Private Over-Threshold Aggregation Protocols

Myungsun Kim[1,⋆], Abedelaziz Mohaisen[2], Jung Hee Cheon[3,⋆], and Yongdae Kim[4]

[1] University of Suwon, Suwon, South Korea
msunkim@suwon.ac.kr
[2] VeriSign Labs, VA, USA
amohaisen@verisign.com
[3] Seoul National University, Seoul, South Korea
jhcheon@snu.ac.kr
[4] Korea Advanced Institute of Science and Technology, Daejeon, South Korea
yongdaek@ee.kaist.ac.kr

Abstract. In this paper, we revisit the private κ^+ data aggregation problem, and formally define the problem's security requirements as both data and user privacy goals. To achieve both goals, and to strike a balance between efficiency and functionality, we devise a novel cryptographic construction that comes in two schemes; a fully decentralized construction and its practical but semi-decentralized variant. Both schemes are provably secure in the semi-honest model. We analyze the computational and communication complexities of our construction, and show that it is much more efficient than the existing protocols in the literature.

Keywords: Privacy-preservation, over-threshold, data privacy, user privacy.

1 Introduction

Of particular interest in many applications is the problem of *computing the over-threshold elements*, elements whose count is greater than a given value, in a *private* manner. A typical application that involves such primitive is network traffic distribution, where n network sensors need to jointly analyze the security alert broadcasted by different sources in order to find suspect sites. In such an application, and without losing generality, each of such sensors has a set of suspects and would like to collaboratively compute the most frequent on each of these sets (e.g., the count greater than κ, referred to as κ^+) without revealing the set of suspects to other sensors with whom she collaborates.

Problem Definition: Let there be n users denoted by $u_i, 1 \leq i \leq n$, and each of them has a private (multi-)set $\mathtt{X}_i$ of cardinality k. For simplicity, assume that the cardinality of each multiset is equal to each other. (We can efficiently handle the case where the cardinality of all multisets are different from each other by adding random elements.) There may exist one or more elements such that $\alpha_{i,j} = \alpha_{i,j'}$ for some $j \neq j'$.

PRIVATE κ^+ AGGREGATION PROBLEM: By the multiplicity of an element of a multiset we mean the number of times it appears in the multiset. Let $\kappa \in \mathbb{N}$, and assume κ

⋆ This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. 2012-0001243).

T. Kwon, M.-K. Lee, and D. Kwon (Eds.): ICISC 2012, LNCS 7839, pp. 472–486, 2013.

has been implicitly predefined among all users. Then the problem at hand is defined as follows: Given n multisets of cardinality k, find a set $Z = \{\alpha_1, \ldots, \alpha_\kappa\} \subset \mathtt{U} = \bigcup_{i=1}^{n} \mathtt{X}_i$ such that (*i*) for all elements $\alpha \in \mathtt{U}$, if α has a multiplicity greater than or equal to κ, then $\alpha \in Z$, (*ii*) no polynomial-time algorithm can learn any element other than the output of a κ^+ protocol, and (*iii*) no polynomial-time algorithm should know which output of the execution belongs to which user [15].

One straightforward technique to solve the problem is to use a trusted third party (TTP), where each user sends his private set to such TTP which performs the κ^+ aggregation task and reports the result back to each user. However, finding such TTP is not always possible in many applications. Moreover, compromising the TTP could lead to a complete privacy loss for all participating users.

Another approach is to use secure multiparty computation (SMC), which allows to securely compute a function over private data where users only learn the result of the function and nothing else. However, despite recent advances in their efficiency, SMC methods still require substantial computation and communication costs, making them impractical for real-world applications that mainly operate on large datasets.

A final approach is to use existing private set-operation protocols such as [14,21,13], especially multiset union protocols. These protocols securely compute all elements appearing in the union of input multisets at least τ times. Here all of them demand a priori-knowledge of the threshold value τ.

Remark 1. There have been many results [22,23,25,26] in the literature with titles containing the term "top-κ". We stress that these protocols produce the greatest κ elements in the union of the given sets in a private manner, and thus are different in their end results from our protocol. For example, there is a secure method for finding the κ-th ranked element in multiple multisets by Aggarwal et al. [1]. Repeatedly applying this protocol we can efficiently find the biggest κ elements in a distributed list.

Our Approach—Informal Descriptions: The most non-trivial part of κ^+ protocols is that we should satisfy two privacy requirements, namely the data privacy and user-privacy, at the same time. First let us take a closer look at e-voting protocols. In e-voting protocols, each ballot is mixed with a shuffle scheme which plays a crucial role in removing linkability between voters and ballots, which would hint user privacy. In fact, in e-voting literature this notion is called unlinkability. In order to emphasize the difference with e-voting protocols, we use the term user privacy. Now assuming that each element in multisets is encrypted and shuffled as in e-voting protocols, all encrypted elements can be decrypted, especially in a threshold manner, while satisfying user privacy. However, all non-κ^+ elements also are revealed, which violates data privacy in our application. Thus we need a way to keep data privacy even when all encrypted elements were decrypted. For this purpose we introduce an efficient function E that commutes with an underlying public-key encryption. More specifically, let Enc be a public-key encryption algorithm and Dec be the corresponding decryption algorithm. We demand that: (i) for all s and for all pk, $E_s \circ \mathsf{Enc}_{pk} = \mathsf{Enc}_{pk} \circ E_s$, and (ii) for all elements α, given $\mathsf{Dec}_{sk}(\mathsf{Enc}_{pk}(E_s(\alpha)))$ no algorithm can efficiently find α without s. We call this notion *double encryption*. In conclusion, our main technique is to shuffle doubly encrypted elements by each user. We should notice that all shuffle algorithms used in e-voting protocols rely on the re-randomizable property of underlying homomorphic encryption (e.g. see [8,17,18,12,11]), but its re-randomization algorithm does not change the plaintexts of input ciphertexts. However, in our protocol a double encryption scheme will change the plaintexts of input ciphertexts, which is the main difference from existing shuffle algorithms.

Summary of Our Results: In this paper, we present a formal definition of private κ^+ protocol and its security. Our operation setting is *fully decentralized* among n users over non-partitioned data. For the efficiency of our protocol, we refrain from using secure multiparty computation and construct an efficient private κ^+ protocol which is both data-private and user-private. Our construction strikes a real balance in the consumed resources and achieved security, and satisfies both privacy requirements. In particular, in its efficiency, our construction is comparable to the work in [3], which achieves its efficiency by giving up decentralized communication model, and in its security guarantees it is comparable to the work [4], which is secure but resources exhaustive. Our protocol on the other hand overcome the limitations and shortcomings of those protocols.

