Private aggregation on untrusted servers with customizable thresholds

Constantinos Patsakis, Michael Clear, Paul Laird

Distributed Systems Group, School of Computer Science and Statistics, Trinity College, Dublin, Ireland
email: {patsakik,clearm,lairdp}@scss.tcd.ie

Abstract. While multiparty computations are becoming more and more efficient, their performance has not yet reached the level needed to be widely deployed for many applications. Nevertheless, the heterogeneous environment of modern computing needs this functionality in order to provide users their right to privacy. For a wide range of applications there is no need for complex computations; operations such as multiplication or addition might be sufficient. In this work we introduce a new multiparty computation protocol (MPC) for multi-round summation whose security is based on DDH in the semihonest model. We also introduce the concept of an *anonymous aggregation system* that combines MPC with "blinded" aggregation so that the aggregate values may remain hidden from the aggregator, and show how to achieve this with our MPC protocol. We give results on the performance of our solution and discuss suitable applications.

Keywords: cryptographic protocols, privacy, anonymity, multiparty computation, aggregation

1 Introduction

The computation environment that a common user is using daily is becoming more and more heterogeneous due to the wide range of interconnected devices that directly or indirectly are being used, e.g. through cloud computing. Therefore, the user environment should be considered hostile in most of the cases. In many scenarios, a user might have to calculate a function in collaboration with other entities, real users or virtual, which cannot be trusted. Hence user input to the function should be hidden in such way that other users cannot disclose his input, while enabling the output of the function to be calculated correctly and efficiently. More formally, we have n entities that want to calculate function $f(x_1, x_2, ...x_n)$ without disclosing $x_i, i \in \{1, 2, ..., n\}$ to any other entity. This is a special case of secure multiparty computation (MPC). In this work, we lower the security barrier of more standard MPC to the setting where we only have honest-but-curious adversaries ; that is, the entities involved are not considered malicious, so they can be expected to strictly follow the protocol, but they might collude in order to expose a user's input.

In this work, we introduce the notion of an *anonymous aggregation system*. In such a system, a set of users wish to allow a specific entity, which we call a consumer, access to certain aggregates (say, sums or products) of their inputs, while keeping their inputs private from each other and the consumer. Furthermore, the users need to communicate with an intermediate that acts as a proxy between the users and the consumer. To reduce bandwidth, it is desirable for this intermediate to perform aggregation. However, such an aggregator may not be trusted to learn the aggregate values, and therefore it is necessary to allow "blinded" aggregation.

A motivating factor for this architecture stems from a limitation with current solutions, namely they do not scale efficiently without the use of trusted third parties. This means that if someone wants to provide anonymous summation with many users, the resultant sum is significantly large. There are two solutions to this. Either he has to increase the key size, imposing a computational overhead, mainly to the users, or alternatively more aggregators have to be deployed. The latter means that due to the current nature of aggregators, sub-sums are disclosed to them. Depending on the nature of the aggregated data, this assumption might not be tolerated. To protect users from localized attacks and blind the result of the aggregation from the aggregator, we separate the entities of aggregator and consumer.

1.1 Main contributions

We introduce the notion of an *anonymous aggregation system*, and provide a concrete construction. The main technical contribution of this work is an extension of the MPC protocol from [21]. Our extension offers a trade-off between collusion tolerance and the number of rounds of aggregation that can be performed from the same public information. We establish precise bounds and prove the protocol secure under the DDH assumption. In contrast, the multi-round protocol in [21] relies on bilinear pairings. In addition, we show how our protocol can be adapted to allow an aggregator to perform aggregation "blindly", thus meeting our requirements of an anonymous aggregation system.

1.2 Organization of this work

The rest of this work is organized as follows. In the next section we provide an overview of the related work, including some of the building blocks for our constructions. Section 3 formalizes an anonymous aggregation system and introduces the main elements of our construction. Section 4 describes our MPC protocol in detail and gives a proof of security. Section 5 provides some experimental results, while Section 6 discusses possible applications. Finally, the article concludes with some remarks and ideas for future work.

2 Related work

The concept of secure multiparty computation was introduced by Yao in [32] with the introduction of the famous millionaires problem. In this problem, two millionaires want to compare their assets in order to verify which one is the richest, but without revealing the actual value of their assets. Some important foundational advances were made soon after such as [33, 17, 11, 3], however it took almost two decades for real-world implementations to became practical [26, 2, 5].

Secure multiparty computation can be broadly divided into schemes which allow arbitrary computations to be performed without leaking information, and specialised protocols for the evaluation of one particular function, such as summation, while keeping individual inputs private.

General-purpose schemes include approaches based on Garbled Circuits (efficient implementations include [24, 25, 23]), Nielsen's protocol using Oblivious Transfer [29] and Oblivious RAM [16, 19]. Other protocols use arithmetic circuits such as those based on the BGW protocol e.g: VIFF [14] and SPDZ which employs fully homomorphic encryption [15].

2.1 Specialised arithmetic multi-party computations

Schemes for specific arithmetic operations trade generality for greater efficiency in calculating their particular operation. Many also allow further trade-offs, such as collusion-resistance against efficiency.

Clifton et al. in [12] proposed a very efficient and elegant scheme for summing the shares of n entities anonymously. Let's assume that n users want to compute the sum of their shares s_i, without disclosing the value of their shares. Starting from user 1, each user adds a random value r_i to the current sum and passes that to the next user. The last user add his value and returns the resulting value to the first one. The first user now subtracts a value r'_1 from the value he is given, so that $r_1 - r'_1 = s_1$. The next users do the same, so at the end, when user n subtracts r'_n the resulting value is the sum. The scheme provides an anonymous sum under the assumption that users do not collude. It is clear that if the previous and the next user of user k collude, then the share s_k can be trivially exposed, however, replacing the ring with an k-pass permuted path without duplicate neighbours increases the number of corrupt parties required to reveal an input to $2k$. In this manner the communicational overhead is traded off against the collusion-tolerance of the scheme.

A significant disadvantage of the share-based anonymous sum protocol is that to achieve collusion tolerance of $n-2$, a total of $n\lfloor\frac{n}{2}\rfloor$ messages must be sent serially, with each node sending and receiving $\lfloor\frac{n}{2}\rfloor$ messages. This is not practical where latency is an issue or where large numbers of nodes are involved. Protocols involving self-cancelling blinding variables alleviate this problem, allowing a constant number of communication rounds for an arbitrary number of participants. Yang *et al.* use two rounds of communication, with the first establishing common key material [31]. Shi *et al.* allow aggregation to be performed with a

single round of comunication [30], however the protocol relies upon the existence of a secret share of zero, which must be supplied in advance by a trusted dealer or created using another protocol.

