Anonymous aggregators and sensor aggregation on untrusted servers

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 reached yet 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 the concepts of *Anonymous Aggregation* and *Anonymous Aggregators*, two lightweight cryptographic primitives that can perform specific private computations efficiently in restricted environments.

Keywords: cryptographic protocols, privacy, anonymity, proxy re-encryption, multiparty computations

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 obfuscated 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.

1.1 Main contributions

In this work *Anonymous aggregation* and *Anonymous Aggregators* are introduced, two lightweight cryptographic primitives that allow users to efficiently perform specific private computations, such as multiplication and addition, under specific constraints. It is shown that the output of the function can be computed correctly if users' inputs have been properly bounded. The main difference

with secure multiparty computation is that we lower the security barrier. In the calculations that are presented in this work all the involved entities 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. An important limitation, which depending on the problem can be circumvented through proper implementation, is that the size of the key increases exponentially with the expected output. However, as it is illustrated in the experiments, this exponential behavior does not prevent the usage of anonymous aggregators in practical problems.

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, such as secure multiparty computation. The third section describes some of the building blocks that are needed to construct *Anonymous aggregation* and *Anonymous Aggregators*. Section §4 introduces the two cryptographic primitives, while in Section §5 some experimental results illustrate their efficiency. 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 [29] 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 [30, 16, 11, 3], however it took almost two decades for real-world implementations to became practical [24, 2, 5].

Currently, the prevailing trends in secure multiparty computation can be categorized as follows, based on chronological order:

- The usage of the original concept from Yao, which is based on computing encrypted binary circuits of the function to be evaluated. Efficient implementations of this approach have only recently been introduced [22, 23, 21].
- Nielsen's approach, introduced in [27]. In this approach, oblivious transfer is being used and it is mainly focused towards Secure two-party computation (2PC), such as computing hash functions.
- The SPDZ approach has been introduced in [13] and extended in [14]. SPDZ is based on the principles of fully homomorphic encryption and can perform general calculations from an arbitrary number of users.

Apart from the aforementioned schemes which realize secure multiparty computation, closely related to this work are others which might not provide all the aforementioned features; nevertheless, in well defined and restricted environments, they can provide more efficient solutions, in terms of performance.

Clifton et al. in [12] proposed a very efficient and elegant scheme for summing the shares of n entities anonymously. Their scheme is illustrated in Figure 1. 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.

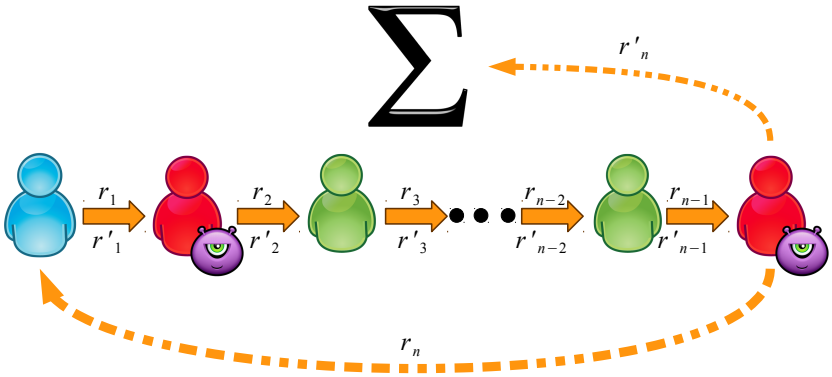

Fig. 1. Clifton et al anonymous sum method.

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 [17] 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 [18] which is discussed in the next section.

3 Building blocks

3.1 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 [18]. 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.

3.2 Re-encryption

Re-encryption is a cryptographic primitive that enables the transition of a ciphertext from one decryption key, to another, without the use of any trusted third party, or disclosing the keys of the users involved. The concept was introduced in 1998 by Blaze, Bleumer and Strauss in [4].

