Practical Private Information Aggregation in Large Networks

Gunnar Kreitz, Mads Dam, and Douglas Wikström

KTH—Royal Institute of Technology
Stockholm
Sweden

Abstract. Emerging approaches to network monitoring involve large numbers of agents collaborating to produce performance or security related statistics on huge, partial mesh networks. The aggregation process often involves security or business-critical information which network providers are generally unwilling to share without strong privacy protection. We present efficient and scalable protocols for privately computing a large range of aggregation functions based on addition, disjunction, and max/min. For addition, we give a protocol that is information-theoretically secure against a passive adversary, and which requires only one additional round compared to non-private protocols for computing sums. For disjunctions, we present both a computationally secure, and an information-theoretically secure solution. The latter uses a general composition approach which executes the sum protocol together with a standard multi-party protocol for a complete subgraph of "trusted servers". This can be used, for instance, when a large network can be partitioned into a smaller number of provider domains.

Keywords. Multi-party computation; Private aggregation; Partial mesh network

1 Introduction

With the continuous increase of network complexity and attacker sophistication, the subject of network and security monitoring becomes increasingly important. Traditionally, organizations have performed network and security monitoring based only on data they can collect themselves. One of the reasons for this is a reluctance to share traffic data and security logs between organizations, as such data is sensitive.

There is much to be gained from collaboration in security monitoring. Attacks range from being targeted at specific individuals or organizations, to global scale attacks such as botnets. Naturally, the response measures depend on the type of attack. The same situation applies to network monitoring, where the complexity of networks, and large amount of applications can make it difficult to distinguish between local and global disruptions with access only to local data.

A natural path towards a solution is to use multi-party computation (MPC) techniques, which have been long studied within the field of cryptography. The goal of MPC is to allow a group of mutually distrusting parties to jointly evaluate a function of their private inputs, while leaking nothing but what can be deduced from the output of the function. Furthermore, protocols built on MPC techniques are generally secure, even if several parties (up to a fraction of the parties involved in the computation) collude to break the privacy of the other participants.

The traditional setting of MPC is one where the number of parties is relatively small and the network is assumed to be full mesh. Sadly, this precludes the immediate application of such techniques in the large, partial mesh networks which are prevalent today.

Recent approaches to monitoring in large networks employ an in-network paradigm [1] whereby monitoring is performed collaboratively by the network nodes themselves, using algorithms based on spanning trees [26,11] or gossiping [24,22]. For these applications, scalability is often taken to mean sub-linear growth in resource consumption growth in the size of the network.

Towards a general solution to the problem of collaborative network and security monitoring we present in this paper efficient protocols for computing sum, max, disjunction, and thresholds in partial mesh networks. These operations are sufficient to implement many of the aggregates of interest in monitoring. Our protocols are efficient, both in terms of message and computational overhead.

We focus in this paper on passive, "honest-but-curious" adversaries whereby attackers are bound to follow the protocol but may collude to learn information about the honest parties' inputs. This is much simpler than the active attack model also considered in multi-party computation and often leads to more efficient protocols. However, it is also a reasonable and attractive model in many practical situations where e.g. side conditions related to traffic observations and arguments of utility can be appealed to to ensure protocol behavior is adhered to.

The security of MPC protocols is commonly characterized by the size of collusions they remain secure against. Such thresholds become less meaningful for protocols, such as ours, which can be used on arbitrary networks. Thus, we analyze security in terms of tolerable adversary structures in the sense of Hirt and Maurer [20], and describe the tolerable structures in terms of graph theoretical properties of the network on which the protocol is executed.

As the need for monitoring is common to many areas, and our protocols are efficient, we believe there is a wide range of applications. We give a few examples of possible applications to set some context for the work.

Example 1 (Collaborative Security Monitoring). The need to aggregate security log information as part of general intelligence gathering is widely acknowledged, cf. [29]. The importance of collaboration is further emphasized by services such as Internet Storm Centre's `www.dshield.org`, where firewall logs can be shared, and aggregate statistics are collected.

Network providers and supervisors have strong interest in accurate security log aggregates, as this will allow more precise estimations of the global security situation, in order to take countermeasures and improve operations. There are, however, important privacy concerns, as log data, even in sanitized form, can reveal significant amounts of critical information concerning internal business and network operations. Previous work has explored techniques such as log anonymization and randomized alert routing to deal with this problem [29,25]. We argue that private aggregation techniques can be used in this scenario to produce practical security aggregates with strong privacy guarantees in near real time.

One application would be to collect aggregate packet- or flow counts to various destination ports. Due to the computational efficiency of our protocols, they could be run directly on network devices such as routers, and without the need to trust a third party.

Example 2 (Anonymous and robust peer-to-peer networks). Consider a peer-to-peer network for anonymous publication and retrieval of files where the network acts as a distributed storage. In this scenario, it could be of interest to compute the number of copies of a file to discover if further duplication of that file is needed, something that could be done by a private computation of a sum. It may also be useful to be able to query for availability of a file without learning any other information than if the file exists in the network or not, which would correspond to a private computation of disjunction.