2.2 One bit multi-party computations

A famous problem in cryptography is the *dining cryptographers problem*, introduced by Chaum in [10]. According to the statement of the problem, three cryptographers dine and the cryptographers are notified by the waiter that their bill has been payed. The payment has been made by one of the cryptographers or by the NSA. The problem is to find anonymously whether the payment has been made by one of them, without disclosing his identity, or by the NSA. While the provided solution manages to anonymously calculate one bit of information (0 if it was paid by the NSA or 1 if it was paid by a cryptographer), one of the main limitations of the proposed protocol, usually named as *DC-net* is that if two cryptographers have paid for the dinner, then they cancel each other out, so the end result is wrong. The protocol has been revised in [18] to allow detection and identification of "cryptographers" that try to cheat.

Trying to extend this result so that one bit of information can be securely calculated without cancellations, Hao and Zieliński introduced another protocol, AV-net, in [20] which is discussed in the next section.

2.3 The Anonymous veto protocol of Hao and Zieliński

Hao and Zieliński introduced a very elegant protocol which manages to patch several vulnerabilities of the dining cryptographers network [20]. The following scenario illustrates the concept underlying their protocol. Due to extreme circumstances, n generals from several countries have to decide whether or not they will invade a certain country. The invasion can only be decided unanimously; thus if a general decides to veto, the invasion is halted. Obviously the pressure is very high and in order to avoid pressure to individuals that decide to veto, the voting has to be made anonymously. Additionally, each party has to be able to compute and verify the result, even if some entities decide to manipulate the result.

The protocol that provides a solution to this problem has two rounds, where the involved parties broadcast their results. Initially, all users decide on a finite cyclic group of prime order q in which the Decision Diffie-Hellman (DDH) problem is known to be intractable, and choose one of its generators g [6]. Every user U_i selects a random secret value $x_i \in \mathbb{Z}_q$ and in the first round every user broadcasts g^{x_i} as well as a proof of knowledge of x_i. Knowing these values, everyone can calculate the following:

$$g^{y_i} = \prod_{j=1}^{i-1} g^{x_j} \prod_{j=i+1}^{n} g^{-x_j}$$

Note that due to the intractability of the DDH problem, the actual values of y_i cannot be evaluated by any party. In the second round, every user broadcasts a

value $g^{c_i y_i}$ along with a knowledge proof for c_i , where:

$$g^{c_i y_i} = \begin{cases} g^{r_i y_i} & if\ U_i\ vetoes. \\ g^{x_i y_i} & if\ U_i\ does\ not\ veto. \end{cases}$$

where $r_i \in \mathbb{Z}_q$ is random and $x_i \neq r_i$.

If no one vetoes then result would be: $\prod_{i=1}^{n} g^{c_i y_i} = 1$ since: $\sum_{i=1}^{n} x_i y_i = 0$. Otherwise, in case of veto: $\prod_{i=1}^{n} g^{c_i y_i} \neq 1$. In either case, the result can be computed by everyone and cannot be tampered by anyone. Finally, the identity of those who have vetoed cannot be deduced.

Kursawe et al. extended the Anonymous veto protocol enabling the aggregation of smart meter values [21]. The concept that they used is quite easy to follow. Since $x_i y_i$ cancel each other out when summed, the authors saw that they could embed a value that cannot be traced, due to the intractability of the DDH problem. Therefore, the exponent of each user becomes $x_i y_i + m_i$. This way, if one multiplies the published values, he will compute:

$$\prod_{i=1}^{n} g^{x_i y_i + m_i} \equiv \prod_{i=1}^{n} g^{x_i y_i} g^{m_i} \equiv \prod_{i=1}^{n} g^{\sum x_i y_i} + g^{\sum m_i} \equiv g^{\sum m_i}$$

This means that if the value $\sum m_i$ is quite small, then it could be easily brute forced, recovering the true value of the sum, without though disclosing the values m_i that each user submitted.

3 High-Level Architecture for Anonymous Aggregation

In what follows, we will first introduce the main actors and desiderata. Afterwards, we will gradually start extending the current protocols, discussing the features that each abstraction introduces.

3.1 Main actors and desiderata

In our model we assume the following entities:

The Users: We have a set of n users $U_1, U_2, ..., U_n$ which have already exchanged their public keys and want to evaluate a function $f(x_1, x_2, ..., x_n)$, so that each user U_i provides input x_i and his input is not disclosed to any other user. Moreover, we assume that we do not have malicious users, yet they are honest but curious. Hence, each user follows the protocol, however, he may try to find more information about others' users input. During the protocol initialization, the users will have to send their privacy preferences, the number of people that have to collude in order to recover their submitted value.

The Consumer: It is the entity that wants to evaluate function f.

The Aggregator: The aggregator can collect all the data from Users in order to help in the evaluation of the function and then forward it to the Consumer. The aggregator has a registry which can be read and written only by the Users and read by the Consumer. We assume that the Aggregator does not behave maliciously and that he is honest but curious.

While the entities are considered distinct, depending on the application scenario some of them can be the same real entity, for instance, the Consumer and the Aggregator can be the same real entity. Next we present a formal definition.

3.2 Formal Definition

An anonymous aggregation scheme consists of a secure multi-party computation (MPC) protocol for a deterministic functionality $f : \mathcal{M}^n \to \mathcal{M}$ over some message space $\mathcal{M}$ together with a means to "blind" the result from an aggregator (and indeed the participants) such that only a designated party can recover the result. We formalize these requirements in the following definition.

