Communication-Efficient and Privacy-Preserving Protocol for Computing Over-Threshold Set-Union

Xuhui Gong, Qiang-sheng Hua(✉), and Hai Jin

National Engineering Research Center for Big Data Technology and System, Services Computing Technology and System Lab, Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, 430074, China
{xuhuigong,qshua,hjin}@hust.edu.cn

Abstract. In a variety of applications, the data items of multiple participants are collected and analyzed, and meanwhile the participants' privacy needs to be protected. This paper studies an over-threshold data aggregation problem, i.e., over-threshold set-union. In our model, we assume there are n participants, an untrusted data aggregator and a proxy, and each participant has a sensitive data item. The over-threshold set-union is normally defined as follows: given a threshold t, the aggregator only learns the data items which appear at least t times in the union of data items of all participants without learning anything else. Existing solutions either suffer from high communication cost or leak the multiplicity information of data items. In order to handle this defect, we present an efficient protocol in the honest-but-curious model by leveraging threshold secret sharing and dual pairing vector spaces. We prove that the proposed protocol not only has $O(n \log^2 n)$ communication complexity which nearly matches the lower bound $\Omega(n/\log n)$ but also protects the data privacy.

Keywords: Over-threshold set-union · Data aggregation · Secure multi-party computation · Privacy

1 Introduction

The privacy-preserving data aggregation problem is of particular interest in many important applications. Most applications need to collect participants' data and compute some algebraic statistics. However, the data items of participants usually include sensitive information and the participants are reluctant to leak the data privacy to anyone. For example [1], in privacy-preserving distributed network monitoring, every node monitors local behavior and a service provider is responsible for a global anomalous monitoring task, and when the number of local anomalous behaviors exceeds a predetermined system threshold, the service provider identifies the anomaly.

There are two major challenges for the privacy-preserving data aggregation problem. One is communication efficiency. In many practical systems, monitoring nodes usually have tight resource constraints. Thus, minimizing communication overhead is a pursued goal of related researchers for designing the protocol. Another issue is privacy-preserving. After a protocol has been executed, the result of the computation is leaked and no adversary can learn more extra information than what can be inferred from the result.

Many types of research have been proposed on the privacy-preserving data aggregation problem. For example, linear aggregation function-SUM [2], and non-linear aggregation functions [2–7]. In this paper, we study a particular privacy-preserving data aggregation problem, i.e., the over-threshold set-union. This problem can be used in many application scenarios, such as the privacy-preserving distributed network monitoring [1]. As far as we know, there are some works [1], [3], [8], [9], [10] that discuss the over-threshold set-union. In [1], [10] the authors present a probabilistic protocol based on polynomial representation of sets and additively homomorphic encryption. In [3], a closely related work is proposed by using secure multi-party computation. But the techniques used in their protocols can lead to high communication complexity ($O(n^2)$). In [8], [9], the authors show cryptographic methodologies to compute the over-threshold set-union. Although their protocols have lower communication complexity, their schemes can reveal the multiplicity of data.

Thus, a natural question is: can we design a protocol which has lower communication complexity and can protect data privacy including the multiplicity of data. A positive answer is presented in this paper. We propose a new probabilistic protocol for this problem. Our protocol can ensure that the data aggregator obtains accurate results with high probability(a probability with at least $1-1/n^{\epsilon}$, where $\epsilon \geq 1$). Our protocol has nearly linear communication complexity regarding the number of participants while protecting the privacy of participants. We summarize the comparison results in Table 1.

Table 1. Complexity Comparisons

Communication	Method	Lower bound	Multiplicity privacy
$O(n^2)$	[1]	$\Omega(n/\log n)^*$ [11]	$\surd$
$O(n^2)$	[3]		$\surd$
$O(n^2)$	[10]		$\surd$
$O(n)$	[8]		$\times$
$O(n)$	[9]		$\times$
$O(n\log^2 n)$	our		$\surd$

*The lower bound on communication complexity does not consider data privacy and security

Our main contributions in this paper are summarized as follows:

– The proposed protocol can precisely calculate the over-threshold set-union with high probability.

– The proposed protocol not only has $O(n \log^2 n)$ communication complexity which nearly matches the lower bound $\Omega(n/\log n)$ but also protects the data privacy.

The remainder of this paper is organized as follows. We introduce our system model, problem definition, the adversary model, and cryptographic background in Section 2. In Section 3, we describe our efficient protocol for the over-threshold set-union problem. We show the detailed analysis for our protocol in terms of correctness, security, and complexity in Section 4. Finally, we conclude this paper in Section 5.

2 PRELIMINARIES

2.1 System Model

In this paper, we adopt the PDA system architecture [8] (see Figure 1), which is composed of three parts: an untrusted data aggregator $\mathcal{A}_{gg}$, a proxy $\mathcal{P}$ and n participants $\{p_1, ..., p_n\}$. Each participant p_i holds a private data item $x_i \in [1, M]$. The data aggregator has a powerful computational and storage capacity and is responsible for computing some aggregate functions (over-threshold set-union in this paper). Let S denote the multiset $\{x_1, ..., x_n\}$, and $[[x_i]]$ denote the number of times of data item x_i which appears in the multiset S. For any set A, let $x \xleftarrow{U} A$ denote that x is sampled uniformly at random from A. We denote by $|A|$ the number of elements in the set A.

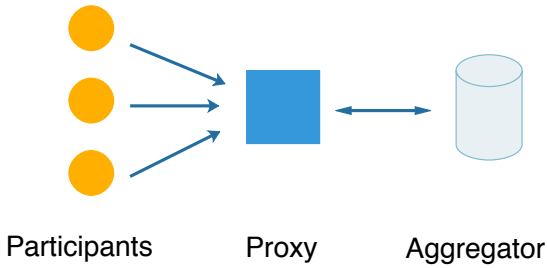

Fig. 1. PDA system architecture

2.2 Problem Definition

In the paper, we mainly investigate the over-threshold set-union problem. Informally speaking, this problem is as follows: the aggregator only learns the data items which appear at least t times in the union of private data items of all participants without obtaining extra knowledge about other data items. The formal definition states as follows.

Definition 1 (Over-Threshold Set-Union) *In our system model, each participant p_i has a data item x_i, and let $S = \{x_1, ..., x_n\}$. Given a threshold t, the aggregator wants to compute the set $\mathcal{U} = \{x'_1, ..., x'_{l'}\}$ without obtaining extra knowledge, where $x'_j \in S, [[x'_j]] \geq t$ for any $j \in \{1, ..., l'\}$.*

2.3 Adversary Model

Like most other related work [1], [8], [9], we consider the honest-but-curious adversaries model. In this model, each party faithfully executes the prescribed actions of the protocol and meanwhile is curious about the private information of other nodes. Moreover, we assume that the participants, proxy and data aggregator do not collude with each other.