More specifically, our scheme does not requires a trusted party to set up the keys. Note that using Paillier cryptosystem [20] in a threshold manner demands some trusted setup. Moreover, our proposed protocol has several desirable features as follows: (1) It has $\mathcal{O}(n^2k)$ computational complexity where n is the number of users and k is the cardinality of each user's set, assuming $\kappa \leq k$, (2) It has $\mathcal{O}(n^2k)$ communication complexity, and (3) It has a linear round complexity in the number of users.

In general, real-world applications has n much smaller than k, which further justifies the efficiency of our protocol. This is, our protocol is beneficial in such real-world applications where the number of participating users is small but the size of their multisets is large. It remains an important open problem to devise a protocol whose round complexity does not depend on the number of users. Then we could make our protocol have $\mathcal{O}(nk)$ computation and communication complexity.

Organization. The rest of this paper is organized as follows. In Section 2, we discuss the related work with extensive analysis and comparison to our work. In Section 3, we outline the preliminaries required for understanding the rest of the paper, including double encryption, our formalism of κ^+ protocol and its security, and cryptographic primitives used in the context of computing the κ^+. In Section 4, we introduce our construction that comes into two forms with different requirements and guarantees and meets data privacy and user privacy. In Sections 5 and 6 we analyze the security and complexity of our work, by proving its security and showing its resources consumption requirements. Finally, we draw concluding remarks and point future work in Section 7.

2 Related Work

There has been plenty of work in the literature to solve the problem of private data aggregation. Such schemes can be classified under three schools of thoughts: fully centralized, fully decentralized, and semi-decentralized. While the centralized schemes assume the existence of a trusted third party (TTP), which makes them of less interest from the cryptographic and practical point of views, the fully decentralized schemes utilize cryptographic primitives and protocols to replace the centralized TTP. Finally, semi-decentralized schemes try to bridge the functional and security gap between other directions. As they are of particular relevance to our work, we limited our discussion to the decentralized and semi-decentralized protocols.

Decentralized solutions to the problem try to replace the centralized TTP with cryptographic constructions, which comes in different forms leading to several directions of research. One direction is based on SMC, as it is the case in [4]. However, at the core of that protocol's drawbacks is its inefficiency: since it uses Yao's garbled circuits [24], the computational resources required for ordinary data sizes is overwhelmingly high. Furthermore, as the datasets become disjoint, the efficiency of such construction decreases sharply. In [4], Burkhart and Dimitropoulos—in what we consider to be the most relevant work to ours—have devised a construction in which *the round complexity is linear to the number of bits* in the data elements. However, due to using sketches to improve the efficiency of subprotocols, the aggregate results are *probabilistic*. Furthermore, while the work in [4] is efficient in terms of its computational complexity, this efficiency comes at the expense of high round complexity. Kissner and Song [14] devised an over-threshold set union protocol—where *a threshold value τ should be given in advance*—to find all elements appearing in the union of input multisets at least τ times. The protocol requires a priori knowledge of the threshold, although operates in a decentralized manner. We compare it to our work in Section 6.

Finally, *semi-decentralized* constructions, represented by the work of Applebaum et al. in [3], aim to enhance the efficiency of fully-decentralized instantiations by adding new entities: proxy and database (DB). However the proxy and the DB are assumed to be semi-honest restricting the possibility of coalition between proxy and DB. This allows to obtain a constant round protocol. While the authors claim that both proxy and DB are expected to act as semi-honest, it might be a strong assumption both theoretically and practically. Furthermore, their scheme extensively relies on oblivious transfer (OT) [16], which is a very expensive public-key primitive since it may require two modular exponentiations per invocation, and run for each bit of the user's data element.

To sum up, Table 1 summarizes properties and efficiency of existing solutions compared with our proposed protocol. Computational complexity is expressed in terms of the number of multiplications over modulo p. For simplicity, we assume that multisets have values less than the prime p. Note that Applebaum et al.'s protocol requires both ElGamal encryption [7] and Goldwasser-Micali encryption [10], but we assume that both encryption systems use the same size modulus.

Table 1. Summary and Comparison

	Comm. Model	Round Cpx	Comp. Cpx	Comm. Cpx
Ours	Fully decentralized	$\mathcal{O}(n)$	$\mathcal{O}(n^2 k \log p)$	$\mathcal{O}(n^2 k \log p)$
[4]	Fully decentralized	$\mathcal{O}(n(n + k \log k) \log p)$	$\mathcal{O}(n^2 k)$	$\mathcal{O}(n^2 k \log p)$
[3]	Semi-decentralized	$\mathcal{O}(1)$	$\mathcal{O}(nk \log^2 p)$	$\mathcal{O}(nk \log p)$

3 Preliminaries

Notation. Let us denote $F(\alpha)$ for the multiplicity (also known as frequency) of an element α in a multiset $\mathtt{X}$ and $F(\mathtt{X})$ for the collection of multiplicities for all elements in the multiset $\mathtt{X}$—here the multiplicity $F(\alpha)$ of an element α refers to how many times the element appears in $\mathtt{X}$. For $n \in \mathbb{N}$, $[1, n]$ denotes the set $\{1, \ldots, n\}$. If A is a probabilistic polynomial-time (PPT) machine, we use $a \leftarrow A$ to denote A which produces output according to its internal randomness. In particular, if U is a set, then $r \xleftarrow{\$} U$ is used to denote sampling from the uniform distribution on U. A function $g : \mathbb{N} \to \mathbb{R}$ is negligible if for every positive polynomial $\mu(\cdot)$ there exists an integer L such that $g(\eta) < 1/\mu(\eta)$ for all $\eta > L$.

3.1 Definitions

Informally, a double encryption is a pair of encryption schemes $\mathcal{E} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ and $\mathsf{E} = (G, E, D)$ such that $\mathsf{Enc}(E(a)) = E(\mathsf{Enc}(a))$. We demand that an encryption scheme E be *deterministic*. The reason is that we need to know a complete distribution of multisets while hiding their elements. See Appendix A for a formal definition of public-key cryptosystem and its standard security definition.

Definition 1 (Double Encryption). *Let $\mathcal{E} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme defined as in Definition 4 with a pair of keys $(pk, sk) \leftarrow \mathsf{KG}(1^\lambda)$ and a message space (resp., ciphertext space) $\mathbf{M}_{pk}$ (resp., $\mathbf{C}_{pk}$). A pair $(\mathcal{E}, \mathsf{E})$ is called* double encryption *if there exists a triple of polynomial-time computable functions, $\mathsf{E} = (G, E, D)$, that satisfies the following properties:*

- *A probabilistic function $G(1^\lambda)$ takes as input a parameter λ, and outputs (s, s') s.t. $\forall s, s'$ and for any $m \in \mathbf{M}_{pk}$, $m = D_{s'}(E_s(m))$, E and D are deterministic.*
- *Commutativity: For all pk, s and for all $m \in \mathbf{M}_{pk}$, $\mathsf{Enc}_{pk}(E_s(m)) = E_s(\mathsf{Enc}_{pk}(m))$ up to the randomness of $\mathsf{Enc}_{pk}(\cdot)$.*
- *For all $c \leftarrow \mathsf{Enc}(m)$, $E_s(m) = \mathsf{Dec}_{sk}(E_s(c))$.*

We give an instantiation of a double encryption scheme in the following example.

