Privacy-Preserving Data Aggregation with Probabilistic Range Validation

Dekker, F.W.; Erkin, Zekeriya

DOI
10.1007/978-3-030-78375-4_4

Publication date
2021

Document Version
Final published version

Published in
Applied Cryptography and Network Security - 19th International Conference, ACNS 2021, Proceedings

Privacy-Preserving Data Aggregation with Probabilistic Range Validation

F. W. Dekker(✉) and Zekeriya Erkin

Cyber Security Group, Delft University of Technology, Delft, The Netherlands
{f.w.dekker,z.erkin}@tudelft.nl

Abstract. Privacy-preserving data aggregation protocols have been researched widely, but usually cannot guarantee correctness of the aggregate if users are malicious. These protocols can be extended with zero-knowledge proofs and commitments to work in the malicious model, but this incurs a significant computational cost on the end users, making adoption of these protocols less likely.

We propose a privacy-preserving data aggregation protocol for calculating the sum of user inputs. Our protocol gives the aggregator confidence that all inputs are within a desired range. Instead of zero-knowledge proofs, our protocol relies on a probabilistic hypergraph-based detection algorithm with which the aggregator can quickly pinpoint malicious users. Furthermore, our protocol is robust to user dropouts and, apart from the setup phase, it is non-interactive.

Keywords: Privacy · Data aggregation · Applied cryptography · Hypergraphs

1 Introduction

Data aggregation gives us many valuable insights into the real world in the form of machine learning [1], participatory sensing [2], software telemetry [3,4], and smart metering [5]. Although the usefulness of these methods depends on the amount of available data, privacy concerns make users reluctant to share their sensitive data with a third party [6,7]. This poses a significant threat to the viability of large-scale data analysis.

To overcome this problem, privacy-preserving data aggregation (PDA) protocols have been proposed which allow an aggregator to calculate statistics on privacy-sensitive data without being able to determine private values. There are various ways to achieve this. For example, several proposals use techniques such as homomorphic encryption [6,8] and secret sharing [9,10] to guarantee that user contributions cannot be decrypted unless they have been aggregated. Other proposals use differential privacy [11–13] to ensure that the connection between the observed value and the actual value is perturbed. Either way, PDA protocols provide the same expressiveness as non-PDA protocols, but without sacrificing user privacy. These guarantees usually come at the cost of increased computational complexity, increased bandwidth usage, or decreased accuracy.

K. Sako and N. O. Tippenhauer (Eds.): ACNS 2021, LNCS 12727, pp. 79–98, 2021.
https://doi.org/10.1007/978-3-030-78375-4_4

A shortcoming of many existing proposals is that they assume that all users are honest-but-curious, for example as in [8,14,15]. As a result, these proposals cannot be used to defend against dishonest users that want to invalidate the aggregate or nudge it in their favour. This means that dishonest users could tamper with their smart meter to reduce their reported electricity consumption [16] or inject false data to increase their score in a ranking system [17]. The aggregator would have been able to detect these attacks by looking at the users' private values, but the privacy-preserving properties of the PDA protocol prevent this.

Transitioning from the honest-but-curious model to the malicious model can be achieved using zero-knowledge proofs and commitments, as suggested in proposals such as [8,13,14]. In particular, range proofs [18] can be used to prove in zero knowledge that a committed value is within a given range. However, range proofs—and zero-knowledge proofs in general—often either require a trusted setup or a significant amount of resources from the user [19]. This makes these approaches unappealing or even infeasible for resource-constrained users.

In this paper, we present an efficient PDA protocol for finding the sum of all private user values at a regular interval. The protocol lets an aggregator probabilistically identify private values that are not within a desired range without the need for zero-knowledge proofs. First, the aggregator divides all users into multiple overlapping groups such that every user is in a unique set of groups. Then, in each interval, each user sends their encrypted values to the aggregator, who determines the sum of private values per group. Finally, the aggregator pinpoints malicious users by looking at the intersection of groups that violate the range. By memorising which groups have out-of-range aggregates, the aggregator can combine detections from different rounds to further enhance its detection rate.

Our protocol boasts several important properties. Firstly, the scheme can be configured to customise the balance between privacy, complexity, and detection rate. For example, one can increase the work the aggregator needs to perform per round to increase the protocol's resistance to user collusions. Secondly, our protocol does not require a trusted setup and is non-interactive apart from the registration phase: Users simply send their encrypted values to the aggregator, who then aggregates and validates asynchronously. Thirdly, our protocol is an efficient solution for aggregators relying on resource-constrained users; users are subject to $\mathcal{O}(\log n)$ complexity per round in the number of users n. Fourthly, the grouping structure of our protocol gives the protocol robustness as the aggregator can continue to operate even when users fail to submit their measurements. Finally, our protocol can be used as a primitive to build complex algorithms such as principal component analysis, singular-value decomposition, and decision tree classifications by writing the inputs as aggregate-sum queries, like in [20].

The remainder of this paper is structured as follows. In Sect. 2 we look at related work. Then, in Sect. 3 we present our protocol in detail, and in Sect. 4 we analyse its security, privacy, complexity, and detection rate. Finally, in Sect. 5 we present our conclusions.

2 Related Work

We discuss various protocols for range validation of malicious inputs. First, we consider PDA protocols that have range validation built in. Then, we consider several alternative approaches not inherent to PDA protocols.

Kursawe [9] proposes a scheme in which the aggregator verifies that all private values are valid by checking that the sum of inputs approximates the true aggregate. However, it cannot identify which user sent the invalid value and requires knowledge of the true aggregate beforehand, which is not always feasible.

Sun et al. [21] present APED, a PDA protocol that detects defective smart meters using a method similar to ours. In APED, a trusted third party divides all users into w random sets of disjoint pairs such that each user is in w pairs at once, and creates a random key k_i for each user i. Then, for each pair of users (i, j), the third party sends $k_{i,j} = -(k_i + k_j)$ to the aggregator. In each round, each user i sends a ciphertext of their measurement, encrypted with the key k_i. After receiving the ciphertexts for that round, the aggregator decrypts the product of the ciphertexts of each pair in one of the w pairing sets of users using that pair's combined key $k_{i,j}$. If a pair cannot be decrypted, at least one of the two users must be defective, and the aggregator will use a different pairing set in the next round. After some rounds, the aggregator infers from the overlap of invalid pairs which users are defective. An extension of the protocol, DG-APED [22], uses groups of arbitrary size. Both protocols have two drawbacks. Firstly, they rely on a trusted third party to create groups and generate key material. Secondly, because the protocols are tailored to defective users, the detection algorithms are unsuitable for users that do not always send invalid users.

Ahadipour, Mohammadi, and Keshavarz-Haddad [23] propose a protocol that reduces the amount of private data the aggregator has access to. Users are divided into disjoint groups, and the aggregator obtains the sum of each group's values in addition to a random subset of the users' private values. The aggregator then looks at the collected private values to determine which users sent invalid values. While this reduces the privacy impact on its users, giving the aggregator access to even a single private value is not tolerable for sensitive data.