2.4 Cryptographic background

Asymmetric Bilinear Pairing Groups Let us denote the asymmetric bilinear pairing groups by $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ which is produced by $\mathcal{G}(1^\lambda)$, where q is a prime, λ is a security parameter, $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are cyclic groups of order q, and e: $\mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$. Let g_1 and g_2 be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. The bilinear map e has the following properties:

1. Bilinearity: $\forall a, b \in \mathbb{Z}_q$, $e({g_1}^a, {g_2}^b) = e(g_1, g_2)^{ab}$;
2. Non-degeneracy: $e(g_1, g_2) \neq 1$.

Definition 2 *(Symmetric External Diffie-Hellman: SXDH Assumption [12]). Given the following distributions* $((q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda)$, g_1, g_2, g_1^a, g_1^b, $Y_\sigma)$ *, where* $Y_0 = g_1^{ab}$, $Y_1 = g_1^{ab+r}$, $a, b, r \xleftarrow{U} \mathbb{Z}_p$, *the advantage* $\boldsymbol{Adv}_{\mathcal{A}}^{\mathrm{SXDH}}$ *that any probabilistic polynomial time (PPT) adversary determines whether* $\sigma = 0$ *or* 1 *is negligible in* λ, *where* $\boldsymbol{Adv}_{\mathcal{A}}^{\mathrm{SXDH}} = |\Pr[\mathcal{A}(1^\lambda, \mathrm{Y}_0) \longrightarrow 1] - \Pr[\mathcal{A}(1^\lambda, \mathrm{Y}_1) \longrightarrow 1]|$.

If $\mathbb{G}_1, \mathbb{G}_2$ reverse the roles, the analogous distributions is also true.

Definition 3 *(External Decisional Linear Assumption: XDLIN [13]) Given* $P_a = ((q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda)$, ${g_1}^\xi$, g_1^κ, $g_1^{\delta\xi}$, $g_1^{\sigma\kappa}$, g_2^ξ, g_2^κ, $g_2^{\delta\xi}$, $g_2^{\sigma\kappa}$, $Y_a)$ *for* $b \in \{1, 2\}$, *where* $Y_0 = g_b^{\delta+\sigma}$, $Y_1 = g_b^{\delta+\sigma+\rho}$, $\delta, \kappa, \sigma, \rho \xleftarrow{U} \mathbb{Z}_p$, *the advantage* $\boldsymbol{Adv}_{\mathcal{A}}^{\mathrm{XDLIN}}$ *that any PPT adversary determines whether* $a = 0$ *or* 1 *is negligible in* λ, *where* $\boldsymbol{Adv}_{\mathcal{A}}^{\mathrm{XDLIN}} = |\Pr[\mathcal{A}(1^\lambda, \mathrm{Y}_0) \longrightarrow 1] - \Pr[\mathcal{A}(1^\lambda, \mathrm{Y}_1) \longrightarrow 1]|$.

Definition 4 (Dual Pairing Vector Spaces [14]) *Dual Pairing Vector Spaces are defined by the tuple* $(q, \mathbb{V}, \mathbb{V}^*, \mathbb{G}_T, \mathbb{A}, \mathbb{A}^*)$ *and can be constructed by* $\mathcal{G}(1^\lambda)$. $\mathbb{V} = \mathbb{G}_1^m$ *and* $\mathbb{V}^* = \mathbb{G}_2^m$ *are* n *dimensional vector spaces and* $\mathbb{V} = (\boldsymbol{a}_1, ..., \boldsymbol{a}_m)$ *and* $\mathbb{V}^* = (\boldsymbol{a}_1^*, ..., \boldsymbol{a}_m^*)$ *are canonical bases of* $\mathbb{V}$ *and* $\mathbb{V}^*$, *respectively, where* $\boldsymbol{a}_i = (0^{i-1}, G_1, 0^{m-i})$, $\boldsymbol{a}_i^* = (0^{i-1}, G_2, 0^{m-i})$, 0^j *represents a line of* j *zeros, e.g.,* $(0^4, 0) = (0, 0, 0, 0, 0)$. $\widetilde{e} : \mathbb{V} \times \mathbb{V}^* \longrightarrow \mathbb{G}_T$ *is pairing and is defined by* $\widetilde{e} = (\boldsymbol{x}, \boldsymbol{y}) = \prod_{i=1}^m e(X_i, Y_i) \in \mathbb{G}_T$, *where* $\boldsymbol{x} = (X_1, ..., X_m) \in \mathbb{V}$ *and* $\boldsymbol{y} = (Y_1, ..., Y_m) \in \mathbb{V}^*$.

Then, given $(q, \mathbb{V}, \mathbb{V}^*, \mathbb{G}_T, \mathbb{A}, \mathbb{A}^*, \widetilde{e})$, we present random dual orthonormal bases $\mathcal{G}_{ob}(1^\lambda, m)$ as follows:

$\psi \xleftarrow{U} Z_p^*, \mathbf{B} = (\chi_{i,j}) \leftarrow GL(m, Z_q), (\phi_{i,j}) \leftarrow \psi(\mathbf{B}^T)^{-1}, \mathbf{b}_i = \sum_{j=1}^m \chi_{i,j}\mathbf{a}_j, \mathbf{b}_i^* = \sum_{j=1}^m \phi_{i,j}\mathbf{a}_j^*, (i = 1, ..., m), \mathbb{B} = (\mathbf{b}_1, ..., \mathbf{b}_m), \mathbb{B}^* = (\mathbf{b}_1^*, ..., \mathbf{b}_m^*), g_T = e(G_1, G_2)^\psi$,

output($\mathbb{B}, \mathbb{B}^*$), where $GL(m, Z_q)$ denotes the general linear group of degree m over Z_p.

For $\mathbf{x} = (X_1, ..., X_m)^T$ and $\mathbb{B} = (\mathbf{b}_1, ..., \mathbf{b}_m)$,, we use $(\mathbf{x})_{\mathbb{B}}$ to denote $\sum_{i=1}^{m} X_i \mathbf{b}_i$. So we have $\widetilde{e}(((\mathbf{x})_{\mathbb{A}}, (\mathbf{y})_{\mathbb{A}^*}) = \prod_{i=1}^{m} e(X_i G_1, Y_i G_2) = e(G_1, G_2)^{\sum_{i=1}^{m} X_i Y_i}$ and $\widetilde{e}((\mathbf{x})_{\mathbb{B}}, (\mathbf{y})_{\mathbb{B}^*}) = \widetilde{e}((\mathbf{Bx})_{\mathbb{A}}, (\psi(\mathbf{B}^T)^{-1}\mathbf{y})_{\mathbb{A}^*}) = e(G_1, G_2)^{\psi \mathbf{Bx} \cdot \mathbf{B}^{-1}\mathbf{y}} = g_T^{\sum_{i=1}^{m} X_i Y_i}$.

