
      $array_k^\le[b] \leftarrow 1$
    end if
  end for
  $c^> \leftarrow$fm_est($bits^>$)
  $c^\le \leftarrow$ fm_est($bits^\le$)

Method 5 CheckSig

Require: input sig and the position indicators k and b
  if $k \neq H(nonce||pk)$ mod K **then**
    return FAIL
  end if
  if $b \neq lsb_0(H(nonce||k||pk))$ **then**
    return FAIL
  end if
  get corresponded pk and v for the signed message
  return verify($H(nonce||k||pk||v), sig, pk$)

Method 6 CheckSum

Require: input $\mathbb{S}$, v
Ensure: decide if the summation regarding v is close to n
  $(c^>, c^\le) \leftarrow$ Estimate($v, \mathbb{S}$)
  return $(1 - \epsilon_c)n \le (c^> + c^\le)$

min/max witnesses are dropped or replaced by the other elements that do not have the min/max values.

CheckSig Given a signature and its position k and b in the FM3, the CheckSig method verifies the signature and also checks if it is placed in the correct position.

CheckSum Given a value v, the CheckSum method computes the FM sketches for predicate "smaller than v" and "larger than or equal to v". Finally, by equation 1 it checks if

$$(1 - \epsilon_c)n \le c^> + c^\le \quad (2)$$

3.4 Query Methods

In this subsection, we show that the FM3 allows various approximate queries, including ranks, quantiles, MADs, and frequent items. In addition, we derive error bounds for each algorithm in the adversarial settings.

For convenience, we define the following notions. Let $V = \{v_i\}$ be the *observed data set*, which may differ from the *true data set* if a fraction of the data is polluted. Arranged in non-decreasing order, we have $v_{s_1} \le v_{s_2} \le ... \le v_{s_n}$, where $(s_1, s_2, ..., s_n)$ is a permutation of $\{i\}_{1 \le i \le n}$. Let $R(v)$ represent the rank of v in set V, i.e. $R(v_{s_i}) = i$. In addition,

Method 7 RankQuery

Require: input v, $\mathbb{S}$
Ensure: return an approximate rank r with verified/failed

$(r \leftarrow c^{>}, c^{\leq}) \leftarrow$ Estimate$(v, \mathbb{S})$
if CheckSum$(v, \mathbb{S})$ **then**
  return $(r$, verified$)$
else
  return $(r$, failed$)$
end if

we define $v_{s_0} \leftarrow -\infty$ and $v_{s_{n+1}} \leftarrow \infty$. We know that:

$$\forall v \in \mathbb{R}, \exists j \text{ such that } v_{s_{j-1}} < v \leq v_{s_j}$$

so we extend the definition of $R(\cdot)$ to define $R(v) = j$. We can interpret $R(v)$ as the number of elements in V that are smaller than v, so it is possible that $v \notin V$. Since $R(\cdot)$ is the observed rank in the "polluted set" V, if there are ρn compromised sources and $\hat{R}(\cdot)$ represents the true rank in the unpolluted set $\hat{V} = \{\hat{v}_i\}$, then

$$|\hat{R}(v) - R(v)| < \rho n \tag{3}$$

Therefore, $R(\cdot)$ is bounded when the fraction of misbehaving sources is less than a given value.

We quantify the error caused by compromised aggregators that perform inflation and deflation attacks on the FM3 Proof Sketch. The effect of misbehaving data generators (attackers who own valid secret keys and can inject bogus values) is bounded by Equation 3. For clarity, in the following we derive the probabilistic bound for $R(\cdot)$, but it is easy to extend it to similarly bound the error of $\hat{R}(\cdot)$. In general, an ϵ-approximation over a polluted set becomes an $(\epsilon + \rho)$-approximation over the original unpolluted set, where ρ is the compromised proportion of the source set.

Before performing any query methods, we need to validate each signature via *checkSig*.

Secure Rank Since the rank of a value v_i in a set V represents the number of values less than v_i, we can easily complete a verifiable rank query by performing the *Estimate Method* with the *CheckSum Method*. RankQuery is outlined in Method 7.