Yang and Li [15] propose a protocol that can identify out-of-range values using re-encryption. The aggregator divides users into disjoint groups, and when it finds that the aggregate of a group is out of range, it re-encrypts and shuffles the values of the violating group and sends them to a random user in that group. The random user decrypts the values and reports which values are out of range. The main drawback of this scheme is that it is especially vulnerable to collusions, as a single collusion between the aggregator and the random user suffices to reveal all private values of an entire group to the aggregator.

Finally, there is a multitude of proposals that assume that users are honest-but-curious, but note that zero-knowledge proofs could be used to perform input validation [8,13,14]. With zero-knowledge proofs, users can mathematically prove that their value is within a particular range without having to reveal their value. Generic zero-knowledge proofs such as SNARKs require a trusted setup, which is often not a realistic assumption. Its cousin, the STARK [24], resolves this problem, but this comes at the cost of increased communication

complexity. Corrigan-Gibbs and Boneh [25] introduce SNIPs to allow users to prove that input is valid according to an arbitrary circuit, but this solution requires a multitude of cooperating servers, of which all must be honest to guarantee correctness and at least one must be honest to guarantee privacy for the user; furthermore, client-side communication costs grow linearly with the complexity of the validation circuit. Range proofs [18] are a specific form of zero-knowledge proof specific to range checking. Even though range proofs such as Bulletproofs [26] are more efficient than generic zero-knowledge proofs, they still incur a relatively high complexity for the users (i.e. the provers) [19], and must also be used in addition to the privacy-preserving data aggregation protocol and a cryptographic link between the two such as a commitment scheme.

3 Probabilistically Range-Limited Private Data Aggregation

We consider a setting with n users and a single aggregator, similar to related work in Sect. 2. Users continuously submit new privacy-sensitive measurements to the aggregator at regular intervals called rounds; we assume that users and the aggregator have access to a synchronised clock. We work in the standard model under the assumption that the discrete log problem is intractable. Some users are malicious and may deviate from the protocol; these are exactly the users the aggregator wants to identify. All other users are honest-but-curious (also known as semi-honest). We assume that the aggregator is honest-but-curious, an assumption made in several other related works including [8,10]. This assumption makes sense in a business-driven setting, in which a malicious aggregator would be faced with negative publicity and a loss in consumer trust if its behaviour were discovered. Still, we allow for collusions between users and the aggregator. We assume that the sets of malicious and colluding users do not change throughout the protocol. Finally, we assume that the security, integrity, and authenticity of all messages is guaranteed. The notation used to describe our protocol is shown in Table 1. Our protocol broadly works as follows.

1. *Registration*: Each user sends a message to the aggregator indicating that they want to register. Once all users have registered, the aggregator divides the users into overlapping groups. It then sends information such as the public parameters and the group configuration to all registered users.
2. *Submission*: Every round, each user creates a new secret share of the value 0 for each group they are in. The user takes copies of their private value and blinds each copy with a different secret share. The user sends the blinded copies in addition to commitments to the secret shares to the aggregator.
3. *Aggregation*: The aggregator verifies that the secret shares of each group sum to 0 and verifies that each user used copies of a single private value, remembering which users and groups failed verification. Next, the aggregator computes the sum of private values of each group, and remembers which groups have aggregates that are out of bounds. Finally, the aggregator combines all group aggregates to find the sum of all private values.

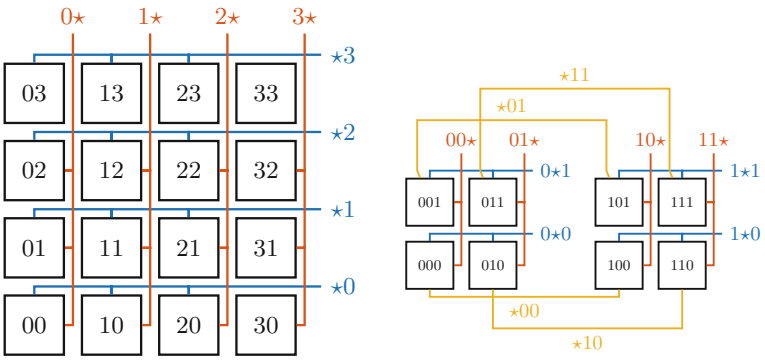
4. *Detection*: Eventually, the aggregator looks back at which groups exhibited malicious activity over the past several rounds, and derives from their overlap which users caused the malicious behaviour. As the protocol progresses, the aggregator is able to identify more and more malicious users.

Table 1. The notation used in the description of our protocol

Symbol	Meaning
n	Number of users
b	Grouping base/radix = users per group
ℓ	Grouping dimensionality = groups per user
$[min, max]$	Valid range of a single private value
g	Generator for commitments
pp	Public parameters, contains all of the above
U	Set of all user identifiers
G	Set of all group identifiers
G_i	Set of identifiers of groups of user i
U_j	Set of identifiers of users of group j
N_i	Set of identifiers of neighbours of user i
(sk_i, pk_i)	User i's key pair
t	Round number
$m_{i,t}$	User i's private value in round t
$c_{i,j,t}$	User i's encryption of $m_{i,t}$ for group j
$M_{j,t}$	Sum of private values of users in group j in round t
M_t	Sum of all private values in round t
$r_{i \to j,t}$	User i's random value for neighbour j in round t
$s_{i,j,t}$	User i's secret share for group j in round t
$d_{i,j,t}$	User i's commitment to $s_{i,j,t}$
V	Set of group identifiers aggregator marked as malicious
W	Set of user identifiers aggregator marked as malicious

3.1 Registration

The goal of the registration phase is to determine the parameters under which the protocol will run and to exchange the necessary information for subsequent rounds. Firstly, the honest-but-curious aggregator chooses a random generator g of an algebraic structure in which the discrete log problem is hard, such as a specific elliptic curve. Additionally, the aggregator chooses application-specific values for n and $min < max$. Then, each user sends a message to the aggregator indicating the desire to participate in the protocol. Once n users have registered, the aggregator sends the public parameters pp and some additional information to all users. The remaining public parameters and additional information are chosen based on the following grouping algorithm and secret sharing scheme.