Threshold Secret Sharing In this paper, we use an (n, t)-threshold secret sharing scheme, i.e., Shamir's secret sharing technique [15]. The secret sharing scheme is that a secret s is split and distributed to n participants, and only t or more than t participants which work together can reconstruct the secret s. For the secret s, it constructs polynomial $h(x) = s + s_1 x + s_2 x^2 + \cdots + s_{t-1} x^{t-1}$, where $s_1, ..., s_{t-1} \xleftarrow{U} Z_q$. Then, it randomly chooses n distinct elements $\alpha_1, ..., \alpha_n$ from Z_q and computes $h(\alpha_1), ..., h(\alpha_n)$ and sends $(\alpha_i, h(\alpha_i))$ to participant p_i for any $i \in [n]$. For any set $B = \{\alpha_{i_1}, ..., \alpha_{i_t}\}$ of size t, the secret s can be recovered as followings: $s = \sum_{j=1}^{t} \Delta_{\alpha_{i_j}, B}(0) h(\alpha_{i_j})$, where $\Delta_{\alpha_{i_j}, B}(0) = \prod_{r \in B, r \neq \alpha_{i_j}} \frac{-r}{\alpha_{i_j} - r}$.

Moreover, we introduce the distributed ElGamal cryptosystem [16] in a threshold manner, which will be adopted in our paper. A simplified description of the cryptosystem between the proxy $\mathcal{P}$ and data aggregator $\mathcal{A}_{gg}$ is described as follows. Let $\mathbb{G}$ be a multiplicative cyclic group of prime order q and g is one of its generators. $\mathcal{P}$ and $\mathcal{A}_{gg}$ choose private keys sk_1 and sk_2 from Z_q, respectively. Then, they publish $y_1 = g^{sk_1}$ and $y_2 = g^{sk_2}$, and compute the public key $pu = y_1 y_2 = g^{sk_1 + sk_2}$. In order to encrypt a message m, any participant chooses $r \xleftarrow{U} Z_q$ and outputs a ciphertext $\mathbf{E}_{pu}[m]=(A, B) = (g^r, my^r)$. To decrypt the ciphertext (A, B), $\mathcal{A}_{gg}$ needs to send A to $\mathcal{P}$. Then, $\mathcal{P}$ computes $A_1 = A^{sk_1}$ and sends $A_1 = A^{sk_1}$ to $\mathcal{A}_{gg}$, and $\mathcal{A}_{gg}$ can obtain the message m by computing $B/(A_1 A^{sk_2})$ (it is not difficult to see that $B/(A_1 A^{sk_2}) = my^r/(g^{rsk_1} g^{rsk_2}) = my^r/g^{(sk_1 + sk_2)r} = my^r/y^r = m$). The distributed ElGamal cryptosystem has an improtant property: re-encryption. That is, given $\mathbf{E}_{pu}[m] = (A, B)$, the proxy can compute another encryption of m: $(Ag^{r'}, By^{r'})$, where $r' \xleftarrow{U} Z_q$, which is still denoted as $\mathbf{E}_{pu}[m]$ for convenience.

3 Efficient Protocol for Threshold-Over Set-Union

3.1 Overview

In the previous schemes [1], [3], [8], [9], [10] they process the data items directly. And yet we use the binary bits of data items to compute the over-threshold set-union. The main idea for our protocol is that only t or more than t ciphertexts which are generated by using the same data item can be decrypted. To achieve the design goal and reduce the communication complexity, we propose a novel protocol based on the threshold secret sharing and dual pairing vector spaces. Our protocol is composed of the four phases as follows: the **System Setup**, **Ciphertext Generation**, **Blind Ciphertext** and **Ciphertext Aggregation**.

3.2 Basic Protocol

We first introduce a basic protocol which can exactly compute the over-threshold set-union with high probability.

System Setup A trusted authority produces some private keys and distributes to the participants, and the data aggregator and proxy generate their own secret keys. Specifically, it selects $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda)$ and generates $(q, \mathbb{V}, \mathbb{V}^*, \mathbb{G}_T, \mathbb{A}, \mathbb{A}^*, \widetilde{e})$, $\{(\mathbb{B}_j, \mathbb{B}_j^*), (\mathbb{C}_j, \mathbb{C}_j^*)\}_{j=1}^3$ from $\mathcal{G}_{ob}(1^\lambda, m)$, where $\mathbb{B}_j = (\mathbf{b}_{j,1}, ..., \mathbf{b}_{j,9}), \mathbb{B}_j^* = (\mathbf{b}_{j,1}^*, ..., \mathbf{b}_{j,9}^*)$ for $j = 1, 2$, $\mathbb{B}_3 = (\mathbf{b}_{3,1}, ..., \mathbf{b}_{3,6l+11}), \mathbb{B}_3^* = (\mathbf{b}_{3,1}^*, ..., \mathbf{b}_{3,6l+11}^*)$, $\mathbb{C}_j = (\mathbf{c}_{j,1}, ..., \mathbf{c}_{j,6l+9}), \mathbb{C}_j^* = (\mathbf{c}_{j,1}^*, ..., \mathbf{c}_{j,6l+9}^*)$ for $j = 1, 2$, $\mathbb{C}_3 = (\mathbf{c}_{3,1}, ..., \mathbf{c}_{3,6l+11}), \mathbb{C}_3^* = (\mathbf{c}_{3,1}^*, ..., \mathbf{c}_{3,6l+11}^*)$. Then it computes $\{(\alpha_{i,j}, h_{i,j}), (\alpha_{i,j}, \overline{h}_{i,j})\}$ for secrets $\{\beta_j, \gamma_j\}$ by using the Shamir's protocol and samples $\{a_1, b_1\}$ from $\{0,1\}^\lambda$ (to be used in pseudo-random function (PRF)), where $i \in [n], j \in [2] = \{1, 2\}$. It sends $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), \{(\mathbb{B}_j, \mathbb{B}_j^*), (\mathbb{C}_j, \mathbb{C}_j^*)\}_{j=1}^3, \{a_1, b_1\}, \{(\alpha_{i,j}, h_{i,j}), (\alpha_{i,j}, \overline{h}_{i,j})\}_{j=1}^2$ to participant p_i for any $i \in [n]$ and $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ to the data aggregator and proxy.