THEOREM 3.1. *Assume $O(\frac{\log(2/\delta_r)}{\epsilon_r^2})$ sketches are used. If a value v satisfies equation 2, then v also satisfies*

$$Pr[|r(v) - R(v)| < \epsilon_r n] > 1 - \delta_r \tag{4}$$

with $\epsilon_r = 2\epsilon_c$ in adversarial environments [12]. That is, for the data point v, the difference between its observed rank $R(v)$ and the estimate rank $r(v)$ is probabilistically bounded in the present of malicious aggregators.

Secure Median and Quantile Method 8 demonstrates how to compute median and other quantiles from S. The idea is that given a FM3 sketch, we can perform binary search over the value domain until we find a value v whose

Method 8 QuantileQuery

Require: input $\mathbb{S}$, quantile q, ϕ
Ensure: return an approximate value v with verified/failed

repeat
  get v by binary search over the value domain
  $(r, ans) \leftarrow$ RankQuery$(v, \mathbb{S})$
until $(|r - qn| \leq \phi n \wedge ans =$ verified$) \vee$ binary search finished
return (v, ans)

rank is close to qn, where q is the quantile we are interested in and $|r(v) - qn| \leq \phi n$ for a constant ϕ. Median is a special case with $q = 0.5$.

THEOREM 3.2. *Given a sketch set satisfying Equation 4, we can construct a function that answers value v to a quantile query q, such that*

$$Pr\left[|R(v) - qn| < \epsilon_q n\right] \geq 1 - \delta_q \tag{5}$$

PROOF. Our quantile query algorithm ensures $|r(v) - qn| \leq \phi n$. Set $\epsilon_q = \phi + \epsilon_r$, and $\delta_q = \delta_r$.

$$\begin{aligned} & Pr\left[|R(v) - qn| < \epsilon_q n\right] \\ \geq & Pr\left[|R(v) - r(v)| + |r(v) - qn| < \epsilon_q n\right] \\ \geq & Pr\left[|R(v) - r(v)| < (\epsilon_q - \phi) n\right] \\ \geq & 1 - \delta_r \\ = & 1 - \delta_q \end{aligned}$$

□

Secure MAD The median is a useful robust measure of the "center" of a distribution. The MAD is a robust measure of the "spread" of the distribution, analogous to the standard deviation. Here we show how to retrieve the MAD from a FM3 S.

The *median absolute deviation* (MAD) of a set of data points $X = \{x_i\}$ is defined as

$$MAD = median(|X - median(X)|)$$

Similar to computing the median, computing the exact MAD requires the involvement of all the original data points.

We first analyze the error bound with respect to the *Median Absolute Deviation of Approximate Median* (MADAM), which is a modified notion of the MAD. We also provide an informal breakdown point analysis showing the difficulty to perturb the MADAM's result away from the true MAD.

$$MADAM = median(|X - median_\epsilon(X)|) \tag{6}$$

The problem of finding the MADAM can be formulated as follows.

Given a set of data points $X = \{x_i\}$ from the domain $(-\infty, \infty)$ and their ϵ-approximate median $x_m = median_\epsilon(X)$,

Method 9 MADQuery

Require: input $\mathbb{S}$, v
Ensure: return an approximate MAD value v with verified/failed
  $v_m = \text{QuantileQuery}(0.5, \mathbb{S})$
  repeat
    get y by binary search over Y domain
    $(r^+, ans^+) \leftarrow \text{RankQuery}(v_m + y, \mathbb{S})$
    $(r^-, ans^-) \leftarrow \text{RankQuery}(v_m - y, \mathbb{S})$
  until $(|r^+ - r^- - 0.5n| \leq \epsilon_y n \wedge ans^+ \wedge ans^-) \vee$ binary search finished
  return $(y, ans^+ \wedge ans^-)$

we can define the *folded set* of points $\{y_i | y_i = |x_i - x_m|\}$ in the domain $[0, \infty)$.