Another application within the realm of peer-to-peer networking would be to implement monitoring of the overlay to enhance quality and research. This could be useful both for overlays with strict anonymity requirements, but also for more traditional file-sharing applications where individual users may still be hesitant to share information on e.g. the amount of data they've uploaded.

Example 3 (Joint control of SCADA systems). A research topic of growing importance is the security of Supervisory Control and Data Acquisition (SCADA) systems, e.g. systems controlling criticial infrastructure such as the electrical grid. Many different entities are involved in running the electrical grid, and they must co-operate to ensure production and consumption is balanced throughout the grid. However, many of the entities are direct competitors, which can prevent collaboration that would involve sharing of business-sensitive data.

Our protocols could be applied to monitor aggregate power flows over various areas of the grid, which is a summation. They could also be applied to compute the disjunction of alert statuses at operators. Then, if one operator has some form of disruptions, other operators would automatically be put on alert and be prepared in case the failure condition affects other parts of the grid. This would decrease the risk of cascading failures by giving early warnings to other operators, without sharing detailed information on the reliability of any individial operator.

We believe that in the scenarios presented above, the assumption of a passive adversary could be reasonable. For network monitoring, there is little to be gained for the participants in disrupting the computation of the aggregated information. In the P2P scenario, attacking monitoring is likely to be uninteresting, but searches and functions ensuring replication may be suitable candidates for protocols with stronger security properties, depending on the nature of the network. In the SCADA scenario, in addition to the small gains from actively manipulating the computations, it's possible that legislation would demand that data was retained for auditing, thus increasing the risk involved in cheating.

1.1 Our Contributions

Firstly, we give a protocol for summation, where we perform a single round of communication to achieve privacy, and then reduce the problem to non-private summation. A single group element is sent in each direction over every link in this extra round. The protocol is similar to a protocol by Chor and Kushilevitz [10], but adapted to a partial mesh network, and with a precise characterization of tolerable adversary structures. It is also similar to the dining cryptographers networks proposed by Chaum [9] which is essentially the same protocol but applied to provide sender untraceability.

Secondly, we present a computationally secure protocol for computing disjunction, based on homomorphic cryptosystem, such as El Gamal [15]. The protocol requires two rounds of communication and then uses a non-private protocol for summation. Computationally, it requires a small number of encryptions and decryptions per neighbor.

We also give a composition structure where the information-theoretically secure protocol for summation is composed with a standard protocol for computing some other function. We show that this can be used for several standard functions in network management, such as disjunction, min/max, or threshold detection. For this composition, there needs to be a complete subgraph K of the network such that no union of two sets from the adversary structure contains K. This is a reasonable assumption in many network monitoring applications where the members of K represent trusted servers appointed by a disjoint collection of network providers. This is similar to the use of trusted aggregation servers in [5,13,7]. The composition essentially performs an efficient and secure "aggregation" of all inputs to some smaller subset of parties who can then run a more expensive protocol with stricter connectivity requirements.

1.2 Related Work

There are general results [18,3] showing that every computable function can be privately evaluated in a multi-party setting, but the protocols involved require a full mesh network between the parties and can be prohibitively expensive to execute.

There are many specialized protocols for computing specific functions in the literature, that are more efficient than the general constructions. Examples of such protocols include an information-theoretically secure protocol for summation by Chor and Kushilevitz [10], and computationally secure protocols for disjunction and maximum by Brandt [6], which uses the homomorphic El Gamal cryptosystem as a building block. While such protocols are more efficient than the general solutions, they are still not scalable in the sense of the previous section. Just sending one message between every pair of parties forces each party to process too many messages.

In most of the works on multi-party computation, the parties are connected in a full mesh network. An article by Franklin and Yung [14] describes how to emulate the missing private channels between parties, and using their construction, protocols built for full mesh networks may also be run on arbitrary networks. However, this emulation can be very expensive, and may not always be possible, depending on what parties an adversary can corrupt.

There has also been research exploring how the network connectivity affects what functions can be computed with information-theoretical privacy. There are results due to Bläser *et al.* [4] and Beimel [2] categorizing the functions that can be computed on 1-connected networks.

The Dining Cryptographers problem, and its solution were discussed by Chaum [9]. They study the problem of creating a channel such that the sender of messages is untraceable and their suggested protocol is similar to our protocol for summation.

A technique that can be applied to sidestep the connectivity and performance issues of traditional MPC solutions is to aggregate data to a small set of semi-trusted nodes, who can then perform the computation. As these servers are few, it is more feasible to connect them with a full mesh network. Examples of such schemes include Sharemind [5], SEPIA [7], and a system by Duan and Canny [13]. These are similar to the protocols we present in Section 5, with a difference being that our protocols perform aggregation while collecting information from the nodes, thus decreasing the load on the servers performing the computation, but limiting what can be computed.