The data aggregator together with proxy compute secret key sk_1 and sk_2, respectively, and they publish common public key pu by running the distributed EIGamal cryptosystem.

Ciphertext Generation In this phase, each participant p_i encrypts its own data item x_i by using private keys.

Specifically, participant p_i first converts x_i into binary vector $\boldsymbol{v}_i = (x_{i,l-1}, ..., x_{i,0})$, where $l = \lceil \log M \rceil$. Let $\boldsymbol{X}_{i,1}$ and $\boldsymbol{X}_{i,2}$ denote the vectors $(\boldsymbol{v}_i, \mathbf{1}, \boldsymbol{v}_i)$ and $(\mathbf{1}, \boldsymbol{v}_i, -2\boldsymbol{v}_i)$, respectively, where $\mathbf{1} = (1, 1, ..., 1)$ which has l elements. $f_{a_1}(\cdot)$ $(f_{b_1}(\cdot))$ is a pseudorandom function of the PRF family $\mathcal{F}_\lambda = \{f_s : \{0,1\}^\lambda \to \{0,1\}^\lambda\}_{s \in \{0,1\}^\lambda}$ that uses a_1 (b_1) as parameter [17]. Then it computes the ciphertexts $\overline{C}_i = \{C_{i,1}, ..., C_{i,6}, C_i\}$ as follows:

$$
\begin{aligned}
C_{i,1} &= (r_{i,1}, h_{i,1}, 0^2, r_{i,2}, r_{i,3}, 0, 0, 0)_{\mathbb{B}_1}, C_{i,2} = (r_{i,4}, r_{i,5}, 0^2, 0, 0, r_{i,6}, r_{i,7}, 0)_{\mathbb{B}_1^*}, \\
C_{i,3} &= (r_{i,4}, \overline{h}_{i,1}, 0^2, \hat{r}_{i,1}, \hat{r}_{i,2}, 0, 0, 0)_{\mathbb{B}_2}, C_{i,4} = (r_{i,1}, \hat{r}_{i,3}, 0^2, 0, 0, \hat{r}_{i,4}, \hat{r}_{i,5}, 0)_{\mathbb{B}_2^*}, \\
C_{i,5} &= (\overline{r}_{i,1} f_{a_1}(r) \cdot \boldsymbol{X}_{i,1}, r_{i,5}, \hat{r}_{i,3}, 1, 0^{3l+3}, \overline{r}_{i,2}, \overline{r}_{i,3}, 0, 0, 0)_{\mathbb{B}_3}, \\
C_{i,6} &= (f_{b_1}(r) \cdot \boldsymbol{X}_{i,2}, h_{i,1}, \overline{h}_{i,1}, \overline{r}_{i,4}, 0^{3l+3}, 0, 0, \overline{r}_{i,5}, \overline{r}_{i,6}, 0)_{\mathbb{B}_3^*}, \\
C_i &= e(G_1, G_2)^{\beta_1 r_{i,5} + \gamma_1 \hat{r}_{i,3} + \overline{r}_{i,4}},
\end{aligned}
$$

where $\{r_{i,j}\}_{j=1}^7, \{\hat{r}_{i,j}\}_{j=1}^5, \{\overline{r}_{i,j}\}_{j=2}^6 \xleftarrow{U} Z_q$, $\overline{r}_{i,1} \xleftarrow{U} Z_{q'}$ $(q' < q/tl)$, and r is a nonce about the message, and $f_{a_1}(r) f_{b_1}(r) \neq 0$. . Similarly, it computes the

following ciphertexts $\overline{D}_i = \{\{D_{i,j}\}_{j=1}^{6}, D_i\}$:.

$$
\begin{aligned}
D_{i,1} &= (s_{i,1} \cdot \boldsymbol{X}_{i,1}, s_{i,2}, h_{i,2}, 0^{3l+2}, s_{i,3}, s_{i,4}, 0, 0, 0)_{\mathbb{C}_1},\\
D_{i,2} &= (s_{i,5} \cdot \boldsymbol{X}_{i,2}, s_{i,6}, s_{i,7}, 0^{3l+2}, 0, 0, s_{i,8}, s_{i,9}, 0)_{\mathbb{C}_1^*},\\
D_{i,3} &= (\hat{s}_{i,1} \cdot \boldsymbol{X}_{i,1}, s_{i,6}, \overline{h}_{i,2}, 0^{3l+2}, \hat{s}_{i,2}, \hat{s}_{i,3}, 0, 0, 0)_{\mathbb{C}_2},\\
D_{i,4} &= (\hat{s}_{i,4} \cdot \boldsymbol{X}_{i,2}, s_{i,2}, \hat{s}_{i,5}, 0^{3l+2}, 0, 0, \hat{s}_{i,6}, \hat{s}_{i,7}, 0)_{\mathbb{C}_2^*},\\
D_{i,5} &= (\overline{s}_{i,1} \cdot \boldsymbol{X}_{i,1}, s_{i,7}, \hat{s}_{i,5}, 1, 0^{3l+3}, \overline{s}_{i,2}, \overline{s}_{i,3}, 0, 0, 0)_{\mathbb{C}_3},\\
D_{i,6} &= (\overline{s}_{i,4} \cdot \boldsymbol{X}_{i,2}, h_{i,2}, \overline{h}_{i,2}, \overline{s}_{i,5}, 0^{3l+3}, 0, 0, \overline{s}_{i,6}, \overline{s}_{i,7}, 0)_{\mathbb{C}_3^*},\\
D_i &= e(G_1, G_2)^{x_i + \beta_2 s_{i,7} + \gamma_2 \hat{s}_{i,5} + \overline{s}_{i,5}},
\end{aligned}
$$

where $\{s_{i,j}\}_{j=1}^{9}, \{\hat{s}_{i,j}, \overline{s}_{i,j}\}_{j=1}^{7} \xleftarrow{U} Z_q$.

Then, p_i handles $\overline{C}_i$ and $\overline{D}_i$ by using pu according to the distributed ElGamal cryptosystem, and obtains $E_i = (\mathbf{E}_{pu}[C_i], \{\mathbf{E}_{pu}[C_{i,j}]\}_{j=1}^{6}, \mathbf{E}_{pu}[\alpha_{i,1}])$ and $E_i' = (\mathbf{E}_{pu}[D_i], \{\mathbf{E}_{pu}[D_{i,j}]\}_{j=1}^{6}, \mathbf{E}_{pu}[\alpha_{i,2}])$.