The original proposal is a tweaked version of the well known ElGamal encryption algorithm [15]. So to initialize the algorithm, we have pick a prime number p of the form $p = 2q + 1$, where q is prime, and a generator g. We assume that

Alice and Bob are the two users that are going to use this algorithm, so Alice and Bob pick randomly a and b respectively, so that $a, b \in \mathbb{Z}_p$. Alice's public key is $g^a mod\ p$ and Bob's is $g^b mod\ p$. If Caroline wants to send message m to Alice, she picks a random value r and computes:

$$c_1 \equiv m g^r mod\ p$$

$$c_2 \equiv g^{ar} mod\ p$$

To decrypt the ciphertext, Alice has to calculate:

$$m \equiv c_1 (c_2^{a^{-1}})^{-1} mod\ p$$

We now assume that Alice goes away and she wants her messages to be forwarded to Bob. However, she does not want to disclose her key to him, or any other entity, as this will reveal all her previous messages. Therefore she asks her email server to use an intermediate key $g^{a^{-1}b}$. If the mail server is given this key, every new mail to Alice will be forwarded to Bob and Bob will be able to decrypt it, even if it was encrypted for him. Additionally, since $g^{a^{-1}b} = (g^{a^{-1}})^b$ nor Alice nor Bob have to disclose their private keys and the intermediate key can be easily calculated by Alice and handed to the mail server.

4 Anonymous aggregation & aggregators

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 Main players 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.

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.

In the proposed model, the entities are clearly distinct just in order to facilitate the description of the methodologies. As it will become apparent, a User can also be the Consumer, or the Consumer and the Aggregator can be the same entities.

4.3 Definitions of the concepts

Going back to the Anonymous veto protocol of Hao and Zieliński that was described in the previous section, we assume for sake of simplicity that user U_1 decides to veto. This means that the end product will not be equal to 1. What will really happen is that we will have:

$$\prod_{i=1}^{n} g^{c_i y_i} \equiv g^{c_1 y_1} \prod_{i=2}^{n} g^{x_i y_i} \equiv g^{r_1} g^{x_1 y_1} \prod_{i=2}^{n} g^{x_i y_i} \equiv g^{r_1} \prod_{i=1}^{n} g^{x_i y_i} \equiv g^{r_1}$$

Thus U_i has not only managed to pass anonymously his veto, but his message g^{r_1}. In the same if way we assume that U_2 had tried to send a his message anonymously, along with the U_1, then clearly the end result would be $g^{r_1} g^{r_2} \equiv g^{r_1+r_2}$. From the definition of the Anonymous veto protocol, the values r_1 and r_2 cannot be recovered, nor the identity of the ones who submitted them. However, the question now becomes whether the value $r_1 + r_2$ can be extracted.

The best way to answer the question is through an example. We assume that $g = 2$ and p is k bits long. If the values r_1 and r_2 are small enough, so that $g^{r_1+r_2}$ can be easily brute forced, then the sum can be easily extracted. Additionally, if we set $g^{r_i} \equiv m_i$, then instead of the sum, the product of small values can be extracted.

Generalizing the previous example, we define the *anonymous aggregation* as follows.

Definition 1. *An anonymous aggregation is a quadruple of algorithms* $AAn = (KeyGen,\ Br,\ Pub,\ Aggr)$*, for key generation, broadcast, publishing and aggregation. More precisely:*

- *The key-generation algorithm* $(p, g, \mathbb{G}) \leftarrow KeyGen$ *outputs a prime number* p*, a group* $\mathbb{G}$ *of order* p *for which the DDH problem is intractable and generator* g *of the group* $\mathbb{G}^*$*.*
- *A randomized broadcasting algorithm* $g^r \leftarrow Br(\mathbb{G})$*.*
- *A one way function* $g^k \leftarrow Pub(\{g^{r_1}, ..., g^{r_n}\}, r, m)$*.*
- *A deterministic algorithm* $f(m_1, ...m_n) \leftarrow Aggr(s)$*.*