Example 1. Let p be a large prime of the form $p = 2q + 1$, where q is also prime. Let $\mathbb{G}_q$ be a subgroup of $\mathbb{Z}_p^\times$ of order q with a generator g. Then a standard CPA-secure ElGamal encryption $\mathcal{E} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ is defined as follows: Selecting $x \xleftarrow{\$} \mathbb{Z}_q$, $\mathsf{KG}(1^\lambda)$ outputs (pk, sk) where $pk := (p, q, g, y = g^x \pmod p)$ and $sk := (p, q, g, x)$. Given a message $m \in \mathbb{G}_q$, the encryption algorithm Enc outputs $c = (g^r, m \cdot y^r)$ for a randomness $r \xleftarrow{\$} \mathbb{Z}_q$. Given an ElGamal ciphertext $c = (u, v)$, the decryption algorithm Dec computes $v \cdot u^{-x}$ using the secret key x. Now $\mathsf{E} = (G, E, D)$ is defined as follows:

- A probabilistic function $G(1^\lambda)$ outputs $(s, s') \in (\mathbb{Z}_q)^2$ such that $ss' = 1 \pmod q$.
- Given $\alpha \in \mathbb{G}_q$, $E : \mathbb{G}_q \to \mathbb{G}_q$ is given by $\alpha \mapsto \alpha^s \pmod p$.
- A deterministic function $D_{s'}(\beta)$ computes $\beta^{s'} \pmod p$.

Then, $(\mathcal{E}, \mathsf{E})$ is a *double encryption*:

- For all values $m \in \mathbb{G}_q$, $m = (m^s)^{s'} \pmod p$.
- For any message $m \in \mathbb{G}_q$, there exists $r' = rs$ s.t. $\left(g^{r'}, (m^s) \cdot y^{r'}\right) = ((g^r)^s, (my^r)^s)$.
- For any ElGamal ciphertext $c = (u, v) \in (\mathbb{G}_q)^2$, where $u = g^r$ and $v = my^r$, $m^s = v^s \cdot (u^s)^{-x} \pmod p$.

We use a standard definition of shuffle given by Nguyen et al. [18]; see details of the definition in [18, Def. 3 & Def. 4] and its extend version. As we mentioned before, our shuffle algorithm takes as input a list of ciphertexts, and outputs a list of *permuted* and *doubly encrypted ciphertexts*. Since a double encryption scheme leads to change the plaintexts of input ciphertexts, we need to devise proving the correctness of the shuffle.

Now we define κ^+ protocol and give its algorithmic description. Throughout the paper, we use Σ_n to denote the set of all permutations on $[1, n]$. A private κ^+ protocol consists of five computable (in polynomial time) algorithms, (Setup, DEncrypt, Shuffle, Aggregate, Reveal), over a double encryption $(\mathcal{E}, \mathsf{E})$, which are explained as follows:

$(pk, sk, s, s') \leftarrow \mathsf{Setup}(1^\lambda)$**:** The setup algorithm takes as an input the security parameter λ, and outputs the public and secret parameters for doubly encrypting input ciphertexts, by invoking $(pk, sk) \leftarrow \mathsf{KG}(1^\lambda)$ and $(s, s') \leftarrow G(1^\lambda)$.

$(E_s(c_1), \ldots, E_s(c_n)) \leftarrow \mathsf{DEncrypt}(pk, s, c_1, \ldots, c_n)$**:** The algorithm DEncrypt takes as input system parameters (pk, s) and a list of ciphertexts $(c_1, \ldots, c_n)$, and produces a list of doubly encrypted ciphertexts $(E_s(c_1), \ldots, E_s(c_n))$.

$\left(E_s(c_{\pi(1)}), \ldots, E_s(c_{\pi(n)})\right) \leftarrow \mathsf{Shuffle}(\pi, E_s(c_1), \ldots, E_s(c_n))$**:** The algorithm Shuffle chooses a random permutation $\pi \in \Sigma_n$ and shuffles the doubly encrypted ciphertexts $(E_s(c_1), \ldots, E_s(c_n))$, and then outputs the mixed list.

$(E_s(\alpha_1), \ldots, E_s(\alpha_\kappa)) \leftarrow \mathsf{Aggregate}\left(pk, sk, E_s\left(c_{\pi(1)}\right), \ldots, E_s\left(c_{\pi(n)}\right)\right)$**:** The algorithm Aggregate takes as input a pair of keys (pk, sk) and a list of permuted, doubly encrypted ciphertexts, and for all $i \in [1, n]$ computes $\mathsf{Dec}_{sk}\left(E_s\left(c_{\pi(i)}\right)\right) = E_s\left(\alpha_{\pi(i)}\right)$. Finally it computes $F\left(E_s\left(\alpha_{\pi(i)}\right)\right), i \in [1, n]$ and outputs only the elements whose multiplicity is greater than or equal than κ. Here $\{E_s(\alpha_1), \ldots, E_s(\alpha_\kappa)\} = \left\{E_s\left(\alpha_{\pi(i)}\right) \middle| F\left(E_s\left(\alpha_{\pi(i)}\right)\right) \geq \kappa\right\}$.

$(\alpha_1, \ldots, \alpha_\kappa) \leftarrow \mathsf{Reveal}\,(pk, s', E_s(\alpha_1), \ldots, E_s(\alpha_\kappa))$**:** The algorithm Reveal outputs the most frequent κ^+ elements by computing $D_{s'}\left(E_s(\alpha_j)\right)$ for all $j \in [1, \kappa]$.

In the next section, we will define the meaning of *secure* κ^+ protocol. Then we describe cryptographic building blocks for constructing a secure κ^+ protocol under proper cryptographic assumptions.