Finally, p_i sends $\{E_i, E_i'\}$ to the proxy.

Blind Ciphertext In this phase, the proxy needs to blind the ciphertexts such that the data aggregator does not learn knowledge about the source of ciphertexts. In fact, communication is very expensive and the participants drop out at any time in many important applications. Thus, denote the index i of the participants p_i that the proxy receives messages by active set $\mathcal{A}$. After receiving messages from $\mathcal{A}$, the proxy performs the blind operation as follows.

It first performs two shuffle operations for the ciphertext $\{E_i\}_{i\in\mathcal{A}}$ and $\{E_i'\}_{i\in\mathcal{A}}$. Specifically, the proxy picks two random permutations π and π' from the set $\mathcal{A}$, and re-encrypts $E = \{E_{\pi(i)}\}_{i\in\mathcal{A}}$ and $E' = \{E'_{\pi(i)'}\}_{i\in\mathcal{A}}$ by using the re-encryption property of distributed ElGamal cryptosystem.

Finally, it sends $\{E, E'\}$ to the data aggregator.

Ciphertext Aggregation In this phase, $\mathcal{A}_{gg}$ computes the over-threshold set-union by using decryption operation. The detailed decryption operation is summarized in Algorithm 1.

More specifically, with the help of the proxy, $\mathcal{A}_{gg}$ decrypts the ciphertexts E and E' and obtains $\{\overline{C}_j, \alpha_{j,1}\}_{j\in\mathcal{A}}$, $\{\overline{D}_j, \alpha_{j,2}\}_{j\in\mathcal{A}}$ (line 1). Next, if $|\mathcal{A}| < t$, the participants' data items do not contain the over-threshold set-union and the data aggregator outputs an empty set ϕ (lines 2-3). Otherwise, they may contain the over-threshold set-union. Any t participants' data items could be the same. $\mathcal{A}_{gg}$ needs to traverse all possible combination of t participants' ciphertexts. Thus, it computes a family of sets $\mathcal{U}_1$ whose elements are all subsets of size t of $\mathcal{A}$ (line 5). Obviously, $|\mathcal{U}_1| = \binom{|\mathcal{A}|}{t}$. Then it randomly chooses an element $\mathcal{E} = \{e_1, ..., e_t\}$ from $\mathcal{A}$ and removes $\mathcal{E}$ from $\mathcal{A}$ and computes B, B' (line 8). Next, $\mathcal{A}_{gg}$ computes $H, F_{i,1}, F_{i,2}, \hat{F}_{i,1}, \hat{F}_{i,2}$ and $\overline{F}_i$ (lines 9). Then, it calculates $\prod_{i\in\mathcal{E}} \left(\frac{F_{i,1}\hat{F}_{i,1}\overline{F}_i}{F_{i,2}\hat{F}_{i,2}}\right)^{\Delta_{\alpha_{i,1},B}(0)}$ and H, and decides whether $\prod_{i\in\mathcal{E}} \left(\frac{F_{i,1}\hat{F}_{i,1}\overline{F}_i}{F_{i,2}\hat{F}_{i,2}}\right)^{\Delta_{\alpha_{i,1},B}(0)} = H$ (line 10).

If the above equation holds, the data aggregator continues to compute $K, J_{i,1}$, $J_{i,2}, \hat{J}_{i,1}, \hat{J}_{i,2}, \overline{J}_i$ and $\hat{U}$ (lines 11-12). To recover U, it needs to compute the discrete logarithm of $\hat{U}$ to the base $e(G_1, G_2)$. By using the Pollard's lambda algorithm [18], this takes expected time $O(\sqrt{M})$ since $x_i \in [1, M]$. Note that if the above equation (line 10) holds, then it means the t ciphertexts contain the same data item which belongs to over-threshold set-union (see Theorem 1). We can obtain the data item by using the above operations.

Finally, to obtain over-threshold set-union, the data aggregator requires at most $\binom{n}{t}$ times of the above computation (lines 6-13), because $\mathcal{U}_1$ contains at most $\binom{n}{t}$ elements when $|\mathcal{A}| = n$.

Algorithm 1: Ciphertext Aggregation: $\mathcal{A}_{gg}$

Input: $\{E_i, \overline{E}_i\}_{i\in\mathcal{A}}, \mathcal{U} \leftarrow \phi$
Output: Over-Threshold Set-Union $\mathcal{U}$

1 with the help of the proxy, $\mathcal{A}_{gg}$ decrypts $\{E_i, \overline{E}_i\}_{i\in\mathcal{A}}$ based on the distributed ElGamal cryptosystem and obtains $\{\overline{C}_j, \alpha_{j,1}\}_{j\in\mathcal{A}}$, $\{\overline{D}_j, \alpha_{j,2}\}_{j\in\mathcal{A}}$;
2 **if** $|\mathcal{A}| < t$ **then**
3 $\quad\mathcal{A}_{gg}$ outputs an empty set ϕ;
4 **else**
5 $\quad\mathcal{A}_{gg}$ computes a family of sets $\mathcal{U}_1$ whose elements are all subsets of size t of $\mathcal{A}$, where $|\mathcal{U}_1| = \binom{|\mathcal{A}|}{t}$;
6 $\quad$**while** $|\mathcal{U}_1| > 0$ **do**
7 $\quad\quad\mathcal{A}_{gg}$ randomly chooses an element $\mathcal{E} = \{e_1, ..., e_t\}$ from $\mathcal{U}_1$;
8 $\quad\quad\mathcal{U}_1 \leftarrow \mathcal{U}_1 - \{\mathcal{E}\}$, $B \leftarrow \{\alpha_{i,1} | i \in \mathcal{E}\}$, $B' \leftarrow \{\alpha_{i,2} | i \in \mathcal{E}\}$;
9 $\quad\quad\mathcal{A}_{gg}$ computes $H = \prod_{j\in\mathcal{E}} C_j$, $F_{i,1} = \prod_{j\in\mathcal{E}} e(C_{i,1}, C_{j,2})$, $F_{i,2} = \prod_{j\in\mathcal{E}} e(C_{i,4}, C_{j,3})$, $\hat{F}_{i,1} = \prod_{j\in\mathcal{E}} e(C_{i,3}, C_{j,4})$, $\hat{F}_{i,2} = \prod_{j\in\mathcal{E}} e(C_{i,2}, C_{j,1})$, $\overline{F}_i = \prod_{j\in\mathcal{E}} e(C_{i,5}, C_{j,6})$;
10 $\quad\quad$**if** $\prod_{i\in\mathcal{E}} \left(\frac{F_{i,1}\hat{F}_{i,1}\overline{F}_i}{F_{i,2}\hat{F}_{i,2}}\right)^{\Delta_{\alpha_{i,1},B}(0)} == H$ **then**
11 $\quad\quad\quad\mathcal{A}_{gg}$ computea $K = \prod_{j\in\mathcal{E}} D_j$, $J_{i,1} = \prod_{j\in\mathcal{E}} e(D_{i,1}, D_{j,2})$, $J_{i,2} = \prod_{j\in\mathcal{E}} e(D_{i,4}, D_{j,3})$, $\hat{J}_{i,1} = \prod_{j\in\mathcal{E}} e(D_{i,3}, D_{j,4})$, $\hat{J}_{i,2} = \prod_{j\in\mathcal{E}} e(D_{i,2}, D_{j,1})$, $\overline{J}_i = \prod_{j\in\mathcal{E}} e(D_{i,5}, D_{j,6})$;
12 $\quad\quad\quad\hat{U} = \left(\prod_{i\in\varepsilon} K_i \left(\frac{J_{i,2}\hat{J}_{i,2}}{J_{i,1}\hat{J}_{i,1}\overline{J}_i}\right)^{\Delta_{\alpha_{i,2},B'}(0)}\right)^{\frac{1}{t}}$, $U = \log \hat{U}$;
13 $\quad\quad\quad\mathcal{U} \leftarrow \mathcal{U} \bigcup U$;
14 **return** $\mathcal{U}$