A number of authors propose additive secret sharing to secure information aggregation in large networks or databases. Privacy schemes similar to the sum protocol used here have been explored in the area of sensor networks and data mining [28,19]. In fact, a very large range of algorithms used in data mining and machine learning, including all algorithms in the statistical query framework [23], can be expressed in a form compatible with additive secret sharing. Several authors have investigated secure aggregation schemes for the case of a centralized aggregator node (cf. [21,27]). A solution with better scalability properties is proposed by Chan *et al.* [8]. There, an additive tree-based aggregation framework is augmented by hash signatures and authenticated broadcast to ensure that, assuming the underlying aggregation tree is already secured, an attacker is unable to induce an honest participant to accept an aggregate which could not be obtained by direct injection of some private data value at the attacking node. Other recent work with similar scope uses Flajolet-Martin sketches for secure counting and random sampling [16].

1.3 Organization of This Paper

We begin by presenting the security and computational model and various definitions in Section 2. We then proceed to outline and prove properties of the protocol for computing sums in Section 3. Then, we give a computationally secure protocol for computing disjunctions in Section 4. We then show a composition structure where the protocol for summation is composed with standard protocols to compute for instance disjunction in Section 5.

2 Model and Definitions

We consider multi-party computation (MPC) protocols for n parties, $P_1, \ldots, P_n$, and denote the set of all parties by $\mathcal{P}$. Each party P_i holds a private input, x_i, and the vector of all inputs is denoted x. The network is modeled as an undirected graph $\mathcal{G} = (\mathcal{P}, \mathcal{E})$ where messages can only be sent between adjacent parties.

For a graph $\mathcal{G} = (\mathcal{P}, \mathcal{E})$, we say that $\mathcal{G}$ is *disconnected* if there exists a pair of vertices such that there is no path between them. For a set of vertices $X \subseteq \mathcal{P}$, we denote by $\mathcal{G} - X$ the subgraph of $\mathcal{G}$ induced by the set of vertices $\mathcal{P} \backslash X$. In other words, $\mathcal{G} - X$ is the graph obtained by deleting all vertices in X and their incident edges from $\mathcal{G}$.

Definition 1 (Separator, set of vertices). *Given a graph $\mathcal{G} = (\mathcal{P}, \mathcal{E})$, a set of vertices $X \subseteq \mathcal{P}$ is called a* separator *of $\mathcal{G}$ if the graph $\mathcal{G} - X$ is disconnected.*

2.1 Adversary Structures

The most common adversary considered in the MPC literature is a threshold adversary corrupting up to a threshold of the parties. More generally, we can allow an adversary corrupting some subset of parties as specified by an *adversary structure* [20].

An adversary structure $\mathcal{Z}$ over $\mathcal{P}$ is a subset of the power set of $\mathcal{P}$, containing all possible sets of parties which an adversary may corrupt. We require that an adversary structure is monotone, i.e., it is closed under taking subsets.

Definition 2 (Separator, adversary structure). *In a network $\mathcal{G} = (\mathcal{P}, \mathcal{E})$, an adversary structure $\mathcal{Z}$ is called a* separator *of $\mathcal{G}$ if some element in $\mathcal{Z}$ is a separator of $\mathcal{G}$.*

From the monotonicity of $\mathcal{Z}$, it follows that if $\mathcal{Z}$ is not a separator of $\mathcal{G}$, then no matter what subset in $\mathcal{Z}$ the adversary chooses to corrupt, every corrupted party will have at least one honest neighbor. More precisely, for every set $C \in \mathcal{Z}$ it must be the case that every party $P \in C$ has at least one neighbor who is not in C. This observation is important for the proof of security of the computationally private protocol for disjunction given in Section 4.

2.2 Security Definition

The security definition of a multi-party computation says that the adversary should not learn anything from the protocol execution except what it can deduce from its inputs and the output of the function the protocol computes.

In the security analysis of our protocols, we only consider passive (*honest-but-curious*), static adversaries in a network with private and reliable channels. The protocols in Sections 3 and 5 are information-theoretically private, and the protocol in Section 4 is computationally private.

We consider information about the network the protocol is executed on to be public knowledge. Our protocols do not depend on honest parties knowing the network structure, but neither do anything to hide that information from the adversary.

We refer to [3,17] for details on security definitions for information-theoretical and computational security of multi-party computation.

2.3 Homomorphic Cryptosystems

A cryptosystem $\mathsf{CS} = (\mathsf{Gen}, \mathsf{E}, \mathsf{D})$ is said to be homomorphic if the following holds.

- Each public key pk output by Gen defines groups of messages, randomness, and ciphertexts, denoted $\mathcal{M}_{pk}$, $\mathcal{R}_{pk}$, $\mathcal{C}_{pk}$ respectively, for which the group operations are efficiently computable.
- For every public key pk, every messages $m_1, m_2 \in \mathcal{M}_{pk}$, and every $r_1, r_2 \in \mathcal{R}_{pk}$: $\mathsf{E}_{pk}(m_1, r_1)\mathsf{E}_{pk}(m_2, r_2) = \mathsf{E}_{pk}(m_1 + m_2, r_1 + r_2)$.