3.2 Security Definition

Ideal Functionality. we define the ideal functionality $\mathcal{F}_{\mathsf{topk}}$ for the κ^+ protocol

Definition 2. *There are a set of n users, $U = \{u_i\}_{i=1}^n$, a trusted party $\mathcal{T}$, and an ideal adversary $\mathcal{S}$ controlling a set of corrupted users $\Upsilon_t = \{u_{i_j}\}_{j=1}^t$ for some $t \in [0, n-1]$. Let $\mathtt{X}_i = \{\alpha_{i,j}\}_{j=1}^{k_i}$ be a multiset of user $u_{i\in[1,n]}$.*

1. *Each user u_i sends $\mathtt{X}_i$ to $\mathcal{T}$.*
2. *$\mathcal{T}$ computes the following functionality, and returns the output Z_l to each $u_{l\in[1,n]}$:*

$$Z_l = \left\{ \alpha_{i,j} \in \bigcup\nolimits_{i\in[1,n]} \mathtt{X}_i \middle| F\left(\alpha_{i,j}\right) \geq \kappa \right\}.$$

Data Privacy. Informally, *data privacy* requires that no user, or coalition of users, should learn anything about honest users' inputs except what can be trivially derived from the output itself. We can easily derive the formal definition for data privacy in κ^+ protocols following the standard privacy definition of existing protocols in the literature; an excellent reference on that is Goldreich's textbook in [9]. More specifically, we use Definition 7.5.1 (resp., Definition 7.5.3) in [9] for the semi-honest model (resp. the malicious model). Roughly speaking, this is formalized by considering an ideal world where $\mathcal{T}$ receives the inputs of the users and outputs the result of the defined functionality. We demand that in the real world application of the protocol–that is, one without the $\mathcal{T}$–no user should learn more information than in the ideal world.

User Privacy. The remaining part to conclude our definitions is user privacy. Let $Z = \{\alpha_1, \ldots, \alpha_\kappa\}$ be an output of a κ^+ protocol. Roughly speaking, no user or coalition of users should gain a non-negligible advantage in distinguishing, for all $\alpha \in Z$, an honest user u_i such that $\alpha \in \mathtt{X}_i$.

Definition 3 (User Privacy). *Let $\Pi_{\kappa,\mathcal{E},\mathsf{E}}$ be a κ^+ protocol defined as in Section 3.1 over a double encryption scheme $(\mathcal{E}, \mathsf{E})$ and $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary.*

Experiment $\mathsf{Exp}_{\mathcal{A}}^{\kappa^+}(\Pi_{\kappa,\mathcal{E},\mathsf{E}}, \lambda)$
  $(pk, sk, s, s') \leftarrow \mathsf{Setup}(\lambda)$;
  $(\mathsf{state}, \Upsilon_t, m_1, \ldots, m_{n-t}) \leftarrow \mathcal{A}_1(pk, n, t)$ *s.t.* Υ_t *is a set of corrupted t users;*
  $\sigma \xleftarrow{\$} \Sigma_n$ *and assign* $m_{\sigma(i)}$ *to each honest i-th user* $u_i \notin \Upsilon_t$;
  $(\alpha_1, \ldots, \alpha_\kappa) \leftarrow \Pi_{\kappa,\mathcal{E},\mathsf{E}}$, *where* $\mathcal{A}_1$ *interacts with the* $n - t$ *honest users;*
  $(i, j) \leftarrow \mathcal{A}_2(pk, m_1, \ldots, m_{n-t}, \mathsf{state})$;

We define the advantage of an adversary $\mathcal{A}$, running in probabilistic polynomial time:

$$\mathsf{Adv}_{\mathcal{A}}^{\kappa^+}(\Pi_{\kappa,\mathcal{E},\mathsf{E}}, \lambda) = \left| \Pr[\sigma(i) = j] - \frac{1}{n-t} \right|.$$

A κ^+ protocol is user-private *if the advantage $\mathsf{Adv}_{\mathcal{A}}^{\kappa^+}(\Pi_{\kappa,\mathcal{E},\mathsf{E}}, \lambda)$ is negligible in the security parameter λ.*

3.3 Cryptographic Assumptions and Tools

Next we outline the cryptographic tools and assumptions we use in our protocol. Let $\mathcal{G}$ be a finite cyclic group of prime order q, and let $g \in \mathcal{G}$ be a generator. Given $h \in \mathcal{G}$, the discrete logarithm problem requires us to compute $x \in \mathbb{Z}_q$ such that $g^x = h$. We denote this (unique) x by $\log_g h$. In particular groups $\mathcal{G}$ and for q large, it is assumed hard to compute x, which is said to be the Discrete Logarithm (DL) assumption.

A stronger assumption is the Decisional Diffie-Hellman (DDH) assumption. Here, given $\mathcal{G}$, a generator g of $\mathcal{G}$, and three elements $a, b, c \in \mathcal{G}$, we are asked (informally) to decide whether there exist x, y such that $a = g^x, b = g^y$, and $c = g^{xy}$. More formally, the DDH assumption states that the following two distributions are computationally indistinguishable: $\{\mathcal{G}, g, g^x, g^y, g^{xy}\}$ and $\{\mathcal{G}, g, g^x, g^y, g^z\}$ where $x, y, z \xleftarrow{\$} \mathbb{Z}_q$.

We extensively use ElGamal encryption defined in Example 1. This scheme secure against CPA attack in a DDH group $\mathbb{G}$; see Appendix A for the CPA security. In addition, we need an efficient scheme which works as follows: When each user holds a shared secret key s_i such that $s = \prod_{i=1}^{n} s_i$, the scheme allows each user to have a share s'_i satisfying $s' = \prod_{i=1}^{n} s'_i$ and $s' = s^{-1} \pmod{q}$ for a public modulus q. Indeed, we may realize the scheme by techniques studied by Algesheimer et al. [2, §5].

4 Our Construction

In this section, we describe our construction for computing the κ^+ elements privately. We begin by considering a basic setting of n users, denoted by $u_1, \ldots, u_n$. Let $\mathtt{X}_i = \{\alpha_{i,1}, \ldots, \alpha_{i,k}\}$ for all $i \in [1, n]$. Each user u_i has its private multiset $\mathtt{X}_i$, and the users wish to jointly compute $\left\{\alpha \in \bigcup_{i=1}^{n} \mathtt{X}_i \middle| F(\alpha) \geq \kappa\right\}$. For simplicity, assume that all elements are in the proper message domain $\mathbf{M}_{pk}$ of an ElGamal encryption scheme, e.g., a finite cyclic subgroup $\mathbb{G}_q$ of $\mathbb{Z}_p$ in which the DDH assumption holds. For a multiset $\mathtt{X} = \{\alpha_1, \ldots, \alpha_k\}$, we denote $\mathtt{X}^s$ as $\{\alpha_1^s, \ldots, \alpha_k^s\}$ for some $s \in \mathbb{Z}_q$. With such notation in mind, we proceed to describe our construction.

