Private Over-threshold Aggregation Protocols over Distributed Databases

Myungsun Kim, Aziz Mohaisen, Jung Hee Cheon, and Yongdae Kim

Abstract—In this paper, we revisit the private over-threshold data aggregation problem. We 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 an efficient cryptographic construction and its proxy-based variant. Both schemes are provably secure in the semi-honest model. Our key idea for the constructions and their malicious variants is to compose two encryption functions tightly coupled in a way that the two functions are commutative and one public-key encryption has an additive homomorphism. We call that double encryption. 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. Specifically, our protocol has linear complexity in computation and communication with respect to the number of users. Its round complexity is also linear in the number of users. Finally, we show that our basic protocol is efficiently transformed into a stronger protocol secure in the presence of malicious adversaries, and provide the resulting protocol's performance and security analysis.

Index Terms—Network traffic distribution, data aggregation, privacy preservation, malicious security

1 Introduction

The problem of *computing the over-threshold elements*, elements whose count is greater than a given value, in a *private* manner is of particular interest in many applications. 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 potential suspect sites. In such application, and without losing generality, each of such sensors has a set of suspects and would like to collaboratively compute the most frequent elements 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.

Formally, let there be n users denoted by $u_i, 1 \leq i \leq n$, and each of them has a private *multiset* $\mathtt{X}_i$ of cardinality k. For simplicity, assume that each of the multisets has the same cardinality.

Private κ^+ Aggregation Problem.

Let $\zeta, \kappa \in \mathbb{N}$, and for a set $\mathtt{X}$ and $\alpha \in \mathtt{X}$, let $F(\alpha)$ denote the number of frequencies (or occurrences) of α in $\mathtt{X}$. Then the problem at hand is defined as follows: given n multisets of cardinality k, find a set $Z = \{\alpha_1, \ldots, \alpha_\zeta\} \subset \mathtt{U} = \bigcup_{i=1}^{n} \mathtt{X}_i$ such that (*i*) for all elements $\alpha \in \mathtt{U}$, if α has the multiplicity greater than or equal to κ, then $\alpha \in Z$, *i.e.*,

$$Z = \left\{ \alpha \in \cup_{i=1}^{n} \mathtt{X}_i \middle| F(\alpha) \geq \kappa \right\},$$

(*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 [28]. As pointed out in [23], using a trusted third party (TTP) to solve the private κ^+ aggregation problem is impractical since it is hard to find such entity in many settings. Also, using secure multiparty computations (SMC) is impractical since they are computationally expensive. A final approach is to use existing private set-operation protocols such as [24], [37]—especially multiset union protocols. These protocols securely compute all elements appearing in the union of input multisets; in particular [24] allows to find all elements whose multiplicity is at least τ. Since these protocols give an output as a set, the output does not have the multiplicity information. While this feature can be beneficial from a privacy standpoint, it risks the functionality of applications relying on the multiplicity of elements, including κ^+ aggregation.

1.1 Our Approach—Informal Descriptions

The non-trivial part of the κ^+ protocols is to satisfy two privacy requirements, namely (ii) and (iii) simultaneously. From the literature, e-voting protocols have similar requirements. In these protocols, each ballot is mixed with a shuffle scheme to remove linkability between voters and ballots. Thus, when each element in a multiset is encrypted and shuffled using e-voting protocols, all encrypted elements can be decrypted while hiding linkability. However, all

- *M. Kim is with the University of Suwon, Hwaseong, South Korea. Email: msunkim@suwon.ac.kr (Corresponding author)*
- *A. Mohaisen is with The State University of New York at Buffalo. EmaiL: mohaisen@buffalo.edu*
- *J. H. Cheon is with Seoul National University, Seoul, South Korea. Email: jhcheon@snu.ac.kr*
- *Y. Kim is with Korea Advanced Institute of Science and Technology, Daejeon, South Korea. Email: yongdaek@ee.kaist.ac.kr*

non-κ^+ elements also are revealed, which violates condition (ii) in our requirements. To this end, we need to find a way to preserve data privacy even when all encrypted elements are decrypted.

In order to achieve this goal, we adopt an efficient function E that commutes with an underlying public-key encryption (E). Roughly speaking, we demand that: (*i*) commutativity: $E \circ \mathsf{E} = \mathsf{E} \circ E$, and (*ii*) double privacy: given $\bar{\bar{\alpha}} = E_s \circ \mathsf{E}_{pk}(\alpha)$, no algorithm can efficiently find α without the secret keys corresponding to s and pk respectively. We call this notion *double encryption*.

Existing shuffle schemes re-randomize input ciphertexts without changing the plaintexts of the input ciphertexts (e.g. [30], [18]). Rather, a double encryption scheme does not preserve the plaintexts of input ciphertexts during executing our protocols, but it still gives a way to recover the plaintexts. In conclusion, our main technique is to shuffle doubly encrypted elements.

1.2 Summary of Our Results

We present a formal definition of a private κ^+ protocol and its security in a system among n users over non-partitioned data. We refrain from using SMC and construct an efficient κ^+ protocol that is secure in the presence of a malicious adversary. In its efficiency, our construction is comparable to [2], which achieves its efficiency by adding to the system an extra entity, called a proxy. On the other hand, the security guarantees of our protocol are comparable to the work in [4], which is secure but expensive.

Our 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. We notice that n is much smaller than k in real-world applications, which further justifies the efficiency of our protocol. However, devising a protocol with a round complexity that does not depend on the number of users remains as an open problem.

Using zero-knowledge techniques, we transform the basic protocol for private over-threshold aggregation into a protocol secure against malicious adversaries. However, since there is a tradeoff between the security and complexity, the protocols comes at a higher computation and communication costs for its stronger security. Nonetheless, the linearity of total complexity in k is still preserved.

Organization. In Section 2, we discuss the related work. In Section 3, we outline the preliminaries, including double encryption, our formalism of κ^+, and cryptographic primitives used in our protocol. In Section 4, we introduce our construction; including the security and complexity analyses in §4.2 and §4.3. In Section 5, we provide a full description of a κ^+ protocol which is secure in the malicious model and analyze its security in the simulation paradigm. Concluding remarks are in Section 6.

2 Related Work

The related work is three directions: general-purpose, special-purpose, and proxy-based. If we assume the existence of a TTP, the cryptographic problem we are considering becomes trivial. Thus all the related work in the literature has been attempting to find a way to replace the TTP while providing security at the same level as when assuming the existence of such a TTP. The general-purpose solutions rely on fundamental theorem of cryptography addressed by Yao [41] and developed by Goldreich et al. [17]. The special-purpose group suggests carefully tuned methods to efficiently solve the problem compared with the general purpose solution. Lastly, the proxy-based schemes introduce some special entities and assign a set of tasks to them, and thus these schemes can achieve a further improved efficiency. However, the results of this line of work makes extra assumptions for the extra entities.

General-purpose approaches. In general, we may not assume the existence of a trusted party whom all participating users have to trust in a real world scenario. For that, the first approach we consider is a general solution based on SMC. The notion of SMC allows n users to create a virtual trusted party. Yao first introduced this notion and a method for performing SMC was developed by Goldriech, Micali and Wigderson in [17]. Their result is called the fundamental theorem of cryptography, stating that assuming trapdoor permutations exist, there exists an SMC protocol for every polynomial-size function. Unfortunately, due to the trade-off between generality and efficiency, we cannot achieve an efficient solution for our problem using this tool.

Special-purpose approaches. There have been a lot of approaches to improve the efficiency of SMC-based general solutions. One key direction is to devise a specific tool for a solution to this cryptographic problem. A closely related work is a protocol proposed by Burkhart and Dimitropoulos [4]. Their solution efficiently operates with respect to its computation complexity, but has two critical drawbacks: if input datasets are disjoint, the accuracy of their construction decreases sharply because the solution is *probabilistic* and their *round complexity is linear in the number of bits* in the data elements.

Another closely related work is to apply Kissner and Song's over-threshold set union protocol [24]. Their protocol allows us to find all elements appearing at least τ times in the union of input multisets. The core idea of their scheme is as follows: each user represents her input $\mathtt{X}_i$ as a polynomial f_i whose roots are in $\mathtt{X}_i$. The roots of the τ-th derivative of $P = \prod_{i=1}^{n} f_i$ give a set consisting of all elements that appear more than τ times in the union. The main shortcoming of this scheme is that each result set does not count the multiplicity of each element. This property may make it difficult to apply the protocol for the κ^+ problem. Specifically, consider a case where one needs to find all elements with the greatest multiplicity. We then need to execute the scheme repeatedly until obtaining the final result. While our protocol does not have such high overhead for repeated execution, it also allows changing τ during any step of the protocol execution; c.f. §4.3 for a detailed comparison.

Proxy-based approaches. Chow *et al.* [8] proposed an efficient scheme extending private set operations. Their scheme introduces two special entities: a randomizer and a computing server, all of which should be semi-honest. One issue with their solution is that it cannot support the aggregate operation over multisets. Another related work is Applebaum *et al.*'s [2] protocol. This solution is based on an efficiency strategy by adding a proxy and database (DB) for the constant round complexity. Both entities are also assumed to be semi-honest to prevent coalition between them. In particular, their scheme extensively relies on oblivious transfer (OT) [29], a computationally expensive public-key primitive which requires two modular exponentiations per invocation and runs for each bit of the user's data element. Furthermore, their protocol extensively uses two semantical secure encryption schemes at the same time: ElGamal encryption [10] together with Goldwasser-Micali encryption [15].

To sum up, Table 1 summarizes properties of the existing solutions compared with our protocol. The computational complexity is expressed as the number of multiplications over modulo p, and assuming that all elements are less than p.

TABLE 1
Summary and Comparison; models are Non-proxy based and Proxy-based.

	Proxy	Round Cpx	Comp. Cpx	Comm. Cpx
Ours	$\times$	$\mathcal{O}(n)$	$\mathcal{O}(n^2 k \log p)$	$\mathcal{O}(n^2 k \log p)$
[4]	$\times$	$\mathcal{O}(n(n + k \log k) \log p)$	$\mathcal{O}(n^2 k)$	$\mathcal{O}(n^2 k \log p)$
[2]	◯	$\mathcal{O}(1)$	$\mathcal{O}(nk \log^2 p)$	$\mathcal{O}(nk \log p)$

Data Aggregation. Data aggregation is an important technique in distributed systems, with many applications to enable efficient utilization of resources. Thus many researchers in wireless and smart grid networks have focused on the problem (*e.g.*, [21], [11], [39], [16], [34], [40], [12], [25], [26], [36]). However, in this paper we use the term "data aggregation" in a different sense. We consider data aggregation as a part of data and information mining process where data is searched, gathered, and presented in a report-based, summarized format to achieve specific business objectives. In particular, we are interested in keeping privacy during the whole process of data aggregation.

3 Background and Main Tools

Let us denote the *multiplicity* of an element α in a multiset $\mathtt{X}$ by $F(\alpha)$ and the collection of multiplicities for all elements in the multiset $\mathtt{X}$ by $F(\mathtt{X})$. For $n \in \mathbb{N}$, we denote the set $\{1, \ldots, n\}$ by $[n]$ and the set of all permutations on $[n]$ by Σ_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. If X is a set, then $x \overset{\$}{\leftarrow} X$ denotes sampling from the uniform distribution on X. We shall write

$$\Pr[x_1 \overset{\$}{\leftarrow} X_1, \ldots, x_n \overset{\$}{\leftarrow} X_n(x_1, \ldots, x_{n-1}) | \varphi(x_1, \ldots, x_n)]$$

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.