It is convenient in our applications to use additive notation for both the group of messages and the group of randomness. However, we do not require that the cryptosystem is "additively homomorphic", e.g., that $\mathcal{M}_{pk} = \mathbb{Z}_m$ for some from integer m. Thus, any homomorphic cryptosystem with sufficiently large message space suffices, e.g., El Gamal. We remark that we do not use the fact that the cryptosystem is homomorphic over the randomness.

3 Computing Sums

We present an information-theoretically secure protocol for computing sums over a finite Abelian group. The protocol is similar to a protocol by Chor and Kushilevitz [10], but adapted to arbitrary networks, and with a precise characterization of tolerable adversary structures. It is also similar to a protocol by Chaum [9], with the difference that we explicitly create shared random secrets by a straightfoward technique and use the protocol for summation rather than sender-untraceability.

When computing sums, privacy comes cheap. We can take any non-private protocol for sums, $\mathtt{NonPrivateSum}(x_1, \ldots, x_n)$, and augment it with a single additional round to turn it into a private protocol. The protocol admits all adversary structures $\mathcal{Z}$ which do not separate the network $\mathcal{G}$. This requirement on the adversary structure is necessary in the information-theoretical setting.

Protocol 1 (Sum). In the protocol for computing $\sum_{i=1}^{n} x_i$ over an Abelian group $\mathcal{M}$, on the network $\mathcal{G} = (\mathcal{P}, \mathcal{E})$, $P_i \in \mathcal{P}$ proceeds as follows:

1. For each neighbor P_j, pick $r_{i,j} \in \mathcal{M}$ randomly and send it to P_j.
2. Wait for $r_{j,i}$ from each neighbor P_j.
3. Compute $s_i = x_i - \sum_{(P_i,P_j)\in\mathcal{E}} r_{i,j} + \sum_{(P_i,P_j)\in\mathcal{E}} r_{j,i}$.
4. Output $\texttt{NonPrivateSum}(s_1, \ldots, s_n)$.

We begin by observing that the protocol correctly computes the sum of the inputs x_i. For every value $r_{i,j}$ sent in step 1 of the protocol, that value is added to s_j and subtracted from s_i, so all $r_{i,j}$ cancel when summing the s_i.

We now show that the protocol is information-theoretically private with respect to passive, static adversaries. We do this by showing that for any non-separating collusion, the remaining s_i values are uniformly random, conditioned on $\sum_{i=1}^{n} s_i = \sum_{i=1}^{n} x_i$.

Theorem 1. *Protocol 1 is information-theoretically private to a passive and static adversary if the adversary structure $\mathcal{Z}$ does not separate the network $\mathcal{G} = (\mathcal{P}, \mathcal{E})$.*

To prove the theorem, we begin by stating a lemma from which the theorem follows immediately.

Lemma 1. *Consider executions of Protocol 1 on a network $\mathcal{G} = (\mathcal{P}, \mathcal{E})$ where: the output $\sum_{i=1}^{n} x_i$, a non-separating collusion $\mathcal{C}$, and the inputs x_i and communication $r_{i,j}, r_{j,i}, s_i$ for $P_i \in \mathcal{C}$ are fixed. For such executions the remaining values s_i for $P_i \in \mathcal{P}\backslash\mathcal{C}$ are uniformly random, conditioned on $\sum_{i=1}^{n} s_i = \sum_{i=1}^{n} x_i$.*

Proof (Theorem 1). The values $r_{i,j}$ sent in the first round are independent of the input. By Lemma 1, for any fixed input and random tapes of a non-separating collusion, and fixed output of the protocol, the remaining messages have the same distribution. $\square$

Proof (Lemma 1). Consider two vectors $s = (s_1, \ldots, s_n)$, $s' = (s'_1, \ldots, s'_n)$, and two vectors of inputs $x = (x_1, \ldots, x_n), x' = (x'_1, \ldots, x'_n)$ such that $\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} x'_i = \sum_{i=1}^{n} s_i = \sum_{i=1}^{n} s'_i$, and $s_i = s'_i, x_i = x'_i$ for all $P_i \in \mathcal{C}$.

Let R denote an $n \times n$ matrix of $r_{i,j}$, where $r_{i,j} = 0$ if (P_i, P_j) is not an edge in $\mathcal{G}$. Define $s(x, R)$ to be the vector of s_i values sent in the protocol when executed on input x with random values R. The value at the ith position of $s(x, R)$ is denoted by $s_i(x, R)$.

We show that the probability of s being sent on input x is equal to the probability of s' being sent on input x'. This is done by, for any tuple of vectors s, s', x, x' constructing a bijective function $f(R)$ such that if $s = s(x, R)$ then $s' = s(x', f(R))$. The function $f(R)$ has the form $f(R) = R + R'$ for an $n \times n$ matrix $R' = (r'_{i,j})_{i,j}$. Furthermore, $r'_{i,j} = 0$ if $P_i \in \mathcal{C}$ or $P_j \in \mathcal{C}$.