For which

$$\prod_{i=1}^{n} Pub(\{g^{r_1}, ..., g^{r_n}\}, r_i, m_i) = g^{\sum_{i=1}^{n} m_i}$$

and $g^{\sum_{i=1}^{n} m_i}$ *can be brute forced.*

As it becomes apparent, in the case of anonymous aggregation, the Consumer and the Aggregator are the same entity. The approach above is followed in [19] to aggregate the measurements from smart meters. For several applications however we might need these two entities to be distinct or there might be scalability or network infrastructure issues. As already discussed the aggregation is bounded up to a specific integer, therefore if the consumers are many, so the sum is expected to be big, the options are two, either increase the size of the key, or install more aggregators. The first solution will introduce a significant performance overhead, while the latter will demand the installation of more aggregators aka more trusted third parties. Moreover, the network infrastructure which might introduce other restrictions due to high mobility and lack of trust. The aforementioned restrictions for instance are common in user-centric networks, so one of the users might have to become the aggregator. Obviously the anonymous aggregation approach will reveal the summation value to a user, which cannot by any chance be considered as a trusted third party from a service provider.

Therefore, we can extend the anonymous aggregation to allow the same functionality, but without disclosing the result to the Aggregator. For this reason one more layer can be added to an anonymous aggregation scheme to provide re-encryption, so an *anonymous aggregator* is a proxy, between the Users and the Consumer.

Modifying the original scheme, we have: Initially, users have agreed on the triplet of values (a, g, p) and they have also agreed with the Consumer to use $g^{a^{-1}b} mod\ p$ for re-encryption, where $g^b mod\ p$ is Consumer's public key.

In the first round each user U_i creates a random value x_i and broadcasts $g^{x_i} mod\ p$. Each user may now calculate:

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

In the second round, each user U_i picks a random value r_i and broadcasts

$$c_{i,1} \equiv M_i g^{r_i} mod\ p$$

$$c_{i,2} \equiv g^{ar_i} mod\ p$$

where:

$$M_i \equiv g^{x_i y_i + m_i}$$

The anonymous aggregator calculates:

$$C_1 \equiv \prod c_{i,1} \equiv \prod M_i g^{r_i} mod\ p \equiv g^{\sum r_i} \prod M_i$$

$$C_2 \equiv \prod c_{i,2} \equiv \prod g^{ar_i} mod\ p \equiv g^{a \sum r_i}$$

Thus the value $\prod M_i$ remains secret to the anonymous aggregator. The anonymous aggregator can re-encrypt (C_1, C_2) for the consumer, who finally, recovers:

$$\prod M_i mod\ p \equiv g^{\sum x_i y_i + m_i} \equiv g^{\sum m_i}$$

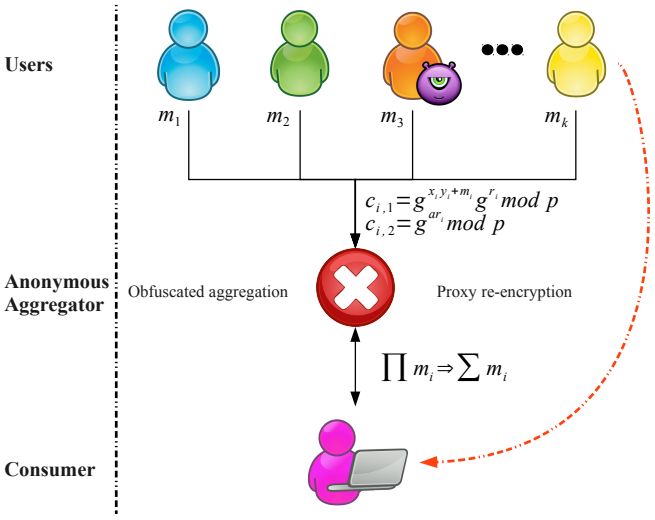

Fig. 2. Anonymous aggregator

The scheme is illustrated in Figure 2.

Due to the intractability of the DDH problem, one can assume that $g^{\sum m_i} \equiv v_i$, so if the values of v_i are small enough so that $\prod v_i < p$ then it becomes apparent that a product can be privately evaluated, or if properly they are properly picked, their individual values can be recovered, without disclosing the identity of the user who selected them. Such examples are going to be discussed in the following section with experimental results. Depending on whether they decide that they want anonymous aggregation or anonymous aggregator, the decryption key can be shared with the aggregator.

4.4 Calculating bigger sums

Using Paillier's cryptosystem we can calculate anonymously sums of bigger integers [28]. In this scenario, we assume that the users have created a key for Paillier's algorithm, sharing the decryption key with the Consumer. Additionally, users agree on a value g that is going to be used for broadcasting their messages.