To compute the MADAM, first consider the relation between $R_x(\cdot)$ and $R_y(\cdot)$, functions that return the observed rank in the X and Y domains respectively. Recall that the rank of a value v equals to the number of data points that are smaller than v. We observe that the data points smaller than v in Y domain, i.e., those points fall into the interval $[0, v]$, are all mapped from points in the interval $[x_m - v, x_m + v]$ in X domain. Therefore, $R_y(v)$ equals the number of points between $x_m - v$ and $x_m + v$ in X domain.

$$R_y(v) = R_x(x_m + v) - R_x(x_m - v) \tag{7}$$

In addition, $r_x(\cdot)$ and $r_y(\cdot)$ are functions that return the approximate rank in X and Y domains, respectively. We define

$$r_y(v) = r_x(x_m + v) - r_x(x_m - v) \tag{8}$$

Hence *the MADAM of X is the median of the folded set Y*. We perform QuantileQuery (Method 8) on the folded set Y for the median, whose value is the approximate MADAM of X.

THEOREM 3.3. *Given an approximate ranking function $r_x(\cdot)$ in domain X with error bound $Pr[|r_x(v) - R_x(v)| < \epsilon_x n] > 1 - \delta_x$, we can construct an approximate MADAM function such that $Pr[|r_y(v) - R_y(v)| < \epsilon_y n] > 1 - \delta_y$, where $\epsilon_y = 2\epsilon_x$ and $\delta_y = 2\delta_x$.*

PROOF. Error bound analysis:

$$\begin{aligned}
&Pr[|r_y(v) - R_y(v)| < \epsilon_y n] \\
=&Pr[|(r_x(x_m + v) - r_x(x_m - v)) \\
&- (R_x(x_m + v) - R_x(x_m - v))| < \epsilon_y n] \\
=&Pr[|(r_x(x_m + v) - R_x(x_m + v)) \\
&- (r_x(x_m - v) - R_x(x_m - v))| < \epsilon_y n] \\
\geq&Pr[|r_x(x_m - v) - R(x_m - v)| \\
&+ |r_x(x_m + v) - R_x(x_m + v)| < 2\epsilon_x n] \\
\geq&Pr[|r_x(x_m - v) - R_x(x_m - v)| < \epsilon_x n] \\
&* Pr[|r_x(x_m + v) - R_x(x_m + v)| < \epsilon_x n] \\
\geq&(1 - \delta_x)^2 \\
\approx&1 - 2\delta_x \\
=&1 - \delta_y
\end{aligned}$$

Method 10 FrequentItemQuery

Require: input $\mathbb{S}$, ϕ
Ensure: return a frequent item set F
  $F \leftarrow \emptyset$
  for all witness values v_w in $\mathbb{S}$ **do**
    $(r, ans) \leftarrow \text{RankQuery}(v_w, \mathbb{S})$
    $(r^+, ans^+) \leftarrow \text{RankQuery}(v_w + 1, \mathbb{S})$
    if $r^+ - r > \phi n \wedge ans \wedge ans^+$ **then**
      $F \leftarrow F \cup \{v_w\}$
    end if
  end for

The equation tells us the inaccuracy of the approximate MADAM algorithm is bounded by the inaccuracy of the approximate median algorithm. □

In addition to the probabilistic guarantee, we evaluate the approximate MADAM by breakdown point analysis. Computing MAD based on the approximate median x_m, where $(0.5 - \epsilon_q)n < R_x(x_m) < (0.5 + \epsilon_q)n$, is no worse than computing MAD over a set of ϵn compromised data sources. Therefore, we know that our approximate result of MAD is still robust as long as $\epsilon_q < 50\%$ (the breakdown point of MAD).

Securely finding frequent items The problem of finding frequent items in streams or distributed data sets is returning a set of items $F = \{x_i \mid f(x_i) > \phi n\}$. Observe that if x is a frequent item, $R(x + 1) - R(x) > \phi n$. Without loss of generality, we assume elements are integers. Finding the exact set of frequent items, similar to finding the exact median, requires linear amount of space [7]. Therefore, researchers focus on solving ϕ'-approximate frequent items problem, where $F = \{x_i | f(x_i) > (\phi - \phi')n\}$ and $\bar{F} = \{x_i | f(x_i) < \phi n\}$. Here we define a new notion of δ-probabilistic ϵ-approximate frequent items problem, which requests a set of approximate frequent items F_a over a set of elements $\{x_i\}$: $\{x_i \mid P[f(x_i) > \epsilon_f n] > 1 - \delta_f\}$, given constants ϵ_f and δ_f.