4.1 Description

Let λ be a security parameter, p be a λ-bit prime such that for some prime q, $p = 2q+1$, and $\mathbb{G}_q$ be a finite cyclic subgroup of $\mathbb{Z}_p^\times$ whose order is q, and g be a generator of $\mathbb{G}_q$.

Setup(1^λ). Each user agrees to a threshold ElGamal encryption $\mathcal{E}$ with a public/private key pair (pk, sk), which are computed as follows. Define $\mathsf{params} := (p, q, g, \mathbb{G}_q)$. Each user selects a value $x_i \xleftarrow{\$} \mathbb{Z}_q$, computes $y_i = g^{x_i}$, and sets $sk = (\mathsf{params}, x_i)$; the public key is then given by $pk := \left(\mathsf{params}, y = \prod_{i=1}^{n} y_i = g^{\sum_{i\in[1,n]} x_i} \pmod{p}\right)$. In addition, all users are distributed a share (s_i, s'_i) such that $s = \prod_{i=1}^{n} s_i$, $s' = \prod_{i=1}^{n} s'_i$, and $s \cdot s' = 1 \pmod{q}$. Notice that in threshold decryption schemes, users generally produce shares of the decrypted element, and during the operation of the schemes if one user sends a uniformly generated share instead of a valid one the decrypted element is uniform. Also, if the decrypted element is uniform, the resulting decryption reveals no information to the users.

DEncrypt. Let $I = \{1, \ldots, n\}$ be a set of indices, and let the power function $E_{s_i}(\alpha) = \alpha^{s_i} \pmod{p}$ which is deterministic.

1. Every user u_i encrypts his multiset $\mathtt{X}_i$ as follows:

$$\mathsf{Enc}_{pk}(\mathtt{X}_i) = \{\mathsf{Enc}_{pk}(\alpha_{i,1}), \ldots, \mathsf{Enc}_{pk}(\alpha_{i,k})\}$$

   where $\mathsf{Enc}_{pk}(\alpha_{i,j}) = (g^{r_{i,j}}, \alpha_{i,j} \cdot y^{r_{i,j}})$ for some randomizer $r_{i,j} \in \mathbb{Z}_q$, and sends $\mathsf{Enc}_{pk}(\mathtt{X}_i)$ to u_1.
2. User u_1 computes $\{E_{s_1}(\mathsf{Enc}_{pk}(\mathtt{X}_1)), \ldots, E_{s_1}(\mathsf{Enc}_{pk}(\mathtt{X}_n))\}$, which is denoted by Y_0.

Shuffle & DEncrypt. For $i \in [1, n]$, u_i receives vector Y_{i-1} and computes a permuted, doubly encrypted version Y_i as follows:

1. $u_{i\neq 1}$ computes

$$E_{s_i}(Y_{i-1}) = \{c_1, \ldots, c_{nk}\}$$
$$= \left\{E_{s_i}\left(E_{s_{i-1}}\left(\cdots E_{s_1}\left(\alpha_{\pi_{i-1}(1)}\right)\cdots\right)\right), \ldots, E_{s_i}\left(E_{s_{i-1}}\left(\cdots E_{s_1}\left(\alpha_{\pi_{i-1}(nk)}\right)\cdots\right)\right)\right\}.$$

   More precisely, here $\alpha_{\pi_{i-1}(\ell)} = \alpha_{\pi_{i-1}\circ\cdots\circ\pi_1(\ell)}$ for all $\ell \in [1, nk]$.
2. u_i chooses a random permutation $\pi_i \in \Sigma_{nk}$, and applies π_i to the list of $c_{\ell\in[1,nk]}$ computed above; denote the result by Y_i.
3. u_i sends Y_i to u_{i+1}; the last user u_n sends Y_n to all users.

Aggregate. Let $\mathtt{U} = \bigcup_{i=1}^{n} \mathtt{X}_i$. Every user has $E_s(\mathsf{Enc}_{pk}(\mathtt{U}))$.

1. Every user participates in a group decryption and obtains

$$E_s(\mathtt{U}) = \left\{E_s\left(\alpha_{\pi(1)}\right), \ldots, E_s\left(\alpha_{\pi(nk)}\right)\right\}$$

   where $\pi = \pi_n \circ \cdots \circ \pi_1$.
2. Every user computes $Z = \{E_s(\alpha) \in E_s(\mathtt{U}) \big| F(E_s(\alpha)) \geq \kappa\}$.

Reveal.

1. For every $E_s(\alpha) \in Z$, user u_i sends its share of $D_{s'_i}(E_s(\alpha))$ to $u_{i'}$.
2. After receiving all the shares, every user u_i computes $\alpha = D_{s'}(E_s(\alpha))$, thereby recovering the κ^+, $\{\alpha \in \mathtt{U} | F(\alpha) \geq \kappa\}$.

Efficiency. The advantage of the above protocol is multifold. First, compared to Kissner and Song's protocol [14], our protocol provides the functionality of finding a threshold value and computing the "over threshold" at the same computation and communication cost—whereas they incur different and higher costs in [14]. Second, compared to the κ^+ protocol described in [4], our protocol has a much better computational complexity. See details in Section 5. In order to present a fair comparison between our proposed protocol and Applebaum et al.'s protocol [3] we devise our protocol for a semi-decentralized model in the next section. The other purpose of our modification is to reduce the round complexity to a constant.

4.2 Semi-decentralized Construction

The most crucial drawback of the previous protocol is its $\mathcal{O}(n)$ round complexity. To avoid this problem, Applebaum et al. introduced two semi-honest users: a *proxy* which shuffles a list of input ciphertexts, and a *database* which aggregates κ^+ elements. Applying the same technique to our protocol, we obtain a constant-round κ^+ protocol.

– Assume that there are n_1 proxies and n_2 databases described as in [3].
– Each database engages in setting up a threshold ElGamal encryption and publishes a public key. Instead of users, all proxies are distributed secret shares $(s_l, s'_l)_{l\in[1,n_1]}$.
– Each user computes a list of ElGamal ciphertexts and sends it to a proxy.
– Each proxy runs DEncrypt and Shuffle, and returns the result to all databases.
– Databases perform group decryption, and get the list of encrypted κ^+ elements
– Finally, all proxies decrypt the encrypted κ^+ list and return the κ^+ to all users.

Compared to [3], our protocol does not require OT operations, nor an extra encryption scheme. Recall that Applebaum et al.'s protocol requires ElGamal encryption and Goldwasser-Micali (GM) encryption: ElGamal encryption is used to encrypt elements in multisets and GM encryption is used to encrypt their multiplicity.