(a) A 4^2-hypermesh, which has 16 nodes and 8 hyperedges

(b) A 2^3-hypermesh (which is the graph of a regular cube), which has 8 nodes and 12 hyperedges

Fig. 1. Examples of hypermeshes

Parameters for the Grouping Algorithm. The grouping algorithm divides users into groups such that the aggregator can pinpoint malicious users based on which groups exhibit malicious behaviour. We base our algorithm on the structure of a hypermesh [27]. A b-ary ℓ-dimensional hypermesh is a hypergraph with b^ℓ nodes, where each node is assigned an ℓ-digit identifier $d_{\ell-1}d_{\ell-2}\dots d_0$ such that $d_i \in [0, b)$ for all $0 \leq i < \ell$. Two nodes are neighbours if and only if their identifiers differ in exactly one digit. Nodes are connected by b-edges, i.e. edges with b endpoints. Edge identifiers have the same format as node identifiers, except that exactly one digit is replaced by the wildcard symbol $\star$. Every edge then connects the b nodes of which the identifier matches that of the edge, ignoring the digit in the wildcard's position. Identifiers can be considered coordinates in an ℓ-dimensional Euclidean space, with $b^{\ell-1}$ edges aligned along each dimension for a total of $\ell b^{\ell-1}$ edges. We give some examples of hypermeshes in Fig. 1.

In our protocol, the aggregator generates a b-ary ℓ-dimensional hypermesh after all n users have registered, with the requirements that $n = b^\ell$, $b \geq 2$, and $\ell \geq 2$. The edges in the hypermesh are then exactly the groups that users are in. Generating such a hypermesh constitutes choosing values for b and ℓ, and assigning to each user a unique identifier in $[0, b^\ell)$, which can be converted to a unique ℓ-digit b-ary identifier. These three variables are sufficient for a user to reconstruct the hypermesh and determine their own position. The ℓ groups that user i is in, denoted G_i, can be found by replacing the respective ℓ digits in i by the wildcard symbol $\star$. The b users in group j, denoted U_j, can be found by replacing the wildcard symbol $\star$ with the respective values $[0, b)$. The neighbours of user i, denoted N_i, can be found by taking the union of $\{j \in U_i \mid G_j\}$, minus i.

Parameters for Secret Sharing. Our scheme uses secret sharing to prevent the aggregator from decrypting ciphertexts unless all ciphertexts of a group have

been aggregated. We apply the procedure for creating 0-sum additive secret shares used in [8] to each group in G. We avoid direct communication between users by forwarding messages through the (honest-but-curious) aggregator, but use public-key encryption to ensure that the aggregator cannot see the actual random values being transmitted. Our goal is to obtain secret shares $s_{i,j,t}$ for each user $i \in U$ in each group $j \in G_i$ in each round t such that

$$\forall j \in G : \sum_{i \in U_j} s_{i,j,t} = 0. \tag{1}$$

While the following description assumes that users exchange random numbers each round, such excessive communication can be avoided by having users exchange seeds for random number generators once during registration.

First, each user i generates an asymmetric key pair (sk_i, pk_i), and includes pk_i when sending the registration message to the aggregator. Once all n users have registered, the (honest-but-curious) aggregator sends to each user i the public keys $\{pk_k \mid k \in N_i\}$. These key pairs can be reused and do not need to be exchanged again in future rounds. Then, in each round t, user i generates a random number $r_{i \to k,t}$ for each neighbour $k \in N_i$, encrypts it with pk_k, and sends this value to the aggregator, who forwards the message to user k. Once user i has obtained $r_{k \to i,t}$ for each neighbour k, user i creates the secret share

$$s_{i,j,t} = \sum_{k \in G_j} (r_{i \to k,t} - r_{k \to i,t}) \tag{2}$$

for each $j \in G_i$. We consider the privacy of this construction in Sect. 4.2. We present a communication diagram that includes registration in Fig. 2.

3.2 Submission

In round t, each user i submits the private value $m_{i,t}$ such that the aggregator can obtain the group aggregates without seeing $m_{i,t}$. We use encryption function $c_{i,j,t} = m_{i,t} + s_{i,j,t}$ to have each user i send $\{c_{i,j,t} \mid j \in G_i\}$ to the aggregator, with the secret share $s_{i,j,t}$ as described in Sect. 3.1. To prevent malicious users from avoiding detection by using a different $m_{i,t}$ in different groups, users must additionally send commitments to their secret shares. We use a simple homomorphic commitment scheme that is computationally binding and computationally hiding: To commit to a value x, a user sends g^x. Then, each user i computes commitments $d_{i,j,t} = g^{s_{i,j,t}}$ and sends $\{(c_{i,j,t}, d_{i,j,t}) \mid j \in G_i\}$ to the aggregator.

3.3 Aggregation

The aggregation phase is asynchronous to user submissions and may be invoked by the aggregator at any time. Before aggregating the submissions for round t, the aggregator verifies for each group $j \in G$ of which all users have submitted their values by checking that

$$\prod_{i \in U_j} d_{i,j,t} = \prod_{i \in U_j} g^{s_{i,j,t}} = g^{\sum_{i \in U_j} s_{i,j,t}} = g^0 = 1 \tag{3}$$

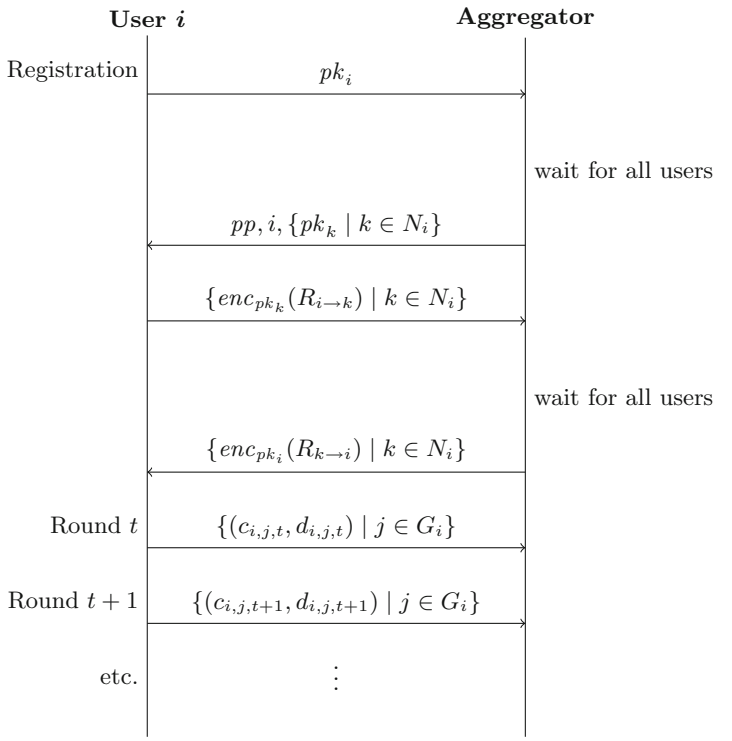


Fig. 2. An overview of the communication in our protocol. To reduce per-round communication, users exchange seeds $R_{i\to k}$ during registration to generate $r_{i\to k,t}$ in round t.

to ensure that users committed to secret shares of the value 0. If a group j fails this check, at least one user in this group must have been malicious, so the aggregator adds j to V. Next, the aggregator constructs for each user i the set

$$\{g^{c_{i,j,t}}(d_{i,j,t})^{-1} \mid j \in G_i\} = \{g^{m_{i,t}+s_{i,j,t}} g^{-s_{i,j,t}} \mid j \in G_i\} = \{g^{m_{i,t}} \mid j \in G_i\} \quad (4)$$

and checks that all values in the set are equal. This ensures that each $c_{i,j,t}$ for user i uses the same $m_{i,t}$. If user i fails this check, all groups in G_i are added to V, effectively marking this user as malicious once the detection algorithm runs. Users that fail to submit measurements similarly have their groups added to V. If desired, a level of lenience can be introduced by only adding these groups once a user fails to submit in multiple rounds.