Definition 1. *An* anonymous aggregation system *for n parties with collusion tolerance t and supporting $\zeta(n,t)$ aggregation rounds of a functionality $f : \mathcal{M}^n \to \mathcal{M}$ is a tuple of algorithms* $\mathsf{AAn} = (\mathsf{ParamGen}, \mathsf{KeyGen}, \mathsf{Init}, \mathsf{Pub}, \mathsf{Aggr}, \mathsf{Recover})$ *for public parameter generation, key generation, initialization of a protocol "session", publication of a value, aggregation (potentially blind aggregation by an aggregator) and a recovery of the result (potentially by a designated consumer). More precisely:*

- **ParamGen** *takes as input a security parameter, and outputs public parameters* PP*, which all entities receive.*
- **KeyGen** *takes as input public parameters* PP *and outputs a public and private key.*
- **Init** *takes as input public parameters* PP*, a number of parties n and a list of n public keys* $\mathsf{pk}_1, \ldots, \mathsf{pk}_n$*, and the public key of a consumer* pk_c*. It outputs information π describing a "session" or execution of the protocol.*
- **Pub** *takes as input protocol "session" π, a party number $i \in \{1, \ldots, n\}$, a round number ρ and an input value $m \in \mathcal{M}$. It outputs a protocol message $P_{\rho,i}$.*
- **Aggr** *takes as input public parameters* PP *and a list of n protocol messages $P_{\rho,1}, \ldots, P_{\rho,n}$. It outputs an element P'.*
- **Recover** *takes as input public parameters* PP*, an element P', and a private key* sk_c *(private key of the consumer). It outputs a result $m' \in \mathcal{M}$.*

Definition 1 encompasses all of the components of an anonymous aggregation system (AAS). Since it is costly for parties to generate, broadcast, and verify keys, an AAS facilitates running multiple "rounds" with the same public keys. How many rounds can be executed with the same public keys is determined as a function ζ of the number of users n and the desired collusion tolerance t. Thus, we write the maximum number of rounds as $\zeta(n,t)$. For security reasons, a party will execute at most $\zeta(n,t)$ rounds.

In order to use the system, it is first necessary to generate the public parameters which are made known to all entities. Let $\mathsf{PP} \leftarrow \mathsf{ParamGen}(\kappa)$ for some security parameter κ. In a given "session" or execution of the protocol, every party will have to be aware of each other's public key. Such public keys are freshly generated for each protocol *session*. Therefore, all parties U_i will run $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{PP})$ to generate a key pair, and then broadcast pk_i. Furthermore, a consumer may also generate a key pair i.e. it may run $(\mathsf{pk}_c, \mathsf{sk}_c) \leftarrow \mathsf{KeyGen}(\mathsf{PP})$. To begin a protocol session, each party runs $\pi \leftarrow \mathsf{Init}(\mathsf{PP}, n, \mathsf{pk}_1, \ldots, \mathsf{pk}_n, \mathsf{pk}_c)$. Note that the consumer's public key may also be pk_i for some $1 \leq i \leq n$, or alternatively, it may be a "null" public key such that everyone ca n recover the results. For a given round ρ, party U_i whose input value in that round is $m_{\rho,i} \in \mathcal{M}$, computes $P_{\rho,i} \leftarrow \mathsf{Pub}(\pi, i, \rho, m_{\rho,i})$ and broadcasts it. The aggregator simply needs to run $P' \leftarrow \mathsf{Aggr}(\mathsf{PP}, P_{\rho,1}, \ldots, P_{\rho,n})$ and hand P'_ρ to the consumer, who can then compute $m'_\rho \leftarrow \mathsf{Recover}(\mathsf{PP}, P'_\rho, \mathsf{sk}_c)$. It should hold that $m'_\rho = f(m_{\rho,1}, \ldots, m_{\rho,n})$.

In this paper, we deal with operations such as summation and multiplication and therefore, the result m'_ρ leaks nothing about the ordering of $m_{\rho,1}, \ldots, m_{\rho,n}$. Hence, the system allows anonymity. If we abstract away the notion of blind aggregation (ignore the consumer's keys pk_c and sk_c), we are left with an MPC protocol. Assuming all parties have pre-agreed a set of parameters generated with $\mathsf{ParamGen}$, the first phase of an MPC protocol can be seen as the generation and broadcasting of public keys (generated with KeyGen) followed by each party running Init and broadcasting the protocol messages obtained from Pub. In the final phase, each party can simply run Aggr followed by $\mathsf{Recover}$ to produce the MPC protocol's output. In the next section, we give an introductory overview of the particular MPC protocol at the heart of the anonymous aggregation system proposed in th is paper. We explore this MPC protocol in more detail in Section 4 where we prove its security.

3.3 Overview of our MPC protocol

Recall the protocol of kursawe et al. [21] described in Section 2.3, which allows n parties to privately compute the sum of their inputs. To extend this protocol, the first step is to generalize the y_i coefficients. Their main property is that they allow $\sum x_i y_i \equiv 0$, nevertheless, there are many other choices fulfilling this property. One could map the y_is used in the protocols of Hao and Zieliński and Kursawe et al. to the following matrix A:

$$A = \begin{bmatrix} 0 & -1 & -1 & -1 & -1 \\ 1 & 0 & -1 & -1 & -1 \\ 1 & 1 & 0 & \cdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \cdots & 0 \end{bmatrix}$$

where the row i represents the coefficients that of x_js that have to be added in order to be multiplied with x_i[1]. The matrix A is a skew-symmetric matrix, that is $-A = A^T$, but clearly any such matrix can be used instead of the proposed one. So far this does not give us any advantages over the standard protocol of Kursawe et al.

Enabling reuse of published information As previously discussed, users could agree upon a skew-symmetric matrix to generate the y_i coefficients. However, if they would like to compute one more summary, then they could not use the previous values. The reason is that an adversary could deduce important information from that. For instance, if they had originally published $g^{x_i y_i + m_{i_1}}$ and then $g^{x_i y_i + m_{i_2}}$, given that m_1 and m_2 are small, the value $m_1 - m_2$ could easily be calculated. It becomes apparent that a new matrix should be generated to protect users' privacy.

It becomes apparent that users could co-operate in the generation of the skew-symmetric matrix, defining their collusion thresholds. Definitely, by lowering the threshold users are making some compromises in their privacy. Nevertheless, this is speeding up the calculations and decreasing the communication overhead. The compromise, depending on the nature of the network, the trust of the users and the importance of protected information can imply a good balance in the user's benefit.