We note that $s = s(x, R)$ holds iff R is such that for each P_i we have $s_i - x_i = \sum_{j=1}^{n} r_{j,i} - r_{i,j}$. Thus, for R' we need precisely that for each P_i we have

$$\sum_{j=1}^{n} (r'_{j,i} - r'_{i,j}) = (s'_i - x'_i) - (s_i - x_i)\,. \tag{1}$$

Since $\mathcal{C}$ is not a separator, there exists a directed spanning tree T that spans the honest parties, $\mathcal{P} \backslash \mathcal{C}$. Let $r'_{i,j} = 0$ if (P_i, P_j) is not an edge in T. We can now fill in R' iteratively during a postorder traversal of T. When a non-root P_i is visited, only $r'_{j,i}$ for its parent P_j is still undefined on the ith row and column of R', and its value is determined by Equation 1.

When the root is visited, R' is completely filled in and we know that Equation 1 holds for all other parties. Consider the sum of Equation 1 over all parties. The left hand side satisfies $\sum_{i=1}^{n} \sum_{j=1}^{n} (r'_{j,i} - r'_{i,j}) = 0$. The right hand side also satisfies $\sum_{i=1}^{n} (s'_i - x'_i) - (s_i - x_i) = 0$ since $\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} x'_i = \sum_{i=1}^{n} s_i = \sum_{i=1}^{n} s'_i$. Since Equation 1 holds for all parties except for the root, it must also hold for the root. $\square$

We would like to remark that the proof of Lemma 1 does not make use of the monotonicity of the adversary structure $\mathcal{Z}$. Thus, if we allow non-monotone adversary structures (for instance, if parties 1 and 2 must always be corrupted jointly), the protocol is still private given that $\mathcal{Z}$ does not separate the network $\mathcal{G}$.

It is intuitively clear that sums cannot be privately computed if $\mathcal{Z}$ separates the network, and this is indeed the case. In [2], Beimel gives a characterization of the functions that can be privately computed in non-2-connected networks, with an adversary structure consisting of all singleton sets, and shows that sums cannot be computed in that setting. Any information-theoretically private protocol computing sums tolerating $\mathcal{Z}$ separating the network can be turned into a protocol violating the bounds given in [2] by standard simulation techniques, and cannot exist.

4 A Computationally Secure Protocol For Disjunction

We now consider the problem of computing the disjunction of all parties' inputs, and present a computationally secure protocol, requiring two rounds of communication and an execution of non-private protocol for summation.

As a building block, we need a cryptosystem $\mathsf{CS} = (\mathsf{Gen}, \mathsf{E}, \mathsf{D})$ that is homomorphic. We further need that the group of messages $\mathcal{M}_{pk}$ is the same group for all keys generated with the same security parameter, κ. For notational convenience, we denote this group $\mathcal{M}$. We require the cryptosystem to have IND-CPA security, i.e., resistance to chosen-plaintext attacks. We relax the correctness requirements slightly, and allow our protocol to incorrectly output `false` with negligible probability $2^{-\kappa}$.

In this protocol, we construct a linear secret sharing of a group element which is zero if all the parties' inputs are `false`, and a uniformly random group element otherwise. The protocol then proceeds by opening the share, which is done by (non-private) summation.

Conceptually, each party contributes either a zero or a random group element, depending on its input. However, it is important that a party does not know the group element representing its own input, as this would allow it to recognize if it was the only party with input `true`. In order to achieve this, we apply homomorphic encryption to allow its neighbors to jointly select how its input is represented.

If the security requirements are relaxed slightly, and it is acceptable that the adversary can learn if any other parties had input `true`, then Protocol 1 can be used instead (with each party herself choosing 0 or a random element as her input).

For ease of notation, we identify `false` with 0, and `true` with 1. In the description of the protocol, we abuse notation slightly and multiply a value by a party's input as a shorthand for including or excluding terms of a sum.

Protocol 2 (Disjunction). In the protocol for computing $\texttt{Or}(x_1, \ldots, x_n)$, where $x_i \in \{0,1\}$, on the network $\mathcal{G} = (\mathcal{P}, \mathcal{E})$, based on a homomorphic cryptosystem $\mathsf{CS} = (\mathsf{Gen}, \mathsf{E}, \mathsf{D})$, $P_i \in \mathcal{P}$ proceeds as follows:

1. Generate a key-pair $(pk_i, sk_i) \leftarrow \mathsf{Gen}(1^\kappa)$.
2. For each neighbor P_j, pick a random element $a_{i,j} \in \mathcal{M}$, and send $pk_i, c_{i,j} = \mathsf{E}_{pk_i}(a_{i,j})$ to P_j.
3. Upon receiving $pk_j, c_{j,i}$ from P_j, pick a random $r_{i,j} \in \mathcal{M}$, and send $c'_{i,j} = \mathsf{E}_{pk_j}(r_{i,j}) + x_i c_{j,i}$ to P_j.
4. Wait for $c'_{j,i}$ to be received from every neighbor P_j, and then compute $s_i = \sum_{(P_i,P_j)\in\mathcal{E}} (\mathsf{D}_{sk_i}(c'_{j,i}) - r_{i,j})$
5. Compute $\texttt{NonPrivateSum}(s_1, \ldots, s_n)$ and output 0 if the sum is the identity, and 1 otherwise.