In the first round, each user U_i creates a random value x_i and broadcasts $g^{x_i} mod\ N^2$. Each user may now calculate:

$$g^{y_i} = \prod_{j=1}^{i-1} g^{x_j} \prod_{j=i+1}^{n} g^{-x_j} mod\ N^2$$

In the second round, each user U_i picks a random value r_i and broadcasts:

$$c_i \equiv g^{x_i y_i} \gamma^{m_i} r_i^N mod\ N^2$$

In order to recover $\sigma = \sum_{i=1}^{n} m_i$, the aggregator calculates the product of c_i, so:

$$\prod_{i=1}^{n} c_i \equiv \prod_{i=1}^{n} g^{x_i y_i} \gamma^{m_i} r_i^N mod\ N^2 \equiv \prod_{i=1}^{n} g^{x_i y_i} \prod_{i=1}^{n} \gamma^{m_i} r_i^N mod\ N^2$$

$$\equiv \gamma^{\sum_{i=1}^{n} m_i} \prod_{i=1}^{n} r_i^N mod\ N^2 \equiv \gamma^{\sigma} R^N mod\ N^2$$

4.5 Security Proof

In order to show that the proposed scheme provides the necessary privacy to the participants, we have to show that the scheme provides privacy against collusions of up to $n-2$ users. It is clear that collusions of $n-1$ users cannot be avoided, e.g. in the case of sum, an adversary knows the end result and the values of $n-1$ users, so the value of the target user can be found trivially. 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, and a secure broadcast channel is also assumed. 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.

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 $I \subseteq \{1, \ldots, m\}$ 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 $|m_1| = \ldots = |m_n|$,*

$$\mathsf{OUTPUT}^{\pi}(m_1, \ldots, m_n) = f(m_1, \ldots, m_n) \tag{1}$$

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:

$$\{\mathsf{VIEW}_I^{\pi}(\boldsymbol{m})\} \underset{C}{\approx} \{\mathcal{S}(I, \boldsymbol{m}_I, f_I(\boldsymbol{m}))\}. \tag{2}$$

Theorem 1. *Under the DDH assumption, our protocol is computationally t-private for all $t \leq n$.*

Proof. Let $h = n - t$ be the number of honest users. If $h \leq 1$, it is trivial to construct a simulator since $\mathcal{S}$ can fully learn $\boldsymbol{m}$ and then simulate all parties. We can write $h = 2w + z$ with $w \geq 0$ and $0 \leq z \leq 1$. Consider the following series of Hybrids.

Hybrid 0: This is the same as the real distribution i.e. the LHS of Equation 2 with the exception that we "simulate" each honest party P_k using input m_k; 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. Without loss of generality, we assume that $1 \leq i \leq j \leq n$. **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. Let m_i and m_j be the inputs of these honest parties. 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. Since we have access to all x_k for each $k \in \{1, \ldots k\} \setminus \{i, j\}$, it is straightforward to replace the term $g^{x_k x_i}$ with $g^{x_k x_i'}$ (resp. for x_j). Let $v_i = g^{x_i' y_i + m_i}$ and $v_j = g^{x_j' y_j + m_j}$ be the protocol messages for parties P_i and P_j respectively.

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 q + 1: Without loss of generality, assume that parties $P_1, \ldots, P_h$ are the honest parties. In this Hybrid, the inputs $m_1, \ldots, m_h$ are replaced by a random partition of $\sum_{k=1}^{h} m_k$, namely the values $s_1, \ldots, s_h$.

An adversary has a zero advantage distinguishing Hybrid q and Hybrid $q+1$. To see this, suppose the adversary could distinguish the hybrids. Then it can determine that some party's input (say P_i) is not s_i (let P_i and P_j be a pair of honest parties considered in one of the previous hybrids). However, this is not possible since $v_i = g^{r'}$ for some uniformly random r', which provides no information about the message (whether it is m_i or s_i). Note that v_j gives no additional information since it can be derived from v_i based on the information known to the adversary.