3.3 Further Improvement

In fact, each p_i generates a new private data item in a certain time interval. If participants use the basic protocol to encrypt their data items in each time

interval, the data aggregator may learn more information than the over-threshold set-union. For example, suppose that p_1 produces the same data item in t time intervals and encrypts it by using the basic protocol, and if the data item does not belong to over-threshold set-union in the t time intervals, the data aggregator can obtain the data item by using the decryption operation. Thus, the basic protocol cannot be applied to this scenario.

To resolve this drawback, we propose our improved algorithm in this section. The main idea is that the ciphertexts can be decrypted only if they are at the same time interval. Specifically, since ciphertexts $C_{i,j}$ and $D_{i,j}$ are computed by using some vectors and random dual orthonormal bases in each time interval, we can use different vectors in different time interval. An efficient method is that we can rearrange the order of the elements in the vectors. For example, $D_{i,1}$ is computed by using vector $(s_{i,1}\cdot \boldsymbol{X}_{i,1}, s_{i,2}, h_{i,2}, 0^{3l+2}, s_{i,3}, s_{i,4}, 0, 0, 0)$, and we use a new vector whose elements are randomly arranged to compute ciphertexts in each time interval. Although the scheme is feasible, there is a disadvantage to this approach. Because the number of elements in vectors is limited. Thus, the number of random permutations is limited. To overcome the weakness, we can add some elements to vectors. In the following, we will illustrate our improved algorithm in detail.

We again run the **System Setup** phase for the trusted authority, but we make the following modifications.

- Add: samples c_i, $d_{i,j} \leftarrow \{0,1\}^\lambda$ $(i = 1, ..., 6, j = 1, ..., 4)$ and sends to all participants;
- Replace $\{(\mathbb{B}_i, \mathbb{B}_i^*), (\mathbb{C}_i, \mathbb{C}_i^*)\}_{i=1}^3$ by $\{(\overline{\mathbb{B}}_i, \overline{\mathbb{B}}_i^*), (\overline{\mathbb{C}}_i, \overline{\mathbb{C}}_i^*)\}_{i=1}^3$: Samples $\{(\overline{\mathbb{B}}_i, \overline{\mathbb{B}}_i^*), (\overline{\mathbb{C}}_i, \overline{\mathbb{C}}_i^*)\}_{i=1}^3 \leftarrow \mathcal{G}_{ob}(1^\lambda, m)$, where $m = 6l + 11 + c_\eta \log n + c_u$;

To rearrange the order of the elements in the vectors in different time intervals, we use a pseudorandom permutation and the Luby-Rackoff construction [19]. Let $\overline{f}_{s_1,s_2,s_3,s_4}()$ be a pseudorandom permutation indexed by s_1, s_2, s_3, s_4 (because it uses four pseudorandom functions), where $\overline{f}_{s_1,s_2,s_3,s_4}() : \{0,1\}^{6l+11+c_\eta \log n + c_u} \times \{0,1\}^\lambda \longrightarrow \{0,1\}^{6l+11+c_\eta \log n + c_u}$, and c_η and c_u are large constants. For a vector (x_1, x_2, x_3), we use positive integers $1, 2, 3$ to label the positions of the elements x_1, x_2, x_3 in the vector from left to right and let $R(x_1, x_2, x_3)$ denote them, i.e., $R(x_1, x_2, x_3) = (1, 2, 3)$. Let $\overline{f}_{s_1,s_2,s_3,s_4}(R(x_1, x_2, x_3))$ denote $\overline{f}_{s_1,s_2,s_3,s_4}(R(x_1, x_2, x_3)) = \overline{f}_{s_1,s_2,s_3,s_4}(1, 2, 3) = (\overline{f}_{s_1,s_2,s_3,s_4}(1), \overline{f}_{s_1,s_2,s_3,s_4}(2), \overline{f}_{s_1,s_2,s_3,s_4}(3))) = (y_1, y_2, y_3))$. Let $\overline{R}(y_1, y_2, y_3)$ denote a new vector which consists of x_1, x_2, x_3 and rearranges the positions of x_1, x_2, x_3 according to y_1, y_2, y_3, respectively. For example, when $(y_1, y_2, y_3) = (3, 2, 1)$, we have $\overline{R}(y_1, y_2, y_3) = (x_3, x_2, x_1)$. For convenience, we use $< k_i(r) >$ to denote $f_{d_{i,1}}(r), f_{d_{i,2}}(r), f_{d_{i,3}}(r), f_{d_{i,4}}(r)$ for $i = 1, ..., 6$.

We again run the **Ciphertext Generation** phase for p_i $(i \in [n])$, but we make the following modifications.

