Privacy Preserving Data Aggregation over Multi-hop Networks

Szilvia Lestyán

***Abstract*—We present a novel privacy-preserving data aggregation protocol in wireless networks composed of short-range devices. These devices provide a collaborative service and conduct privacy-preserving computations to obtain the aggregated result of their secret inputs. Our solution uses secure multi-party primitives as well as a new distributed perturbation technique to guarantee strong differential privacy against untrustworthy aggregators.**

***Keywords*—Privacy Preserving Data Mining, Secure Multiparty Computation, Differential Privacy**

I. INTRODUCTION

Although Internet is the prevalent communication network today connecting billions of different devices worldwide, there are still several practical cases where Internet connectivity is scarce and expensive, such as surveillance and monitoring of rural areas. In such applications, multi-hop wireless networks using short-range communications still provide a cheaper and compelling alternative to a global networking infrastructure. For example, wireless sensor networks are deployed for the purpose of monitoring agricultural areas in order to facilitate more responsive intervention and to optimise maintenance tasks with the aim to increase productivity [1]. However, different producers can compete with each other, and hence security, and in particular, confidentiality is of primary concern.

As a motivational scenario, consider vineyards[1] where sensors measure different regional characteristics such as the pH of the soil or its mineral composition. Across a large territory, there are several wineries, where winemakers use different types of fertilisers for their field. The composition of fertilisers is considered as trade secret among winemakers, thus revealing some characteristics of the soil can also reveal this confidential information. Moreover, for geological surveys as well as various consultancy services, different organizations periodically collect statistics about a larger territory over multiple vineyards, belonging to different producers, e.g., in order to measure soil contamination. Such monitoring service can also be beneficial for the producers as they may lack of expertise to deeply analyze the quality of the soil. Importantly, these organizations are only interested in aggregate measurements over multiple vineyards, and less concerned with sensor readings of a single producer. To this end, organizations can use mobile base stations, which move along the perimeter of multiple vineyards without approaching the fields [2]. These base stations initiate the aggregation protocol through the sensor nodes they can reach via short-range communication. That is, a base station builds an aggregation tree having the base station as root and simply fan-in[2] the values from leaves to the root (see Figure 1).

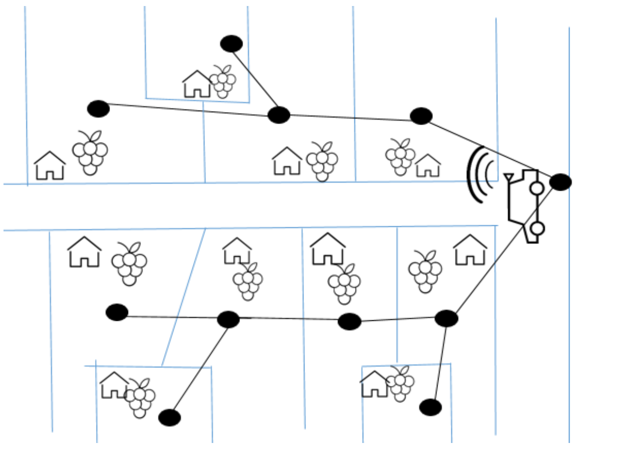

Fig. 1: Aggregation on vineyards. A mobile base station stops at the perimeter of the area and builds an aggregation tree, where the base station is the aggregator (root) node.

However, producers do not trust each other or the organization providing the monitoring service. In particular, they do not want neither another producer nor the organization to learn any of their sensor readings. The above simple aggregation protocol does not provide such privacy guarantees. In fact, this protocol cannot guarantee privacy even in the presence of semi-honest participants who follow the aggregation protocol faithfully but may learn any private sensor reading from the received messages. For example, a parent of a node in the aggregation tree, where both nodes belong to different producers, can immediately learn the measurement of its children, or an eavesdropper can capture a sensor's incoming and outgoing aggregates and easily calculate the measurement of the sensor.

In this paper, we propose a privacy-preserving aggregation protocol for the above scenario. In particular, our protocol guarantees that (1) sensor nodes cannot learn each other's readings, (2) a passive eavesdropper cannot infer any measurement

Manuscript submitted on December 3, 2016.

Szilvia Lestyan is with CrySyS Lab, Dept. of Networked Systems and Services, Budapest Univ. of Technology and Economics; email: lestyan@crysys.hu

[1]http://smartvineyard.com

[2]Fan-in is the algorithm where the values are sent from leaves to the root and gradually aggregated at each inner node, therefore the total sum appears at the root at the end of the algorithm.

in the network, (3) the aggregator node, who is untrusted, cannot learn any individual sensor reading in the network. For this purpose, sensor nodes employ secure multiparty computation as well as add noise to their readings in order to provide strong differential privacy guarantees against the aggregator. The variance of the noise is calibrated so that the aggregator can still learn the aggregate but any of its constituent measurement. We propose a novel distributed noise generation algorithm based on the geometric divisibility of the Laplace distribution, which provides increased robustness as well as flexibility over state-of-the-art solutions.

Our contributions are summarised as follows.