After the aggregator has completed its verifications, aggregation can start. For each group $j \in G$, the aggregator calculates

$$M_{j,t} = \sum_{i \in U_j} c_{i,j,t} = \sum_{i \in U_j} (m_{i,t} + s_{i,j,t}) = \sum_{i \in U_j} m_{i,t}. \quad (5)$$

If an aggregate $M_{j,t}$ is not in the range $[b \cdot min, b \cdot max]$, at least one user must have sent a value that is not in $[min, max]$, so the aggregator adds j to V. This check can be adjusted to support use cases in which ranges differ per user or per round by checking that the aggregate is in the sum of the users' current ranges.

The sum of all private values can be calculated by taking the sum of all group aggregates. However, the aggregator should refrain from including invalid groups. Therefore, the aggregator calculates

$$M_t = \frac{\sum_{j \in G \setminus V} M_{j,t}}{\ell}, \tag{6}$$

which is the average of the total sums along each of the hypermesh's ℓ dimensions, excluding groups in V. This approximates the sum of only the honest-but-curious users; if all users behave honestly this approximation is perfect. If desired, the aggregator can estimate the sum of all users by including a fake group aggregate for each group in V based on the average of $\{M_{j,t} \mid G \setminus V\}$.

3.4 Detection

The detection algorithm lets the aggregator identify which users are malicious. Throughout the protocol and across rounds, the aggregator adds groups that exhibit malicious behaviour to the set V. In particular, the set V contains all groups in which at least one user sent a wrongly constructed secret share or sent different private values to different groups in the same round, and contains a subset of groups in which at least one user sent an out-of-bounds value. By looking at the overlaps of groups in V, the aggregator can infer which users caused the malicious behaviour; users that are in exactly ℓ different groups in V are malicious and are added to W. Over time, the set V becomes more and more complete until all groups containing malicious users have been detected. We prove that this method does not result in false-positive detections in Sect. 4.1, even if some malicious users collude. We analyse the detection rate in Sect. 4.4.

4 Analyses

4.1 Security Analysis

In this section we prove that the aggregator does not incorrectly identify users, we prove that malicious users cannot submit different measurements to different groups, and we analyse the impact of missing users to the aggregate.

Proof of No False Positives. It is important that the aggregator correctly identifies which users are malicious. We prove that malicious users cannot frame an honest-but-curious user, even if they coordinate the values they send.

Theorem 1. *In our protocol, the aggregator will never identify an honest-but-curious user as a malicious user if there are fewer than ℓ malicious users.*

Proof. For the sake of contradiction, let there be an honest-but-curious user whom the aggregator falsely identifies as malicious. Then this user must be in ℓ groups of V, so this user shares ℓ groups with malicious users. Because a group contains those users that differ by exactly one digit, two users can at most share a single group. The wrongly-identified user must therefore share groups with ℓ different malicious users. However, by assumption of the theorem's antecedent, there are strictly fewer than ℓ malicious users. Therefore, the honest-but-curious user could not have been identified as a malicious user. □

Proof of Aggregate Consistency. Users blind their private measurements $m_{i,t}$ using secret shares $s_{i,j,t}$ to obtain $c_{i,j,t}$. It is important that the aggregator verifies that a user's $c_{i,j,t}$ values use the same underlying $m_{i,t}$, malicious users could avoid detection by causing inconsistencies between aggregates otherwise. We show that it is infeasible for users to do this under our security model, regardless of how many users are malicious. Working in the standard model, every user i sends $(c_{i,j,t}, d_{i,j,t})$ for each $j \in G_i$ to the aggregator, constructed in any way the users want. Let $s_{i,j,t} = dlog_g(d_{i,j,t})$ and $m_{i,j,t} = c_{i,j,t} - s_{i,j,t}$ for all users i and for all $j \in G_i$, regardless of whether values are constructed honestly.

Theorem 2. *In our protocol, a malicious user i cannot send messages in round t to the aggregator such that $m_{i,j,t} \neq m_{i,j',t}$ for any two groups $j, j' \in G_i$ such that the aggregator's verification does not fail, assuming that the discrete log problem is intractable in the group generated by g.*

Proof. Firstly, if either user i or any neighbour $k \in N_i$ fails to send their messages, the aggregator's verification fails right away and the malicious user does not succeed. Now, it follows from the aggregator's verification of Eq. 3 that $\sum_{i \in U_j} s_{i,j,t} = 0$. Subsequently, we know from the verification of Eq. 4 that, for fixed $i \in U$ and $t \in \mathbb{N}$, all $c_{i,j,t} - s_{i,j,t}$ for $j \in G_i$ are equal. Therefore, by definition of $m_{i,j,t}$, all $m_{i,j,t}$ for fixed $i \in U$ and $t \in \mathbb{N}$ are also equal. □

Impact of Missing User Values. The influence of malicious users on M_t decreases as the aggregator adds more groups to V. At the same time, groups in V contain honest-but-curious users. We quantify the effect that malicious users have on the correctness of the total aggregate.

Each user effectively contributes their measurement ℓ times, and, by Theorem 2, each contribution is the same. An ideal protocol would remove only the ℓ contributions of each malicious user. Our protocol also removes the $(b-1)\ell$ contributions of each malicious user's neighbours. The total impact of any set of fewer than ℓ malicious users is greatest when these malicious users do not share any groups, in which case V contains $(\ell - 1)\ell$ groups. The aggregator then removes $b(\ell-1)\ell$ contributions instead of the optimal $(\ell-1)\ell$; a factor of b more than optimal. With a total of ℓb^ℓ contributions amongst all users, the effect of malicious users on M_t therefore diminishes as ℓ increases.

4.2 Privacy Analysis

We argue that our protocol is a secure data summation protocol in the setting described in Sect. 3. In particular, we argue that when executing the protocol using a b^ℓ-hypermesh, both the joint view of any set of users and the joint view of the aggregator and a set of fewer than $(b-1)^\ell$ users do not leak any information about honest-but-curious users' inputs, besides what can be inferred from the group aggregates. We should note that we assume that each group with an honest-but-curious user also contains at least one other non-colluding honest-but-curious user so as to prevent trivial attacks on the aggregates. This assumption is naturally present in many group-based aggregation schemes, including [8,10,14]. Also recall that the aggregator is honest-but-curious and will therefore assign users to random positions honestly.