A function $\mu : \mathbb{N} \to \mathbb{R}$ is *negligible* if for every positive polynomial $p(\cdot)$ there exists an integer L such that $\mu(\lambda) < 1/p(\lambda)$ for all $\lambda > L$. Let $X = \{X(\alpha, \lambda)\}_{\alpha \in \{0,1\}^*, \lambda \in \mathbb{N}}$ and $Y = \{Y(\alpha, \lambda)\}_{\alpha \in \{0,1\}^*, \lambda \in \mathbb{N}}$ be distribution ensembles. We say that X and Y are *computationally indistinguishable*, which is denoted by $X \overset{c}{\approx} Y$, if for every polynomial-time algorithm $\mathcal{D}$ there exists a negligible function $\mu(\cdot)$ such that for every $\alpha \in \{0,1\}^*$ and $\lambda \in \mathbb{N}$

$$|\Pr[\mathcal{D}(X(\alpha, \lambda)) = 1] - \Pr[\mathcal{D}(Y(\alpha, \lambda)) = 1]| < \mu(\lambda)$$

3.1 Definitions

We begin by reviewing the definitions of public-key encryption (PKE) [15].

Definition 1 (PKE): $\mathcal{PKE} = (\mathsf{KG}, \mathsf{E}, \mathsf{D})$ is a *public-key encryption scheme* if KG, E, and D are polynomial-time algorithms defined as follows:

- KG, given a security parameter λ, outputs a pair of keys (pk, sk), where pk is a public key and sk is a secret key. We denote this by $(pk, sk) \leftarrow \mathsf{KG}(1^\lambda)$. The key pk also describes the plaintext and ciphertext message spaces, $\mathbf{M}_{pk}$ and $\mathbf{C}_{pk}$, respectively.
- E, given the public key pk and a plaintext m, outputs a ciphertext c encrypting m, which is denoted by $c \leftarrow \mathsf{E}_{pk}(m)$. To emphasize the randomness r used for encryption, we denote this encryption by $c \leftarrow \mathsf{E}_{pk}(m; r)$.
- D, given the public key pk, secret key sk and a ciphertext message c, outputs a plaintext m

such that there exists randomness r for which $c = \mathsf{E}_{pk}(m; r)$; otherwise outputs $\perp$. We denote this by $m \leftarrow \mathsf{D}_{sk}(c)$. We omit pk for simplicity.

For a public-key- encryption scheme $\mathcal{PKE}$ and an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, we consider the semantic security game as follows:

1. $(pk, sk) \leftarrow \mathsf{KG}(1^\lambda)$
2. $(\mathsf{state}, m_0, m_1) \leftarrow \mathcal{A}_1(pk)$, such that $|m_0| = |m_1|$
3. $c \leftarrow \mathsf{E}_{pk}(m_b)$, where $b \overset{\$}{\leftarrow} \{0, 1\}$
4. $b' \leftarrow \mathcal{A}_2(\mathsf{state}, c)$

It is said that $\mathcal{A}$ wins the game by a probability $\mathsf{Adv}^{\mathsf{cpa}}_{\mathcal{PKE},\mathcal{A}}(\lambda)$ if $b = b'$.

Definition 2 (Semantic Security): A public-key cryptosystem $\mathcal{PKE} = (\mathsf{KG}, \mathsf{E}, \mathsf{D})$ with a security parameter λ is called *semantically secure* if for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there is a negligible function μ such that $\mathsf{Adv}^{\mathsf{cpa}}_{\mathcal{PKE},\mathcal{A}}(\lambda) \leq \frac{1}{2} + \mu(\lambda)$.

Another important tool that we exploit in our construction is *double* encryption over a public-key encryption $\mathcal{PKE}$. Roughly speaking, a double encryption scheme is a pair of encryption schemes $\mathcal{PKE} = (\mathsf{KG}, \mathsf{E}, \mathsf{D})$ and $\mathcal{E} = (G, E, D)$ such that $\mathsf{E} \circ E = E \circ \mathsf{E}$.

Definition 3 (Double Encryption): Let $\mathcal{PKE}$ be a PKE scheme defined as above. For a PKE scheme $\mathcal{PKE}$, a pair $(\mathcal{PKE}, \mathcal{E})$ is called *double encryption* if there exists a triple of polynomial-time computable functions, $\mathcal{E} = (G, E, D)$, that satisfies the following properties:

- A probabilistic function $G(1^\lambda)$ takes as input a parameter λ, and outputs (s, t) s.t. $\forall s, t$ and $\forall\, m \in \mathbf{M}_{pk}$: $m = D_t \circ E_s(m)$, where E and D are deterministic.
- $\forall pk, s$, $\forall m \in \mathbf{M}_{pk}$: $\mathsf{E}_{pk} \circ E_s(m) = E_s \circ \mathsf{E}_{pk}(m)$ up to the randomness of $\mathsf{E}_{pk}(\cdot)$.
- $\forall\, c \in \mathsf{E}_{pk}(m)$, $E_s(m) = \mathsf{D}_{sk} \circ E_s(c)$.

We note that the definition of double encryption does not require semantic security, however the second property implicitly implies that the security of a PKE scheme gets preserved between commuting operations. In addition, the second property does not require the original plaintext to be preserved.

In addition to double encryption, our construction uses a verifiable shuffle scheme as a sub-protocol. Informally, a shuffle of a list of ciphertexts $c_1, \ldots, c_n$ is a newly updated set of a list of ciphertexts $c'_1, \ldots, c'_n$ with the same plaintexts in permuted order.

Definition 4 (Shuffle): Let $\mathcal{PKE}$ be a PKE scheme as in Definition 1. A *shuffle* is a pair of polynomial-time algorithms $(\mathcal{PKE}, \mathsf{Shuffle})$ where:

- $\mathsf{Shuffle}$, given a public key pk, a list of ciphertexts $(c_1, \ldots, c_n)$ and a random permutation π on $[n]$, outputs a list of ciphertexts $(c'_1, \ldots, c'_n)$.

Then we say that a shuffle $(\mathcal{PKE}, \mathsf{Shuffle})$ is verifiable if there exists a proof system (P, V) such that, given a public key pk, a list of input ciphertexts $(c_1, \ldots, c_n)$ and a list of output ciphertexts $(c'_1, \ldots, c'_n)$, it proves that $\mathsf{Shuffle}$ is correct.

Putting together a double encryption scheme and a shuffle scheme, we get an *over-threshold (κ^+) aggregation* protocol. Below, we sketch such a protocol:

- **Setup**: The setup algorithm takes as input a security parameter λ and outputs the public and secret parameters by invoking $(pk, sk) \leftarrow \mathsf{KG}(1^\lambda)$ and $(s, t) \leftarrow G(1^\lambda)$.
- **Double Encryption**: In this phase, the users takes as input the system parameters (pk, s) and a list of elements $(\alpha_1, \ldots, \alpha_n)$, and produces a list of doubly encrypted ciphertexts $(\bar{\bar{\alpha}}_1, \ldots, \bar{\bar{\alpha}}_n)$.
- **Shuffle**: In this shuffle phase, each user chooses a random permutation $\pi \in \Sigma_n$ and shuffles the doubly encrypted ciphertexts $(\bar{\bar{\alpha}}_1, \ldots, \bar{\bar{\alpha}}_n)$, and then outputs the mixed list such that $\forall i$: $\mathsf{D}_{sk}(\alpha'_i) = \mathsf{D}_{sk}(\bar{\bar{\alpha}}_{\pi(i)})$.
- **Aggregate**: In this aggregate phase, the users compute $\tilde{\alpha}_{\pi(i)} = \mathsf{D}_{sk}\left(\alpha'_i\right) = \mathsf{D}_{sk}\left(\bar{\bar{\alpha}}_{\pi(i)}\right) = E_s\left(\alpha_{\pi(i)}\right)$ for all $i \in [n]$, using the list of permuted, doubly encrypted ciphertexts. Then each user outputs all elements such that $F\left(\tilde{\alpha}_{\pi(i)}\right) \geq \kappa$.
- **Reveal**: In the reveal phase, each user outputs the most frequent κ elements by computing $D_t\left(\tilde{\alpha}_j\right)$ for all $j \in [\zeta]$.

We remark that a shuffle scheme works on a list of doubly encrypted ciphertexts. The reason for shuffling is that for an n-tuple ciphertext $\langle \bar{\bar{\alpha}}_i = E_s \circ \mathsf{E}_{pk}(m_i) \rangle_{i \in [n]}$, and because $\bar{\bar{\alpha}}_i$ can be written by $\mathsf{E}_{pk} \circ E(m_i)$ for each i, and thus the n-tuple list is just a list of ciphertexts under E, we need to apply shuffling after performing double encryption.

3.2 Security Definition

We prove that our protocol is secure against *malicious adversaries* that may arbitrarily deviate from the protocol specification. Following the standard technique, we analyze the security by comparing what an adversary can do in a real model to what it can do in an ideal model. In the ideal model, the computation is carried out by an incorruptible TTP to which the users send their inputs over a secure channel. The TTP then performs operations on the inputs and sends the output to each user. Roughly speaking, the protocol is *secure* if any adversary interacting in the real-model protocol can do no more harm than it could do in the ideal model. We only consider a *static* adversary who can corrupt a fixed number of users before executing the protocol.

Let f be an n-party functionality, $\mathcal{A}$ be a polynomial-time algorithm, and $\Upsilon \subsetneq \{u_i\}_{i \in [n]}$ be a set of corrupted users. Then the ideal execution of f on inputs $(\alpha_1, \ldots, \alpha_n)$, auxiliary input z to $\mathcal{A}$, and security parameter λ is defined as the outputs of the honest users and the adversary from this execution, which is denoted by $\mathtt{IDEAL}_{f, \mathcal{A}(z), \Upsilon}(\alpha_1, \ldots, \alpha_n; \lambda)$.

On the other hand, in the real model there is no TTP and the users exchange rounds of communication with each other. Let $f, \mathcal{A}$, and Υ be as described earlier, and $\wp$ be an n-party protocol for computing f. The real-model execution of $\wp$ on inputs $(\alpha_1, \ldots, \alpha_n), z$, and λ is denoted by $\mathtt{REAL}_{\wp,\mathcal{A}(z),\Upsilon}(\alpha_1, \ldots, \alpha_n; \lambda)$ and defined as the outputs of the honest users and the adversary $\mathcal{A}$ from executing $\wp$.

Security following the Simulation Paradigm. Informally speaking, the standard definition asserts that a secure protocol (in the real model) emulates the ideal-model execution (where there exists a TTP). A formal definition is given below.

Definition 5 (Secure Protocol): Let f and $\wp$ be as above. Then, we say that protocol $\wp$ *securely computes* f in the malicious model if for any PPT adversary $\mathcal{A}$ for the real model, there exists a PPT adversary $\mathcal{S}$ for the ideal model, such that for all $\Upsilon \subsetneq \{u_i\}_{i\in[n]}$,

$$\left\{\mathtt{IDEAL}_{f,\mathcal{A}(z),\Upsilon}(\alpha_1, \ldots, \alpha_n; \lambda)\right\} \overset{c}{\approx} \left\{\mathtt{REAL}_{\wp,\mathcal{A}(z),\Upsilon}(\alpha_1, \ldots, \alpha_n; \lambda)\right\}.$$

In practice, our construction will use secure sub-protocols. A standard method of hiding the details of a sub-protocol computing an ideal functionality g is to work in a *hybrid model* in which users do not only communicate directly with each other but also have access to a TTP $\mathcal{T}_{\mathrm{g}}$ for g. Thus, when executing a protocol $\wp$ that uses a sub-protocol for securely computing g, the users run $\wp$ and access $\mathcal{T}_{\mathrm{g}}$ instead of invoking the sub-protocol. Then, the g-*hybrid execution* of $\wp$ on inputs $(\alpha_1, \ldots, \alpha_n), z$, and λ is defined as the outputs of the honest users and $\mathcal{A}$ from the hybrid execution of $\wp$ with $\mathcal{T}_{\mathrm{g}}$.