- We propose a novel mechanism to compute the sum of measurements (e.g. pH value of the soil, temperature, etc.) of multiple nodes in a privacy preserving and distributed manner assuming semi-honest adversaries. Our solution relies on secure multi-party primitives combined with homomorphic encryption. In addition, we noise the aggregate to prevent the untrusted aggregator from learning individual sensor readings in the network and hence providing formal privacy guarantees.
- We introduce a novel technique for privacy preserving distributed noise generation. We use the geometric infinite divisibility property of the Laplace distribution to preserve ϵ-differential privacy. This new noise generation method provides increased robustness and flexibility on distributed systems over earlier solutions.

II. Related Work

Our view of privacy preserving and computational model derived from a multi-party approach. For a survey of privacy preserving data mining see e.g. [3] and [4], and on general multi-party computation see [5] or [6] The basic idea of Secure Multiparty Computation (SMPC) is that a computation is secure if at the end of the computation, no party knows anything except its own input and the results (*privacy*). Secure two party computation was first investigated by Yao [7], and was later generalised to multiparty computation [5], [8]. These works all use a similar methodology: the function f to be computed is first represented as a combinatorial circuit, and then the parties run a short protocol for every gate in the circuit.

The aim of secure multiparty computation is to enable parties to carry out such distributed computing tasks in a secure manner. Whereas distributed computing [9] [10] classically deals with questions of computing under the threat of machine crashes and other inadvertent faults, secure multiparty computation is concerned with the possibility of deliberately malicious behavior by some adversarial entity.

On the combination of SMPC and graph algorithms, see a weighted case [11], where a protocol with which a set of n stores, selling l products between them, participate in joint computation to securely determine c_{jk}, the number of times product j and product k have sold together in all stores combined (without revealing any information about the products that any one store, individually, sells).

Others have only considered some route-planning in a similar setting: for privacy-preserving computation of APSD (all pairs shortest distance) and SSSD (single source shortest distance) see [12]; and in [13] a private computation for collision-avoiding route planning is introduced.

A comparative study has been fairly recently written on the problem of secure data aggregation in a distributed setting while preserving differential privacy for the aggregated data [14]. In their paper, they show the infinite divisibility of the Laplace distribution, and generate partial noises by drawing random variables from the *gamma*, the *Gauss* and one *Laplace* distributions.

In order to read on the application of the Laplace distribution based on the gamma distribution see [15]; where a privacy-preserving smart metering scheme that guarantees users' privacy while still preserving the benefits and promises of smart metering is proposed.

III. Applied Model

A. Network Model

Let $P_1, P_2, ..., P_N$ be parties (i.e., producer) owning private measurements $x_1, x_2, ..., x_N \in \mathbb{R}$. The parties wish to apply a function to the joint set $\bigcup x_i$ without revealing any unnecessary information about their individual values. That is, the only information learned by P_i about x_{-i} (where x_{-i} is any other measurement except x_i) is that which can be learned from the output of the algorithm, and vice-versa. We *do not assume any trusted third party* who computes the joint output on the raw data.

We also assume that a unique label is given to each party (or nodes). We do not assume a peer-to-peer system to be available, e.g. it is not necessary for all the parties to be directly connected. A channel between two parties is bidirectional and *first in first out* (FIFO), i.e. the messages received in the order in which they have been sent. Each party is represented by a single sensor node in the network. In case a producer deploys multiple sensors over his vineyard in our motivational scenario (see Section I), a single sensor is selected per vineyard which collects all measurements over the vineyard and can be reached by either the base station or by a sensor of a neighboring vineyard. Therefore, the topology of the parties can be represented with an *arbitrary strongly connected graph*.

We could also view the model as a multi-hop ad-hoc network, where nodes cooperate to form a network without using any infrastructure such as access points or base stations. Instead, nodes forward packets to each other, allowing communication among nodes outside wireless transmission range. For a survey on attacks on multi-hop ad-hoc networks see [16].

The channels between any two parties can be *secure* or *insecure* as well. A secure channel is a way of transferring data that is resistant to overhearing and tampering. In case of an insecure channel an eavesdropper can overhear any message (ciphertext) from any existing channel and try to decipher it.

Each party P_i has a set of neighbors, denoted N_i, this set contains the identities (labels) of these parties. This is the only knowledge a node (participant) can have of the global graph, e.g. it cannot "see" any other nodes besides its direct neighbours, it does not even know the total number of participants (only if it is the result of a particular protocol).

This model is *partially synchronous* (timing-based), i.e. we assume some restrictions on the relative timing of events, but execution is not completely lock-step as it is in the synchronous model. These models are the most realistic, but they are also the most difficult to program. Algorithms designed using knowledge of the timing of event can be efficient, but they can also be fragile in that they will not run correctly if the timing assumptions are violated.