The algorithm finds elements x, such that $r(x+1) - r(x) > \phi n$.

THEOREM 3.4. *If element x satisfying $r(x + 1) - r(x) > \phi n$, and the sketches satisfying Equation 4, then $P[f(x) > \epsilon_f n] > 1 - \delta_f$, where $\epsilon_f = \phi - 2\epsilon_r$ and $\delta_f = 2\delta_r$.*

PROOF.

$$\begin{aligned}
&Pr[f(x) > \epsilon_f n] \\
=&Pr[R(x + 1) - R(x) > (\phi - 2\epsilon_r)n] \\
\geq&Pr[r(x + 1) - r(x) > \phi n] \\
&* Pr[|R(x + 1) - r(x + 1)| < \epsilon_r n] \\
&* Pr[|R(x) - r(x)| < \epsilon_r n] \\
\geq&1 * (1 - \delta_r)^2 \\
\approx&1 - 2\delta_r \\
=&1 - \delta_f
\end{aligned}$$

□

4. VERIFIABLE ORDER STATISTICS FOR SECURE AGGREGATION

Having introduced the FM3 structure and the approximate order statistic queries, we now present a secure in-network aggregation protocol for order statistics with three features: 1) single communication pass, 2) logarithmic communication, and 3) approximate answers with guaranteed error bound. The first two features come from the nature of the FM3, and the third feature is proved in Section 3.

We evaluate the order-statistic aggregation protocol in Section 5. We study its behavior in the absence of attack, and quantify the degree of security during different attack scenarios.

The distributed aggregation protocol produces a public verifiable result with size $O(\ln a \ln n)$. Given a quantile q, our protocol guarantees an estimate value v, where $Pr[|R(v) - qn| < \epsilon n] > 1 - \delta$, and $R(v)$ is the observed rank of v in the source data set.

4.1 Problem Statement

Each data source i has its own public/private key pair $\{pk_i, sk_i\}$ and a unique ID_i, $1 \leq i \leq n$. (For example, we can use the public keys as the unique IDs.) We assume the existence of PKI and key management schemes handling key issuing, revocation, authentication, etc. In each session, i will generate a value v_i, which either answers a current query, e.g. asking each monitoring host for the amount of bad packets coming from a suspicious IP address, or emits spontaneously at scheduled time, e.g. the hosts are configured to report daily bandwidth usage at 12 am. Note that these observed values may have already been polluted, so if the true answer to the query is $\hat{v}_i$, it is possible $\hat{v}_i \neq v_i$ for some i.

Protocol Description There are four phases in each query session: Request Dissemination, Data Initialization, Aggregation, and Verifiable Query.

Request Dissemination The querier disseminates a request Q to n data sources. The message body should contain a description qd and other necessary parameters by which the data sources can initialize their sketches locally. The querier assigns an unique taskID for this particular aggregation task. $m = \{\text{qd, taskID, params}\}$. The querier, who has a public/private key pair (pk_q, sk_p), signs the message:

$$Q = \{m, sign(H(m), sk_q)\}$$

We assume the request is transmitted via a reliable protocol. A packet loss is the same as a inflation attack – from the querier's view. Therefore, a reliable routing protocol ensures a high success rate in a benign environment.

This phase can be omitted when the sources report values spontaneously at scheduled time. The data sources use a previous or default setting as their parameter, and use a synchronized counter as the nonce. The counter is increased after each session.

Data Initialization Each data generator i performs the *Init Method* and *Add Method* to generate a sketch set containing a value v_i.

Aggregation Aggregation requires no secret information. Any party who receives multiple sets of sketches can be the aggregator performing the *Aggregate Method*.

Due to the untrusted aggregators, our protocol does not require the aggregator to verify the signatures and the correctness of the bit position. The entire verification process can be done by the querier. However, it is more bandwidth-efficient if the benign aggregators detect incorrect signatures at an earlier stage.

The aggregation process is duplicate-insensitive. The result remains the same even if a sketch set is aggregated more than once. This property enables a more flexible aggregation topology. Rather than an aggregation tree, where each source has only one route to the root, it allows multi-path aggregation, where each source can have multiple routes to the root. The multi-path topology is more robust and easier to construct than a tree-based topology. [21, 23]

Verifiable Query After the querier receives his aggregation result, a sketch set $\mathbb{S}$, he verifies the signatures and takes out all invalid signatures by the *CheckSig Method*. He can then perform order statistic queries, such as quantiles and MADs, on the sketch set.