Firstly, we consider the joint view of any set of users $U_A \subset U$. The view consists only of the public parameters pp, the users' private data, and the public keys pk_i and random seeds $r_{i \to k,t}$ other users have sent to users in U_A. Confidentiality is trivial because the view does not contain any data derived from the private values $m_{i,t}$ of any user $i \notin U_A$.

Next, we consider the joint view of the honest-but-curious aggregator and any set $U_A \subset U$ of fewer than $(b-1)^\ell$ users. The view consists of the same data as before, now in addition to the aggregator's private information and the data that are sent to the aggregator. We proceed to dissect the implications of this view. Firstly, malicious users in U_A differ only from honest-but-curious users in U_A in that they can interact dishonestly with other users, but this does not give them an advantage. If a malicious user refuses to interact with or sends malformed data to a user, then this user halts and privacy is maintained. Otherwise, if a malicious user sends non-random data to user i, then this is no worse than an honest-but-curious user in U_A sharing their data with the aggregator. Secondly, users that are not in U_A receive sensitive information through the aggregator, but privacy is ensured by encrypting data such that the decryption key is not in the adversary's view. Thirdly, the private values $m_{i,t}$ of user $i \notin U_A$ are masked using the secret shares $s_{i,j,t}$ constructed from values $r_{i \to k,t}$. Because at least one user $k \neq i$ of each group $j \in G_i$ is not in U_A, both $r_{i \to k,t}$ and $r_{k \to i,t}$ are chosen honestly and remain unknown to the adversary. Because additive secret sharing is trivially secure, the secret shares $s_{i,j,t}$ properly mask $m_{i,j,t}$. Finally, we observe that each submission occurs in multiple linearly dependent aggregates, which is equivalent to a system of linear equations. We prove that it is infeasible for the adversary to solve this system because it is not full rank.

Theorem 3. *The rank of the incidence matrix of a b^ℓ-hypermesh is $b^\ell - (b-1)^\ell$. (Equivalently, the number of unknowns in the incidence matrix is $(b-1)^\ell$.)*

Proof. We model the incidence matrix such that each row describes a group and each column describes a user. We construct the incidence matrix recursively, similar to how the hypermesh itself can be constructed. Given a b-ary 1-dimensional hypermesh, its incidence matrix $C_{b,1}$ is a $(1 \times b)$-matrix containing only 1s. Then, a b-ary ℓ-dimensional hypermesh can be constructed from b copies of the b-ary

$(\ell-1)$-dimensional hypermesh, where all nodes are additionally connected to their counterparts in the other copies using b-edges. This allows us to construct the incidence matrix $C_{b,\ell}$ for $\ell > 1$ as

$$\begin{bmatrix} C_{b,\ell-1} & 0 & \dots & 0 \\ 0 & C_{b,\ell-1} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & C_{b,\ell-1} \\ I_{b^{\ell-1}} & I_{b^{\ell-1}} & \dots & I_{b^{\ell-1}} \end{bmatrix}, \tag{7}$$

where each 0 represents a matrix of the same size as $C_{b,\ell-1}$ containing only 0s, and I_x denotes an identity matrix of size $x \times x$.

We now use complete induction on ℓ to prove that $\operatorname{rank}(C_{b,\ell}) = b^\ell - (b-1)^\ell$. For the base case, we take $\ell = 1$ and find that $\operatorname{rank}(C_{b,1}) = 1$, which matches our theorem:

$$b^\ell - (b-1)^\ell = b - (b-1) = 1. \tag{8}$$

For the recursive case, take as our induction hypothesis that $r = \operatorname{rank}(C_{b,\ell-1}) = b^{\ell-1} - (b-1)^{\ell-1}$. We write $C_{b,\ell}$ in column echelon form as follows to determine its rank. Firstly, consider the column operations necessary to write $C_{b,\ell-1}$ in column echelon form, and apply them to each instance of $C_{b,\ell-1}$ in $C_{b,\ell}$. Note that this also transforms the $I_{b^{\ell-1}}$s located beneath the $C_{b,\ell-1}$s. After applying these steps, each instance of $C_{b,\ell-1}$ has $b^{\ell-1} - r$ empty columns on the right, while each instance of $I_{b^{\ell-1}}$ has no zero columns because it is full rank. The rightmost $b^{\ell-1} - r$ columns of each $I_{b^{\ell-1}}$ are now identical, however, and have nothing but 0s above them. As such, we cancel out these columns except in the rightmost instance of $I_{b^{\ell-1}}$ using simple column operations. This cancels out $(b-1)(b^{\ell-1} - r)$ columns, while all other columns are non-zero. After moving these zero columns to the right of the matrix, $C_{b,\ell}$ is in column echelon form. The rank of $C_{b,\ell}$ is then the number of non-zero columns, which is

$$b^\ell - (b-1)\left(b^{\ell-1} - r\right) = b^\ell - (b-1)\left(b^{\ell-1} - \left(b^{\ell-1} - (b-1)^{\ell-1}\right)\right) \tag{9}$$

$$= b^\ell - (b-1)(b-1)^{\ell-1} = b^\ell - (b-1)^\ell, \tag{10}$$

proving our theorem. □

With fewer than $(b-1)^\ell$ users in the view, the adversary always has at least one unknown in this system. To give an intuition into the growth of $(b-1)^\ell$, consider Fig. 3, where we show the maximum ratio of users that may collude with the aggregator as a function of b and ℓ without breaking confidentiality. For example, a system with $b = \ell = 2$ could not tolerate a single colluding user, while a system with $b = \ell = 5$ could tolerate up to $\frac{4^5}{5^5} \approx 33\%$ of all users colluding. As the number of groups per user grows, the collusion resistance decreases. This can be compensated for by increasing the number of users per group, but, as we discuss in Sect. 4.4, this decreases the detection rate.

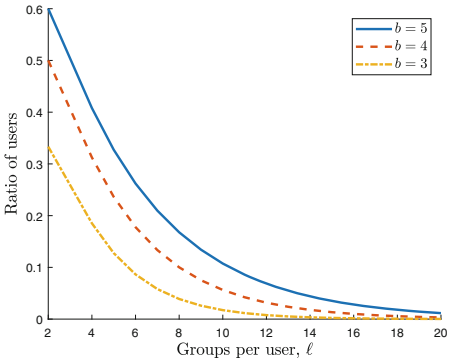
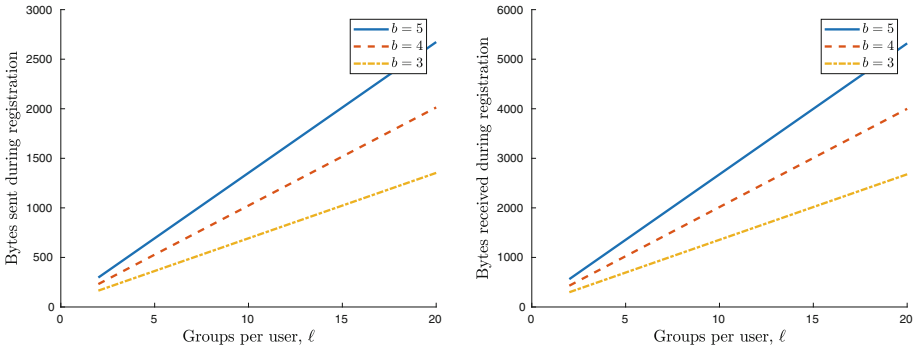


Fig. 3. Maximum proportion of users that can collude with the aggregator as a function of b (users per group) and ℓ (groups per user)



(a) Outgoing communication (b) Incoming communication

Fig. 4. Per-user communication during registration. We assume 4 bytes per (masked) private value and 256-bit EC-cryptography. With point compression, this results in 33-byte keys, ciphertexts (for seeds), and commitments.