B. Adversary Model

The adversary is assumed to be *semi-honest* and *static*, malicious adversaries are also partly considered [17], [18]. Therefore, semi-honest parties faithfully follow the protocol specification, yet attempt to learn additional information by analyzing the messages received during the protocol execution. Although the semi-honest adversarial model is weaker than the malicious model (where a party may arbitrarily deviate from the specification, it is often a realistic one. This is because deviating from a specified protocol which may be buried in a complex application is a non-trivial task. Moreover, producers do not collude with the potentially malicious aggregator.

C. Privacy and Security Model

It is assumed that a protocol execution can be attacked by an external entity, or even by a subset of the participating parties. The aim of this attack may be to learn private measurement or cause the result of the computation to be incorrect. In order to avoid this, every node can send its output to the trusted party, who performs the computation. But this is unlikely to happen in our scenario, thus we use and design algorithms where the same result can be achieved without using a trusted party. Different definitions of security for multiparty computation have been proposed, in this paper we are going to use the following [18]:

- *Privacy:* No party should learn anything more than its prescribed output. In particular, the only information that should be learned about other parties' inputs is what can be derived from the output itself. For example, in an auction where the only bid revealed is that of the highest bidder, it is clearly possible to derive that all other bids were lower than the winning bid. However, this should be the only information revealed about the losing bids.
- *Correctness:* Each party is guaranteed that the output that it receives is correct.
- *Independence of inputs:* Corrupted parties must choose their inputs independently of the honest parties' inputs.
- *Guaranteed output delivery:* Corrupted parties should not be able to prevent honest parties from receiving their output. For example, the adversary should not be able to disrupt the computation by carrying out a *denial of service* attack.
- *Fairness:* Corrupted parties should receive their outputs if and only if the honest parties also receive their outputs.

D. Differential Privacy

The above guarantees still allow the untrusted aggregator to learn individual sensor readings from the aggregate. Indeed, knowing a few measurements in the network (e.g., the aggregator may deploy extra sensors in the observed area to replicate the measurements) may help the aggregator to obtain a more accurate approximation of the remaining measurements.

Differential privacy ensures that the removal or addition of a single measurement from the network does not (substantially) affect the outcome of any analysis performed on the set of all measurements (such as the output of an aggregate function). Roughly speaking, this means that even if the aggregator learns all constituent measurements of the aggregate except one, it will not be able to infer this unknown measurement if the aggregate itself is differential private.

Suppose two databases D_1 and D_2, which *differ in at most one record* (measurement), where one is a proper subset of the other and the larger database contains just one additional measurement [19].

Definition 1 (Differential Privacy)**.** *A randomised algorithm $\mathcal{A}$ gives ϵ-differential privacy if for all data sets D_1 and D_2 differing on at most one record, and all $S \subset Range(\mathcal{A})$,*

$$Pr[\mathcal{A}(D_1) \in S] \leq e^{\epsilon} \times Pr[\mathcal{A}(D_2) \in S]$$

The probability is taken is over the coin tosses of $\mathcal{A}$.

The above definition guarantees that if one participant's data is removed from the dataset no outputs (and thus consequences of outputs) would become significantly more or less likely (up to ϵ). That is, all possible values of the aggregate are almost equally likely with D_1 and D_2. If ϵ is small, we have stronger privacy guarantee as the output probabilities become closer.

To provide differential privacy, the output of f (i.e., the aggregate) needs to be randomised, for example, by adding noise to that where the noise variance is calibrated to the sensitivity of the aggregate.

Definition 2 (Global sensitivity)**.** *Global sensitivity S_f of $f : D \to \mathbb{R}$ is the maximum absolute valued difference between a function's maxima and minima on neighboring datasets:*

$$S(f) = \max_{D_1, D_2} |f(D_1) - f(D_2)|$$

where D_1 and D_2 differ in a single entry.

It has been shown that by perturbing the output of a function f, we are able to reach ϵ-differential privacy [19]. The perturbation shall be a random noise added to the value of f, furthermore the distribution of the noise is dependent on the global sensitivity of f.

Theorem 1 (Laplace Mechanism)**.** *For all $f : D \to \mathbb{R}$, the following algorithm $\mathcal{A}$ is ϵ-differential private:*

$$\mathcal{A}(D) = f(D) + \mathcal{L}(S(f)/\epsilon)$$

where $\mathcal{L}(\lambda)$ is an independently generated random variable following the Laplace distribution with probability density function $g(x) = \frac{1}{2\lambda} e^{-\frac{|x|}{\lambda}}$ and $S(f)$ denotes the global sensitivity of f.

In our case, f represents the aggregate function which is the sum of sensor measurements, and its sensitivity is the maximum of any measurement that a sensor can take. If this value is too large, then every measurement can be truncated to a pre-defined threshold t by each sensor node, hence ensuring that the global sensitivity is at most t in the whole network. Therefore, to guarantee ϵ-differential privacy, we need to add a random value to the aggregate which is sampled form a Laplace distribution with zero mean and variance $2S(f)^2/\epsilon^2$.