Thus, we argue that in order to minimize the bandwidth overhead and communication between users, instead of agreeing on a matrix A, users on the initialization of the scheme agree on a random seed that is used to generate a series of matrices A_ρ. However, we must allow each A_ρ's independent entries to be uniformly random over $\mathbb{Z}_p$ instead of over $\{-1, 1\}$ in order to avoid a straightforward linear algebra attack. Instead of using the public keys of the parties (i.e. the g^{x_i}) for a single aggregation, we observe that we can re-use them for multiple aggregation "rounds" provided we use a different skew-symmetric matrix A in each round, where the independent entries of each A is uniformly random over $\mathbb{Z}_p$. We use a hash function to derive the matrix A for each round. in each round.

Nevertheless, since the reuse of g^{x_i} in many rounds, might enable collusion attacks, the amount of matrices that can be generated is bounded by $\frac{n-t}{2}$, where t is the minimum threshold. More details on the latter bound are given in the Section 4. The advantage of allowing users to reuse their published values is that it decreases the computational and communication cost. Rather than going through the first step of the algorithm again, generating, publishing g^{x_i} and downloading the output of the other users, users recompute locally the new instance of matrix A and compute the new values of g^{y_i}. It turns out that Kursawe et al. propose a variant of their protocol that similarly supports multiple rounds, but they rely on bilinear pairings to achieve this. Our protocol instead relies only on DDH, but the number of rounds we can support is at most $\frac{n-2}{2}$.

An extension, beyond the scope of this paper, sees the matrix take on an additional role, in which users can use it to indicate which entities, or at least

[1] It is easy to see that $A \times [x_1, x_2, \cdots, x_n]$ returns the coefficients that each x_i has to be multiplied to generate the according y_i.

how many, must collude in order to recover their input. This information is then used to insert zero or non-zero values in the appropriate cells of the matrix, reducing the computational load per user, at the cost of a diminished number of rounds. To understand how this achieved, it has to be noted that the more zeros one introduces to his rows, the fewer published values he has to multiply in order to calculate his base value for each round of the protocol.

3.4 Blind Aggregation

Recall that in Section 3.2, we decomposed an *anonymous aggregation system* into two parts: MPC and blind aggregation. In this section, we describe how our MPC protocol can be combined with a means to provide the latter. Given the public key g^c of a consumer C, we show how each party can "blind" his protocol messages using g^c. Moreover, an aggregator is still able to compress the *blinded* protocol messages of all parties into a *blinded* element, from which only C can recover the result using her private key c.

In our definition of an anonymous aggregation system in Section 3.2, the Init algorithm accepted the public key of a consumer as input. For our system, let this be g^c where c is the consumer's private key. In each round, a user U_i who runs our MPC protocol produces a protocol message M_i. For more details on the structure of M_i, see the discussion in the next section. For our purposes here it is sufficient to say that aggregation is performed by computing $\prod_{i=1}^{n} M_i$. In each aggregation round the users do not just send M_i as they would in the "bare" MPC protocol, but:

$$c_{i,1} = M_i g^{r_i}$$
$$c_{i,2} = g^{cr_i}$$

where r_i is uniformly random in $\mathbb{Z}_p$. The aggregator will therefore calculate:

$$C_1 = \prod c_{i,1} = \prod M_i g^{r_i} = g^{\sum r_i} \prod M_i$$
$$C_2 = \prod c_{i,2} = \prod g^{ar_i} = g^{c \sum r_i}$$

Thus the value $\prod M_i$ remains secret to the aggregator. The consumer can then recover $\prod M_i$ by computing

$$\prod M_i = C_1 \cdot C_2^{-c^{-1}}$$

where c^{-1} denotes the inverse of c in $\mathbb{Z}_p^*$. Therefore, the consumer has the same view as a party in standard MPC.

4 Multiparty Computation Protocol for Aggregation

In this section, we first introduce some notation and then provide a formal description of our multiparty computation protocol (MPC) for secure summation in the semihonest model. Note that this section focuses exclusively on our MPC protocol.

4.1 Notation

Let k be an integer. We denote the contiguous set of integers $\{1, \ldots, k\}$ by $[k]$. Let X and Y be distributions. The notation $X \underset{C}{\approx} Y$ denotes the fact that both distributions are computationally indistinguishable to any probabilistic polynomial time (PPT) algorithm.

4.2 Protocol Description

Our protocol builds on the work in [21] to add support for multiple "rounds" of aggregation using the same public keys generated by all parties in the initial stage. Moreover, instead of relying on additional assumptions to achieve this (as is the case in [21] where bilinear groups are employed), our construction provides security in the semihonest model for $\mathsf{max}(1, \lfloor \frac{n-t}{2} \rfloor)$ rounds where t is the collusion tolerance. This is optimal for our techniques.

Let $A \in \mathbb{Z}_p^{k \times k}$ be a skew-symmetric matrix with uniformly random entries in $\mathbb{Z}_p$. Each row of A represents a quadratic polynomial over $\mathbb{Z}_p$ in k unknowns. There are $k(k-1)/2$ possible monomials. Thus, A can be transformed into a coefficient matrix $B \in \mathbb{Z}_p^{k \times k(k-1)/2}$. We write this as $B = \mathsf{coeff}(A)$. It follows that $\mathsf{rank}(B) = k - 1$. Therefore, no linear combination of $k-1$ equations yields 0.

Let $H : \{0,1\}^* \to \mathbb{Z}_p$ be a hash function. We define a function $\chi : \mathbb{Z}_p \times \mathbb{Z} \to \mathbb{Z}_p^{n \times n}$ that takes a random seed and a round number, and outputs a pseudorandom skew-symmetric matrix over $\mathbb{Z}_p$. Let $s \in \{0,1\}^*$ be a seed. The skew-symmetric matrix $A^{(i)}$ for round i is generated as follows:

- set $A^{(i)}_{j,k} \leftarrow H(s \parallel i \parallel j \parallel k)$.
- set $A^{(i)}_{k,j} \leftarrow -A^{(i)}_{j,k}$

for every j, k satisfying $1 \leq j < k \leq n$. The remaining entries of $A^{(i)}$ are set to zero. By construction, $A^{(i)}$ is skew-symmetric. Furthermore, $\mathsf{rank}(\mathsf{coeff}(A^{(i)})) = n - 1$.

Suppose there are n parties and the desired collusion tolerance is $T \leq n - 2$. Then the protocol can accommodate $\ell \leq \frac{\lfloor n-T \rfloor}{2}$ independent aggregations. We call each aggregation a "round". We use the term *stage* to describe what is commonly referred to as a *round* in the multiparty computation literature. Let $\beta > 0$ denote the size of the message space i.e. every party chooses her input for a given round from the set $\{0, \ldots, \beta\}$. Therefore, the sums are bounded from above by $n\beta$.