Here we briefly presented the standard definition of security as used in our protocols. Readers may refer to [14, Chap. 7] for more details.

3.3 Cryptographic Assumptions

Let $\mathbb{G}$ be a finite cyclic group of large prime order q, and let $g \in \mathbb{G}$ be a generator. Given $h \in \mathbb{G}$, the discrete logarithm (DL) problem requires us to compute $x \in \mathbb{Z}_q$ such that $g^x = h$. A formal definition is given as follows:

Definition 6 (DL Assumption): We say that the *discrete logarithm problem is intractable* with respect to $\mathbb{G}$ if for every PPT algorithm $\mathcal{A}$ there exists a negligible function μ in λ such that $\Pr_{g,x}\left[\mathcal{A}\left(\mathbb{G}, q, g, g^x\right) = x\right] \leq \mu(\lambda)$.

A stronger variant of the above assumption is called the decisional Diffie-Hellman (DDH) assumption. The DDH problem says that given $\mathbb{G}$ and three elements $a, b, c \in \mathbb{G}$, we are asked to decide whether there exist $x, y \in \mathbb{Z}_q$ such that $a = g^x, b = g^y$, and $c = g^{xy}$. Formally, the DDH assumption is stated in the following.

Definition 7 (DDH Assumption): We say that the *decisional Diffie-Hellman problem is intractable* with respect to $\mathbb{G}$ if for every PPT algorithm $\mathcal{A}$ there exists a negligible function μ in λ such that:

$$\left|\Pr\left[\mathcal{A}\left(\mathbb{G}, q, g, g^x, g^y, g^z\right) = 1\right] - \Pr\left[\mathcal{A}\left(\mathbb{G}, q, g, g^x, g^y, g^{xy}\right) = 1\right]\right| \leq \mu(\lambda),$$

where the probabilities are taken over the choices of g and $x, y, z \in \mathbb{Z}_q$.

3.4 The ElGamal PKE and Its Distributed Version

We extensively use the ElGamal encryption scheme [10]. Formally, ElGamal PKE is semantically secure assuming the hardness of the DDH problem. We describe the distributed version here. For settings consisting of multiple users, we need to consider the threshold scheme.

Let $\mathbb{G}$ be a cyclic group where the DDH assumption holds. Let x_i be a secret key and y_i be a corresponding public key of a user u_i. The public key $y = \prod_{i=1}^n y_i = g^{\sum_{i=1}^n x_i} = g^x$ is known to all users, and encryption is as in the standard ElGamal PKE. To ensure correct behavior, each user must prove knowledge of her corresponding x_i by running the zero-knowledge proof $\wp_{\mathsf{DL}}$ on (g, y_i), as specified in §3.6. Further, we denote the key generation protocol by $\wp_{\mathsf{KG}}$ which is related with the ideal functionality $\mathcal{F}_{\mathsf{KG}}$.

For decryption, a user u_j sends a request for decryption containing $c = (u, v)$ to all other users. On receiving the request, all other users compute a decryption share $d_i = u^{x_i}$ and send it to the user u_j. Upon receiving all decryption shares, the user computes the message as $m = \frac{v}{\prod_{i=1}^n d_i} = \frac{v}{u^{\sum_{i=1}^n x_i}} = v/u^x$. We denote the associated ideal functionality by $\mathcal{F}_{\mathsf{D}}$. We note that since if one knows $r = \log_g u$ he can reconstruct $m = v/h^r$, a user who encrypted m can prove knowledge of plaintext m by proving knowledge of r.

3.4.1 Instantiating a Double Encryption Scheme

We give an instance of a double encryption scheme using the standard ElGamal PKE $\mathcal{PKE} = (\mathsf{KG}, \mathsf{E}, \mathsf{D})$. Let us specify $\mathcal{E} = (G, E, D)$ as follows:

- A probabilistic function $G(1^\lambda)$ outputs $(s, t) \in (\mathbb{Z}_q)^2$ such that $st = 1 \pmod q$.
- A function $E : \mathbb{G} \to \mathbb{G}$ takes as input $\alpha \in \mathbb{G}$ and s, and outputs $\beta = \alpha^s \mod p$.
- Given $\beta \in \mathbb{G}$ and t, a function $D : \mathbb{G} \to \mathbb{G}$ outputs $\beta^t \mod p$.

The following lemma states that $(\mathcal{PKE}, \mathcal{E})$ is a double encryption scheme.

Lemma 1: Let $\mathcal{PKE}$ and $\mathcal{E}$ be as above. Then, $(\mathcal{PKE}, \mathcal{E})$ is a double encryption scheme.

Proof: We can easily verify that $(\mathcal{PKE}, \mathcal{E})$ satisfies the conditions of double encryption. First we see that for all values $m \in \mathbb{G}_q$, $m = (m^s)^t \pmod p$.

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

3.5 Pedersen's Commitment Scheme

We specify the type of commitment we are using. First, we assume a PPT key generation algorithm to output a public commitment key ck which specifies a message space $\mathbf{M}_{ck}$, a randomizer space $\mathbf{R}_{ck}$ and a commitment space $\mathbf{C}_{ck}$. We assume an efficient commitment function $\mathsf{com}_{ck} : \mathbf{M}_{ck} \times \mathbf{R}_{ck} \to \mathbf{C}_{ck}$. We write the operations as $\hat{\mathsf{m}} = \mathsf{com}(m; r)$ for $r \overset{\$}{\leftarrow} \mathbf{R}_{ck}$. We sometimes omit the randomness r if there is no ambiguity.

In this work, we use a perfectly hiding commitment scheme. Informally speaking, the commitment scheme is *hiding* if a commitment $\hat{\mathsf{m}}$ does not reveal which message is inside it. In particular, our construction requires the commitment scheme to be perfectly hiding. As a scheme with these properties, we use the Pedersen's commitment scheme using the same underlying group $\mathbb{G}$ used for the ElGamal PKE [35].

3.6 Zero-knowledge Proofs

To prevent malicious activities, all users should demonstrate that they follow the protocol honestly. To achieve this, our construction utilizes zero-knowledge proofs of knowledge (ZKP). All our proofs are Σ-protocols that prove knowledge of a witness that a given statement is true. We have an efficient transformation of any Σ-protocol into a ZKP, and examples can be found in [20]. (This transformation demands additionally five exponentiations.) All ZKPs used in our construction have been well studied elsewhere [14].

Let $\mathbb{G}, p, q, g$ be defined as above and pk (resp., ck) be the public key of underlying PKE (resp., commitment scheme). Details of ZKPs used in our scheme are as follows:

- $\wp_{\mathsf{DL}}$, for demonstrating that given $(\mathbb{G}, p, q, g, h)$, a prover knows the discrete logarithm to the base g of h [38].

$$\mathcal{R}_{\mathsf{DL}} = \{\langle \mathbb{G}, p, q, g, y\rangle | \exists x \in \mathbb{Z}_q \text{ s.t. } y = g^x\}.$$

- $\wp_{\mathsf{PT}}$, for proving for a given ElGamal ciphertext $c = \mathsf{E}_{pk}(m; r)$ that she knows the corresponding plaintext m (see §3.4).

$$\mathcal{R}_{\mathsf{PT}} = \{\langle pk, c\rangle | \exists m \in \mathbb{G}, r \in \mathbb{Z}_q \text{ s.t. } c = \mathsf{E}_{pk}(m; r)\}.$$

- $\wp_{\mathsf{com}}$, for demonstrating that a prover committing to a message m, proves that he knows the message m [32].

$$\mathcal{R}_{\mathsf{com}} = \{\langle ck, \hat{\mathsf{m}}\rangle | \exists m \in \mathbb{G}, \exists r \in \mathbb{Z}_q \text{ such that } \hat{\mathsf{m}} = \mathsf{com}_{ck}(m; r)\}.$$

- $\wp_{\mathsf{INV}}$, for demonstrating that a prover who committed to secrets (s, t), proves that $st = 1$ without revealing s, t [1].

$$\mathcal{R}_{\mathsf{INV}} = \{\langle ck, \hat{\mathsf{s}}, \hat{\mathsf{t}}\rangle | \exists s, t \in \mathbb{Z}_q \text{ such that } \hat{\mathsf{s}} = \mathsf{com}_{ck}(s) \wedge \hat{\mathsf{t}} = \mathsf{com}_{ck}(t) \wedge s \cdot t = 1\}.$$

- $\wp_{\mathsf{DE}}$, for proving a correct double encryption, i.e., that given (pk, ck, c), a prover knows the discrete logarithm s of c_2 to the base c_1 where $\hat{\mathsf{s}} = \mathsf{com}_{ck}(s)$ and $c_1, c_2 \in \mathbf{C}_{pk}$ [6].

$$\mathcal{R}_{\mathsf{DE}} = \{\langle pk, ck, \hat{\mathsf{s}}, c_1, c_2\rangle | \exists s \in \mathbb{Z}_q \text{ such that } c_2 = c_1^s \wedge \hat{\mathsf{s}} = \mathsf{com}_{ck}(s)\}.$$

- $\wp_{\pi}$, for demonstrating that a set of ElGamal ciphertexts $L' = \{c'_i\}_{i\in[n]}$ is a random permutation and rerandomization of another set $L = \{c_i\}_{i\in[n]}$. We have a number of potential proofs in the literature; the most recent solution was given by Bayer and Groth [3] with sub-linear communication cost.

$$\mathcal{R}_{\pi} = \{\langle pk, L, L'\rangle | \exists \pi \in \Sigma_n, \exists r_i \in \mathbb{Z}_q \text{ such that } \forall i \in [n], c'_{\pi(i)} = c_i \cdot \mathsf{E}_{pk}(1; r_i)\}.$$

4 A Secure κ^+ Protocol in the Semi-Honest Model

In this section, we describe our construction for computing κ^+ elements privately. For a set of n users, denoted by $u_1, \ldots, u_n$, let us denote u_i's private set by $\mathtt{X}_i = \{\alpha_{i,1}, \ldots, \alpha_{i,k}\}$. Then the users wish to jointly compute $Z = \{\alpha \in \bigcup_{i=1}^n \mathtt{X}_i | F(\alpha) \geq \kappa\}$.

4.1 Details of the Protocol

Let λ be a security parameter. As above, let p be a λ-bit prime such that for some prime q, $p = 2q+1$, $\mathbb{G} = \langle g\rangle$ is a finite cyclic subgroup of $\mathbb{Z}_p^\times$ whose order is q. We use the double encryption scheme $(\mathcal{PKE}, \mathcal{E})$ given in §3.4.1 over $\mathsf{params} = (\mathbb{G}, p, q, g)$. For simplicity, we assume all elements are in the domain $\mathbb{G}$ of the ElGamal PKE.

With such notation, we proceed to describe our construction.

- **Setup.** For $i \in [n]$, each user u_i
  1) selects a value $x_i \overset{\$}{\leftarrow} \mathbb{Z}_q$, computes $y_i = g^{x_i}$, and sets $sk = (\mathsf{params}, x_i)$ and $pk = \left(\mathsf{params}, y = \prod_{i=1}^n y_i = g^{\sum_{i\in[n]} x_i}\right)$, for a threshold ElGamal encryption $\mathcal{PKE}$ with a public/private key pair (pk, sk).
  2) computes a pair of secret keys (s_i, t_i) such that $s_i t_i = 1 \pmod q$.
- **DEncrypt.**
  1) Every user u_i first encrypts his multiset $\mathtt{X}_i$ as $\bar{\mathtt{X}}_i = \{\bar{\alpha}_{i,j}\}_{j\in[k]}$, where $\bar{\alpha}_{i,j} = \mathsf{E}_{pk}(\alpha_{i,j}; \gamma_j)$ for some $\gamma_j \overset{\$}{\leftarrow} \mathbb{Z}_q$. Then he sends the set $\bar{\mathtt{X}}_i$ to u_1.