Intuitively, if a single measurement can substantially change the output of f, then larger noise needed to be introduced to "hide" the contribution of a single record (sensor) to the aggregate. However, larger noise also deteriorates utility as the final aggregate will be inaccurate. This is a fundamental trade-off between utility and privacy: larger/smaller noise yields stronger/weaker privacy and smaller/larger utility. There seems to be no free lunch. On the other hand, the relative error of the aggregate (or "signal-to-noise ratio") can also be decreased without degrading privacy by aggregating more sensor readings. This is because the aggregate (sum) becomes larger while the added Laplace noise is still calibrated to the global sensitivity which remains unchanged by adding more readings to the aggregate.

IV. Building Blocks

In this section, we introduce some basic building blocks that are used in our solution.

A. Privacy Preserving Primitive: Secure Sum

In SMPC, each participant holds onto a number of their own, and they would like to compute the sum of their inputs. The aggregator – one of the parties – generates a random number R, adds R to its local value and sends the result to the next party. All participants add their local value to the received number. Finally the aggregator receives the sum, subtracts R from the result and broadcasts the result. This guarantees that no one besides the aggregator will learn the correct sum of the values.

B. Homomorphic Encryption

A homomorphism is a structure-preserving map between two algebraic structures. Using homomorphic encryption, computations can be carried out on ciphertext, thus generating an encrypted result which, when decrypted, matches the result of operations performed on the plaintext. We use the Paillier cryptosystem [20] in this paper. In this scheme, if the public key is the modulus m and the base g, then the encryption of a message x is

$$Enc(x) = g^x r^m \mod m^2$$

for some random $r \in \{0, \dots, m-1\}$. The homomorphic property is then

$$\begin{aligned} Enc(x_1) \cdot Enc(x_2) &= (g^{x_1} r_1^m)(g^{x_2} r_2^m) \bmod m^2 \\ &= g^{x_1+x_2}(r_1 r_2)^m \bmod m^2 \\ &= Enc(x_1 + x_2) \end{aligned}$$

C. Robust DFS

We build an aggregation tree using a distributed version of the depth-first search (DFS) algorithm. We need to create univoque routes between all nodes in order to avoid redundant packet channelling and to support aggregation. We must note that this problem is reminiscent of the secure routing problem in wireless or distributed networks, which has been widely studied in the literature [16]. Here, we only provide a basic solution which fits our goal. In particular, a DFS tree results in a definite order of the messages which property is indispensable for our solution to obtain the correct aggregates. The distributed version of the BFS algorithm can be found in [9] [10], the *Robust DFS* algorithm described below is analogous to the BFS one:

Building a Depth First Search Tree:

1) At any point during execution, there are some nodes that are "marked", initially just i_0, the root. The root sends a *search* message at the first round to one of its neighbours.
2) At any round, if an unmarked node receives a *search* message, it marks itself, sets and notifies its parent with a *child* message, and sends *non-child* message to those nodes from which it received a *search* message in earlier rounds.
3) After this, the node sends the *search* message to one of its neighbours.
4) This continues until a *search* message reaches a leaf node. A leaf node realises that it is indeed a leaf-node by receiving *non-child* messages from all of its children candidates (or have only one neighbour).
5) When a node declares itself as a leaf it sends an *end* message back to its parent who then chooses another neighbour of its own and waits for the next *end* message. When it received messages from all of its neighbours, the node sends the *end* it to its parent.
6) The algorithm ends when the root could also send the *end* message.

REMARK: In the non-secure version of the above protocol, a node also sends the list of its parent and children to neighboring nodes, therefore every other node in the graph may be able to reconstruct the tree (or the whole graph) using the received lists of parents and children.

Robustness: The network can lose some nodes due to power failure, hardware or software complications and many more that can cause the node to be detached from the network. Moreover, nodes can potentially be mobile, such as in wireless mobile ad-hoc or vehicular networks. To increase robustness against these failures and potential node mobility, we extend the above algorithm as follows. When a node detects that one (or) some of its neighbours are disconnected, it does the following:

1) if it is a child node, it does nothing;
2) if it is a parent, then
   a) if there is another node during the tree-building phase who was second to become its parent, i.e. sent the node

a *Search* message, then the node notifies them about the change and becomes its child;

b) if there is none like the above the node notifies one of its children to search for a new parent and they switch roles, i.e. the child becomes the parent and vice-versa;

c) if none of the above succeed, or the node has no other neighbours, then it becomes an isolated node (or tree).

Security in the presence of semi-honest adversaries:

- *Privacy:* Complies. No node learns anything about the global graph, they learn only their parent and children nodes, i.e. the local output.
- *Independence of inputs*: Complies trivially.
- *Output delivery:* Complies trivially.
- *Correctness:* Complies trivially.
- *Fairness:* Complies trivially.

Security in the presence of malicious adversaries: Consider the scenario when a node intentionally deviates from the protocol. The only thing it can do without being detected is to set several nodes as its parents, which violates the property of correctness because the malicious node becomes the child of some nodes of which it should not be thereby corrupting their output. In particular, circles may be created in the output graph, which is not a tree and hence an incorrect output of the protocol.