A public seed s is deterministically derived from the users' public keys generated in the first stage of the protocol. Alternatively, s may be pre-agreed or collaboratively generated. In the security proof, it is assumed to be unique for a given protocol execution.

The "public parameters" used in the protocol consist of a description of a cyclic group $\mathbb{G}$ of order p together with a generator g of $\mathbb{G}$. It is assumed that

the Decisional Diffie-Hellman (DDH) problem is intractable in $\mathbb{G}$. These public parameters $\mathsf{PP} = (\mathbb{G}, g, p)$ are known to all parties P_i.

The protocol proceeds in the following stages:

1. **Setup:** Party P_i generates a secret key $x_i \in \mathbb{Z}_p$ and computes her public key $u_i = g^{x_i} \in \mathbb{G}$. She broadcasts u_i.
2. For every $r \in \{1, \ldots, \ell\}$:
   Round r:
   - Party P_i chooses her input $m_i^{(r)} \in \{0, \ldots, \beta\}$.
   - Compute $A^{(r)} \leftarrow \chi(s, i)$.
   - Compute $w \leftarrow \prod_{j \in 1}^{n} u_j^{A_{i,j}^{(r)}} \in \mathbb{G}$.
   - Compute $v_i^{(r)} \leftarrow w^{x_i} \cdot g^{m_i^{(r)}} \in \mathbb{G}$.
   - Broadcast $v_i^{(r)}$.
3. **Output:** The protocol produces an output of ℓ elements, namely the sum of the inputs in each round. To compute the sum σ_r for round r:
   - Compute $z \leftarrow \prod_{j=1}^{n} v_j^{(r)}$.
   - Use Pollard's Lambda algorithm to compute the discrete log $\sigma_r \in \{0, \ldots, n\beta\}$ of z with respect to g in G. The time complexity of Pollard's lambda algorithm is $\sqrt{n\beta}$.
   - The final output is $(\sigma_1, \ldots, \sigma_\ell)$.

It can be easily observed that for any $1 \leq r \leq \ell$,

$$\prod_{j=1}^{n} v_j^{(r)} = g^{\sum_{j=1}^{n} m_j^{(r)}} \tag{1}$$

4.3 Security

In order to show that the proposed protocol provides the necessary privacy to the participants, we have to show that it provides privacy against collusions of up to t users when executed with at most $\mathsf{max}(1, \lfloor \frac{n-t}{2} \rfloor)$ rounds. Intuitively, suppose $n-2$ users collude, then it should not be possible for the colluding users to learn anything about the 2 honest users' inputs beyond their sum. If $n-1$ users collude, then we expect them to learn the honest party's input. So for the case of $n - 1 \leq t \leq n$, there is no privacy requirement, and thus these trivial cases are easily handled in meeting our security definition below.

We adopt the standard simulation-based definition of security in the semi-honest model with static adversaries where secure channels are assumed to exist between all pairs of parties along with a secure broadcast channel. We base our definition below on Definition 2.1 in [1]. Here we consider only computational security, and relax the more standard definition to deterministic functionalities with a single output, since this paper is concerned with aggregation. Note that this definition is general enough to accommodate multi-round aggregation as provided by our protocol.

Let $\boldsymbol{m} \in (\{0,1\}^*)^n$ be a vector of the inputs from each party and let π be a protocol. We define $\mathsf{OUTPUT}^{\pi}(m_1, \ldots, m_n)$ as the final aggregated result

computed with protocol π from the input vector $\boldsymbol{m}$. Furthermore, we define the view of a party P_i in the execution of protocol π with input vector $\boldsymbol{m}$ as

$$\mathsf{VIEW}_i^{\pi}(\boldsymbol{x}) = (m_i, r_i, \mu_i^{(1)}, \ldots, \mu_i^{(\ell)})$$

where m_i is party $P_i's$ input, r_i is its random coins and $\mu_i^{(1)}, \ldots, \mu_i^{(\ell)}$ are the ℓ protocol messages it received during the protocol execution. Similarly, the combined view of a set of $I \subseteq \{1, \ldots, n\}$ parties is denoted by $\mathsf{VIEW}_I^{\pi}(\boldsymbol{x})$.

Definition 2 (t-privacy of n-party protocols for deterministic aggregation functionalities). *Let $f : (\{0,1\}^*)^n \to (\{0,1\}^*)$ be a deterministic n-ary functionality and let π be a protocol. We say that π t-privately computes f if for every $\boldsymbol{m} \in (\{0,1\}^*)^n$ where $|m_1| = \ldots = |m_n|$,*

$$\mathsf{OUTPUT}^{\pi}(m_1, \ldots, m_n) = f(m_1, \ldots, m_n) \tag{2}$$

and there exists a PPT algorithm $\mathcal{S}$ such that for every $I \subset [n]$ with $|I| \leq t$, and every $\boldsymbol{m} \in (\{0,1\})^n$ where $|m_1| = \ldots = |m_n|$, it holds that:

$$\{\mathsf{VIEW}_I^{\pi}(\boldsymbol{m})\} \underset{C}{\approx} \{\mathcal{S}(I, \boldsymbol{m}_I, f_I(\boldsymbol{m}))\}. \tag{3}$$

The correctness condition given in 2 above holds for our protocol from Equation 1.