5 Security Analysis

Theorem 1 (Correctness). *In the private top-κ protocol in sec. 4.1, every honest user learns the joint set distribution of all users' private inputs, i.e., each element $E_s(\alpha)$ such that $\alpha \in \bigcup_{i=1}^{n} \mathtt{X}_i$ and the number of times it appears, with overwhelming probability.*

Proof. Each player learns a randomly permuted joint multiset $E_s(\mathtt{U}) = \left\{E_s\left(\alpha_{\pi(1)}\right), \ldots, E_s\left(\alpha_{\pi(nk)}\right)\right\}$. We know that $|\mathtt{U}^s| = nk$. Since π is a permutation, for each $E_s\left(\alpha_{\pi(\ell)}\right)$ and for all $\ell \in [1, nk]$, there exist a pair of the unique index ℓ^* such that

$$\begin{aligned}\ell^* &= \pi^{-1}(\ell)\\ &= \pi_n^{-1}(\ell) \circ \cdots \circ \pi_1^{-1}(\ell).\end{aligned}$$

Namely, $E_s\left(\alpha_{\pi(\ell)}\right)$ is a unique blinded version of $\alpha_{\ell^*} \in \bigcup_{i=1}^{n} \mathtt{X}_i$. Moreover, $\forall \ell, \ell^* \in [1, nk]$, $\alpha_\ell = \alpha_{\ell^*}$ if and only if $E_s(\alpha_\ell) = E_s(\alpha_{\ell^*})$ with overwhelming probability. □

Corollary 1. *In the private top-κ protocol in Section 4.1, every honest user learns the κ^+ in the union of private multisets with overwhelming probability.*

Now we show that our protocol satisfies the privacy requirements in the semi-honest model. Let $\mathcal{T}$ be a trusted party in the ideal world which receives the private input multiset $\mathtt{X}_i$ of size k from user u_i for $i \in [1, n]$, and then returns to every user the joint multiset distribution $\{F(\alpha)\}$ for all $\alpha \in \bigcup_{i=1}^{n} \mathtt{X}_i$.

Theorem 2 (Data Privacy). *Assume that the threshold ElGamal encryption $\mathcal{E} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ is secure against CPA. In the private top-κ protocol in Section 4.1, any coalition of less than n semi-honest users learn no more information than would be given by using the same private inputs in the ideal-world model with $\mathcal{T}$.*

Proof. We assume that the ElGamal encryption scheme is CPA-secure, and so each user learns only

$$\begin{aligned}&\mathsf{Enc}_{pk}(\mathtt{X}_1), \ldots, \mathsf{Enc}_{pk}(\mathtt{X}_n);\\ &E_{s_1}(\mathsf{Enc}_{pk}(\mathtt{X}_1)), \ldots, E_{s_1}(\mathsf{Enc}_{pk}(\mathtt{X}_1)), \ldots, E_{s_{i-1}}(\mathsf{Enc}_{pk}(\mathtt{X}_1)), \ldots, E_{s_{i-1}}(\mathsf{Enc}_{pk}(\mathtt{X}_n));\\ &\quad\vdots\\ &E_{s_{i-1}}(\cdots E_{s_1}(\mathsf{Enc}_{pk}(\mathtt{X}_1))\cdots), \ldots, E_{s_{i-1}}(\cdots E_{s_1}(\mathsf{Enc}_{pk}(\mathtt{X}_n))\cdots)\end{aligned}$$

during an execution. At the end of the protocol all users further know $E_s(\mathsf{Enc}_{pk}(\mathtt{U}))$ where $\mathtt{U} = \bigcup_{i=1}^{n} \mathtt{X}_i$, and for some $\gamma_{\ell \in [1, nk]} \in \mathbb{Z}_q$

$$\begin{aligned}E_s(\mathsf{Enc}_{pk}(\mathtt{U})) &= \{E_s(\mathsf{Enc}_{pk}(\mathtt{X}_1)), \ldots, E_s(\mathsf{Enc}_{pk}(\mathtt{X}_n))\}\\ &= \left(g^{\gamma_1}, \left(\alpha_{\pi(1)}\right)^s \cdot y^{\gamma_1}\right), \ldots, \left(g^{\gamma_{nk}}, \left(\alpha_{\pi(nk)}\right)^s \cdot y^{\gamma_{nk}}\right).\end{aligned}$$

Note that π is a composition of random permutations and is unknown to all users, as the maximum coalition size is smaller than n. That is, if there exists at least an honest user, then a composition of random permutations $\pi = \pi_n \circ \cdots \circ \pi_1$ is a random permutation because at least a permutation $\pi_{i\in[1,n]}$ is secure. What is more, note that s is uniformly distributed and unknown to all users for the same reason. As s is uniformly distributed for any user inputs and π is random, no user or coalition can learn more than a set of re-randomized ElGamal encryptions. As s is uniformly distributed, a group decryption of ElGamal encryptions reveals no more than

$$\{E_s(\alpha_\ell)\}_{\ell\in[1,nk]} = E_s\left(\bigcup_{i=1}^{n} \mathtt{X}_i\right) = E_s(\mathtt{U}).$$

We know the fact that $F(\alpha) = F(\alpha^s)$ for two multisets $\mathtt{X}$ and $E_s(\mathtt{X}) \in (\mathbb{G}_q)^k$, for all $s \in \mathbb{Z}_q$ and for all $\alpha \in \mathtt{X}$. Hence we see that

$$F(E_s(\mathtt{U})) = F\left(E_s\left(\bigcup_{i=1}^{n} \mathtt{X}_i\right)\right) = F\left(\bigcup_{i=1}^{n} \mathtt{X}_i\right) = F(\mathtt{U}),$$

which can be derived from the output returned by $\mathcal{T}$ in the ideal-world model. □

Theorem 3 (User Privacy). *Assume that the threshold ElGamal encryption* Enc *is CPA-secure. The private top-*κ *protocol in Section 4.1 is user-private against any coalition of less than* n *semi-honest users.*

Proof. Assume that there is at least an honest user in the system, and that the threshold ElGamal encryption $\mathcal{E} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ is CPA-secure. After performing $\mathsf{DEncrypt}$ and $\mathsf{Shuffle}$ algorithms, every user obtains a collection of ElGamal encryptions $\{c_1, \ldots, c_{nk}\}$. By the second assumption, the adversary cannot learn any further information except that which encryptions have been sent from which users. Running these algorithms, each user should raise the power of the received encryptions with his shared secret s_i. Namely, each user holds the modified list of the encryptions,

$$\{E_{s_i}(c_1), E_{s_i}(c_2), \ldots, E_{s_i}(c_{nk})\}.$$

Next the user should apply his private permutation π_i to the list to transform it to

$$\left\{E_{s_i}\left(c_{\pi(1)}\right), E_{s_i}\left(c_{\pi(2)}\right), \ldots, E_{s_i}\left(c_{\pi(nk)}\right)\right\}.$$