Since Hybrid $q+1$ 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 2, it follows that there exists an algorithm $\mathcal{S}$ that can simulate the real distribution. □

5 Performance

The proposed schemes are very efficient. In the first round each user has to make one modular exponentiation and in the second one n multiplications to find $g^{x_C y_i}$ one modular exponentiation to calculate g^{m_i} and one more modular multiplication. Hence 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 of the two primitives a Python implementation has been made and the results that are reported were made on a system with an Intel® Core™ i7-2600 CPU at 3.40GHz processor, 16GB of RAM and running on 64 bit Ubuntu GNU/Linux kernel 3.8.0-31. The evaluated schemes that are presented are based on groups $\mathbb{Z}_p$ where p is 512 bits long. The reported results refer to the mean values of 1000 experiments. The prime number generation takes on average 13.7 Mac's. The reported results calculate all the needed computations for all the users, without threading.

In our first experiment, we assume that we have 100 users, each of which picks a random number from one to ten. On average, the result can be retrieved in 88.11ms.

In the second experiment we calculate private products. We assume that have 10 users, each of which selects a random 32 bit integer. The product can be recovered in 7.7 ms .

In the last experiment we perform a decision problem with 100 participants, so that each of them votes Yes/No. On average this decision can be made privately within 8.6ms. Detailed results of the experiments above are illustrated in Table 5.1.

	Average	**Min**	**Max**	**StDev**
Private Sum	88.11	87.11	89.11	1.42
Decisions	86.04	84.69	87.38	1.90
Private product	7.7	7.46	7.93	0.33

Table 1. Experimental results, time in ms.

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 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 modulo prime 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 retrieve easily 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 publicise 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 organisation in planning contingency measures or allocating resources to resolution, however the possibility that a node may have been corrupted and 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 do 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, 20, 25, 26] 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 two cryptographic primitives, namely the Anonymous Aggregation and Anonymous Aggregators, which can be considered an extension of secure multiparty computations. Contrary to the latter, the two introduced primitives do not offer security from malicious data manipulation, but are focused on honest but curious users. In this context Anonymous Aggregation and Anonymous Aggregators offer an efficient alternative to secure multiparty computations that can be easily deployed and offer private computations to a wide range of applications. Compaired to other protocols, the proposed ones allow scalability without the use of trusted third parties or computational overhead by using longer keys. On the contrary, the role of the aggregator can be delegated to one of the users or "local" aggregators can be introduced without exposing the users' feedback.

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. Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In *Advances in CryptologyEUROCRYPT'98*, pages 127–144. Springer, 1998.
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. 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/`.
14. Ivan Damgard, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P. Smart. Practical covertly secure mpc for dishonest majority or: Breaking the spdz limits. Cryptology ePrint Archive, Report 2012/642, 2012. `http://eprint.iacr.org/`.
15. Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *Information Theory, IEEE Transactions on*, 31(4):469–472, 1985.
16. 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.
17. Philippe Golle and Ari Juels. Dining cryptographers revisited. In *Advances in Cryptology-Eurocrypt 2004*, pages 456–473. Springer, 2004.
18. Feng Hao and Piotr Zieliński. A 2-round anonymous veto protocol. In *Security Protocols*, pages 202–211. Springer, 2009.
19. Klaus Kursawe, George Danezis, and Markulf Kohlweiss. Privacy-friendly aggregation for the smart-grid. In *Privacy Enhancing Technologies*, pages 175–191. Springer, 2011.
20. 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.
21. Yehuda Lindell. Fast cut-and-choose based protocols for malicious and covert adversaries. *IACR Cryptology ePrint Archive*, 2013:79, 2013.
22. 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.
23. Yehuda Lindell and Benny Pinkas. Secure two-party computation via cut-and-choose oblivious transfer. *Journal of cryptology*, 25(4):680–722, 2012.
24. Dahlia Malkhi, Noam Nisan, Benny Pinkas, and Yaron Sella. Fairplay-secure two-party computation system. In *USENIX Security Symposium*, pages 287–302, 2004.
25. 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.
26. 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.
27. 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.
28. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in cryptologyEUROCRYPT99*, pages 223–238. Springer, 1999.
29. Andrew Chi-Chih Yao. Protocols for secure computations. In *FOCS*, volume 82, pages 160–164, 1982.
30. 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.