Lemma 1. *Let $m = \lfloor k/2 \rfloor$. Let $A^{(1)}, \ldots, A^{(m)}$ be skew-symmetric $k \times k$ matrices with uniformly random entries in $\mathbb{Z}_p$. Let $B^{(1)} = \mathsf{coeff}(A^{(1)})[1, \ldots, k-1], \ldots, B^{(m)} = \mathsf{coeff}(A^{(m)})[1, \ldots, k-1]$ where the notation $[1, \ldots, k-1]$ signifies the first $k-1$ rows of the matrix. Let $M = (B^{(1)}; \ldots; B^{(m)}) \in \mathbb{Z}_p^{m(k-1) \times m(k-1)}$ be the joint matrix consisting of $k-1$ rows from each of the m coefficient matrices. Then $\mathsf{Pr}[\mathsf{rank}(M) \neq m(k-1)] \leq \frac{\mathsf{poly}(k)}{p}$.*

Proof. We can rearrange the rows of M such that the t-th block $M^{(t)}$ consists of the t-th rows of the coefficient matrices. In each such row, there are only $k-1$ nonzero entries. Eliminating the zero columns results in an $m \times (k-1)$ matrix $M^{(t)\prime}$ with independent and uniformly random elements from $\mathbb{Z}_p$. Since $m < k-1$, the probability that $M^{(t)\prime}$ is linearly independent is at least the probability that its left $m \times m$ submatrix is linearly independent.It is mentioned in [4] (the result is due to Cooper [13]) that the probability that an $m \times m$ random matrix over $\mathbb{Z}_p$ is linearly independent is at least

$$\prod_{i=1}^{m}(1 - \frac{1}{p^i}).$$

Now the probability that this does not hold is bounded by $\frac{m}{p}$. Observe that if $M^{(t)}$ is linearly independent for all $1 \leq k-1$ then so is M, since each submatrix $M^{(t)}$ contains a unique column that is zero in all other submatrices, provided that a submatrices's unique column is nonzero. The probability of the latter not

holding is $\frac{k-1}{p^m}$. Therefore, an upper bound on the probability of M not being linearly independent is

$$\frac{k-1}{p^m} + \frac{m(k-1)}{p} = \frac{\mathsf{poly}(k)}{p}.$$

□

.

Theorem 1. *Under the DDH assumption, our multi-round protocol is computationally t-private for all $t \leq n$ with at most $\mathsf{max}(1, \lfloor (n-t)/2 \rfloor)$ rounds in the random oracle model.*

Proof. Let $\ell \leq \mathsf{max}(1, \lfloor (n-t)/2 \rfloor)$ be the number of rounds. Let $h = n - t$ be the number of honest users. If $h \leq 1$, it is trivial to construct a simulator $\mathcal{S}$ since $\mathcal{S}$ can fully learn $\boldsymbol{m}$ and then simulate all parties. Therefore, we assume that $h \geq 2$. Let $w = h(h-1)/2$. Consider the following series of Hybrids.

Hybrid 0: This is the same as the real distribution i.e. the LHS of Equation 3 with the exception that we "simulate" each honest party P_k using input $m_k^{(\rho)}$; therefore we have access to x_k.

For $1 \leq q \leq w$: Hybrid q involves two honest parties which we denote by P_i and P_j. Their equations share the monomial $x_i x_j$. There are $w = h(h-1)/2$ such monomials and the goal of each Hybrid q is to replace the q-th monomial with a uniformly random element.

Hybrid q: The changes between Hybrid q and Hybrid $q-1$ involve changing the protocol messages of the honest parties P_i and P_j in all ℓ rounds. Let $m_i^{(\rho)}$ and $m_j^{(\rho)}$ be the inputs of these honest parties in round ρ. Generate a uniformly random integer $r \in \{0, \ldots, p-1\}$ and replace all occurrences of $g^{x_i x_j}$ by g^r in the computation of the second messages in all rounds.

Hybrid $q-1$ and Hybrid q are computationally indistinguishable under the DDH assumption. Hybrid $q-1$ involves the DDH instance $(g, g^{x_i}, g^{x_j}, g^{x_i x_j})$ and Hybrid q involves the DDH instance $(g, g^{x_i}, g^{x_j}, g^r)$ where x_i, x_j and r are uniformly distributed in $\{0, \ldots, p-1\}$. A non-negligible advantage distinguishing between Hybrid 0 and Hybrid 1 implies a non-negligible advantage against DDH.

Hybrid w + 1: (where $w = h(h-1)/2$) H is modelled as a random oracle and as such the skew-symmetric matrices contain uniformly random elements in $\mathbb{Z}_p$. In this Hybrid, we program H such that the joint coefficient matrix $M \in \mathbb{Z}_p^{\ell(n-t-1) \times (n-t)(n-t-1)/2}$ formed from the coefficient matrix in every round is linearly independent. By Lemma 1, the probability of M not being linearly independent when generated as in the real world is at most $\frac{\mathsf{poly}(n-t)}{p}$. Because p is superpolynomial in the security parameter, an adversary has a negligible chance between distinguishing Hybrid $w+1$ and Hybrid w.

Hybrid w + 2: Without loss of generality, assume that parties $P_1, \ldots, P_h$ are the honest parties. For all $1 \leq i < h$ and $1 \leq \rho \leq \ell$, replace the protocol message $v_i^{(\rho)}$ of party P_i in round ρ with $g^{r_i^{(\rho)}} \cdot g^{m_i^{(\rho)}}$ for uniformly random $r_i^{(\rho)} \in \mathbb{Z}_p$. Furthermore, for every $1 \leq \rho \leq \ell$, replace the protocol message $v_h^{(\rho)}$

with $g^{-\sum_{j=1}^{h-1} r_j^{(\rho)} + m_h^{(\rho)}}$. Due to the linear independence of the coefficient matrix $M \in \mathbb{Z}_p^{\ell(n-t-1)\times(n-t)(n-t-1)/2}$, distinguishing between Hybrid $w+2$ and Hybrid $w+1$ is impossible.

Hybrid w + 3 Finally, in this Hybrid, the inputs $m_1^{(\rho)}, \ldots, m_h^{(\rho)}$ are replaced by a random partition of $\sum_{k=1}^{h} m_k^{(\rho)}$, namely the values $s_1^{(\rho)}, \ldots, s_h^{(\rho)}$ for every $\rho \in \{1, \ldots, \ell\}$.

An adversary has a zero advantage distinguishing Hybrid $w+3$ and Hybrid $w+2$. To see this, suppose the adversary could distinguish the hybrids. Then it can determine that some party's input (say P_i) in some round ρ is not $s_i^{(\rho)}$. But $v_i^{(\rho)} = g^{r'}$ for some uniformly random r', which provides no information about the message (whether it is $m_i^{(\rho)}$ or $s_i^{(\rho)}$). Note that $v_h^{(\rho)}$ gives no additional information since it can be derived from the information known to the adversary (recall that the sum of each round is known).