2) u_1 computes $\langle E_{s_1}(\bar{\alpha}_{i,j})\rangle_{i\in[n],j\in[k]}$. We denote this by Y_0.

- **Shuffle & DEncrypt.** For $i \in [n]$, u_i receives a vector Y_{i-1}, where Y_{i-1} is a vector of ciphertexts generated by user u_{i-1}. Using Y_{i-1}, each user computes his shuffled and doubly encrypted version Y_i as follows:
  1) $u_{i\neq 1}$ computes
  $$\begin{aligned} E_{s_i}(Y_{i-1}) &= \langle \bar{\bar{\alpha}}_1, \ldots, \bar{\bar{\alpha}}_{nk}\rangle \\ &= \left\langle E_{s_i}\left(\cdots E_{s_1}\left(\bar{\alpha}_{\pi_{i-1}(\ell)}\right)\cdots\right)\right\rangle_{\ell\in[nk]}. \end{aligned}$$
  2) u_i chooses a random permutation $\pi_i \in \Sigma_{nk}$ along with randomizers $\{\gamma_\ell\}$ where $\gamma_\ell \in \mathbb{Z}_q$ for all $\ell \in [nk]$, and applies it to the vector of $\bar{\bar{\alpha}}_\ell$; we 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.** For $Y_n = (\alpha'_1, \ldots, \alpha'_{nk})$, all users
  1) Jointly decrypt each element of Y_n, obtaining $\tilde{\mathtt{U}} = \{\tilde{\alpha}_1, \ldots, \tilde{\alpha}_{nk}\}$.
  2) Each user computes
  $$\tilde{Z} = \left\{\langle\tilde{\alpha}, F(\tilde{\alpha})\rangle \,|\tilde{\alpha} \in \tilde{\mathtt{U}} \wedge F(\tilde{\alpha}) \geq \kappa\right\}.$$
- **Reveal.** Let $\tilde{Z}_0 := \tilde{Z}$ and $|\tilde{Z}| = \zeta$.
  1) For each $i \in [n-1]$, u_i computes $\tilde{Z}_i = \{\langle D_{t_i}(\tilde{\alpha}), F(\tilde{\alpha})\rangle\}_{\langle\tilde{\alpha},F(\tilde{\alpha})\rangle\in\tilde{Z}_{i-1}}$ and sends it to u_{i+1}.
  2) Lastly, u_n computes
  $$\begin{aligned} \tilde{Z}_n &= \{\langle D_{t_n}(\tilde{\alpha}), F(\tilde{\alpha})\rangle\}_{\langle\tilde{\alpha},F(\tilde{\alpha})\rangle\in\tilde{Z}_{n-1}} \\ &= \{\langle\alpha_j, F(\alpha_j)\rangle\}_{j\in[\zeta]} \end{aligned}$$
  and broadcasts $Z = \tilde{Z}_n$ to all other users.

Finally, each user obtains a κ^+ set Z, including all elements $\alpha \in \mathtt{U}$ such that $F(\alpha) \geq \kappa$.

4.2 Security Analysis

We prove that our protocol is secure in the semi-honest model and analyze its efficiency. The following theorem states the correctness of our protocol.

Theorem 1 (Correctness): In the private κ^+ protocol in §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.

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

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

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, $\forall \ell, \ell^* \in [nk]$, $\alpha_\ell = \alpha_{\ell^*}$ if and only if $E_s(\alpha_\ell) = E_s(\alpha_{\ell^*})$. □

Corollary 1: In the private κ^+ protocol in §4.1, every honest user learns all κ^+ elements in the union of private multisets.

Now we show that our protocol satisfies the privacy requirements in the semi-honest model. To this end, we define the ideal functionality for κ^+ aggregation as follows.

Definition 8 (Ideal Functionality $\mathcal{F}_{\mathsf{topk}}$): Consider a set of n users $\{u_i\}_{i\in[n]}$, a trusted party $\mathcal{T}$, and an ideal adversary $\mathcal{S}$ controlling a set of corrupted users $\{u_{i_1}, \ldots, u_{i_\nu}\}$ for some $0 \leq \nu < n$. Again let u_i's input $\mathtt{X}_i = \{\alpha_{i,j}\}_{j\in[k]}$ and $\mathtt{U} = \bigcup_{i\in[n]}\mathtt{X}_i$.

1) Each user u_i sends $\mathtt{X}_i$ to $\mathcal{T}$.
2) $\mathcal{T}$ computes the following functionality, and returns the outputs Z_i and M_i to each $u_{i\in[n]}$, where $Z_i = \{\langle\alpha, F(\alpha)\rangle\,|\alpha \in \mathtt{U} \wedge F(\alpha) \geq \kappa\}$ and $M_i = \{F(\alpha)|\alpha \in \mathtt{U}\}$.

Next, we show that the protocol indeed privately computes $\mathcal{F}_{\mathsf{topk}}$ for any coalition of semi-honest users. Intuitively, a user does not get any computational knowledge from the execution because, for any communication, every user sees only a collection of (doubly) encrypted and permuted encryptions. Before proving the protocol's privacy, we prove some supporting lemmas.

Lemma 2: Let $E, \alpha \in \mathbb{G}$ and $s_{i\in[n]}$ be defined as above. Assuming $F(\alpha_i) = F(\alpha_j)$ for all $i, j \in [n]$, for any $r \xleftarrow{\$} \mathbb{G}$, we have

$$\begin{pmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \alpha'_1 & \alpha'_2 & \cdots & \alpha'_n \end{pmatrix} \overset{c}{\approx} \begin{pmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \alpha'_1 & \alpha'_2 & \cdots & E_{s_n}(r) \end{pmatrix}$$

where $\alpha_i = E_{s_i}(\alpha), \alpha'_i = E_{s_i}\left(\alpha'_{i-1}\right)$ with $\alpha'_1 = \alpha_1$ for each $i \in [n]$.

Proof: Let us first denote the distribution of the left-side $2 \times n$-tuple by $L_n := \left(\begin{smallmatrix} \alpha_1 & \cdots & \alpha_n \\ \alpha'_1 & \cdots & \alpha'_n \end{smallmatrix}\right)$ and the distribution of the right-side $2 \times n$-tuple by $R_n := \left(\begin{smallmatrix} \alpha_1 & \cdots & \alpha_n \\ \alpha'_1 & \cdots & E_{s_n}(r) \end{smallmatrix}\right)$. If L_n and R_n are distinguishable by a PPT algorithm $\mathcal{A}$, $(\mathbb{G}, q, g, g^x, \alpha_n, \alpha'_n)$ and $(\mathbb{G}, q, g, g^x, \alpha_n, E_{s_n}(r))$ are distinguishable from the following algorithm that takes as input $(\mathbb{G}, q, g, g^\alpha, a, b)$:

1) Let $x \xleftarrow{\$} \mathbb{Z}_q$. For $i \in [n-1]$, set $\alpha = g^x, \alpha_i = E_{s_i}(\alpha)$ and $z_i = E_{s_i}(z_{i-1})$ with $z_1 = \alpha_1$.
2) Let $\alpha_n = a$ and $z_n = b$.
3) Submit the $2 \times n$-tuple
$$\begin{pmatrix} \alpha_1 & \cdots & \alpha_{n-1} & \alpha_n \\ z_1 & \cdots & z_{n-1} & z_n \end{pmatrix}$$
to algorithm $\mathcal{A}$ and output what algorithm $\mathcal{A}$ produces.

For $i \in [n-1]$, we have $z_i = g^{x\prod_{j=1}^{i} s_j}$ and all α_i's are indistinguishable from uniformly random since the DDH assumption holds in $\mathbb{G}$. Hence the distribution of the tuple given to $\mathcal{A}$ is indistinguishable from L_n when $(\mathbb{G}, q, g, g^x, a, b)$ is distributed as $(\mathbb{G}, q, g, g^x, \alpha_n, \alpha'_n)$, and from R_n when $(\mathbb{G}, q, g, g^x, a, b)$

is distributed as $(\mathbb{G}, q, g, g^x, \alpha_n, E_{s_n}(r))$. Therefore the assumption that L_n and R_n are distinguishable leads to the contradiction the DDH assumption in $\mathbb{G}$. This completes the proof. □

Lemma 3: Let E, E, α and $s_{i\in[n]}$ be defined as above. Then for any $r \xleftarrow{\$} \mathbb{G}$ and $\bar{\alpha} \in \mathsf{E}_{pk}(\alpha)$, we have

$$\begin{pmatrix} \bar{\bar{\alpha}}_1 & \bar{\bar{\alpha}}_2 & \cdots & \bar{\bar{\alpha}}_n \\ \alpha'_1 & \alpha'_2 & \cdots & \alpha'_n \end{pmatrix} \overset{c}{\approx} \begin{pmatrix} \bar{\bar{\alpha}}_1 & \bar{\bar{\alpha}}_2 & \cdots & \bar{\bar{\alpha}}_n \\ \alpha'_1 & \alpha'_2 & \cdots & E_{s_n}(\bar{r}) \end{pmatrix}$$

where $\bar{r} = \mathsf{E}_{pk}(r), \bar{\bar{\alpha}}_i = E_{s_i}(\bar{\alpha}), \alpha'_i = E_{s_i}\left(\alpha'_{i-1}\right)$ with $\alpha'_1 = \bar{\bar{\alpha}}_1$ for each $i \in [n]$.

Proof: It is easy to prove the indistinguishability between two $2\times n$-tuples by using the same techniques as in Lemma 2. As above, denote the the distribution of the left-side $2 \times n$-tuple by $\bar{L}_n$ and the distribution of the right-side $2 \times n$-tuple by $\bar{R}_n$. If $\bar{L}_n$ and $\bar{R}_n$ are distinguishable from some PPT algorithm $\bar{\mathcal{A}}$, then we can construct the following distinguisher that can differentiate $\mathsf{E}_{pk}(\alpha^s)$ and $\mathsf{E}_{pk}(r^{s_n})$, which takes pk as input:

1) As above, let $x \xleftarrow{\$} \mathbb{Z}_q$. For $\in [n-1]$, $\alpha = g^x, \bar{\alpha} = \mathsf{E}_{pk}(\alpha), \bar{\bar{\alpha}}_i = E_{s_i}(\bar{\alpha})$, and $z_i = E_{s_i}(z_{i-1})$ with $z_1 = \bar{\bar{\alpha}}_1$.
2) Send $m_0 = \alpha^s$ and $m_1 = r^{s_n}$, where $s = \prod_{i=1}^n s_i$, to a challenger.
3) Get an encryption $c_b = \mathsf{E}_{pk}(m_b)$ from the challenger.
4) Let $\bar{\bar{\alpha}}_n = E_{s_n}(\alpha)$. Then choose a random bit $b' \in \{0, 1\}$, submit the tuple
$$\begin{pmatrix} \bar{\bar{\alpha}}_1 & \cdots & \bar{\bar{\alpha}}_n \\ z_1 & \cdots & c_{b'} \end{pmatrix}$$
to algorithm $\bar{\mathcal{A}}$, and output whatever it produces.