- Replace $\{\{C_{i,j}\}_{j=1}^6, C_i\}$ by $\{\{\widetilde{C}_{i,j}\}_{j=1}^6, \widetilde{C}_i\}$:
  $\widetilde{C}_{i,1} = \overline{R}\Big(\overline{f}_{<k_1(r)>}\big(R(r_{i,1}, h_{i,1}, 0^2, r_{i,2}, r_{i,3}, 0, 0, 0, f_{c_1}(1), ..., f_{c_1}(t'))\big)\Big)_{\overline{\mathbb{B}}_1}$,

$\widetilde{C}_{i,2} = \overline{R}\Big(\overline{f}_{<k_1(r)>}\big(R(r_{i,4}, r_{i,5}, 0^2, 0, 0, r_{i,6}, r_{i,7}, 0, u_{i,1}, ..., u_{i,t'})\big)\Big)_{\overline{\mathbb{B}}_1^*}$,
$\widetilde{C}_{i,3} = \overline{R}\Big(\overline{f}_{<k_2(r)>}\big(R(r_{i,4}, \overline{h}_{i,1}, 0^2, \hat{r}_{i,1}, \hat{r}_{i,2}, 0, 0, 0, f_{c_2}(1), ..., f_{c_2}(t'))\big)\Big)_{\overline{\mathbb{B}}_2}$,
$\widetilde{C}_{i,4} = \overline{R}(\overline{f}_{<k_2(r)>}\big(R(r_{i,1}, \hat{r}_{i,3}, 0^2, 0, 0, \hat{r}_{i,4}, \hat{r}_{i,5}, 0, \hat{u}_{i,1}, ..., \hat{u}_{i,t'})\big)\Big)_{\overline{\mathbb{B}}_2^*}$,
$\widetilde{C}_{i,5} = \overline{R}\Big(\overline{f}_{<k_3(r)>}\big(R(\overline{r}_{i,1} f_{a_1}(r) \cdot \boldsymbol{X}_{i,1}, r_{i,5}, \hat{r}_{i,3}, 1, 0^{3l+3}, \overline{r}_{i,2}, \overline{r}_{i,3}, 0, 0, 0,$
$f_{c_3}(1), ..., f_{c_3}(t''))\big)\Big)_{\overline{\mathbb{B}}_3}$,
$\widetilde{C}_{i,6} = \overline{R}\Big(\overline{f}_{<k_3(r)>}\big(R(f_{b_1}(r) \cdot \boldsymbol{X}_{i,2}, h_{i,1}, \overline{h}_{i,1}, \overline{r}_{i,4}, 0^{3l+3}, 0, 0, \overline{r}_{i,5}, \overline{r}_{i,6}, 0,$
$\overline{u}_{i,1}, ..., \overline{u}_{i,t''})\big)\Big)_{\overline{\mathbb{B}}_3^*}$,
$\widetilde{C}_i = e(G_1, G_2)^{\beta_1 r_{i,5} + \gamma_1 \hat{r}_{i,3} + \overline{r}_{i,4}}$
where $t' = 6l + 2 + c_\eta \log n + c_u$, $t'' = c_\eta \log n + c_u$, and $u_{i,j}$ $\hat{u}_{i,j}$, $\overline{u}_{i,j'} \leftarrow Z_q$ for $j = 1, ..., t' - 1$ and $j' = 1, ..., t'' - 1$, and $\sum_{j=1}^{t'} f_{c_1}(j) u_{i,j} = 0$, $\sum_{j=1}^{t'} f_{c_2}(j) \hat{u}_{i,j} = 0$, $\sum_{j'=1}^{t''} f_{c_3}(j') \overline{u}_{i,j'} = 0$

– Replace $\{\{D_{i,j}\}_{j=1}^6, D_i\}$ by $\{\{\widetilde{D}_{i,j}\}_{j=1}^6, \widetilde{D}_i\}$:
$\widetilde{D}_{i,1} = \overline{R}\Big(\overline{f}_{<k_4(r)>}\big(R(s_{i,1} \cdot \boldsymbol{X}_{i,1}, s_{i,2}, h_{i,2}, 0^{3l+2}, s_{i,3}, s_{i,4}, 0, 0, 0, f_{c_4}(1), ...,$
$f_{c_4}(\hat{t}'))\big)\Big)_{\overline{\mathbb{C}}_1}$,
$\widetilde{D}_{i,2} = \overline{R}\Big(\overline{f}_{<k_4(r)>}(R(s_{i,5} \cdot \boldsymbol{X}_{i,2}, s_{i,6}, s_{i,7}, 0^{3l+2}, 0, 0, s_{i,8}, s_{i,9}, 0, j_{i,1}, ...,$
$j_{i,\hat{t}'}))\Big)_{\overline{\mathbb{C}}_1^*}$,
$\widetilde{D}_{i,3} = \overline{R}\Big(\overline{f}_{f_{<k_5(r)>}}\big(R(\hat{s}_{i,1} \cdot \boldsymbol{X}_{i,1}, s_{i,6}, \overline{h}_{i,2}, 0^{3l+2}, \hat{s}_{i,2}, \hat{s}_{i,3}, 0, 0, 0, f_{c_5}(1), ...,$
$f_{c_5}(\hat{t}'))\big)\Big)_{\overline{\mathbb{C}}_2}$,
$\widetilde{D}_{i,4} = \overline{R}\Big(\overline{f}_{<k_5(r)>}\big(R(\hat{s}_{i,4} \cdot \boldsymbol{X}_{i,2}, s_{i,2}, \hat{s}_{i,5}, 0^{3l+2}, 0, 0, \hat{s}_{i,6}, \hat{s}_{i,7}, 0, \hat{j}_{i,1}, ...,$
$\hat{j}_{i,\hat{t}'})\big)\Big)_{\overline{\mathbb{C}}_2^*}$,
$\widetilde{D}_{i,5} = \overline{R}\Big(\overline{f}_{f_{<k_6(r)>}}\big(R(\overline{s}_{i,1} \cdot \boldsymbol{X}_{i,1}, s_{i,7}, \hat{s}_{i,5}, 1, 0^{3l+3}, \overline{s}_{i,2}, \overline{s}_{i,3}, 0, 0, 0, f_{c_6}(1), ...,$
$f_{c_6}(\hat{t}''))\big)\Big)_{\overline{\mathbb{C}}_3}$,
$\widetilde{D}_{i,6} = \overline{R}\Big(\overline{f}_{<k_6(r)>}\big(R(\overline{s}_{i,4} \cdot \boldsymbol{X}_{i,2}, h_{i,2}, \overline{h}_{i,2}, \overline{s}_{i,5}, 0^{3l+3}, 0, 0, \overline{s}_{i,6}, \overline{s}_{i,7}, 0, \overline{j}_{i,1}, ...,$
$\overline{j}_{i,\hat{t}''})\big)\Big)_{\overline{\mathbb{C}}_3^*}$,
$\widetilde{D}_i = e(G_1, G_2)^{x_i + \beta_2 s_{i,7} + \gamma_2 \hat{s}_{i,5} + \overline{s}_{i,5}}$,
where $\hat{t}' = 2 + c_\eta \log n + c_u$ and $\hat{t}'' = c_\eta \log n + c_u$, and $j_{i,k}$ $\hat{j}_{i,k}$, $\overline{j}_{i,k'} \leftarrow Z_q$ for $k = 1, ..., \hat{t}' - 1$ and $k' = 1, ..., \hat{t}'' - 1$, and $\sum_{k=1}^{\hat{t}'} f_{c_4}(k) j_{i,k} = 0$, $\sum_{k=1}^{\hat{t}'} f_{c_5}(k) \hat{j}_{i,k} = 0$, $\sum_{k'=1}^{\hat{t}''} f_{c_6}(k') \overline{j}_{i,k'} = 0$