The verifiability aspects of the FM3 can identify aggregation-level misbehavior. Source-level misbehavior (i.e., poisoning) causes different degrees of undetectable damage on different statistics. As we pointed out in Section 1.1, our aggregation functions are robust against source-level misbehavior: unless the adversary's attack rate exceeds the breakdown point of the estimator, their ability to perturb the result is constrained by the values of unpoisoned data. We provide experimental results in Section 5.

5. EVALUATION

To evaluate the efficiency and security properties of the FM3 Proof Sketch, we implemented the FM3 and its query algorithms in C++.

5.1 Setup

In order to study the practical use of FM3 sketches, we need to tune some parameters: the number of bits per sketch, the number of sketches to use, and the ϵ and δ settings that we can realistically achieve. To explore these parameters, we ran a series of experiments on both real and synthetic data.

Data Source The synthetic data set contains 100,000 randomly-generated values, uniformly distributed in [0, 99,999]. The real data from Intel Berkeley Research Lab contains 98,884 temperature values from three adjacent sensor motes. The temperature values form a normal distribution and are in between $16°C$ and $50°C$.

Parameters We study the parameter settings for sketches and error bounds empirically, using both synthetic and real

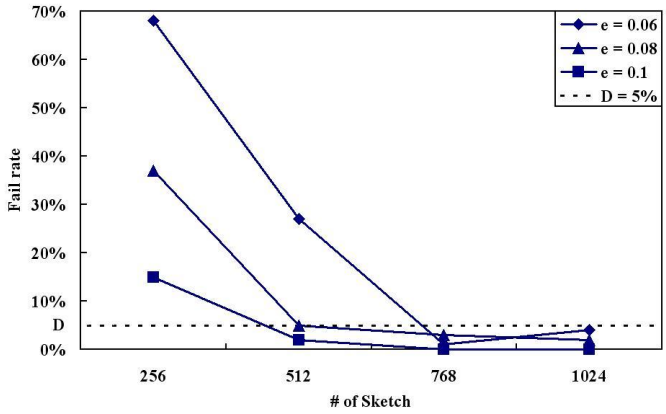

Figure 2: The relation between the number of sketches and the false alarm rate

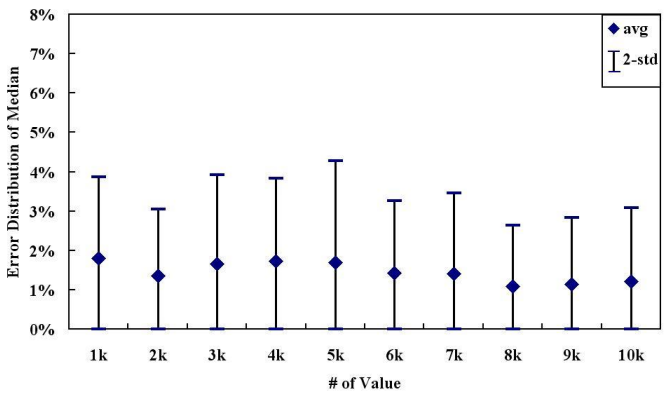

Figure 3: The error distribution of medians

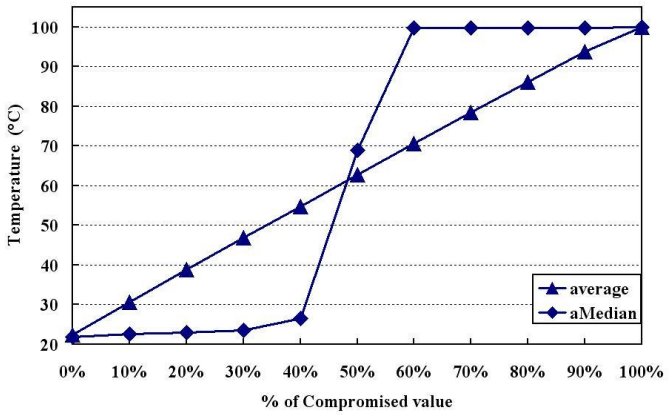

Figure 4: Approximate medians and observed means under a data pollution attack

data. We show the mean value of the estimators, along with variance, based on multiple experiments for each setting of the parameters. For synthetic data, we randomize the generation function (i.e. a pseudorandom number generator) across experiments. For real data, we use a statistical resampling technique known as bootstrapping to simulate the effect of running many independent experiments, and allow the derivation of variance estimates [9].