Since Hybrid $w+3$ no longer relies on the honest parties' messages, and all other information needed to construct the distribution can be derived from the simulators' inputs in Equation 3, it follows that there exists an algorithm $\mathcal{S}$ that can simulate the real distribution. □

5 Performance

The proposed system is very efficient. In the initial stage, each user has to make one modular exponentiation to generate her public key. Then for each round, she has to perform $n-1$ hash function evaluations in addition to n exponentiations and n multiplications in the group $\mathbb{G}$. If blind aggregation is also employed, two more exponentiations and a single multiplication are required. Note that the cost for each individual user is linear in the number of users. Similarly, the aggregator has to perform n multiplications to recover the value and probably a brute force attack on the size of $\sum m_i$. So the time complexity of participation is linear in the number of users, while the time complexity of disclosure is $O(\sqrt{\sum m_i})$ if Pollard's lambda algorithm is used for recovery of the sum. The complexity is dominated by one or other of these factors depending on the system parameters.

5.1 Experimental results

In order to test the efficiency, we implemented our MPC protocol from Section 4 using Sage. Our results here refer to the cost of this protocol. The results that are reported were obtained on a system with an Intel Core i5-3340M processor clocked at 2.70 GHz, 4GB of RAM and running on 64 bit Debian GNU/Linux 3.2.41. The evaluated schemes that are presented are based on a multiplicative subgroup $\mathbb{Z}_q^*$ where $q = 2p+1$ for a randomly generated 512-bit prime p. The reported results refer to the mean values of 1000 experiments. In the experiment, the computation time for each user in a given stage of the protocol is calculated (as mentioned above, this relates to the MPC protocol only). We consider three stages: key generation, round evaluation and result recovery. The latter includes

aggregation and extraction of the aggregate value. We evaluate the protocol for both private sum and private product. For the former, result recovery entails Pollard's lambda algorithm. For key generation, /dev/urandom was used.

Each experiment was run 1000 times for $n = 100$ users. Each user's input value in all cases was randomly sampled in the range $\{0, \ldots, 10,000\}$ (i.e. $\beta = 10,000$). For all stages, we measure the cost of the computation on one user's side. The results are illustrated in Table 5.1. ' Suppose a satisfactory collusion

	Average	**Min**	**Max**	**StDev**
Key Generation	0.166	0.157	0.370	0.009
Round Computation (Our protocol)	27.19	26.42	28.67	0.42
Round Computation (Kursawe et al.)	0.88	0.32	2.24	0.37
Result Recovery (Sum)	10.15	7.18	40.78	4.66
Result Recovery (Product)	0.082	0.078	0.115	0.003

Table 1. Experimental results, time in ms

tolerance is 1/3; that is, $t = 33$ for $n = 100$, then we can securely evaluate $\lfloor \frac{n-t}{2} \rfloor = 33$ rounds. Although each round is more expensive than that of Kursawe et al., we amortize key generation (which is comparitively not too expensive from the table), potential key verification and bandwidth for key broadcasts over a large number of rounds.

6 Applications

If we assume that the users are fair and follow the protocol without trying to maliciously manipulate their messages, then many applications can be achieved with the proposed schemes.

6.1 Collective Decision-making

A typical example is the case of collective decision-making, essentially e-voting where no party has a vested interest in any outcome, on an untrusted server. Typical e-voting schemes depend on a trusted third party for performing the elections. However, using anonymous aggregation, the election can be made on an untrusted server. The case of a Yes/No decision is quite straightforward and an evaluation in terms of time requirements is provided in the previous section. Additionally, users can select from κ options. To achieve this, if the prime modulus is p and we have n users, we need $prime[\kappa-1]^n < p$, where $prime[i]$ denotes the i^{th} prime and $prime[0] = 1$. In this case, we map each of the κ candidates to one of the first κ values of $prime[x]$. To cast the vote for option j, user i on the second round broadcasts:

$$prime[j]g^{x_i y_i} mod\ p$$

Thus the aggregator will easily retrieve the number $v = \prod prime^{k_j}$, where $\sum k_j = n$. Since $prime[\kappa-1]^n < p \Rightarrow v < p$. In order to recover the votes, we have to factor v which is a smooth number, divided only by $prime[x], x \in \{0, ...\kappa-1\}$, which can be done efficiently.

Only non-contentious decisions can be made with this protocol, as in any contentious election or decision, there is no way to prevent ballot-stuffing, and some vote rigging may even be undetectable. It is not reasonable to assume non-malicious behaviour for contentious issues or elections. There are many instances where it is in the interests of all parties to a decision to determine the honest answer to a question. Examples include when the appropriate action for a system to take is based on sensed data, which may contain errors, so it is important to determine whether a state being reported by some nodes in a wireless sensor network is the most prevalent state.

6.2 Anonymous Statistics

It is frequently of value to gather aggregate information from a population who do not wish to disclose the relevant information regarding themselves. It may be in everyone's interests to make the information available to all parties, including themselves, but the desire to avoid revealing sensitive information may override the benefit derived from having the accurate information available to everyone. Examples may include determining the prevalence of threats, those affected by which would not like to publicize their vulnerability. This could include physical preparedness of homes, business premises or military installations against invasion or intrusion, or the patch state of critical nodes following discovery of a vulnerability in their software. Knowledge of how many systems are potentially vulnerable is highly valuable to the organization in planning contingency measures or allocating resources to resolution, however the possibility that a node may have been corrupted an d leaking information makes it unacceptable to have nodes directly provide this kind of information directly, as it could be used to direct attacks exploiting the vulnerability.

6.3 Privacy Preserving Collaborative Filtering

Due to the wide growth of e-commerce, automatic recommender systems and more specially Collaborative Filtering have become standard components in many services. In order to enable more private solutions, Privacy Preserving Collaborative Filtering has been introduced [9, 8]. The proposed protocol can allow users to send their preferences preserving their privacy, without the use of a trusted third party.

6.4 Urban-scale sensor aggregation

As already discussed, current state of the art does not allow large-scale aggregation. One solution is to have a large key, but this introduces a significant

performance cost, mainly on the user side. The other solution is to install more aggregators. While this will keep the size of the key short enough, it will demand the introduction of many trusted third parties (the local aggregators). However, in smart cities the environment is very heterogeneous[7, 22, 27, 28] this translates to many installations from many parties. The proposed solution though manages to minimize this cost by hiding the local summaries from the aggregators, therefore many services can use them without needing to install separate ones.

7 Conclusions

In this work we introduced the notion of an anonymous aggregation system and presented a practical realization. As a building block of the latter, we presented an MPC protocol that allows multiple rounds of aggregation to be performed by lowering the collusion threshold. We showed this to be secure in the semihonest model assuming the hardness of DDH. Coupled with a method of blind aggregation, our proposed system offers scalability and input privacy without the use of trusted third parties.