For $i \in [n-1]$, we have

$$\begin{aligned} z_i &= E_{s_i}(z_{i-1}) = E_{s_i} \circ E_{s_{i-1}} \circ \cdots \circ E_{s_1}(\bar{\alpha}) \\ &= E_{s_i} \circ E_{s_{i-1}} \circ \cdots \circ E_{s_1} \circ \mathsf{E}_{pk}(\alpha) \\ &= \mathsf{E}_{pk} \circ E_{s_i} \circ E_{s_{i-1}} \circ \cdots \circ E_{s_1}(\alpha) = \mathsf{E}_{pk}\left(\alpha^{\prod_{j=1}^i s_j}\right), \end{aligned}$$

and all $\bar{\bar{\alpha}}_i$'s are indistinguishable from a uniformly random distribution, based on the semantic security of the public-key encryption. However, this contradicts the assumption that the public-key encryption is semantically secure. □

We are now ready to prove the main theorem. A formal argument is given next.

Theorem 2: Assume that the threshold ElGamal encryption E is semantically secure. The protocol in §4.1 privately computes the n-party functionality $\mathcal{F}_{\mathsf{topk}}$ against any coalition of less than n semi-honest users.

Proof: We construct a simulator, denoted by $\mathcal{S}$, for the view of the semi-honest users. Interestingly, it is easy to construct the simulator despite the complexity of the description of the protocol. Let $f(X) = (f(x_1), \ldots, f(x_k))$ for a set $X = \{x_1, \ldots, x_k\}$ where $f \in \{\mathsf{E}, E, D\}$. Given an index set of semi-honest users, $J = \{i_1, \ldots, i_\nu\}$ (with $\nu < n$), the public key pk, and a sequence of their inputs $\mathtt{X}_{i_1}, \ldots, \mathtt{X}_{i_\nu}$ and the outputs of the protocol Z and M, the simulator proceeds as follows.

1) Let $I := [n]\backslash J$. For each $i \in I$, choose $\pi_i \xleftarrow{\$} \Sigma_{nk}$ and a pair of secret keys, $(s_i, t_i) \xleftarrow{\$} (\mathbb{Z}_q)^2$ such that $s_i t_i = 1$.
2) For all honest users, choose their new, random multisets $\mathcal{X}_{i\in I}$, satisfying that $\{\langle\alpha, F(\alpha)\rangle \,|\alpha \in (\mathtt{X}_{j\in J} \cup \mathcal{X}_{i\in I}) \wedge F(\alpha) \geq \kappa\}$ is equal to Z and the distribution of the union $\mathtt{X}_{j\in J} \cup \mathcal{X}_{i\in I}$ is identically distributed to M.
3) For each $j \in J$ and $i \in I$, compute $\bar{\mathtt{X}}_j = \mathsf{E}_{pk}(\mathtt{X}_j)$ and $\bar{\mathcal{X}}_i = \mathsf{E}_{pk}(\mathcal{X}_i)$. Output $\bar{\mathtt{X}}_j$ for $j \neq 1$ and $j \in J$.
4) If $j = 1$ and $j \in J$, select $s_1 \xleftarrow{\$} \mathbb{Z}_q$ and compute $\bar{\bar{\mathtt{X}}}_j = E_{s_1}(\bar{\mathtt{X}}_j)$ for each $j \in J$ and $\bar{\bar{\mathcal{X}}}_i = E_{s_1}(\bar{\mathcal{X}}_i)$ for each $i \in I$. Let $Y_0 = (\bar{\bar{\mathtt{X}}}_{j\in J}, \bar{\bar{\mathcal{X}}}_{i\in I})$. Then apply a random permutation π_1 and randomizers $\gamma_{1,j\in J}$ to Y_0 and output the result Y_1. If $i = 1$ and $i \in I$, compute Y_0 and Y_1 using s_1 and π_1 generated in Step 1 and $\gamma_{1,i\in I}$, in sequence.
5) For each pair $(i, j) \in I \times J$ such that $j = i+1$ and $0 < i < n$, after selecting $s_j \xleftarrow{\$} \mathbb{Z}_q$ and $\pi_j \xleftarrow{\$} \Sigma_{nk}$, computing $E_{s_j}(Y_i)$, and applying π_j to it, output the result vector Y_j. For each $(i', i) \in [n] \times I$ such that $i = i' + 1$ and $0 < i' < n$, compute Y_i using the corresponding s_i and π_i. (For simplicity, all randomizers were omitted).
6) For each $i \in I$ and $j \in J$, compute $\tilde{\mathtt{U}} = E_s(\mathtt{X}_j) \cup E_s(\mathcal{X}_i)$ where $s = \prod_{i\in I} s_i \cdot \prod_{j\in J} s_j$ and output $\tilde{\mathtt{U}}$ and $\tilde{Z} = \{\langle\tilde{\alpha}, F(\tilde{\alpha})\rangle | \tilde{\alpha} \in \tilde{\mathtt{U}} \wedge F(\tilde{\alpha}) \geq \kappa\}$.
7) Let $\tilde{Z}_0 := \tilde{Z}$. For each $(i, j) \in I \times J$ such that $j = i + 1$ and $0 < i < n$, compute $\tilde{Z}_j = D_{t_j}(\tilde{Z}_i)$ where $t_j = s_j^{-1} \mod q$ and output $\tilde{Z}_j$. For each $(i', i) \in [n] \times I$ such that $i = i' + 1$ and $0 < i' < n$, compute $\tilde{Z}_i$ with t_i of Step 1.

We claim that the output of the protocol executions can be computed by the simulator as a polynomial function of Z and $\tilde{\mathtt{U}}$. Furthermore, we claim that for every such J, every $(\mathtt{X}_{i_1}, \ldots, \mathtt{X}_{i_\nu})$, and for every possible outcome Z from the ideal functionality $\mathcal{F}_{\mathsf{topk}}$, it holds that the conditional distribution of the simulator's outputs is computationally indistinguishable from the conditional distribution of the users' view in J.

It is also trivial to show that by the simulator's choices in Step 2 and Lemma 2, the simulator outputs the same Z and $\tilde{Z}$, as the protocol and $\tilde{\mathtt{U}}$ in Step 6 of the simulator are computationally indistinguishable from $\tilde{\mathtt{U}}$ in the real execution.

Turning to the conditional distributions, we show that the view of the simulator is computationally indistinguishable from the view of users in J. Without loss of generality, we can assume that $u_1 \in \Upsilon$. The view of users in $J = \{i_1, \ldots, i_\nu\}$ is

$$\Big\{\mathsf{R}_1(\bar{\mathtt{X}}_{i\in[n]}), \mathsf{R}_2(Y_{i_1}), \ldots, \mathsf{R}_{\nu+1}(Y_{i_\nu}), \mathsf{R}_{\nu+2}(\tilde{Z}_{i_1}), \quad \ldots, \mathsf{R}_\eta(\tilde{Z}_{i_\nu})\Big\}$$

where R denotes the real transcript of the protocol and $\eta = 2\nu + 1$. The simulated transcript produced by the simulator $\mathcal{S}$ is

$$\Big\{\mathsf{S}_1(\bar{\mathtt{X}}_{j\in J} \cup \bar{\mathcal{X}}_{i\in I}), \mathsf{S}_2(Y_{i_1}), \ldots, \mathsf{S}_{\nu+1}(Y_{i_\nu}), \mathsf{S}_{\nu+2}(\tilde{Z}_{i_1}), \ldots, \mathsf{S}_\eta(\tilde{Z}_{i_\nu})\Big\},$$

where S denotes the simulated transcript. We then prove a hybrid argument over the simulated views for the protocol. First, let us define the hybrid distribution $H^{(\ell)}$ in which the first ℓ outputs are simulated and the last $\eta - \ell$ are real. Formally, let $H^{(\ell)}$ denote the distribution:

$$\Big\{\mathsf{S}_1(\bar{\mathtt{X}}_{j\in J} \cup \bar{\mathcal{X}}_{i\in I}), \mathsf{S}_2(Y_{i_1}), \ldots, \mathsf{S}_\ell(Y_{i_{\ell-1}}), \mathsf{R}_{\ell+1}(Y_{i_\ell}), \ldots, \mathsf{R}_{\nu+1}(Y_{i_\nu}), \mathsf{R}_{\nu+2}(\tilde{Z}_{i_1}), \ldots, \mathsf{R}_\eta(\tilde{Z}_{i_\nu})\Big\}.$$

We now prove that $H^{(0)} \overset{c}{\approx} H^{(\eta)}$ by showing that for all $\ell \in [\eta]$, $H^{(\ell)} \overset{c}{\approx} H^{(\ell+1)}$. For the sake of contradiction, assume the opposite, and choose ℓ so that $H^{(\ell)} \overset{c}{\not\approx} H^{(\ell+1)}$. These two distributions differ in only one term, so there must be a polynomial-time distinguisher for

$$\begin{cases} \mathsf{S}_1(\bar{\mathtt{X}}_{j\in J} \cup \bar{\mathcal{X}}_{i\in I}) \text{ and } \mathsf{R}_1(\bar{\mathtt{X}}_{i\in[n]}) & \text{if } \ell = 1, \\ \mathsf{S}_\ell(Y_{i_{\ell+1}}) \text{ and } \mathsf{R}_\ell(Y_{i_{\ell+1}}) & \text{if } 2 \le \ell \le \nu + 1, \\ \mathsf{S}_\ell(\tilde{Z}_{\ell-\eta+\nu}) \text{ and } \mathsf{R}_\ell(\tilde{Z}_{\ell-\eta+\nu}) & \text{if } \nu + 2 \le \ell \le \eta \end{cases}$$

However, the first case contradicts the semantic security assumption of ElGamal encryption, the second case contradicts Lemma 3, and the last case contradicts Lemma 2. This implies that no such polynomial-time distinguishers exists. Finally, because the simulator runs in linear time, it meets the requirement for a polynomial time simulation. This completes the proof. □

4.3 Efficiency Analysis

In the following we give a detailed analysis of the running time and the space requirements of the protocol. We base our protocol on ElGamal encryption and the power function with primes $|p| = 1024, |q| = 160$. To measure the overhead at each user, we count the number of exponentiations using a 1024-bit modulus.

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

Detailed Comparison. We consider three protocols for a comparison with the literature: Kissner and Song (KS) protocol [24], Burkhart and Dimitropoulos (BD) protocol [4], and Applebaum et al. protocol.

- **KS protocol.** Since KS does not provide a way for finding a threshold value τ that separates the κ-th from the $(\kappa + 1)$-th items, we do not know the actual complexity required for computing τ. However, 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. The DB protocol does not use Yao's garbled circuit evaluation, but it extensively utilizes two special-purpose sub-protocols–`equality` and `lessthan` (see [9], [31]). However, in [5] it is shown that comparing two shared secrets computational intensive. Thus, BD uses 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 runs two key algorithms:
  - *Finding a correct threshold value* τ: the round complexity of the algorithm is
  $$(\log k(2 \log p + 10) + \log p + 2 \log p + 10)\, nk$$
  - *Resolving collisions*: the round complexity of this algorithm is
  $$\frac{n(n-1)}{2} \log p + 2(n-1) \log p + 10(n-1)$$

  Thus, the total round complexity is $\mathcal{O}(n(n + k \log k) \log p)$ for hash tables of size $\log k$ and U of cardinality nk. Their protocol requires $4\left(\frac{n(n-1)}{2}k + k(n-1)\right)$ multiplications in $\mathbb{Z}_p^\times$.
- **Applebaum *et al.*'s protocol.** Let us use $\mathcal{O}_p(\cdot)$ and $\mathcal{O}_N(\cdot)$ to denote the complexity using modulus prime p and modulus composite N, respectively. Assume all elements are integers less than p and the maximum multiplicity is less than $\log \log p$. This protocol's computations are as follows:
  - Interactive computation between users and the proxy (defined in §2): First, users should

TABLE 2
Complexity Analysis

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

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 encrypts his ElGamal ciphertexts using ElGamal encryption once more, which requires $2nk$ exponentiations over $\mathbb{Z}_p^\times$.
- Aggregation by Database: The most computationally-intensive parts are ElGamal and GM decryption. Since the database receives two types of ElGamal ciphertexts, it has to perform $2nk$ exponentiations over $\mathbb{Z}_p^\times$. GM decryption requires $2nk \log\log p$ exponentiations over $\mathbb{Z}_N^\times$.