The protocol is efficient, both in terms of computational resources and communication. Each party needs to perform two encryptions, one decryption and one ciphertext multiplication per neighbor. The first encryption does not depend on the input, and can be performed off-line. The communication overhead of the protocol is two rounds, in addition to performing a (non-private) summation.

Theorem 2. *Protocol 2 for computing the disjunction of n bits on a network $\mathcal{G} = (\mathcal{P}, \mathcal{E})$, gives the correct output if it is* `false`*, and gives an incorrect output with probability $2^{-\kappa}$ when the correct output is* `true`*.*

Proof. Consider the sum

$$\sum_{i=1}^{n} s_i = \sum_{i=1}^{n} \sum_{(P_i,P_j)\in\mathcal{E}} (x_j a_{i,j} + r_{j,i} - r_{i,j}) = \sum_{(P_i,P_j)\in\mathcal{E}} x_j a_{i,j} \,.$$

If all x_j are 0, clearly the sum is 0. Otherwise, it is a sum of uniformly random group elements, and thus has uniformly random distribution. In particular, with probability $1 - 2^{-\kappa}$ it is non-zero. □

4.1 Privacy

Theorem 3. *If the cryptosystem* CS *is* (t, ϵ)*-IND-CPA secure, then no adversary running in time* $t - t'$*, for a small* t'*, can violate the privacy of Protocol 2 with advantage more than* $\frac{n^2}{4}\epsilon$*.*

The proof of Theorem 3 begins like the proof of Theorem 1 with a combinatorial lemma similar to Lemma 1, essentially saying that unless the adversary learns something about the values $a_{i,j}$ from seeing them encrypted, it cannot violate the privacy of Protocol 2. Given the lemma, we apply a hybrid argument to prove the security of the protocol.

Lemma 2. *Consider executions of Protocol 2 on a connected network* $\mathcal{G} = (\mathcal{P}, \mathcal{E})$ *with input* x *such that* $x_i =$ `true` *for at least one* P_i*, and where a collusion* $\mathcal{C} \in \mathcal{Z}$ *from a non-separating adversary structure* $\mathcal{Z}$*, and communication* $a_{i,j}, r_{i,j}, r_{j,i}, s_i$ *for* $P_i \in \mathcal{C}$ *is fixed. For such executions, the values* s_i *for* $P_i \in \mathcal{P} \backslash \mathcal{C}$ *have a uniform and independent distribution.*

Proof (Theorem 3). We begin with the observation that if all the parties have input `false`, then the protocol behaves exactly as Protocol 1 with zeroes as inputs and by Lemma 1, then the honest parties' s_i will be uniformly random conditioned on $\sum_{i=1}^{n} s_i = 0$.

First, consider the case where the inputs of all corrupted parties are `false`. In this case, a simulator that independently samples the pk_i, $c_{i,j}$, $r_{i,j}$ and s_i included in the adversary's view, conditioned only on $\sum_{i=1}^{n} s_i = 0$ if the output is `false`, or $\sum_{i=1}^{n} s_i \neq 0$ otherwise perfectly simulates the protocol to the adversary, by the previous observation and Lemma 2. Thus, in this case, the adversary cannot violate the privacy of the protocol.

Now, consider the case when at least one of the corrupted parties has input `true`. We begin by constructing a simulator S_0 that randomly selects inputs and $a_{i,j}$ for all honest parties, conditioned on the output matching the output it should simulate. It then follows the protocol to simulate the adversary's view.

We now construct hybrid simulators, S_k, working like S_0 but replacing the first k ciphertexts $c_{i,j}$ in the adversary's view by random ciphertexts. It follows from the (t, ϵ)-IND-CPA security of $\mathsf{E}_{pk_i}(x)$ that no adversary running in time $t - t'$, for some small t' required to run the simulator S_k, can distinguish between the views simulated by S_k and S_{k-1}.

Assume that the adversary's view includes T ciphertexts $c_{i,j}$, so the view simulated by S_T contains no information on the $a_{i,j}$ sent by honest nodes to corrupted nodes. There can be at most $(n/2)^2$ edges between honest and corrupted nodes, so $T \leq n^2/4$. By Lemma 2, the distribution of simulated $r_{i,j}$ and s_i values is exactly the same as in a real execution, so the view simulated by S_T contains no information on the honest parties inputs. □

Proof (Lemma 2). Consider the following mental experiment, where we modify an execution of the protocol in two steps.

MODIFICATION 1. For each neighbor P_j of P_i we subtract $x_i c_{j,i}$ from $c'_{i,j}$ in Step 3 of the protocol and add $x_i a_{j,i}$ to s_i in Step 4 of the protocol. It is easy to see that this does not change the distribution of either s_i or $\mathsf{D}_{sk_i}(c'_{i,j})$ for any neighbor P_j.

MODIFICATION 2. Remove all encryptions and decryptions. This transforms Steps 3-5 of the protocol into an execution of Protocol 1, where P_i holds the input $\sum_{j=1}^{n} x_i a_{j,i}$.