Unless specified, all experiments use 512 sketches, $\epsilon_c = 0.08$, and $\delta = 0.05$ for median estimators. We run 100 times of simulation for each experiment.

Figure 2 shows the relation between the number of sketches and the false positive rate for verification. $e = \epsilon_c$ is the error factor in Equation 2. In this experiment, we consider that a sketch "fails" if there exists a witness value which fails the *CheckSum* test. The failure testing is conservative because the FM3 verifiable order statistic queries only require a small subset of values passing the *CheckSum* test. As shown in Figure 2, 512 sketches can achieve a 95% correct rate, i.e. with only 5% false alarms.

5.2 Security evaluation

In Section 3, we have shown the error bounds of approximate medians in misbehaving aggregator settings. In this experiment, we show how many sources can perturb an aggregated order statistic by a significant amount.

First we show that the probabilistic error is small and irrelevant to the sizes of population in a benign environment. The second experiment compares the robustness of medians and means by varying the proportion of polluted sources.

No attack: control group Figure 3 shows the error distribution of medians in a benign environment. The error $|r(v) - R(v)|/n$ is kept less than 4% as the population goes from 10,000 to 100,000. The result shows that, in a benign environment, the FM3 performs much better than the worst case scenario with error bound $\epsilon_q = 0.16$ (by Eq. 5).

Source-level misbehavior For the source-level misbehavior, we consider an optimally aggressive attacker who moves values from one extreme of the order to the other, e.g. from the bottom k to the top k. In this experiment we use the sensor temperature data. The attacker raises the bottom k values to 100. Figure 4 shows the errors in the value domain. Median performs well before approaching the breakdown point. There is a huge jump when the proportion of the compromised sources goes beyond 50%. On the other hand, the average gets worse in a stable pace. Ideally, averages have $1/n$ breakdown point. However, since the values are limited within range [0, 100], the average is bounded by the value range. However, the range check in practice may require a run-time enforcement technique, which may introduce security vulnerabilities as well.

5.3 Performance evaluation

We evaluate the time and space usage of FM3 Proof Sketch using Intel Core 2 Duo 2GHz workstation with 2G memory. We tested our default setting with different number of values, and tested different number of sketches aggregating 100,000 randomly-generated data. We summarizes the results in Table 2 and Table 3.

Table 2: Performance Comparison based on the number of values

Number of values	1k	2k	3k	4k	5k	6k	7k	8k	9k	10k
Stored witness data	4730	5760	6355	6785	7115	7375	7610	7810	7975	8125
Compression ratio	2.11	3.47	4.72	5.9	7.03	8.14	9.2	10.24	11.29	12.31
Total time (s)	1.6	3.1	4.6	6	7.6	9	10.6	12.3	14	15.2

Table 3: Performance comparison based on the number of sketches

Number of sketches	128	256	512
Stored witness data	2540	4575	8125
Compression ratio	39.37	21.86	12.31
Total time (s)	6	9.2	15.2

6. DISCUSSION

In this section, we discuss a possible generalization of our problem space. We also propose several alternatives for deflation detection instead of knowing the universe size. Finally, we compare our work to previous work.

6.1 Generalization

Our protocol supports a more general query class of "exclusive multiple choice" counting queries, where each source says "yes" to one and only one out of a set of a queries.