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

In order to present a fair comparison with our protocol, we devise our protocol for a proxy-based model in the following subsection.

4.4 A Proxy-based Construction

The most crucial drawback of the above 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 can also obtain a constant-round κ^+ protocol with the same security with modifications as follows:

- Assume that there are n_1 proxies and n_2 databases described as in [2].
- 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 [2], our protocol does not require OT operations, nor an extra encryption scheme. Recall that Applebaum et al.'s protocol requires to use both the ElGamal encryption scheme and the Goldwasser-Micali (GM) encryption scheme: the ElGamal encryption scheme is used to encrypt elements in multisets, but the GM encryption scheme is used to encrypt their multiplicity.

5 A κ^+ Protocol with Malicious Adversaries

Now, we present our protocol secure against malicious adversaries and by realizing the ideal functionality $\mathcal{F}_{\mathsf{topk}}$. We first describe privacy issues raised by a malicious adversary, which will be addressed in the construction. We then provide a full description of a secure κ^+ protocol in the malicious model and end with the security analysis.

5.1 Possible Attacks by a Malicious Adversary

In the following, we outline several privacy issues raised by allowing a malicious adversary to the basic protocol in §4.1. Recall that $U = \{u_i\}_{i\in[n]}$ denotes the set of all users and $\Upsilon \subsetneq U$ is used to further identify the set of corrupted users.

1) It is clearly easy for a corrupted user to construct his multiset by copying an honest user's input. For example, a user $u \in \Upsilon$ obtains an encrypted input $\bar{\mathtt{X}}_i$ of an honest user u_i through a public channel. Since ElGamal encryption is homomorphic, she can re-randomize the output of u_i so that she can submit it as the encryptions of his input multiset without detection.
   To address these problems, we introduce a ZKP for verifying whether a user knows the corresponding plaintext m to an ElGamal ciphertext $c \in \mathsf{E}_{pk}(m)$.
2) Corrupted players may deviate from the protocol by producing their outputs by applying an incorrect permutation or a value different from a secret key s_i fixed in the setup phase of the protocol.
   These problems can be addressed by using ZKPs specified in §3.6. First, given an ElGamal ciphertext $c \in \mathsf{E}_{pk}(m)$, when a user u_i computes $\bar{\bar{c}} = E_{s_i}(c)$ she need to first prove that she raised c exactly to the power of s_i. In addition, given a list of ElGamal ciphertexts, $L = \{c_i\}_{i\in[n]}$, she must produce another list of ElGamal ciphertexts, $L' = \{c'_i\}_{i\in[n]}$, with a proof that there exists a permutation $\pi \in \Sigma_n$ satisfying $\mathsf{D}_{sk}(c'_{\pi(i)}) = \mathsf{D}_{sk}(c_i)$ for all $i \in [n]$, .

5.2 The Protocol Specification

We are now ready to describe a protocol that securely computes $\mathcal{F}_{\mathsf{topk}}$, allowing an adversary to behave maliciously. The main component of this protocol is a careful combination of the basic protocol and ZKPs. However, introducing these zero-knowledge protocols leads to changes in the basic protocol. To ensure a better readability of the protocol, we add two symbols to the protocol specification as follows: a star symbol ($\star$) for indicating that a step is newly added and a dagger symbol (†) for that a step is modified from the basic protocol's description. In the following, $\wp_{\mathsf{topk}}$

denotes the real-world protocol corresponding to the ideal functionality $\mathcal{F}_{\mathsf{topk}}$.

Protocol $\wp_{\mathsf{topk}}$ — Secure Over-threshold Aggregation

- *Inputs.* $\mathtt{X}_i = \{\alpha_{i,1}, \ldots, \alpha_{i,k}\}$ with $|\mathtt{X}_i| = k$ for all $i \in [n]$.
- *System parameters.* Let $\mathsf{params} = (\mathbb{G}, p, q, g)$ be as defined in §4. In addition, we simply assume that Pedersen's commitment scheme uses the exactly same elements, and the keys are selected so that the message spaces are compatible.
- *Protocol actions.*
  - **Setup.** For each user $u_{i\in[n]}$:
    - †1) computes $y_i = g^{x_i}$ and $h_i = g^{x'_i}$ for randomly chosen $x_i, x'_i \in \mathbb{Z}_q$, and sends (y_i, h_i) to other users with a proof of knowledge of $\log_g y_i$ and $\log_g h_i$ using the ZKP protocol for $\mathcal{R}_{\mathsf{DL}}$. The key for the Pedersen commitment scheme is $h = \prod_{i=1}^n h_i$ and the public key for the ElGamal PKE is $y = \prod_{i=1}^n y_i$.
    - †2) chooses a pair of secret keys (s_i, t_i) where $s_i t_i = 1$ and publishes commitments $\hat{\mathsf{s}}_i = \mathsf{com}_{ck}(s_i)$ and $\hat{\mathsf{t}}_i = \mathsf{com}_{ck}(t_i)$ for the respective keys while proving that $s_i t_i = 1$ by invoking the ZKP protocols for $\mathcal{R}_{\mathsf{INV}}$ and $\mathcal{R}_{\mathsf{com}}$.
  - **DEncrypt.**
    - †1) Every user u_i computes $\bar{\mathtt{X}}_i$ as in the basic protocol, but sends it to all other users while proving the knowledge of each element by invoking the ZKP protocol for $\mathcal{R}_{\mathsf{PT}}$.
    - †2) Every user checks if all ZKPs were computed correctly, and aborts otherwise. The user u_1 then computes Y_0 as in the basic protocol, proving the knowledge of s_1 using the ZKP protocol for $\mathcal{R}_{\mathsf{DE}}$.
  - **Shuffle & DEncrypt.**
    - 1) Each user $u_{i\geq 2}$ computes $E_{s_i}(Y_{i-1})$ as in the basic protocol.
    - 2) Each user $u_{i\in[n]}$ gets Y_i by applying $\pi_i \xleftarrow{\$} \Sigma_{nk}$ and $\gamma_{i,\ell} \xleftarrow{\$} \mathbb{Z}_q$ to $\bar{\bar{\alpha}}_\ell$ for all $\ell \in [nk]$.
    - †3) Each user $u_{i\in[n]}$ sends Y_i to all other users, and proves knowledge of $(s_i, \pi_i, \{\gamma_{i,\ell}\})$ by using ZKPs for $\mathcal{R}_{\mathsf{DE}}$ and $\mathcal{R}_\pi$, respectively.
    - ⋆4) Upon receiving Y_i along with correct double encryption and shuffle proofs, all other users engage in zero-knowledge executions $\wp_{\mathsf{DE}}$ and $\wp_\pi$ with u_i for which u_i proves that Y_i were computed correctly. If an error is found, the protocol is aborted
  - **Aggregate.** As in the basic protocol, all users obtain $\tilde{Z} = \{\langle\tilde{\alpha}, F(\tilde{\alpha})\rangle | \tilde{\alpha} \in \tilde{\mathtt{U}} \wedge F(\tilde{\alpha}) \geq \kappa\}$ by participating in a group decryption.
  - **Reveal.** Let $\tilde{Z}_0$ be as in the basic protocol. For each $i \in [n]$:
    - †1) u_i computes $\tilde{Z}_i$ as in the basic protocol and sends it to all other users, proving the knowledge of t_i by invoking the ZKP protocol for $\mathcal{R}_{\mathsf{DE}}$.
    - ⋆2) Upon receiving $\tilde{Z}_i$, all other users perform the ZKP protocol $\wp_{\mathsf{DE}}$ with u_i for which u_i proves that she correctly computed $\tilde{Z}_i$ from her committed key t_i and her input $\tilde{Z}_{i-1}$. If an error is found, the protocol is aborted.

  Finally if all checks with u_n succeed, all users get a κ^+ set $Z = \{\alpha \in \mathtt{U} | F(\alpha) \geq \kappa\}$.

5.2.1 Group Construction

Group setup. We first need to discuss how users participate in an arbitrary group. In our work, users make a group together with the help of a center $\mathcal{C}$ which receives the IDs of users who would like to currently join in our protocol. Because we allow even the center to misbehave maliciously, we apply a group construction protocol which is provably secure against such an attacker. One concrete example is Lindell and Waisbard's scheme [27, §5.2.2]. For $n = 20$, their scheme ensures a center to maliciously build a group with probability 10^{-48}.

Electing a group leader. The next issue we discuss is how users elect a group leader after building a group. Indeed, leader election is a fundamental problem in distributed computing. There are numerous protocols for addressing this problem (see, for example, [33]] for the semi-honest case; there have also been extensions that deal with corrupted users [13], [22]). In particular, Katz and Koo suggested a leader election protocol running in a constant round, using moderated verifiable secret sharing [22].

5.2.2 Efficiency

On top of the overhead of the basic protocol, the additional cost for the protocol $\wp_{\mathsf{topk}}$ is required for performing the ZKP protocols. Table 3 summarizes the complexities of selected ZKP protocols. We evaluated our scheme using ElGamal encryption and Pedersen commitments with primes p, q where $q | p-1$, $|q| = 160$, $|p| = 1024$. In particular we considered a ZKP protocol for correct shuffle by Groth [19]. Since all of them are a special honest verifier zero-knowledge argument of knowledge, we need to transform the used protocol into a standard ZKP protocol, which requires additional computation and communication cost. However, because this transformation does not increase the big-O complexities, we ignored this cost in our evaluation. Moreover, we did not consider some optimized variants for the three zero-knowledge protocols.

5.3 Security Analysis

We now proceed to prove that the protocol $\wp_{\mathsf{topk}}$ is secure in the presence of malicious adversaries. The

TABLE 3
Complexities of Zero-knowledge Protocols (ZKP). Communication is in bits at the prover's side.

	$\mathcal{R}_{\mathsf{DL}}$	$\mathcal{R}_{\mathsf{DE}}$	$\mathcal{R}_{\mathsf{PT}}$	$\mathcal{R}_{\pi}$
Prover	n	$2n^2k + n\kappa$	n^2k	$0.4nk$
Verifier	$2n$	$4n^2k + 2n\kappa$	$2n^2k$	$0.5nk$
Comm.	$1184n$	$2368n^2k + 1184n\kappa$	$1184n^2k$	$720nk$

following is our main theorem stating this result.

Theorem 3: Assuming the threshold ElGamal encryption $\mathcal{PKE}$ is semantically secure, the hardness of the DDH and DL problems, and that all of the previously specified ZKPs cannot be forged, then the protocol $\wp_{\mathsf{topk}}$ described above securely computes $\mathcal{F}_{\mathsf{topk}}$ in the presence of malicious adversaries.

Proof: It is clear that for the case where no users are corrupted, the output is correct. Our proof is in a hybrid model where a TTP is used to compute an ideal functionality $\mathcal{F}_{\mathsf{KG}}$ which outputs a public value (pk, ck) for Pedersen's commitment scheme and the ElGamal PKE, $\mathcal{F}_{\mathsf{D}}$ for decryption and the ZKPs for $\mathcal{R}_{\mathsf{DL}}, \mathcal{R}_{\mathsf{PT}}, \mathcal{R}_{\mathsf{com}}, \mathcal{R}_{\mathsf{INV}}, \mathcal{R}_{\mathsf{DE}}$ and $\mathcal{R}_{\pi}$.