4.3 Complexity Analysis

We quantify the complexity of our protocol in terms of the number of users n. Then, we compare our protocol to a selection of related PDA protocols. We express complexity in terms of the amount of encryptions, decryptions, multiplications, exponentiations, additions, subtractions, and outgoing messages, separately for each user and the aggregator, similar to the analysis in [10].

Complexity of Our Protocol. Firstly, note that in our protocol, $b = n^{\frac{1}{\ell}}$. This amount is maximal when $\ell = 2$, so we say that b is $\mathcal{O}(\sqrt{n})$. Meanwhile, $\ell = \log_b(n)$ is $\mathcal{O}(\log n)$. We show a time diagram of our protocol in Fig. 2.

During the registration, each user sends one encrypted seed for each neighbor and a fixed-size key to the aggregator, resulting in an outgoing communication

complexity of $\mathcal{O}(\sqrt{n}\log n)$ per user. Meanwhile, each user receives one key and one encrypted seed per neighbor, for an incoming communication complexity of $\mathcal{O}(\sqrt{n}\log n)$ per user. We visualize registration communication complexity in Fig. 4. Later, in each round, each user sends for each group it is in a constant-size message containing a masked plaintext and a commitment, for a communication complexity of $\mathcal{O}(\log n)$. Users do not receive anything during rounds. Creating a submission requires one commitment and one masked private value for each of the user's groups, for a total of $\mathcal{O}(\log n)$ exponentiations and $\mathcal{O}(\log n)$ additions per user per round.

Table 2. Complexity analysis of several privacy-sensitive data aggregation protocols, separated by party: User (U) or Aggregator (A), given total number of users n and range size 2^r.

Protocol	[15]		[25]		[26]		Ours	
Aggregation	✓		✓				✓	
Detection	✓		✓		✓		✓	
Robust			✓		✓		✓	
Topology	Tree		Arbitrary		Arbitrary		Hypermesh	
Group	ElGamal		FFT field		EC		EC	
Party	U	A	U	A	U	A	U	A
Enc	$\mathcal{O}(1)$	$\mathcal{O}(1)$	–	–	–	–	–	–
Dec	$\mathcal{O}(1)$	$\mathcal{O}(1)$	–	–	–	–	–	–
Mult	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(r\log r)$	$\mathcal{O}(r\log r)$	–	–	–	$\mathcal{O}(n\log n)$
Exp	$\mathcal{O}(1)$	$\mathcal{O}(1)$	–	–	$\mathcal{O}(r)$	$\mathcal{O}(nr)$	$\mathcal{O}(\log n)$	$\mathcal{O}(n\log n)$
Add	–	–	–	–	–	–	$\mathcal{O}(\log n)$	$\mathcal{O}(n\log n)$
Sub	–	–	–	–	–	–	–	–
Com	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(\log r)$	$\mathcal{O}(1)$	$\mathcal{O}(r)$	$\mathcal{O}(nr)$	$\mathcal{O}(\sqrt{n}\log n)$	$\mathcal{O}(n\sqrt{n}\log n)$

The aggregator forwards each user message during the registration, resulting in a factor of n more communication. After the registration, however, the aggregator does not need to communicate with users other than sending acknowledgements. During aggregation, the aggregator verifies user inputs, requiring one exponentiation and one multiplication for each group for each user, for a total of $n\ell b^{\ell-1}$ of either operation. The calculation of the aggregate itself requires only that the aggregator sums together all $n\ell b^{\ell-1}$ submissions. The detection phase does not require complex operations, as the aggregator need only find which users are in ℓ groups of V.

Comparison to Related Protocols. We compare our protocol to a selection of related PDA protocols, as shown in Table 2. Our analysis is subject to several limitations. Firstly, because our protocol is tailored to identifying malicious users, we restrict our analysis to detection protocols for malicious users, thus also excluding APED [21] and DG-APED [22]. Secondly, in our analysis of the protocol in [25], we assume that the number of multiplication gates is linear in the

size of the range, which corresponds to the size of an integer comparison circuit. Finally, for the protocol in [15], we assume a binary tree topology for simplicity, and include operations related to the detection sub-protocol for fairness.

The protocol in [15] provides by far the lowest complexity by validating in a decentralised fashion, but requires long periods of interactivity and has the weakest security model: An honest-but-curious aggregator and any single user can collude to obtain all private values. Prio [25] and Bulletproofs [26] have a complexity that depends on the size r of the valid range; meanwhile, our complexity is independent of r. Additionally, with Bulletproofs, the size of the range must be of the form $[0, 2^r)$ for some natural number r, whereas our protocol supports arbitrary ranges, as does Prio. Finally, Bulletproofs can verify user submissions in bulk, but only if all users have the same valid range. Otherwise, the verification complexity grows linearly with the number of different ranges. While an alternative would be to verify the widest range in bulk, this is not practical. Our protocol supports different ranges for all users without an increase in complexity, instead affecting the detection rate, as we discuss in Sect. 4.4.

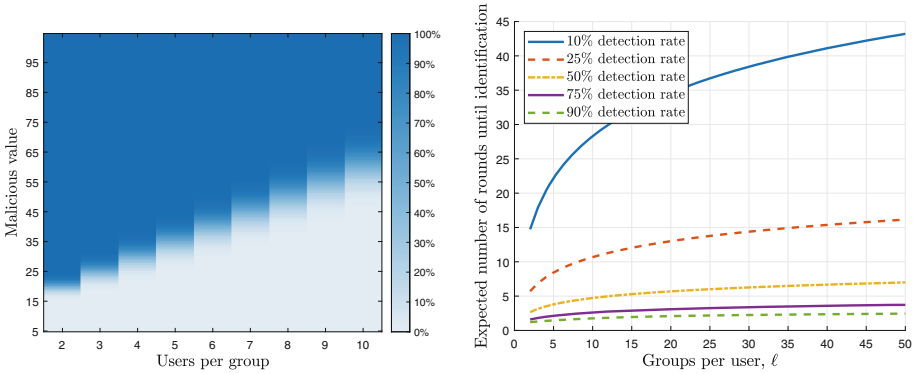
We conclude that the complexities of these protocols must be considered in the context of the application. If users have different, personalised use cases, the computation and communication complexities of our protocol scale better than competing protocols.