Note that we add some elements to the vectors used to compute ciphertexts such that the vectors have the same number of elements. Moreover, the added elements do not affect the result of executed protocol. For $O(\log n)$ ele-

ments, we have that the number of different permutations is $O((\log n)!)$. By using Stirling's approximation, we have $O((\log n)!) \sim O(\sqrt{2\pi \log n}(\frac{\log n}{e})^{\log n})$. On the other hand, $O((\log n)^{\log n}) = O(e^{\log n \log\log n})$. Thus, we have $O((c_\eta \log n)!) = O(e^{(\log\log n - 1)c_\eta \log n}) = O(n^{c_\eta \log\log n})$.

Finally, the proxy and data aggregator again run the **Blind Ciphertext** and **Ciphertext Aggregation** phases to decrypt ciphertexts and do not make any modifications.

4 Protocol Analysis

4.1 Correctness

Theorem 1. *In our protocol, the data aggregator can accurately obtain the over-threshold set-union.*

Proof. We omit here due to limited space. A detailed proof can be found in [20].

4.2 Privacy

We discuss the privacy of our protocol in this section. Since the private data items of participants are encrypted by using the distributed ElGamal encryption scheme, other participants and the proxy cannot decrypt them and obtain the data privacy information based on the semantic security assumption. We only consider whether the aggregator can obtain extra information than what can be deduced from the over-threshold set-union or not.

Theorem 2. *Our protocol is secure under the SXDH and XDLIN assumptions.*

Proof. We omit here due to limited space. A detailed proof can be found in [20].

4.3 Complexity

It is not difficult to see that the communication complexity of **System Setup**, **Ciphertext Generation** and **Blind Ciphertext** of our protocol are $O(n \log^2 n)$, $O(n \log n \log p)$ and $O(n \log n \log p)$, respectively. Thus, the total communication complexity of our protocol is $O(n \log^2 n)$.

5 Conclusions

In this paper, we propose a novel communication-efficient and privacy-preserving protocol for the over-threshold set-union problem. Compared with previous schemes, our protocol has nearly optimal communication complexity and meanwhile can protect data privacy. In future work, we will focus on how to protect the sensitive information of the final aggregation result.

Acknowledgements

This work is supported in part by the National Natural Science Foundation of China Grant No. 61972447.

References

1. Lea Kissner and Dawn Song. Privacy-preserving set operations. In *CRYPTO*, pages 241–257. Springer, 2005.
2. Qinghua Li and Guohong Cao. Efficient and privacy-preserving data aggregation in mobile sensing. In *ICNP*, pages 1–10. IEEE, 2012.
3. Martin Burkhart and Xenofontas Dimitropoulos. Fast privacy-preserving top-k queries using secret sharing. In *ICCCN*, pages 1–7. IEEE, 2010.
4. Zhipeng Cai, Xu Zheng, and Jiguo Yu. A differential-private framework for urban traffic flows estimation via taxi companies. *IEEE Trans. Ind. Inform.*, 15(12):6492–6499, 2019.
5. Zhipeng Cai and Xu Zheng. A private and efficient mechanism for data uploading in smart cyber-physical systems. *IEEE Trans. Netw. Sci. Eng.*, 2018.
6. Zhipeng Cai and Zaobo He. Trading private range counting over big iot data. In *ICDCS*, pages 144–153. IEEE, 2019.
7. Xu Zheng and Zhipeng Cai. Privacy-preserved data sharing towards multiple parties in industrial iots. *IEEE J. Sel. Areas Commun*, 38(5):968–979, 2020.
8. Benny Applebaum, Haakon Ringberg, Michael J Freedman, Matthew Caesar, and Jennifer Rexford. Collaborative, privacy-preserving data aggregation at scale. In *PETS*, pages 56–74. Springer, 2010.
9. Myungsun Kim, Aziz Mohaisen, Jung Hee Cheon, and Yongdae Kim. Private over-threshold aggregation protocols over distributed datasets. *IEEE Trans. Knowl. Data Eng.*, 28(9):2467–2479, 2016.
10. Ji Young Chun, Dong Hoon Lee, and Ik Rae Jeong. Privacy-preserving range set union for rare cases in healthcare data. *IET Commun*, 6(18):3288–3293, 2012.
11. David P Woodruff and Qin Zhang. When distributed computation is communication expensive. *Distrib. Comput.*, 30(5):309–323, 2017.
12. Allison Bishop, Abhishek Jain, and Lucas Kowalczyk. Function-hiding inner product encryption. In *ASIACRYPT*, pages 470–491. Springer, 2015.
13. Masayuki Abe, Melissa Chase, Bernardo David, Markulf Kohlweiss, Ryo Nishimaki, and Miyako Ohkubo. Constant-size structure-preserving signatures: Generic constructions and simple assumptions. *J. Cryptology*, 29(4):833–878, 2016.
14. Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *CRYPTO*, pages 191–208. Springer, 2010.
15. Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.
16. Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. *J. Cryptology*, 20(1):51–83, 2007.
17. Claude Castelluccia, Aldar CF Chan, Einar Mykletun, and Gene Tsudik. Efficient and provably secure aggregation of encrypted data in wireless sensor networks. *ACM TOSN*, 5(3):1–36, 2009.
18. Alfred J Menezes, Jonathan Katz, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 1996.
19. Michael Luby and Charles Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM J. Comput.*, 17(2):373–386, 1988.
20. Xuhui Gong, Qiang-Sheng Hua, and Hai Jin. Communication-efficient and privacy-preserving protocol for computing over-threshold set-union. [Online]. Available: https://qiangshenghua.github.io/papers/otsu-full.pdf.