Let $J = \{i_1, \ldots, i_\nu\}$ denote the set of corrupted users. Let $\mathcal{A}$ denote an adversary controlling users with indexes in J. We construct a simulator $\mathcal{S}$ that extracts $\mathcal{A}$'s input $\mathtt{X}_{j\in J}$ and outputs a κ^+ set Z. A full description of the simulator is as follows:

1) $\mathcal{S}$ is given $\mathtt{X}_{j\in J}, Z, M$ and $\mathcal{A}$'s auxiliary input z, and invokes $\mathcal{A}$ on these inputs.
2) $\mathcal{S}$ receives from $\mathcal{A}$, $(x_{j\in J}, x'_{j\in J})$ for the ideal functionality of $\mathcal{R}_{\mathsf{DL}}$ and stores $(x_{j\in J}, x'_{j\in J})$ only if for each $j \in J$, $y_j = g^{x_j}$ and $h_j = g^{x'_j}$. Otherwise $\mathcal{S}$ aborts and sends $\perp$ to the TTP for $\mathcal{F}_{\mathsf{topk}}$.
3) Let $I := [n]\backslash J$. $\mathcal{S}$ chooses $x_i, x'_i \in (\mathbb{Z}_q)^2$ uniformly at random and sends $y_i = g^{x_i}$ and $h_i = g^{x'_i}$ for each $i \in I$. It then emulates the TTP abd computes $\mathcal{R}_{\mathsf{DL}}$.
4) For each $j \in J$, $\mathcal{S}$ receives from $\mathcal{A}$, (s_j, t_j) for the ideal computation of $\mathcal{R}_{\mathsf{INV}}$ and records it only if $s_j t_j = 1$; otherwise $\mathcal{S}$ aborts and sends $\perp$ to $\mathcal{F}_{\mathsf{topk}}$.
5) For each $i \in I$, $\mathcal{S}$ chooses $(\mathtt{s}_i, \mathtt{t}_i) \xleftarrow{\$} (\mathbb{Z}_q)^2$ such that $\mathtt{s}_i\mathtt{t}_i = 1$ and sends $\hat{\mathtt{s}}_i = \mathsf{com}(\mathtt{s}_i)$ and $\hat{\mathtt{t}}_i = \mathsf{com}(\mathtt{t}_i)$. It then emulates the TTP computing $\mathcal{R}_{\mathsf{com}}$ and $\mathcal{R}_{\mathsf{INV}}$.
6) From $\mathcal{A}$, $\mathcal{S}$ receives the encrypted sets $\bar{\mathtt{X}}_{j\in J}$ and $\mathtt{X}_{j\in J}$ (upon the execution of $\mathcal{R}_{\mathsf{PT}}$), and verifies if each $\mathtt{X}_{j\in J}$ in this step is equal to each $\mathtt{X}_{j\in J}$ received in Step 1. If there is any difference between the two sets, $\mathcal{S}$ sends $\perp$ to $\mathcal{F}_{\mathsf{topk}}$ and aborts. $\mathcal{S}$ sends $\mathtt{X}_{j\in J}$ to the TTP for $\mathcal{F}_{\mathsf{topk}}$ and receives a set Z and the multiplicity distribution of the union, M, as the answer.
7) $\mathcal{S}$ chooses each random set $\mathcal{X}_{i\in I}$ such that both Z and M are derived from $\mathtt{X}_{j\in J} \cup \mathcal{X}_{i\in I}$ by adding random elements in $\mathbb{G}$. $\mathcal{S}$ computes the encryptions $\bar{\mathcal{X}}_{i\in I}$. $\mathcal{S}$ sends all encryptions to $\mathcal{A}$ and then emulates the TTP for $\mathcal{R}_{\mathsf{PT}}$.
8) For each user $u_{j\in J}$, $\mathcal{S}$ receives Y_j from $\mathcal{A}$ along with $(s_j, \pi_j, \{\gamma_{j,\ell}\}_{\ell\in[nk]})$ for the execution of $\mathcal{R}_{\mathsf{DE}}$ and $\mathcal{R}_{\pi}$, respectively. $\mathcal{S}$ checks that Y_j is computed from Y_{j-1}, and terminates otherwise. If $1 \in J$, $\mathcal{S}$ computes Y_0 using $\bar{\mathtt{X}}_{j\in J} \cup \bar{\mathcal{X}}_{i\in I}$ and s_1 received from $\mathcal{A}$. For each $u_{i\in I}$, $\mathcal{S}$ computes Y_i according to the protocol while emulating the ideal functionalities of $\mathcal{R}_{\mathsf{DE}}$ and $\mathcal{R}_{\pi}$ by sending Y_i to $\mathcal{A}$, where it uses a random permutation $\sigma_i \in \Sigma_{nk}$ and randomizers.
9) $\mathcal{S}$ receives inputs for the functionality $\mathcal{F}_{\mathsf{D}}$ from $\mathcal{A}$ and gives $\tilde{Z}$ and M to $\mathcal{A}$ where $\tilde{Z} = E_s(Z)$ for $s = \prod_{i\in I} \mathtt{s}_i \cdot \prod_{j\in J} s_j$.
10) For each $u_{j\in J}$, $\mathcal{S}$ receives $\tilde{Z}_j$ from $\mathcal{A}$ together with t_j for the functionality of $\mathcal{R}_{\mathsf{DE}}$. It checks that $\langle D_{t_j}(\tilde{\alpha}), F(\tilde{\alpha})\rangle \in \tilde{Z}_j$ for all $\langle \tilde{\alpha}, F(\tilde{\alpha})\rangle \in \tilde{Z}_{j-1}$ and aborts otherwise. For each $u_{i\in I}$, $\mathcal{S}$ computes $\tilde{Z}_i$ according to the protocol and emulates the functionality of $\mathcal{R}_{\mathsf{DE}}$ by sending $\tilde{Z}_i$ to $\mathcal{A}$.

As when proving Theorem 2, we compare the simulated execution to a hybrid execution where a TTP is used to computed the ideal functionalities $\mathcal{F}_{\mathsf{KG}}$ and $\mathcal{F}_{\mathsf{D}}$, and the ZKPs for $\mathcal{R}_{\mathsf{DL}}, \mathcal{R}_{\mathsf{PT}}, \mathcal{R}_{\mathsf{INV}}, \mathcal{R}_{\mathsf{DE}}$ and $\mathcal{R}_{\pi}$. To prove that $\mathcal{A}$'s outputs in the hybrid and simulated executions are computationally indistinguishable, we construct a sequence of hybrid games and show that the corresponding random variables $H^{(\ell)}_{\mathcal{A}(z)}(\mathtt{X}_{i\in[n]}, \lambda)$ that consist of the output of $\mathcal{A}$ in hybrid game $H^{(\ell)}$ are computationally indistinguishable.

Game $H^{(0)}$: The simulated execution.

Game $H^{(1)}$: The simulator $\mathcal{S}_1$ does not interact with the TTP for $\mathcal{F}_{\mathsf{topk}}$ and is given u_i's real input $\mathtt{X}_i$ instead for each $i \in I$. $\mathcal{S}_1$ works exactly as $\mathcal{S}$ except that in Step 7 of the simulation, it uses the encryptions in conjunction with the sets $\mathtt{X}_{i\in I}$ (whereas $\mathcal{S}$ uses $\mathcal{X}_{i\in I}$). That is, the only difference between the two games is that $\mathcal{S}$ uses random sets to complete Z and M, whereas $\mathcal{S}_1$ uses $\mathtt{X}_{i\in I}$.

We claim that the distributions in these two executions are computationally indistinguishable. Let $\mathtt{X}'_i = \mathtt{X}\backslash Z$ and $\mathcal{X}'_i = \mathcal{X}\backslash Z$ for each $i \in I$, and let $\alpha' \in \mathtt{X}'_i$ and $\alpha'' \in \mathcal{X}'_i$ have the same position for some $i \in I$. Note that E_{pk} is semantically secure. Assuming $\mathcal{A}$ distinguishes the views of games $H^{(0)}$ and $H^{(1)}$, we can construct an adversary $\mathcal{A}_{\mathsf{E}}$ that breaks the semantic security assumption in $\mathbb{G}$ with the same probability. This contradicts the semantic security assumption of the ElGamal PKE. Hence,the random variables $H^{(0)}_{\mathcal{A}(z)}$ and $H^{(1)}_{\mathcal{A}(z)}$ are computationally indistinguishable.

Game $H^{(2)}$: The simulator $\mathcal{S}_2$ is identical to $\mathcal{S}_1$ except that it commits to the inputs (s_i, t_i) of each honest user $u_{i\in I}$ from the start instead of computing the commitments based on random values $(\mathtt{s}_{i\in I}, \mathtt{t}_{i\in I})$. As com_{ck} is a perfectly hiding commitment scheme,

$H^{(1)}_{\mathcal{A}(z)}$ and $H^{(2)}_{\mathcal{A}(z)}$ are identically distributed.

Game $H^{(3)}$: The simulator $\mathcal{S}_3$ works exactly as $\mathcal{S}_2$ except that in Step 8 of the simulation, it uses (s_i, π_i) to compute Y_i, whereas $\mathcal{S}_2$ uses $(\mathtt{s}_i, \sigma_i)$. Note that com_{ck} is a perfectly hiding commitment scheme and the double encryption scheme $(\mathcal{PKE}, \mathcal{E})$ is commutative, i.e., $E_{s_i} \circ \cdots \circ E_{s_1} \circ \mathsf{E}_{pk} = E_{s_i} \circ \cdots \mathsf{E}_{pk} \circ E_{s_1} = \cdots = \mathsf{E}_{pk} \circ E_{s_i} \circ \cdots \circ E_{s_1}$. Thus, the only difference between the two games is that for some $(i, j) \in I \times J$ with $j = i + 1$, $\mathcal{S}_3$ sends $Y_i = \langle \mathsf{E}_{pk}(E_s(\alpha_{\pi_i(\ell)})) \rangle_{\ell \in [nk]}$ to $\mathcal{A}$ where $s = \prod_{l=1}^{i} s_i$, whereas $\mathcal{S}_2$ sends Y_i to $\mathcal{A}$ by applying $s = \prod_{j \in I, j<i} s_j \cdot \prod_{i' \in I, i' \le i} \mathtt{s}_{i'}$ instead. We claim that $H^{(2)}_{\mathcal{A}(z)}$ and $H^{(3)}_{\mathcal{A}(z)}$ are computationally indistinguishable. The proof follows from the security of the shuffle because Y_i in both games is a list of mixed and re-randomized ElGamal ciphertexts. Let $\mathcal{A}_\pi$ denote a PPT adversary who is given (pk, Y_{i-1}, Y_i) and wishes to find π_i and $\{r_\ell\}_{\ell \in [nk]}$ such that $\alpha''_\ell = \alpha'_{\pi_i(\ell)} \cdot \mathsf{E}_{pk}(1; r_\ell)$ where $\alpha'_\ell \in Y_{i-1}, \alpha''_\ell \in Y_i$. Using the same technique used in Lemma 3, we can build $\mathcal{A}_\pi$ that breaks the shuffle with the same probability that A distinguishes the views of two games. This leads to a contradiction to the DDH assumption by Lemma 3, and completes the argument.