Complexity: The time complexity is at most $2 \cdot diam$ rounds, (where *diam* is the longest shortest path between any two nodes). The transmitted messages are composed of $3(e-1)$ *search* messages, where e denotes the number of edges, i.e., neighboring node pairs, one *child* or *non-child message* and one *end* message between each neighboring pair of nodes. Thus the time complexity is $\mathcal{O}(diam)$ and the communication complexity is $\mathcal{O}(e)$.

D. Distributed Noise Generation

To achieve differential privacy, Theorem 1 suggests that Laplace noise with scale $\lambda = S(f)/\epsilon$ needs to be added to the value of the aggregate function f. A natural question arises: *Which node should add this noise to the aggregate?* As the aggregator is untrusted, the sensor nodes themselves need to add the required amount of noise in a distributed manner. A naive solution would be that a single node is selected to inject all the Laplace noise. However, this approach requires the cooperation of nodes which can be expensive. Also, this makes the protocol less robust against privacy attacks as the selected single node may be malfunctioning (i.e., do not add the noise) or already left the network.

Instead, in our solution, each node *probabilistically and independently* decides whether it adds some noise share to this aggregate such that the sum of these added noise shares yields the required amount Laplace noise needed to guarantee differential privacy. For that, we rely on the following property of the Laplace distribution [21]:

Definition 3 (Geometric Infinite Divisibility)**.** *A random variable Y (and its probability distribution) is said to be* ***geometric infinitely divisible*** *if for any $p \in (0,1)$ it satisfies the relation:*

$$Y \overset{\mathrm{d}}{=} \sum_{i=1}^{\mu_p} Y_p^{(i)}$$

where μ_p is a geometric random variable with mean $1/p$, and the random variables $Y_p^{(i)}$ are independent and identically distributed for each p, and μ_p and $Y_p^{(i)}$ are independent.

The Laplace distribution exhibits the above geometric infinite divisibility, which is shown by the following theorem [21].

Theorem 2. *Let Y possess a Laplace distribution $\mathcal{L}(\lambda)$ with zero mean. Then, Y is geometric infinitely divisible and for any $p \in (0,1)$ the above holds with $Y_{\mu_p}^{(i)} \sim \mathcal{L}(\lambda\sqrt{p})$.*

V. SECURE AGGREGATION

Our solution combines the *Secure Sum* primitive (in Section IV-A) with homomorphic encryption (in Section IV-B) and distributed noise injection to preserve differential privacy (in Section IV-D). A DFS tree is assumed to be already built (as described in Section IV-C) before running our aggregation protocol.

A. Basic protocol

The operation of the aggregator and sensor nodes are shown in Algorithm 1 and 2, respectively.

Algorithm 1 Secure Aggregation: Aggregator node

```
1: Generate public-secret key pairs $(pk_1, sk_1)$ and $(pk_2, sk_2)$
2: $p := 1/N$
3: $G := 0$
4: for all child $j \in [1, m]$ do
5:     Send $\{G,\ p,\ pk_1,\ pk_2\}$ to a child $j$
6:     Receive $\{G_j, c_{j1}, c_{j2}\}$ from child $j$
7:     $G := \vee_{i=1}^{j} G_i$
8: $c_1 := \prod_{j=1}^{m} c_{j1} = Enc_{pk_1}\left(\sum_{i=1}^{N} r_i + \sum_{i=1}^{N} x_i + \sum_{i=1}^{N} Y_i\right)$
9: $c_2 := \prod_{j=1}^{m} c_{j1} = Enc_{pk_2}(\sum_{i=1}^{N} r_i)$
10: Decrypt $(c_1, c_2)$ to retrieve the noisy aggregate $R$, where
11:     $R = Dec_{sk_2}(c_2) - Dec_{sk_1}(c_1) = \sum_{i=1}^{N} x_i + \sum_{i=1}^{N} Y_i$
12:     $\sum_{i=1}^{N} Y_j \sim \mathcal{L}(S/\varepsilon)$
```

We assume that the aggregator (root) node knows the size N of the network. First, the root generates two pairs of homomorphic asymmetric keys (pk_1, sk_1) and (pk_2, sk_2), then sets the noise parameter p (see Definition 2) to $1/N$ and sends the public keys along with p to its children. The children forward this message to their children and so forth, until the message reaches the a leaf node.

When a leaf receives the message of the aggregator from its parent, it tosses a biased coin which results in head with probability p (denoted by $G = 1$ in Alg. 2). If the result of the coin toss is 0, the node generates a Laplace noise with scale $S\sqrt{p}/\epsilon$, where S denotes the global sensitivity of the