4.4 Detection Rate Analysis

Values submitted by honest-but-curious users in the same group as a malicious user may coincidentally compensate for the malicious transgression. As a result, our detection algorithm is probabilistic. In this section we analyse how the detection rate varies as a function of the protocol's parameters. In our analysis we model each honest-but-curious user's value as a truncated binomial distribution X with $\mu = \frac{min+max}{2}$ and a support of $[min, max]$. For the sake of illustration, we use $\sigma = 2$, $min = 5$, and $max = 15$. We model the sum of n independent honest-but-curious users' values, denoted X_n, by approximating X with a non-truncated binomial distribution, multiplying the distribution by n, and truncating this distribution to the range $[n \cdot min, n \cdot max]$.

Detection Rate of a Single Malicious User. Consider a system with a single malicious user i who submits the out-of-range measurement m. We assume that $m > max$, without loss of generality because X and X_{b-1} are symmetrical. Recall that user i is detected only once all ℓ groups in G_i are in V.

First, we consider the detection rate of an individual group. The aggregate of a group $j \in G_i$ does not exceed its upper bound if and only if $M_{j,t} = X_{b-1} + m \leq b \cdot max$, or, equivalently, if $X_{b-1} \leq b \cdot max - m$. We illustrate the probability that this relation holds as a function of m and b in Fig. 5a. The figure shows that fixing a particular detection rate results in the corresponding malicious value growing linearly with the group size. Note that the detection rate is exactly 0% at $m = max$ and exactly 100% at $m = b(max - min) + min$.



(a) Probability that a group of given size transgresses the valid range, given the value of the single malicious user in that group, with each honest value distributed as X

(b) Expected number of rounds until a given number of the malicious user's groups has been detected by the aggregator, given the per-group detection rate for any single round

Fig. 5. Detection rate of a single malicious user

We can thus model the detection rate of a group as a geometric variable to express the expected number of rounds until it is detected. Because the groups G_i overlap only in user i, their detection rates are independent for fixed m. The expected number of rounds until all ℓ groups have been detected at least once is then the expected maximum of ℓ iid geometric variables, which is [28]

$$f(\ell, p) = \sum_{k=0}^{n} \left(\binom{\ell}{k} p^k (1-p)^{\ell-k} (1 + f(\ell - k, p)) \right), \tag{11}$$

where ℓ is the number of groups and p is the per-round detection probability of each group. Figure 5b shows $f(\ell, p)$ for various combinations of ℓ and p. We conclude that increasing the number of groups per user necessitates a higher per-group detection rate to retain the number of expected rounds, which can be done by reducing the group size, for example.

Detection Rate of Multiple Malicious Users. When a group contains multiple malicious users, these users can either intensify or diminish the sum effect they have on their group's aggregate. This means that, depending on the usage scenario, multiple malicious users either become harder to detect (if malicious users have equal reason to transgress the range in either direction) or easier to detect (if malicious users have more reason to transgress the range in a particular detection). Therefore, our protocol is best suited for applications where users are most likely to transgress in a particular direction.

We can reuse our results from Sect. 4.4 to quantify the detection rate of a group with multiple malicious users. Given a group of size b with n malicious

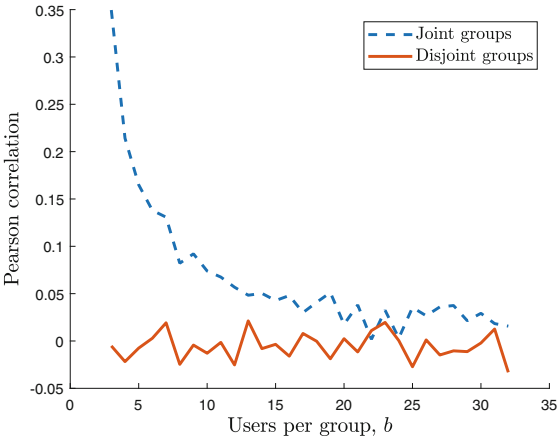


Fig. 6. The correlation of the detection rate of two groups, each with a different malicious user and overlapping in one honest-but-curious user. Simulated in Matlab by sampling honest values from truncated normal distribution $\mathcal{N}(10, 9)$ with support $[8, 19]$. Malicious users send $10 + 5b$, which ensures the groups are not always detected. Correlation was calculated with 5000 trials per group size.

users, the detection rate of the sum of malicious values m is

$$\Pr[X_{b-n}+m \leq b \cdot max] = \Pr[X_{b-n}+m-(n-1)\cdot max \leq (b-(n-1))\cdot max]. \quad (12)$$

That is, this detection rate is the same as that of a single malicious user that sends the value $m-(n-1)\cdot max$ in a group with only $b-(n-1)$ users.

Users may coordinate the malicious values they send to avoid being detected by the aggregator in some groups. However, it follows from Eq. 12 that complete avoidance is possible only if the sum of their values is valid. Because values are consistent between groups by Theorem 2, this type of avoidance detection requires that the sum effect on the total aggregate is valid, so malicious users do not gain any significant advantages by working together.

An important observation regarding the interplay of group aggregates is that malicious users that do not share a group may still have an overlap in the users that they share groups with. In this case, the detection rates of these groups become covariant because of the common user. As shown in Fig. 6, the impact of this covariance depends on the group size b and quickly becomes negligible. Therefore, the expected number of rounds until detection as expressed in Eq. 5b holds for multiple users up to covariance.

5 Conclusion

Data aggregation is an immensely useful tool for various applications, but introduces a number of privacy concerns. Existing privacy-preserving data aggregation protocols tend to assume that the users are honest-but-curious rather than

malicious, or use zero-knowledge proofs, which impose significant computational requirements on the users. Either way, adoption of these much-needed protocols is difficult. We present a data aggregation protocol that probabilistically detects out-of-range user values without giving the aggregator access to these values. Our protocol imposes only $\mathcal{O}(\log n)$ per-round computational complexity on its users without relying on expensive cryptography. The protocol is also robust to missing data because it can exclude any number of groups that have exhibited malicious behaviour. Furthermore, given b^ℓ users for positive integers b and ℓ, the aggregator will not misidentify an honest-but-curious user as malicious as long as there are strictly fewer than ℓ malicious users. Finally, our protocol continues to guarantee privacy even when up to $(b-1)^\ell$ users collude with the aggregator.