Game $H^{(4)}$: The simulator $\mathcal{S}_4$ uses $t_{i \in I}$ instead of $\mathtt{t}_{i \in I}$. Note that every $\tilde{Z}_{i \in [n]}$ is a list of ElGamal ciphertexts. By the same argument the random variables $H^{(3)}_{\mathcal{A}(z)}$ and $H^{(4)}_{\mathcal{A}(z)}$ are computationally indistinguishable.

Game $H^{(5)}$: The hybrid execution. The output distribution in $H^{(4)}$ is identical to the output in the hybrid execution since the only difference is that the honest users performs all computations by themselves in the latter game.

Clearly, $\mathcal{S}$ runs in probabilistic polynomial time. This completes the proof. □

6 Conclusion

In this paper we have looked at the problem of finding the κ^+ elements 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 by analyzing its precise computational and communicational cost. Moreover, we provided a full proof showing that our protocol is secure in the presence of semi-honest adversaries.

Since the semi-honest protocols commonly have critical security restrictions, by requiring every adversary to follow the instructions specified in the protocol, we transformed our basic protocol into a stronger κ^+ protocol which is also secure in the presence of malicious adversaries. In addition to a full description of our protocol with malicious adversaries, we proved that the protocol is secure within the simulation paradigm.

In the future, we will look into converting the Zero-Knowledge Proofs from their present interactive variant into Non-Interactive Zero Knowledge Proofs through the Fiat-Shamir heuristic, which will improve the communication complexity of our protocols.

Acknowledgment

Myungsun Kim was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. 2014-R1A1A2058377), the work of Aziz Mohaisen was supported in part by a grant from the University at Buffalo, Jung Hee Cheon was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No. B0717-16-0098, Development of homomorphic encryption for DNA analysis and biometry authentication), and Yongdae Kim was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (No. NRF-2014M3C4A7030648). The preliminary version of this paper was presented in Information Security and Cryptology–ICISC 2012–15th International Conference, Seoul, Korea, November 28–30, 2012.

References

[1] M. Abe, R. Cramer, and S. Fehr. Non-interactive distributed-verifier proofs and proving relations among commitments. In Y. Zheng, editor, *Advances in Cryptology-Asiacrypt*, LNCS 2501, pages 206–223, 2002.

[2] B. Applebaum, H. Ringberg, M. Freedman, M. Caesar, and J. Rexford. Collaborative, privacy-preserving data aggregation at scale. In M. Atallah and N. Hopper, editors, *PETS*, LNCS 6205, pages 56–74, 2010.

[3] S. Bayer and J. Groth. Efficient zero-knowledge argument for correctness of a shuffle. In D. Pointcheval and T. Johansson, editors, *Advances in Cryptology-Eurocrypt*, LNCS 7237, pages 263–280, 2012.

[4] M. Burkhart and X. Dimitropoulos. Fast privacy-preserving top-*k* queries using secret sharing. In *IEEE ICCCN*, 2010.

[5] M. Burkhart, M. Strasser, D. Many, and X. Dimitropoulos. SEPIA: Privacy-preserving aggregation of multi-domain network events and statistics. In *USENIX Security*, 2010.

[6] J. Camenisch. Proof systems for general statements about discrete logarithms. Technical Report TR 260, Dept. of Computer Science, ETH Zurich, 1997.

[7] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Y. Desmedt, editor, *Advances in Cryptology-Crypto*, LNCS 839, pages 174–187, 1994.

[8] S. Chow and J.-H. Lee and L. Subramanian. Two-party computation model for privacy-preserving queries over distributed databases. In *NDSS*, 2009.

[9] I. Damgård, M. Fitzi, E. Kiltz, J. B. Nielsen, and T. Toft. Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In S. Halevi and T. Rabin, editors, *TCC*, pages 285–304, 2006.

[10] T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and D. Chaum, editors, *Advances in Cryptology-Crypto*, LNCS 196, pages 10–18, 1984.

[11] Y.-C. Fan and A. L. P. Chen. Efficient and robust schemes for sensor data aggregation based on linear counting. *IEEE Trans. Parallel Distrib. Syst.*, 21(11):1675–1691, 2010.

[12] Y.-C. Fan and A. L. P. Chen. Energy efficient schemes for accuracy-guaranteed sensor data aggregation using scalable counting. *IEEE Trans. Knowl. Data Eng.*, 24(8):1463–1477, 2012.

[13] P. Feldman and S. Micali. An optimal probabilistic protocol for synchronous Byzantine agreement. SIAM Journal on Computing, 26:873933, 1997.

[14] O. Goldreich. *The foundations of cryptography: Volume 2–Basic Applications*. Cambridge University Press, 2004.

[15] S. Goldwasser and S. Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 1984.

[16] M. Groat, W. He, and S. Forrest. KIPDA: k-indistinguishable privacy-preserving data aggregation in wireless sensor networks. In *INFOCOM*, pages 2024–2032, 2011.

[17] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, pages 218229, 1987.

[18] J. Groth. A verifiable secret shuffle of homomorphic encryptions. In Y. Desmedt, editor, *PKC*, LNCS 2567, pages 145–160, 2003.

[19] J. Groth. A verifiable secret shuffle of homomorphic encryptions. *Journal of Cryptology*, 23:546–579, 2010.

[20] C. Hazay and Y. Lindell. *Efficient Secure Two-Party Protocols – Techniques and Constructions*. Springer, 2010.

[21] W. He, X. Liu, H. Nguyen, K. Nahrstedt, and T. Abdelzaher. PDA: privacy-preserving data aggregation in wireless sensor networks. In *INFOCOM*, pages 2045–2053, 2007.

[22] J. Katz and C.-Y. Koo. On expected constant-round protocols for Byzantine agreement. In *Advances in Cryptology-Crypto*, pages 445-462, 2006.

[23] M. Kim, A. Mohaisen, J. H. Cheon, and Y. Kim. Private over-threshold aggregation protocols. In T. Kwon, M.-K. Lee, and D. Kwon, editors, *ICISC 2012*, LNCS 7839, pages 472–486, 2012.

[24] L. Kissner and D. Song. Privacy-preserving set operations. In V. Shoup, editor, *Advances in Cryptology-Crypto*, LNCS 3621, pages 241–257, 2005.

[25] Q. Li and G. Cao. Efficient and privacy-preserving data aggregation in mobile sensing. In *ICNP*, pages 1–10, 2012.

[26] Y.-H. Lin, S.-Y. Chang, and H.-M. Sun. CDAMA: concealed data aggregation scheme for multiple applications in wireless sensor networks. *IEEE Trans. Knowl. Data Eng.*, 25(7):1471–1483, 2013.

[27] Y. Lindell and E. Waisbard. Private web search http://.

[28] A. Mohaisen, D. Hong, and D. Nyang. Privacy in location based services: Primitives toward the solution. In *NCM*, 2008.

[29] M. Naor and B. Pinkas. Oblivious transfer with adaptive queries. In M. Wiener, editor, *Advances in Cryptology-Crypto*, LNCS 1666, pages 573–590, 1999.

[30] C. Neff. A verifiable secret shuffle and its application to e-voting. In *ACM CCS*, pages 116–125, 2001.

[31] T. Nishide and K. Ohta. Multiparty computation for interval, equality, and comparison without bit-decomposition protocol. In T. Okamoto and X. Wang, editors, *PKC*, LNCS 4450, pages 343–360, 2007.

[32] T. Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In E. Brickell, editor, *Advances in Cryptology-Crypto*, LNCS 740, pages 31–53, 1992.

[33] G. Peterson. An $O(nlogn)$ unidirectional distributed algorithm for the circular extrema problem. ACM Trans. on Programming Languages and Systems, 4(4):758762, 1982.

[34] H. Özgür Tan, I. Korpeoglu, and I. Stojmenovic. Computing localized power-efficient data aggregation trees for sensor networks. *IEEE Trans. Parallel Distrib. Syst.*, 22(3):489–500, 2011.

[35] T. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In J. Feigenbaum, editor, *Advances in Cryptology-Crypto*, LNCS 576, pages 129–140, 1991.

[36] C. Rottondi, G. Verticale, and C. Krauß. Distributed privacy-preserving aggregation of metering data in smart grids. *IEEE JSAC*, 31(7):1342–1354, 2013.

[37] Y. Sang and H. Shen. Efficient and secure protocols for privacy-preserving set operations. *ACM Transactions on Information and System Security (TISSEC)*, 13(1):9:1–9:35, 2009.

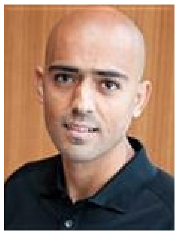
[38] C.-P. Schnorr. Efficient identification and signatures for smart cards. In G. Brassard, editor, *Advances in Cryptology-Crypto*, LNCS 435, pages 239–252, 1989.

[39] J. Shi, R. Zhang, Y. Liu, and Y. Zhang. PriSense: privacy-preserving data aggregation in people-centric urban sensing systems. In *INFOCOM*, pages 758–766, 2010.

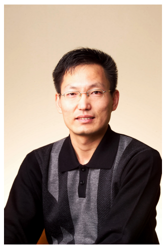
[40] X. Xu, X.-Y. Li, X. Mao, S. Tang, and S. Wang. A delay-efficient algorithm for data aggregation in multihop wireless sensor networks. *IEEE Trans. Parallel Distrib. Syst.*, 22(1):163–175, 2011.

[41] A. Yao. Protocols for secure computations. In *FOCS*, pages 160–164, 1982.

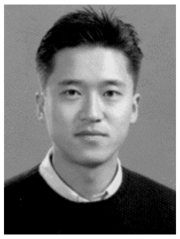
Myungsun Kim received the B.S. degree in computer science and engineering from Sogang University, Seoul, Korea, in 1994 and the M.S. degree in computer science and engineering from the Information and Communications University (ICU), Daejeon, in 2002. He received the Ph.D. degree in mathematics from Seoul National University (SNU), Seoul, in 2012. Currently, he is an assistant professor in Department of Information Security, University of Suwon. He was with the Digital Media Research and Development Center, Samsung Electronics, until 2008. His research interests include multiparty computation in cryptography



Aziz Mohaisen is an Assistant Professor in the Department of Computer Science and Engineering at SUNY Buffalo. He obtained the M.S. and Ph.D. degrees in Computer Science from the University of Minnesota, both in 2012. Previously, he was a senior research scientist at Verisign Labs from 2012 to 2015 and a researcher at the Electronics and Telecommunication Research Institute (ETRI) in South Korea from 2007 to 2009. His research interests are in the areas of networked systems, systems security, data privacy, and applied cryptography. He is a member of ACM and a senior member of IEEE.



Jung Hee Cheon is a professor in the department of mathematical sciences of Seoul National University (SNU) and a director of the cryptographic hard problems research initiatives. He received his B.S. and Ph.D. degrees in mathematics from KAIST in 1991 and 1997, respectively. After working as a senior researcher in ETRI and a visiting scientist in Brown University, he was an assistant professor in Information and Communications University (ICU) for two years. His research interests include computational number theory, cryptography and information security. He has been on program committees of many international conferences including Crypto, Eurocrypt, and Asiacrypt. He received the best paper award in Asiacrypt 2008.



Yongdae Kim is a professor in the Department of Electrical Engineering at KAIST. He received PhD degree from the computer science department at the University of Southern California under the guidance of Gene Tsudik. Prior to join KAIST, he was a faculty member in the Department of Computer Science and Engineering at the University of Minnesota - Twin Cities. He received NSF career award and McKnight Land-Grant Professorship Award from University of Minnesota in 2005. Currently, he is serving as a steering committee member of NDSS (Network and Distributed System Security Symposium). His current research interests include security issues in various systems such as cyber physical systems, mobile/ad hoc/sensor/cellular networks, social networks, storage systems, and anonymous communication systems.