References

1. Gilad Asharov and Yehuda Lindell. A full proof of the bgw protocol for perfectly-secure multiparty computation. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:36, 2011.
2. Assaf Ben-David, Noam Nisan, and Benny Pinkas. Fairplaymp: a system for secure multi-party computation. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 257–266. ACM, 2008.
3. Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 1–10. ACM, 1988.
4. I. F. Blake and C. Studholme. Properties of random matrices and applications. Unpublished report available at `http://www.cs.toronto.edu/~cvs/coding`, 2006.
5. Peter Bogetoft, Dan Lund Christensen, Ivan Damgård, Martin Geisler, Thomas P Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, et al. Multiparty computation goes live. *IACR Cryptology ePrint Archive*, 2008:68, 2008.
6. Dan Boneh. The decision diffie-hellman problem. In *Algorithmic number theory*, pages 48–63. Springer, 1998.
7. Francesco Calabrese, Massimo Colonna, Piero Lovisolo, Dario Parata, and Carlo Ratti. Real-time urban monitoring using cell phones: A case study in rome. *Intelligent Transportation Systems, IEEE Transactions on*, 12(1):141–151, 2011.
8. Fran Casino, Josep Domingo-Ferrer, Constantinos Patsakis, Domenec Puig, and Agusti Solanas. Privacy preserving collaborative filtering with k-anonymity through microaggregation. In *e-Business Engineering (ICEBE), 2013 IEEE 10th International Conference on*, pages 490–497. IEEE, 2013.
9. Fran Casino, Constantinos Patsakis, Domenec Puig, and Agusti Solanas. On privacy preserving collaborative filtering: Current trends, open problems, and new issues. In *e-Business Engineering (ICEBE), 2013 IEEE 10th International Conference on*, pages 244–249. IEEE, 2013.

10. David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1:65–75, 1988.
11. David Chaum, Claude Crépeau, and Ivan Damgard. Multiparty unconditionally secure protocols. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 11–19. ACM, 1988.
12. Chris Clifton, Murat Kantarcioglu, Jaideep Vaidya, Xiaodong Lin, and Michael Y Zhu. Tools for privacy preserving distributed data mining. *ACM SIGKDD Explorations Newsletter*, 4(2):28–34, 2002.
13. C. Cooper. On the rank of random matrices. *Random Struct. Algorithms*, 16:2000, 2000.
14. I. Damgård, M. Geisler, M. Krøigaard, and J. Nielsen. Asynchronous Multiparty Computation: Theory and Implementation. *Public Key Cryptography*, pages 160–179, 2009.
15. I. Damgard, V. Pastro, N.P. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. Cryptology ePrint Archive, Report 2011/535, 2011. `http://eprint.iacr.org/`.
16. O. Goldreich. Towards a Theory of Software Protection and Simulation by Oblivious RAMs. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 182–194, 1987.
17. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 218–229. ACM, 1987.
18. Philippe Golle and Ari Juels. Dining cryptographers revisited. In *Advances in Cryptology-Eurocrypt 2004*, pages 456–473. Springer, 2004.
19. S. Dov Gordon, Jonathan Katz, Vladimir Kolesnikov, Fernando Krell, Tal Malkin, Mariana Raykova, and Yevgeniy Vahlis. Secure two-party computation in sublinear (amortized) time. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 513–524, New York, NY, USA, 2012. ACM.
20. Feng Hao and Piotr Zieliński. A 2-round anonymous veto protocol. In *Security Protocols*, pages 202–211. Springer, 2009.
21. Klaus Kursawe, George Danezis, and Markulf Kohlweiss. Privacy-friendly aggregation for the smart-grid. In *Privacy Enhancing Technologies*, pages 175–191. Springer, 2011.
22. Nicholas D Lane, Shane B Eisenman, Mirco Musolesi, Emiliano Miluzzo, and Andrew T Campbell. Urban sensing systems: opportunistic or participatory? In *Proceedings of the 9th workshop on Mobile computing systems and applications*, pages 11–16. ACM, 2008.
23. Yehuda Lindell. Fast cut-and-choose based protocols for malicious and covert adversaries. *IACR Cryptology ePrint Archive*, 2013:79, 2013.
24. Yehuda Lindell and Benny Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In *Advances in Cryptology-EUROCRYPT 2007*, pages 52–78. Springer, 2007.
25. Yehuda Lindell and Benny Pinkas. Secure two-party computation via cut-and-choose oblivious transfer. *Journal of cryptology*, 25(4):680–722, 2012.
26. Dahlia Malkhi, Noam Nisan, Benny Pinkas, and Yaron Sella. Fairplay-secure two-party computation system. In *USENIX Security Symposium*, pages 287–302, 2004.
27. A. Manzoor, C. Patsakis, M. Bouroche, S. Clarke, V. Cahill, J. McCarthy, and G. Mullarkey. Data sensing and dissemination framework for smart cities. *Proceedings of MobilWare 2013, November 11-12, Bologna, Italy.*, 2013.

28. A. Manzoor, C. Patsakis, A Morris, J. McCarthy, G. Mullarkey, H. Pham, S. Clarke, V. Cahill, and M. Bouroche. CityWatch: Exploiting Sensing Data to Manage Cities Better. *Transactions on Emerging Telecommunication Technologies*, 2014.
29. Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A new approach to practical active-secure two-party computation. In *Advances in Cryptology–CRYPTO 2012*, pages 681–700. Springer, 2012.
30. E. Shi, R. Chow, T. H. H. Chan, Dawn Song, and Eleanor Rieffel. Privacy-Preserving Aggregation of Time-Series Data. Technical report, UC Berkeley, 2011.
31. Z. Yang, S. Zhong, and R. N. Wright. Privacy-Preserving Classification of Customer Data Without Loss of Accuracy. In *SIAM International Conference on Data Mining*, pages 1–11, 2005.
32. Andrew Chi-Chih Yao. Protocols for secure computations. In *FOCS*, volume 82, pages 160–164, 1982.
33. Andrew Chi-Chih Yao. How to generate and exchange secrets. In *Foundations of Computer Science, 1986., 27th Annual Symposium on*, pages 162–167. IEEE, 1986.