At the end of running the algorithms, all users get a permuted and doubly encrypted list

$$\left\{E_s\left(c_{\pi(1)}\right), E_s\left(c_{\pi(2)}\right), \ldots, E_s\left(c_{\pi(nk)}\right)\right\}$$

where the permutation $\pi = \pi_n \circ \cdots \circ \pi_1$ and $s = \prod_{i=1}^{n} s_i$. As there exists at least an honest user, even when $n-1$ users collude, s is uniformly distributed and unknown to all users and π is a random permutation. This completes the proof of the claim. □

Theorem 4. *Assuming that the threshold ElGamal encryption is CPA-secure and the DL assumption holds, the proposed top-*κ *protocol is secure in the semi-honest model.*

Proof. We complete the proof of security by Theorem 2 and Theorem 3. □

6 Efficiency Analysis

The private κ^+ protocol has not yet been implemented, but we give a detailed analysis of the running time and space requirements as follows. We base our protocol on ElGamal encryption and the power function with primes $|p| = 1024, |q| = 160$. To measure users' overhead, we count the number of exponentiations using a 1024-bit modulus.

Table 2. Complexity Analysis

	Comp. Cpx (expo.)	Comm. Cpx (bits)	Rounds Cpx
Setup	n	$n \log p$	1
DEncrypt & Shuffle	$4nk + 2n^2k$	$2(n-1)k \log p + 2n^2 k \log p$	n
Aggregate	n^2k	$2n^2k \log p$	1
Reveal	$n\kappa$	$n\kappa \log p$	$n-1$

In Table 2 we show a summary of the complexity result for our proposed protocol. The total computational complexity is dominated by DEncrypt and Shuffle algorithms. Putting the computational complexities together shows that total computation complexity is $\mathcal{O}(n^2k)$ in $\mathcal{O}(n)$ rounds. The proposed protocol has $\mathcal{O}(n^2k \log p)$ bits in total. It is impossible to directly compare our protocol with Applebaum et al.'s protocol, since it runs in the semi-decentralized model, we just present the computational complexity.

Comparison. We consider three protocols: Kissner and Song (KS) protocol [14], Burkhart and Dimitropoulos (BD) protocol [4], and Applebaum et al. protocol.

Based on KS Protocol. We first compare our work with a KS-based κ^+ protocol. As mentioned earlier, since it does not provide a way for finding τ, we do not know computational and communication complexity in computing τ. Assuming τ is given, their protocol has $\mathcal{O}(n^2k)$ computation complexity in $\mathcal{O}(n)$ rounds.

BD Protocol. In turn, we give a comparison with BD protocol. To our knowledge, it is the only fully decentralized κ^+ protocol that does not use Yao's garbled circuit evaluation. Their protocol utilizes two special-purpose sub-protocols–`equality` and `lessthan` (see [6,19]), but in [5] as the authors pointed out, comparison of two shared secrets is very expensive and computational intensive. Thus, they use a computationally efficient version of the basic sub-protocols as follows: `equality` requires $\log p$ rounds and `lessthan` requires $(2 \log p + 10)$ rounds. Their protocol consists of two key ingredients as follows:

- Finding the correct τ: takes $(\log k(2 \log p + 10) + \log p + 2 \log p + 10)\, nk$ rounds.
- Resolving collisions: Requires $\frac{n(n-1)}{2} \log p + 2(n-1) \log p + 10(n-1)$ rounds.

Note that BD protocol also should know τ as in KS protocol. Hence, the total round complexity is $\mathcal{O}(n(n + k \log k) \log p)$ for hash tables of size $\log k$ and U of nk. We find their protocol takes $4\left(\frac{n(n-1)}{2}k + k(n-1)\right)$ multiplications in $\mathbb{Z}_p^{\times}$.

Applebaum et al. Protocol. Let us use $\mathcal{O}_p(\cdot)$ to denote complexity using modulus prime p and $\mathcal{O}_N(\cdot)$ complexity using modulus composite N. Assume all elements are integers less than p and the maximum multiplicity is less than $\log\log p$.
Their major computation-intensive parts are as follows:

– Interactive computation between Users and Proxy: First, users should run a protocol for oblivious evaluation of pseudorandom function by communicating with proxies, then encrypt the result with ElGamal encryption. This requires $n(k(2\log p + 2) + 2k)$ exponentiations over $\mathbb{Z}_p^\times$. Also, users should encrypt the multiplicity of each element with GM encryption, requiring $nk\log\log p$ multiplications over $\mathbb{Z}_N^\times$. Finally each user doubly encrypts their elements using ElGamal encryption. This requires $2nk$ exponentiations over $\mathbb{Z}_p^\times$.
– Aggregation by Database: The most computationally-intensive part is ElGamal and GM decryption. Since database receives two types of ElGamal ciphertexts, it performs $2nk$ exponentiations over $\mathbb{Z}_p^\times$. GM decryption requires $2nk\log\log p$ exponentiations over $\mathbb{Z}_N^\times$.

Thus, the complexity is $\mathcal{O}_p(nk\log p) + \mathcal{O}_N(nk\log\log p)$ exponentiations.

7 Conclusion

In this paper we have looked at the problem of finding the κ^+ element securely, and formally defined what it means for a protocol to be a secure κ^+ protocol. We developed two protocols, with varying operation overhead, analyzed their security, and demonstrated their practicality. In the near future, we will investigate two directions. First, since our constructions' security is proven in the semi-honest model—which is rationalized by the application domain, we will investigate constructions that are provably secure in the malicious model, and their potential applications. Second, as the shuffling algorithm in our current construction requires sending messages among players in a relay manner, we will consider the practical and security aspects of a construction that relies on sending such messages in a broadcast manner.

Acknowledgement. We thank Burt Kaliski for his feedback on an earlier version.