We index these queries from q_1 to q_n. Similarly, in each bit position, we keep two signatures that have the highest and lowest *indexes* of the query q_i. (In the previous construction, we keep two signatures that have the highest and lowest *values*.) To compute the count of $\bigcup_{i\le k\le j} q_k$, the administrator retrieves the sketches $S(j)$ for $\bigcup_{1\le k\le j} q_k$ and $S(i)$ for $\bigcup_{1\le k<i} q_k$. $|\bigcup_{i\le k\le j} q_k| = |S(j)| - |S(i)|$. The reason we cannot compute $|q_i|$ directly is when a q_i event is aggregated into the sketches, it may be masked by other queries that have higher/lower index than the q_i has. We can only retrieve the accumulated counting of those patterns with index larger or smaller than i. Compared with the naive solution that obtains each counting separately, the accumulative solution requires only $O(\ln a)$ times of space instead of $O(a)$.

6.2 Universe Size

The verifier requires the knowledge of universe size to detect deflation. It is possible to have other ground truth as the basis of detection. For example, a message in a high-density broadcasting network often has many duplication. Attackers have to find out all the clones in order to terminate the message.

6.3 Previous Work

Earlier work addresses the secure aggregation problem with one of two general approaches. The first class consists of cryptographic solutions [4, 11, 22, 27, 33] that focus on intended attacks. The cryptographic schemes either assume a rigid aggregation topology in the network, or are specialized to count and average functions. The other class of solutions involve resilient network or protocol designs [23, 26] for mitigating unintended failures. Though these schemes improve resiliency against link/node failures, they are insecure against malicious aggregators.

In addition to compromising data aggregators, attackers can perturb results by pollution attacks that provide bad data at the input to the aggregation process. Outlier detection [19, 29,30] can help detect the abnormal data sources. However, outlier detection schemes rely on trusted intermediators or servers to detect the anomaly.

Wagner [31] quantified how much damage an attacker can cause actively under various statistical models, and suggested the use of aggregation functions from robust statistics to mitigate the impact of this class of attacks.

Approximate query processing improves communication or space efficiency by giving up a certain degree of accuracy [1–3,5]. Several approximate counting algorithms are proposed [12, 20, 24]. Of particular note in the context of secure distributed aggregation, Garofalakis et al. [12] proposed Proof Sketches, a one-pass verifiable counting scheme based on FM sketches [10]. Proof Sketches provide a direct solution for order statistic queries. However, the solution, which counts elements that fall in different value ranges, requires a prior knowledge of the data distribution to determine the value ranges for a better estimation, because the error is bounded by the maximum number of values that fall into the ranges. Moreover, the solution requires linear additional space regarding the number of value ranges.

Researchers have also worked on summarizing approximate order statistics over data streams [8, 13, 15, 25] and distributed data sets [6, 14]. In particular, Greenwald and Khanna [14] have proposed a one-pass algorithm for tracking quantiles in distributed environments. Their algorithm provides ϵN bound with only sub-linear communication, but to date there is no extension of that work to an adversarial context.

7. CONCLUSION

In this paper, we present the FM3 Proof Sketch, an efficient distributed aggregation protocol that answers a variety of approximate queries with guaranteed error bounds, while remaining secure against misbehaving aggregators, and robust to misbehaving data sources. Our protocol only requires a single communication pass and $O(\ln a \ln n)$ communication. To our knowledge, this is the first efficient protocol that can handle both these types of attacks.

8. REFERENCES

[1] B. Arai, G. Das, D. Gunopulos, and V. Kalogeraki. Efficient approximate query processing in peer-to-peer networks. *IEEE Transactions on Knowledge and Data Engineering*, 19(7):919–933, 2007.

[2] B. Babcock, S. Chaudhuri, and G. Das. Dynamic sample selection for approximate query processing. In *SIGMOD '03*, pages 539–550, New York, NY, USA, 2003. ACM.

[3] K. Chakrabarti, M. Garofalakis, R. Rastogi, and K. Shim. Approximate query processing using wavelets. *The VLDB Journal*, 10(2-3):199–223, 2001.

[4] H. Chan, A. Perrig, and D. X. Song. Secure hierarchical in-network aggregation in sensor networks. In *ACM Conference on Computer and Communications Security*, pages 278–287, 2006.

[5] G. Cormode and M. N. Garofalakis. Sketching streams through the net: Distributed approximate query tracking. In *VLDB*, pages 13–24, 2005.

[6] G. Cormode, M. N. Garofalakis, S. Muthukrishnan, and R. Rastogi. Holistic aggregates in a networked world: Distributed tracking of approximate quantiles. In *SIGMOD Conference*, pages 25–36, 2005.