References

1. Bonawitz, K., et al.: Practical secure aggregation for privacy-preserving machine learning. In: Proceedings of the ACM Conference on Computer and Communications Security, New York, New York, USA, pp. 1175–1191. ACM Press (2017). ISBN 9781450349468. https://doi.org/10.1145/3133956.3133982
2. Burke, J., et al.: Participatory sensing. In: The 4th ACM Conference on Embedded Networked Sensor Systems, p. 5 (2006). https://escholarship.org/uc/item/19h777qd
3. Fanti, G., Pihur, V., Erlingsson,Ú.: Building a RAPPOR with the unknown: privacy-preserving learning of associations and data dictionaries. Proc. Priv. Enhancing Technol. **2016**(3), 41–61 (2016). ISSN 2299–0984. https://doi.org/10.1515/popets-2016-0015
4. Bittau, A., et al.: PROCHLO: strong privacy for analytics in the crowd. In: SOSP 2017 - Proceedings of the 26th ACM Symposium on Operating Systems Principles, New York, New York, USA, pp. 441–459. ACM Press (2017). ISBN 9781450350853. https://doi.org/10.1145/3132747.3132769
5. LeMay, M., Gross, G., Gunter, C.A., Garg, S.: Unified architecture for large-scale attested metering. In: Proceedings of the Annual Hawaii International Conference on System Sciences, pp. 1–10 (2007). ISSN 15301605. https://doi.org/10.1109/HICSS.2007.586
6. Garcia, F.D., Jacobs, B.: Privacy-friendly energy-metering via homomorphic encryption. In: Cuellar, J., Lopez, J., Barthe, G., Pretschner, A. (eds.) STM 2010. LNCS, vol. 6710, pp. 226–238. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22444-7_15
7. Christin, D.: Privacy in mobile participatory sensing: current trends and future challenges. J. Syst. Software **116**, 57–68 (2016). ISSN 01641212. https://doi.org/10.1016/j.jss.2015.03.067
8. Erkin, Z., Tsudik, G.: Private computation of spatial and temporal power consumption with smart meters. In: Bao, F., Samarati, P., Zhou, J. (eds.) ACNS 2012. LNCS, vol. 7341, pp. 561–577. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31284-7_33
9. Kursawe, K.: Some Ideas on Privacy Preserving Meter Aggregation. Radboud Universiteit Nijmegen, Technical report, ICIS-R11002, pp. 1–15 (2010)

10. Erkin, Z.: Private data aggregation with groups for smart grids in a dynamic setting using CRT. In: 2015 IEEE International Workshop on Information Forensics and Security, WIFS 2015 - Proceedings, vol. 30, pp. 1–6. IEEE, 11 2015. ISBN 9781467368025. https://doi.org/10.1109/WIFS.2015.7368584
11. Rastogi, V., Nath, S.: Differentially private aggregation of distributed time-series with transformation and encryption. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, New York, New York, USA, pp. 735–746. ACM Press (2010). ISBN 9781450300322. https://doi.org/10.1145/1807167.1807247
12. Ács, G., Castelluccia, C.: I Have a DREAM! (DiffeRentially privatE smArt Metering). In: Filler, T., Pevný, T., Craver, S., Ker, A. (eds.) IH 2011. LNCS, vol. 6958, pp. 118–132. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-24178-9_9
13. Shi, E., Hubert Chan, T.-H., Rieffel, E., Chow, R., Song, D.: Privacy-preserving aggregation of time-series data. In: Annual Network & Distributed System Security Symposium (NDSS) (2011)
14. Kursawe, K., Danezis, G., Kohlweiss, M.: Privacy-friendly aggregation for the smart-grid. In: Fischer-Hübner, S., Hopper, N. (eds.) PETS 2011. LNCS, vol. 6794, pp. 175–191. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22263-4_10
15. Yang, L., Li, F.: Detecting false data injection in smart grid in-network aggregation. In: 2013 IEEE International Conference on Smart Grid Communications, SmartGridComm 2013, pp. 408–413. IEEE, October 2013. ISBN 9781479915262. https://doi.org/10.1109/SmartGridComm.2013.6687992
16. McLaughlin, S., Holbert, B., Fawaz, A., Berthier, R., Zonouz, S.: A multi-sensor energy theft detection framework for advanced metering infrastructures. IEEE J. Sel. Areas Commun. **31**(7), 1319–1330 (2013). ISSN 07338716. https://doi.org/10.1109/JSAC.2013.130714
17. Lie, D., Maniatis, P.: Glimmers: resolving the privacy/trust quagmire. In: Proceedings of the Workshop on Hot Topics in Operating Systems - HOTOS, volume Part F1293, New York, New York, USA, 2017, pp. 94–99. ACM Press. ISBN 9781450350686. https://doi.org/10.1145/3102980.3102996
18. Boudot, F.: Efficient proofs that a committed number lies in an interval. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 431–444. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45539-6_31
19. Morais, E., Koens, T., van Wijk, C., Koren, A.: A survey on zero knowledge range proofs and applications. SN Appl. Sci. **1**(8), 1–17 (2019). ISSN 2523–3963. https://doi.org/10.1007/s42452-019-0989-z
20. Blum, A., Dwork, C., McSherry, F., Nissim, K.: Practical privacy: the SulQ framework. In: Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp. 128–138 (2005). https://doi.org/10.1145/1065167.1065184
21. Sun, R., Shi, Z., Lu, R., Lu, M., Shen. , X.: APED: an efficient aggregation protocol with error detection for smart grid communications. In: GLOBECOM - IEEE Global Telecommunications Conference, pp. 432–437 (2013). https://doi.org/10.1109/GLOCOM.2013.6831109
22. Shi, Z., Sun, R., Lu, R., Chen, L., Chen, J., Shen, X.S.: Diverse grouping-based aggregation protocol with error detection for smart grid communications. IEEE Trans. Smart Grid **6**(6), 2856–2868 (2015). ISSN 19493053. https://doi.org/10.1109/TSG.2015.2443011

23. Ahadipour, A., Mohammadi, M., Keshavarz-Haddad, A.: Statistical-based privacy-preserving scheme with malicious consumers identification for smart grid, pp. 1–9, April 2019
24. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Scalable, transparent, and post-quantum secure computational integrity. IACR Cryptol. ePrint Arch., 2018:46 (2018). URL https://eprint.iacr.org/2018/046.pdf
25. Corrigan-Gibbs, H., Boneh, D.: Prio: private, robust, and scalable computation of aggregate statistics. In: Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017, pp. 259–282, March 2017. ISBN 9781931971379
26. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: short proofs for confidential transactions and more. In: Proceedings - IEEE Symposium on Security and Privacy, volume 2018-May, pp. 315–334. IEEE, May 2018. ISBN 9781538643525. https://doi.org/10.1109/SP.2018.00020
27. Szymanski, T.: "Hypermeshes": optical interconnection networks for parallel computing. J. Parall. Distrib. Comput. **26(1)**, 1–23 (1995). ISSN 07437315. https://doi.org/10.1006/jpdc.1995.1043
28. Eisenberg, B.: On the expectation of the maximum of IID geometric random variables. Stat. Probabil. Lett. **78**(2), 135–143 (2008). ISSN 01677152. https://doi.org/10.1016/j.spl.2007.05.011