References

1. Aggarwal, G., Mishra, N., Pinkas, B.: Secure computation of the k^{th}-ranked element. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 40–55. Springer, Heidelberg (2004) 473
2. Algesheimer, J., Camenisch, J., Shoup, V.: Efficient computation modulo a shared secret with application to the generation of shared safe-prime products. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 417–432. Springer, Heidelberg (2002) 479
3. Applebaum, B., Ringberg, H., Freedman, M.J., Caesar, M., Rexford, J.: Collaborative, privacy-preserving data aggregation at scale. In: Atallah, M.J., Hopper, N.J. (eds.) PETS 2010. LNCS, vol. 6205, pp. 56–74. Springer, Heidelberg (2010) 474, 475, 480, 481
4. Burkhart, M., Dimitropoulos, X.: Fast privacy-preserving top-k queries using secret sharing. In: IEEE ICCCN (2010) 474, 475, 480, 483

5. Burkhart, M., Strasser, M., Many, D., Dimitropoulos, X.: SEPIA: Privacy-preserving aggregation of multi-domain network events and statistics. In: USENIX Security (2010) 483
6. Damgård, I., Fitzi, M., Kiltz, E., Nielsen, J.B., Toft, T.: Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 285–304. Springer, Heidelberg (2006) 483
7. El Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985) 475
8. Furukawa, J., Sako, K.: An efficient scheme for proving a shuffle. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 368–387. Springer, Heidelberg (2001) 474
9. Goldreich, O.: The foundations of cryptography. Cambridge University Press (2004) 478
10. Goldwasser, S., Micali, S.: Probabilistic encryption. J. Comput. Syst. Sci. (1984) 475, 486
11. Groth, J.: A verifiable secret shuffle of homomorphic encryptions. J. of Cryptology (2010) 474
12. Groth, J., Lu, S.: Verifiable shuffle of large size ciphertexts. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 377–392. Springer, Heidelberg (2007) 474
13. Hong, J., Kim, J.W., Kim, J., Park, K., Cheon, J.H.: Constant-round privacy preserving multiset union. In: Cryptology ePrint Archive, 2011/138 (2011) 473
14. Kissner, L., Song, D.: Privacy-preserving set operations. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 241–257. Springer, Heidelberg (2005) 473, 475, 480, 483
15. Mohaisen, A., Hong, D., Nyang, D.: Privacy in location based services: Primitives toward the solution. In: NCM (2008) 473
16. Naor, M., Pinkas, B.: Oblivious transfer with adaptive queries. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 573–590. Springer, Heidelberg (1999) 475
17. Neff, C.: A verifiable secret shuffle and its application to e-voting. In: ACM Conference on Computer and Communications Security, pp. 116–125 (2001) 474
18. Nguyen, L., Safavi-Naini, R., Kurosawa, K.: Verifiable shuffles: A formal model and a paillier-based efficient construction with provable security. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 61–75. Springer, Heidelberg (2004) 474, 477
19. Nishide, T., Ohta, K.: Multiparty computation for interval, equality, and comparison without bit-decomposition protocol. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 343–360. Springer, Heidelberg (2007) 483
20. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999) 474
21. Sang, Y., Shen, H.: Efficient and secure protocols for privacy-preserving set operations. ACM Transactions on Information and System Security (TISSEC) 13(1), 9:1–9:35 (2009) 473
22. Vaidya, J., Clifton, C.: Privacy-preserving top-k queries. In: ICDE (2005) 473
23. Xiong, L., Chitti, S., Liu, L.: Top$k$ queries across multiple private databases. In: International Conference on Distributed Computing Systems (ICDCS), pp. 145–154 (2005) 473
24. Yao, A.: Protocols for secure computations. In: FOCS, pp. 160–164 (1982) 475
25. Zhang, R., Shi, J., Liu, Y., Zhang, Y.: Verifiable fine-grained top-k queries in tiered sensor networks. In: INFOCOM, pp. 2633–2641 (2010) 473
26. Zhang, R., Zhang, Y., Zhang, C.: Secure top-k query processing via untrusted location-based service providers. In: INFOCOM, pp. 1170–1178 (2012) 473

A Basic Definitions

We first give a formal definition of a public key cryptosystem and then its standard security definition. We shall write

$$\Pr[x_1 \stackrel{\$}{\leftarrow} X_1, x_2 \stackrel{\$}{\leftarrow} X_2(x_1), \ldots, x_n \stackrel{\$}{\leftarrow} X_n(x_1, \ldots, x_{n-1}) : \varphi(x_1, \ldots, x_n)]$$

to denote the probability that when x_1 is drawn from a certain distribution X_1, and x_2 is drawn from a certain distribution $X_2(x_1)$, possibly depending on the particular choice of x_1, and so on, all the way to x_n, the predicate $\varphi(x_1, \ldots, x_n)$ is true.

Definition 4. *A* public-key cryptosystem $\mathcal{E}$ *is a 3-tuple of PPT algorithms* $(\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ *such that*

1. *The key generation algorithm* KG *takes as input the security parameter* λ *and outputs a pair of keys* (pk, sk)*. For given* pk*, the message space* $\mathbf{M}_{pk}$ *and the randomness space* $\mathbf{R}_{pk}$ *are uniquely determined.*
2. *The encryption algorithm* Enc *takes as input a public key* pk *and a message* $m \in \mathbf{M}_{pk}$*, and outputs a ciphertext* $c \in \mathbf{C}_{pk}$ *where* $\mathbf{C}_{pk}$ *is a finite set of ciphertexts. We write this as* $c \leftarrow \mathsf{Enc}_{pk}(m)$*. We sometimes write* $\mathsf{Enc}_{pk}(m)$ *as* $\mathsf{Enc}_{pk}(m, r)$ *when the randomness* $r \in \mathbf{R}_{pk}$ *used by* Enc *needs to be emphasized..*
3. *The decryption algorithm* Dec *takes as input a private key* sk *and a ciphertext* c*, and outputs a message* m *or a special symbol* $\perp$ *which means failure.*

We say that a public-key cryptosystem $\mathcal{E}$ is *correct* if, for any key-pair $(pk, sk) \leftarrow \mathsf{KG}(\lambda)$ and any $m \in \mathbf{M}_{pk}$, it is the case that: $m \leftarrow \mathsf{Dec}_{sk}(\mathsf{Enc}_{pk}(m))$.

Definition 5 ([10]). *A public-key cryptosystem* $\mathcal{E} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ *with a security parameter* λ *is called to be* semantically secure (IND-CPA secure) *if after the standard CPA game being played with any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, the advantage* $\mathsf{Adv}^{\mathsf{cpa}}_{\mathcal{E},\mathcal{A}}(\lambda)$*, formally defined as*

$$\left| \Pr_{b,r} \left[\begin{array}{l} (pk, sk) \leftarrow \mathsf{KG}(\lambda), (\mathsf{state}, m_0, m_1) \leftarrow \mathcal{A}_1(pk), \\ c = \mathsf{Enc}_{pk}(m_b; r) : b \leftarrow \mathcal{A}_2(\mathsf{state}, m_0, m_1, c) \end{array} \right] - \frac{1}{2} \right|,$$

is negligible in λ *for all sufficiently large* λ*.*

In the experiment above, when we allow $\mathcal{A}_1$ to query the decryption oracle, if the advantage $\mathsf{Adv}^{\mathsf{cca2}}_{\mathcal{E},\mathcal{A}}(\lambda)$ is negligible, we say $\mathcal{E}$ is IND-CCA1 secure, in short, CCA1 secure.