[7] G. Cormode and M. Hadjieleftheriou. Finding frequent items in data streams. *Proc. VLDB Endow.*, 1(2):1530–1541, 2008.

[8] G. Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. In *LATIN*, pages 29–38, 2004.

[9] A. C. Davison and D. V. Hinkley. *Bootstrap Methods and Their Application*. Cambridge University Press, October 1997.

[10] P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci.*, 31(2):182–209, 1985.

[11] K. B. Frikken and I. Joseph A. Dougherty. An efficient integrity-preserving scheme for hierarchical sensor aggregation. In *WiSec '08*, pages 68–76, New York, NY, USA, 2008. ACM.

[12] M. N. Garofalakis, J. M. Hellerstein, and P. Maniatis. Proof sketches: Verifiable in-network aggregation. In *ICDE*, pages 996–1005, 2007.

[13] M. Greenwald and S. Khanna. Space-efficient online computation of quantile summaries. In *SIGMOD Conference*, pages 58–66, 2001.

[14] M. Greenwald and S. Khanna. Power-conserving computation of order-statistics over sensor networks. In *PODS*, pages 275–285, 2004.

[15] S. Guha and A. McGregor. Approximate quantiles and the order of the stream. In *PODS*, pages 273–279, 2006.

[16] F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel. *Robust Statistics – The Approach Based on Inuence Functions*. Wiley, 1986.

[17] P. Huber. Robust statistics, 1981.

[18] 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.

[19] Y. Kotidis, V. Vassalos, A. Deligiannakis, V. Stoumpos, and A. Delis. Robust management of outliers in sensor network aggregate queries. In *MobiDE '07*, pages 17–24. ACM, 2007.

[20] M. G. Kuhn. Probabilistic counting of large digital signature collections. In *SSYM'00: Proceedings of the 9th conference on USENIX Security Symposium*, pages 6–6. USENIX Association, 2000.

[21] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: A tiny aggregation service for ad-hoc sensor networks. In *OSDI*, 2002.

[22] A. Mahimkar and T. S. Rappaport. SecureDAV: a secure data aggregation and verification protocol for sensor networks. *Global Telecommunications Conference, 2004. GLOBECOM '04. IEEE*, 4:2175–2179 Vol.4, 29 Nov.-3 Dec. 2004.

[23] A. Manjhi, S. Nath, and P. B. Gibbons. Tributaries and deltas: Efficient and robust aggregation in sensor network streams. In *SIGMOD Conference*, pages 287–298, 2005.

[24] G. S. Manku and R. Motwani. Approximate frequency counts over data streams. In *VLDB*, pages 346–357, 2002.

[25] G. S. Manku, S. Rajagopalan, and B. G. Lindsay. Approximate medians and other quantiles in one pass and with limited memory. In *SIGMOD Conference*, pages 426–435, 1998.

[26] S. Nath, P. B. Gibbons, S. Seshan, and Z. R. Anderson. Synopsis diffusion for robust aggregation in sensor networks. *TOSN*, 4(2), 2008.

[27] B. Przydatek, D. X. Song, and A. Perrig. SIA: secure information aggregation in sensor networks. In *SenSys*, pages 255–265, 2003.

[28] P. J. Rousseeuw and C. Croux. Alternatives to the median absolute deviation. *Journal of the American Statistical Association*, (88), 1993.

[29] B. Sheng, Q. Li, W. Mao, and W. Jin. Outlier detection in sensor networks. In *MobiHoc '07*, pages 219–228. ACM, 2007.

[30] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos. Online outlier detection in sensor data using non-parametric models. In *VLDB '06*, pages 187–198. VLDB Endowment, 2006.

[31] D. Wagner. Resilient aggregation in sensor networks. In *SASN*, pages 78–87, 2004.

[32] P. Yalagandula and M. Dahlin. A scalable distributed information management system. In *SIGCOMM*, pages 379–390, 2004.

[33] Y. Yang, X. Wang, S. Zhu, and G. Cao. SDAP: a secure hop-by-hop data aggregation protocol for sensor networks. In *MobiHoc*, pages 356–367, 2006.