From Lemma 1 we conclude that with the two modifications, the s_i are independently distributed conditioned on $\sum_{i=1}^{n} s_i = \sum_{i=1}^{n} \sum_{j=1}^{n} x_i a_{j,i}$, but the right side of this equation is randomly distributed when some $x_i = 1$ and $a_{i,j}$ for some neighbor P_j is randomly distributed. From the conditions of the lemma, we know there is at least one P_i such that $x_i = 1$, and from the monotonicity of $\mathcal{Z}$ and that it is non-separating, we know that every party has an honest neighbor. Thus, the s_i are uniformly and independently distributed. This concludes the proof. □

4.2 Computing the Maximum

In the setting with passive adversaries, it is easy to construct a protocol for computing the maximum by repetition and parallel composition of a protocol for disjunctions.

Assume the inputs are integers of ℓ bits. We can then compute the disjunction of the most significant bits of all parties' inputs, which is also the most significant bit of the maximum of the inputs. We then proceed to the next most significant bit. When a party learns that its input is smaller than the maximum (its input was 0 and the output was 1), it participates with input 0 in the remaining protocol executions.

Several bits can be handled in parallel to reduce the number of rounds at the cost of more protocol executions. To find the maximum of k bits, one can run $2^k - 1$ parallel disjunction computations, where the parties set their inputs based on if their k most significant bits represent an integer greater or equal to $2^k - 1, 2^k - 2, \ldots, 1$, respectively. Thus, to find the maximum of ℓ-bit integers, one can run $\lceil \ell/k \rceil$ rounds of protocols for disjunction, with $2^k - 1$ protocol executions in each round.

5 General Composition

Many functions can be computed as a function of the sum of inputs of the parties. Examples include disjunction, counting and threshold functions. In this context, a threshold function is a function returning `true` if the sum of inputs exceeds some threshold and `false` otherwise.

We can combine our Protocol 1 with standard protocols (which assume full mesh communications) to construct information-theoretically secure protocols for computing such functions. The benefit of this approach is that information-theoretical security is achieved in a partial mesh network while maintaining efficiency. Another approach would have been to simulate the missing edges (e.g., with the techniques from [14]) and then immediately using standard protocol, but this approach is generally more expensive in terms of communication.

By this composition, we essentially run a cheap protocol to "accumulate" the inputs of most parties and then let some small subset of parties run a more expensive protocol and jointly act as a trusted party. This can be useful when performing computations with a large number of parties where some subset can be trusted not to collude with each other. This can be compared to the trusted servers in [5,13,7].

Executing the standard protocol requires a complete network, so this construction is only applicable when $\mathcal{G}$ contains a subgraph K that is complete. Furthermore, tolerable adversary structures $\mathcal{Z}$ are those that do not separate the graph, and which, restricted to K, are tolerable by the standard protocol being used. For most protocols, the requirement will be that no two subsets in $\mathcal{Z}$ cover K, or using notation from [20], the predicate $Q^{(2)}(\mathcal{Z}|_K, K)$ must hold.

Protocol 1 constructs a secret sharing of the sum of the parties inputs and then opens it. When we adapt the protocol for composition, we only construct the secret sharing, and accumulate the sum (still shared) in the nodes in K.

As an example, we give an information-theoretically secure protocol for disjunction. Here, we let each party input 0 or 1 (for `false` and `true`) and then use a protocol by Damgård *et al.* [12] for comparison.

Protocol 3 (Disjunction). In the protocol for computing $\texttt{Or}(x_1, \ldots, x_n)$ where $x_i \in \{0, 1\}$ on the network $\mathcal{G} = (\mathcal{P}, \mathcal{E})$ with a set $K \subseteq \mathcal{P}$ of designated parties, $P_i \in \mathcal{P}$ proceeds as follows:

1. For each neighbor P_j, pick $r_{i,j} \in \mathbb{Z}_p$ randomly and send it to P_j.
2. Wait for $r_{j,i}$ from each neighbor P_j.
3. Compute $s_i = x_i - \sum_{(P_i,P_j)\in\mathcal{E}} r_{i,j} + \sum_{(P_i,P_j)\in\mathcal{E}} r_{j,i}$.
4. Compute $s = \sum_{P_j \notin K} s_j$ using `NonPrivateSum`.
5. If in K, execute comparison protocol from [12] to test if $s + \sum_{P_j \in K} s_j = 0$.

Theorem 4. *Protocol 3 is information-theoretically private to a passive and static adversary if the adversary structure $\mathcal{Z}$ does not separate the network $\mathcal{G} = (\mathcal{P}, \mathcal{E})$ and there is a complete subgraph $K \subseteq \mathcal{G}$ such that no two sets in $\mathcal{Z}$ cover K.*

Proof. The values $r_{i,j}$ are independent of the input. By the restriction on $\mathcal{Z}$ there must be at least one party in K not corrupted by the adversary. By Lemma 1 we know that the s_i values input to `NonPrivateSum` are uniform and independent. Thus, the adversary gains no information from these, and by the composition theorem [17, Theorem 7.5.7], we conclude that the protocol is private. □

6 Conclusion

In this paper we have given efficient protocols for privately evaluating summation and disjunction on any network topology. The ability to privately evaluate these two basic primitives have applications in several widely varying contexts. As the most expensive part of our protocols is the task of non-private summation, privacy comes very cheaply.