Algorithm 2 Secure Aggregation: Non-aggregator (sensor) node

```
1: Receive $\{G, p, pk_1, pk_2\}$ from the parent
2: for all child $j \in [1, m]$ do
3:     Send $\{G, p, pk_1, pk_2\}$ to a child $j$
4:     Receive $\{G_j, c_{j1}, c_{j2}\}$ from child $j$
5:     $G := \vee_{i=1}^{j} G_i$
6: if $G = 1$ then
7:     $Y := 0$
8: else
9:     $G \sim \mathcal{B}(p)$, where $\mathcal{B}$ is the Bernoulli distribution
10:    if $G = 0$ then
11:        $Y \sim \mathcal{L}(S\sqrt{p}/\epsilon)$
12:    else
13:        $Y := 0$
14: Generate $r$ uniformly at random
15: Send $(G, c_1, c_2)$ to the parent, where
16:     $c_1 := Enc_{pk_1}(r + x + Y) \cdot \prod_{j=1}^{m} c_{j1}$
17:     $c_2 := Enc_{pk_2}(r) \cdot \prod_{j=1}^{m} c_{j2}$
```

sum (see Theorem 1), which is the maximum value of any sensor measurement (or its truncation threshold). Afterwards, the node generates a random number r, which is added to the node measurement x and encrypted with key pk_1 to get c_1. In addition, r is also encrypted with the other key pk_2 to get c_2. The encrypted messages c_1 and c_2 along with the result G of the coin toss are sent back to the parent as a reply. Any intermediate node between a leaf and the aggregator repeats the same steps as the leaf node after receiving all reply messages from its children, except that it also checks if it receives $G = 1$ from any of its children, that is, any node in the corresponding subtree observed head by executing the coin toss. If so, it will not add Laplace noise to its own measurement before encryption. As a result, the sensor nodes fan-in the values along the edges of the built DFS tree. When the root receives all reply messages from its children it aggregates them likewise all other sensor nodes, then decrypts the received ciphertexts with secret keys sk_1 and sk_2. After decryption, the aggregator removes the random value $\sum_{i=1}^{N} r_i$ from the aggregate by subtracting $Dec_{sk_1}(c_1)$ from $Dec_{sk_2}(c_2)$, thus obtaining the aggregated measurements with the Laplace noise. Specifically,

$$c_1 = \prod_{i=1}^{m} c_{i1} = Enc_{pk_1}\left(\sum_{i=1}^{N} r_i + \sum_{i=1}^{N} x_i + \sum_{i=1}^{N} Y_i\right)$$

$$c_2 = \prod_{i=1}^{m} c_{i1} = Enc_{pk_2}\left(\sum_{i=1}^{N} r_i\right)$$

where x_i and r_i are the measurement and random value generated by node i, respectively, and Y_i is the noise share (which is 0 or follows $\mathcal{L}(S\sqrt{p}/\epsilon)$) added to the aggregate by node i. Therefore,

$$Dec_{sk_2}(c_2) - Dec_{sk_1}(c_1) = \sum_{i=1}^{N} x_i + \sum_{i=1}^{N} Y_i$$

and hence the aggregator obtains the noisy aggregate which is ϵ-differential private due to Theorem 2 and 1. In particular, each node repeats the coin tossing until the first node succeeds to get 1 (head), which then notifies the rest of the nodes that they do not need to generate more noise, hence $Y = 0$ for all subsequent nodes. In other words, we have a geometrically distributed number of random values drawn from the Laplace distribution with scale $\mathcal{L}(S\sqrt{p}/\epsilon)$, which means that $\sum_{i=1}^{N} Y_i$ follows a Laplace distribution with $\mathcal{L}(S/\epsilon)$ based on Theorem 2. Moreover, as $p = 1/N$, all nodes generate Laplace noise exactly once with large probability. Nevertheless, p is a parameter which can be set depending on node failures hence providing increased robustness and flexibility over prior works.

Notice that, in the geometric distribution, we must have infinite possibilities to succeed, which might not be the case if each node tosses the coin exactly once (i.e., with some positive, albeit small probability none of the nodes have head after finishing the above protocol, which means that the Laplace noise shares will not sum up to the required Laplace noise needed for ϵ-differential privacy). To alleviate this problem, we allow the nodes to make as many rounds as needed for one successful coin toss in the DFS tree. Therefore, we need to repeat the above protocol until at least one head occurs at any node where each node adds Laplace noise to the aggregate (without adding their measurement x to the random r in Line 16 of Alg. 2).

Motivation of geometric divisibility: There have been proposed several schemes for distributed noise generation to guarantee differential privacy [14], all of them are based on the divisibility of the Laplace distribution. The reason we chose geometric divisibility for our model lies in its flexibility. In particular, any network nodes may fail from time to time due to various reasons; if we used any technique described in [14], the failed nodes would not add noise to the sum which would imply extra noise generation tasks from other nodes, or the resetting of the parameters of the distribution. One workaround for this problem could be to select a subset of all nodes for the noise generation task, but this is difficult for distributed systems. In our scheme, there is no need for such coordination; upon the detection of the failure of a child node, the parent can report its own measurement towards its parent without any modification of the protocol. This is because G in Algorithm 2 is drawn independently at each sensor node. Moreover, the probability p of this biased coin toss can also be flexibly adjusted depending on the anticipated number of node failures which also provides stronger robustness.