We believe that the question of which functions can be efficiently privately evaluated in arbitrary network topologies is an interesting topic for further study.

References

1. The FP7 4WARD project. `http://www.4ward-project.eu/`.
2. Amos Beimel. On private computation in incomplete networks. *Distributed Computing*, 19(3):237–252, 2007.
3. Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, pages 1–10. ACM, 1988.
4. Markus Bläser, Andreas Jakoby, Maciej Liskiewicz, and Bodo Manthey. Private computation: k-connected versus 1-connected networks. *J. Cryptology*, 19(3):341–357, 2006.
5. Dan Bogdanov, Sven Laur, and Jan Willemson. Sharemind: A framework for fast privacy-preserving computations. In Sushil Jajodia and Javier López, editors, *ESORICS*, volume 5283 of *Lecture Notes in Computer Science*, pages 192–206. Springer, 2008.
6. Felix Brandt. Efficient cryptographic protocol design based on distributed el gamal encryption. In Dongho Won and Seungjoo Kim, editors, *ICISC*, volume 3935 of *Lecture Notes in Computer Science*, pages 32–47. Springer, 2005.
7. Martin Burkhart, Mario Strasser, Dilip Many, and Xenofontas Dimitropoulos. SEPIA: Privacy-preserving aggregation of multi-domain network events and statistics. In *19th USENIX Security Symposium*, Washington, DC, USA, August 2010.
8. Haowen Chan, Adrian Perrig, and Dawn Xiaodong Song. Secure hierarchical in-network aggregation in sensor networks. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM Conference on Computer and Communications Security*, pages 278–287. ACM, 2006.
9. David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *J. Cryptology*, 1(1):65–75, 1988.
10. Benny Chor and Eyal Kushilevitz. A communication-privacy tradeoff for modular addition. *Inf. Process. Lett.*, 45(4):205–210, 1993.
11. M. Dam and R. Stadler. A generic protocol for network state aggregation. In *Proc. Radiovetenskap och Kommunikation (RVK)*, 2005.
12. Ivan Damgård, Matthias Fitzi, Eike Kiltz, Jesper Buus Nielsen, and Tomas Toft. Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In Shai Halevi and Tal Rabin, editors, *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 285–304. Springer, 2006.
13. Yitao Duan and John F. Canny. Practical private computation and zero-knowledge tools for privacy-preserving distributed data mining. In *SDM*, pages 265–276. SIAM, 2008.

14. Matthew K. Franklin and Moti Yung. Secure hypergraphs: Privacy from partial broadcast. *SIAM J. Discrete Math.*, 18(3):437–450, 2004.
15. Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.
16. Minos N. Garofalakis, Joseph M. Hellerstein, and Petros Maniatis. Proof sketches: Verifiable in-network aggregation. In *ICDE*, pages 996–1005. IEEE, 2007.
17. Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, New York, NY, USA, 2004.
18. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229. ACM, 1987.
19. Wenbo He, Xue Liu, Hoang Nguyen, Klara Nahrstedt, and Tarek F. Abdelzaher. PDA: Privacy-preserving data aggregation in wireless sensor networks. In *INFOCOM*, pages 2045–2053. IEEE, 2007.
20. Martin Hirt and Ueli M. Maurer. Player simulation and general adversary structures in perfect multiparty computation. *J. Cryptology*, 13(1):31–60, 2000.
21. Lingxuan Hu and David Evans. Secure aggregation for wireless networks. In *Workshop on Security and Assurance in Ad hoc Networks*, page 384. IEEE Computer Society, 2003.
22. Márk Jelasity, Alberto Montresor, and Özalp Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Trans. Comput. Syst.*, 23(3):219–252, 2005.
23. Michael J. Kearns. Efficient noise-tolerant learning from statistical queries. In *STOC*, pages 392–401, 1993.
24. David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-based computation of aggregate information. In *FOCS*, pages 482–491. IEEE Computer Society, 2003.
25. Patrick Lincoln, Phillip A. Porras, and Vitaly Shmatikov. Privacy-preserving sharing and correlation of security alerts. In *USENIX Security Symposium*, pages 239–254. USENIX, 2004.
26. Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. TAG: A tiny aggregation service for ad-hoc sensor networks. In *OSDI*, 2002.
27. Bartosz Przydatek, Dawn Xiaodong Song, and Adrian Perrig. SIA: secure information aggregation in sensor networks. In Ian F. Akyildiz, Deborah Estrin, David E. Culler, and Mani B. Srivastava, editors, *SenSys*, pages 255–265. ACM, 2003.
28. Matthew Roughan and Yin Zhang. Secure distributed data-mining and its application to large-scale network measurements. *SIGCOMM Comput. Commun. Rev.*, 36(1):7–14, 2006.
29. Adam J. Slagell and William Yurcik. Sharing computer network logs for security and privacy: A motivation for new methodologies of anonymization. *CoRR*, cs.CR/0409005, 2004.