B. Extension to a malicious aggregator

In Algorithm 1, the aggregator can easily decrypt the partial sums, corresponding to each of its children, before forwarding that to other children, thus it can learn the partial sums of the aggregate. To overcome this problem, the *children* of the aggregator can add additional random value that cancel out when their partial sums are summed. For example, if node P_i adds $+R_{i,j}$, then another child P_j of the root node needs to add $-R_{i,j}$ to its partial sum[3]. This way the aggregator must add all

[3]A similar method is used in [15]

the partial sums belonging to each of its children in order to get the correct aggregate. This implies that a secure channel needs to be established between any pair of children to agree on the value of $R_{i,j}$. For example, the children of the aggregator can derive a shared secret key K, using any key exchange protocol such as Diffie-Hellman, which is known to all the children of the aggregator except the aggregator[4]. Then, similarly to [15], $R_{i,j}$ can be computed as $R_{i,j} = PRF(K, P_i, P_j)$ where $PRF(\cdot)$ is a pseudo-random function.

If the aggregator is malicious, it can also misbehave by lying to one of its child node that $G = 1$ (i.e., it falsely claims that there has been a sensor node before whose coin toss resulted in head). This causes less Laplace noise to be added in the network than what is required to achieve ϵ-differential privacy. We can however easily amend Algorithm 1 to resist against such attack by using key K shared between all children of the aggregator. In particular, the children of the aggregator can compute a message authentication code (MAC) using K on the value of G and attach this MAC to the message sent to the aggregator. This MAC is required to be forwarded by the aggregator to other children as part of the message in Line 5 of Algorithm 1, and hence, any child node can detect, by verifying the MAC, if the aggregator modifies G.

C. Analysis

First, consider the requirements of security based on secure multi-party computation described in our *Privacy and Security Model*.

Security in the presence of semi-honest adversaries:

- *Privacy:* Complies. No node learns anything more than its prescribed output. The root learns some additional information - the partial sums of its children and the partial sums of the random numbers-, but it cannot derive the individual inputs.
- *Independence of inputs:* Complies. All inputs are encrypted, one node has to break the encryption in order to learn anything.
- *Output delivery:* Complies trivially.
- *Correctness:* Complies trivially.
- *Fairness:* Complies trivially.

Security in the presence of malicious adversaries: The privacy of the above SMPC still complies. If the root is the adversary unfortunately even independence fails, since the root can in any way alter the result depending on the received sum. Moreover, with this kind of an adversary we cannot guarantee any other requirement. Although, if an inner node is malicious independence complies, but the rest are do not regarding the subtree under the malicious node.

The reason for adding random numbers to the output in the algorithm is the following. Assume that we expect the inputs to be in a closed, short interval, taking discrete values. In such a case anyone could try all values by applying *brute-force* attack to get the information. The pairs of keys are used for the following reason. If we used no encryption, noise nor random numbers we could face several attack scenarios. For example, if the inputs were very diverse, then a partial sum at an inner node would not carry any information about the underlying sub-tree or the global graph for a node. However if the values are similar one could easily approximate the size of a subtree or even the whole graph. Finally, if an eavesdropper had access to one node's incoming and outgoing messages it could easily calculate the node's input, and also the partial sum sent to the node.

The second key which encrypts the (sum of) random numbers is necessary because we can eliminate the first attack mentioned above. The first key - which encrypts the sum of the measurements, the noise and the random number(s) - is used for preventing the eavesdropper's attack, but if we used solely this key, then an eavesdropper could still capture the incoming and the outgoing messages of one node, encrypt the random numbers with the broadcasted key, and by simple subtraction learn that nodes measurement. Thus the need for two different homomorphic encryption keys.

Differential privacy: According to Theorem 2 on the geometric divisibility of the Laplace distribution with $\mathcal{L}(\lambda)$, the algorithm generates μ_p values drawn from the Laplace distribution with parameters $\mathcal{L}(\lambda\sqrt{p})$, where μ_p is a random variable having geometric distribution. The sum of these values has distribution $\mathcal{L}(s)$. The aggregated result with the noise is in line 11 of algorithm 1:

$$\sum_{i=1}^{N} x_i + \sum_{j=1}^{\mu_p} Y_j = \sum_{i=1}^{N} x_i + Y_{\mathcal{L}(\lambda)}$$

where Y has Laplace distribution $\mathcal{L}(\lambda)$.

We used a biased coin with probability $p = 1/N$, because the geometric distribution is the probability distribution of the number X of Bernoulli trials needed to get one success, supported on the set $1, 2, 3, ...$; therefore we get a geometric random variable number of nodes. Therefore the nodes align in a sequence of the DFS tree and keep on sequentially flipping a biased coins independently from each other until they get the first success. Furthermore, we want to set the probability of success of the geometric distribution to $1/N$, where N is the number of nodes in the graph. Thus the expected value of a random variable with geometric distribution is $1/p = N$, hence with high probability we are going to have all the nodes in the graph adding noise to their output exactly once. Since the geometric distribution has infinite support we cannot limit the number of trials to N. Therefore if all nodes have already tossed a coin but none succeeded, then we must restart the experiment from the first node until we succeed, thus we gain the infinite support of trials.

In summary, since Theorem 2 is fulfilled by the algorithm it follows that differential privacy is preserved.

VI. Conclusion

In this paper we have presented a new method for preserving ϵ-differential privacy in a distributed sensor system. We have

[4]Notice that sensor nodes (i.e., producers) do not collude with the aggregator (i.e., organization)

presented our model as a strongly connected graph having processing units at the nodes. These participants want to engage in a privacy preserving computation to gain the aggregated result of their measurements. We have applied secure multiparty computation protocols as basic building blocks to preserve security; moreover, we have introduced a new distributed noise generation protocol, where we used the geometric infinite divisibility of the Laplace distribution. This distribution has a property, namely the geometrically distributed number of noise segments, that we utilised to make our protocol more robust against node failures, and flexible in the expected value of the number of nodes participating in the noise generation.

Building an even more secure version of an algorithm as always is a challenge. Here the next step could be to create secure versions under the assumption of malicious nodes. Considering pseudo-random generators and synchronization in order to test a node's honesty is a possible approach.

Future work: This paper has focused on the feasibility of privacy preserving data aggregation over Multi-hop networks. Performance evaluation of the proposed algorithms in both wireless and wired multi-hop networks, incorporating energy efficiency and computational constraints, constitutes important future work.

Acknowledgment

First, I would like to express my special thanks of gratitude to Gergely Ács and to Gergely Biczók who provided insight and expertise that greatly assisted my work. Second, I would like to thank my M.Sc. thesis adviser, András Lukács for his interesting ideas and guidance.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE communications magazine*, vol. 40, no. 8, pp. 102–114, 2002.

[2] J. Luo and J.-P. Hubaux, "Joint mobility and routing for lifetime elongation in wireless sensor networks," in *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 3. IEEE, 2005, pp. 1735–1746.

[3] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu, "Tools for privacy preserving distributed data mining," *ACM SIGKDD Explorations Newsletter*, vol. 4, no. 2, pp. 28–34, 2002.

[4] C. C. Aggarwal and S. Y. Philip, *A general survey of privacy-preserving data mining models and algorithms*. Springer, 2008.

[5] Y. Lindell and B. Pinkas, "Secure multiparty computation for privacy-preserving data mining," *Journal of Privacy and Confidentiality*, vol. 1, no. 1, p. 5, 2009.

[6] R. Cramer and I. Damgård, *Multiparty Computation, an Introduction*. Basel: Birkhäuser Basel, 2005, pp. 41–87. [Online]. Available: http://dx.doi.org/10.1007/3-7643-7394-6_2

[7] A. C. Yao, "Protocols for secure computations," in *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*. IEEE, 1982, pp. 160–164.

[8] O. Goldreich, "Secure multi-party computation," *Manuscript*, 1998.

[9] M. Raynal, *Distributed Algorithms for Message-Passing Systems*. Berlin, Germany: Springer-Verlag, 2013.

[10] N. A. Lynch, *Distributed Algorithms*. The Morgan Kaufmann Series in Data Management Systems, 1996.

[11] T. Raeder, M. Blanton, N. V. Chawla, and K. Frikken, "Privacy-preserving network aggregation," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2010, pp. 198–207.

[12] J. Brickell and V. Shmatikov, "Privacy-preserving graph algorithms in the semi-honest model," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2005, pp. 236–252.

[13] K. B. Frikken and M. J. Atallah, "Privacy preserving route planning," in *Proceedings of the 2004 ACM workshop on Privacy in the electronic society*. ACM, 2004, pp. 8–15.

[14] S. Goryczka, L. Xiong, and V. Sunderam, "Secure multiparty aggregation with differential privacy: a comparative study," in *Proceedings of the Joint EDBT/ICDT 2013 Workshops*. ACM, 2013, pp. 155–163.

[15] G. Ács and C. Castelluccia, "I have a dream!(differentially private smart metering)," in *International Workshop on Information Hiding*. Springer, 2011, pp. 118–132.

[16] Y. C. Hu and A. Perrig, "A survey of secure wireless ad hoc routing," *IEEE Security and Privacy*, pp. 28–39, 2004.

[17] L. Kissner and D. Song, "Privacy-preserving set operations," in *Advances in Cryptology–CRYPTO 2005*. Springer, 2005, pp. 241–257.

[18] B. P. Yehuda Lindell, "Privacy preserving data mining," *Annual International Cryptology Conference*, pp. 36–54, 2000.

[19] C. Dwork, "Differential privacy: A survey of results," in *International Conference on Theory and Applications of Models of Computation*. Springer, 2008, pp. 1–19.

[20] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 223–238, 1999.

[21] T. K. Kotz, Samuel and K. Podgorski., *The Laplace Distribution and Generalizations*. Springer, 2001.

Szilvia Lestyán received B.Sc. and M.Sc. degrees both in applied mathematics from the Eötvös Loránd University, Budapest, Hungary, in 2013 and in 2015 respectively. Currently she is applying for a doctorate course (PhD) in informatics at the Budapest University of Technology and Economics, where she will conduct research at the Laboratory of Cryptography and System Security (Crysys Lab) in the field of machine learning in privacy and security.