Efficient Anonymous Message Submission

Xinxin Zhao, Lingjun Li, Guoliang Xue, and Gabriel Silva
Arizona State University

***Abstract*—In online surveys, many people are not willing to provide true answers due to privacy concerns. Thus, anonymity is important for online message collection. Existing solutions let each member blindly shuffle the submitted messages by using the IND-CCA2 secure cryptosystem. In the end, all messages are randomly shuffled and no one knows the message order. However, the heavy computational overhead and linear communication rounds make it only useful for small groups. In this paper, we propose an efficient anonymous message submission protocol aimed at a practical group size. Our protocol is based on a simplified secret sharing scheme and a symmetric key cryptosystem. We propose a novel method to let all members secretly aggregate their messages into a message vector such that a member knows nothing about other members' message positions. We provide a theoretical proof showing that our protocol is anonymous under malicious attacks. We then conduct a thorough analysis of our protocol, showing that our protocol is computationally more efficient than existing solutions and results in a constant communication rounds with a high probability.**

I. Introduction

In the real world, anonymity has always been an important societal issue. As more and more people discover the digital world and find the need for anonymous participation, this issue is becoming increasingly important in this new society. For example, a company wants to collect opinions from its employees about its managers using an online system. However, employees may be scared to reveal their true opinions in fear of retribution from their managers if they were to find that they negatively answered the survey. As another example, consider a health research group who wants to collect individual health information from a survey group. The participants might not willing to provide their health information if their identities will be exposed during the information submission process. These examples show that a mechanism for anonymous message submission is thus necessary and significant.

It is easy to anonymously submit messages in the real world when compared with the same task in a public network. For example, in the real world, the collector just hands out the questionnaires and lets the group members fill them out. When the members submit their questionnaires, they first put them in a ballot box and then shake the box to shuffle the sheets before they submit them to the collector. However, if the collector wants to do it online, it becomes challenging because 1) a secure ballot box, which is actually a third storage facility, is hard to find, and 2) the shaking and shuffling activities are under all members' surveillance, which is difficult to implement in a distributed network.

In [27], Yang *et al.* proposed the first solution for online anonymous message submission. Bickell and Shmatikov revisited this problem and proposed a collusion resistant anonymous data collection protocol in [5]. Recently, Corrigan-Gibbs and Ford improved the protocol in [5] to make it accountable and added the ability to send messages of variable lengths [9]. Anonymity in the above protocols is guaranteed by "shuffling", which is done by the group members themselves. Using the IND-CCA2 secure cryptosystem, existing collusion resistant solutions [5] [9] can only be used in a small and private group, and have linear communication rounds in the group size.

The objective of this work is therefore to find an efficient construction of a collusion resistant anonymous message submission protocol for a group of a practical scale, whose size is usually on a scale as large as ten thousand. Although accountability is a desirable property for anonymous message submission, our work focuses on improving the performance of the anonymous message submission, which can then be extended to be accountable by using standard audit trail technique [9]. We propose a novel technique which secretly aggregates the group members' messages into a message vector so that each member is unaware of other members' message positions. The security primitives employed in our protocol are the simplified secret sharing scheme and a symmetric key cryptosystem, both of which can be efficiently implemented and thus help our protocol fit a practical scale. This work aims to promote the practical application of anonymous message submission, which will then increase the privacy protection and anonymity in the network.

Our contributions are as follows:

1) We propose a novel anonymous position application technique, which can help every group member find a position in a position vector in a manner that every member is oblivious to the positions of the other members. This technique can be done in a few communication rounds with a high probability of success (2 rounds with a 95% success probability).
2) We propose an efficient anonymous message submission protocol for groups with a practical scale. Our protocol runs in polynomial time in the group size and constant communication rounds with a high probability.
3) We prove that our protocol is anonymous and collusion resistant under malicious attacks. The security of our protocol is based on the security of the secret sharing scheme and the symmetric key cryptosystem.
4) We analyze the performance of our protocol from the aspects of security and efficiency. Our simulation results show that our protocol is more efficient than the existing works, especially when the group size is large. More specifically, the computation of the collector in our protocol can achieve linear time complexity compared to the polynomial time complexity in [5].

The rest of our paper is organized as follows. We formulate

Xinxin Zhao, Lingjun Li, Guoliang Xue and Gabriel Silva are all affiliated with Arizona State University, Tempe, AZ 85287. E-mail: {zhao.xinxin, li.lingjun, xue, gsilva}@asu.edu. This research was supported in part by ARO grant W911NF-09-1-0467 and NSF grant 0901451. The information reported here does not reflect the position or the policy of the federal government.

the problem in Section II. In Section III, we present the related work. We present preliminary techniques in Section IV. We demonstrate the protocol construction in Section V. We analyze our protocol from the security and efficiency aspects in Section VI and show our simulation results in Section VII.

II. Problem Formulation

In this section, we present the network model, the threat model, and define the security objectives in the context of anonymous message submission.

A. Network Model

Our model consists of $N+1$ parties: a set $\mathcal{M}$ consisting of N group members, and a collector C who collects all of the group members' messages. The group members submit their messages to the collector without exposing their identities. The initiator of the protocol could be either the collector, when he wants to collect messages, or a group of members, when they want to submit their messages. Note that if a group of members initiate the protocol, this group must consist of at least three members.

Before the protocol execution, all group members must agree on a canonical order of their group IDs, $\{m_1, m_2, \cdots, m_N\}$. We do not assume the existence of a trusted third party during our protocol execution. Our anonymous message submission protocol runs in a completely distributed way. All the communications are carried out in secure channels. A secure channel can be easily set up by using any public key cryptosystem.

B. Threat Model

There are two types of adversaries: *semi-honest* adversaries and *malicious* adversaries [12] [13]. A semi-honest attack happens when the collector colludes with some of the group members. They honestly follow the protocol execution but try to infer some information other than their own inputs and outputs. The malicious adversary can eavesdrop on the communication channel, and perform one of the following actions: modifying or replying honest members' messages, or injecting his own messages.

In this paper, we consider the security of our protocol in the malicious model. Since some members and the collector can refuse to participate in or abort from the protocol permanently, we do not claim that our protocol always finishes with the right result, which is the same as in [5] and [9]. Instead, we prove that the security properties can be preserved when the protocol terminates under malicious attacks.

C. Security Objectives

In general, our work has the same security objectives as in [5], and we restate them here.

- Anonymity: If k $(k \leq N-2)$ out of the N $(N \geq 3)$ group members collude with the collector, the collector cannot infer the identity of the honest members.
- Authenticity: The protocol should guarantee that when the protocol terminates, either the collector receives honest messages from honest members, or he is notified that the honest messages are modified or substituted.
- Confidentiality: If the collector is honest, k $(k \leq N-1)$ dishonest colluding members cannot learn the content of the honest messages.

We formalize our notion of anonymity for an anonymous message submission protocol when k out of the N group members are dishonest by utilizing an "anonymization game" [5]. The anonymization game is played between an adversary and an oracle with a security parameter 1^λ. The adversary plays the roles of the collector and the k dishonest members, while the oracle plays the roles of the honest members. The adversary is assumed to not know the secret positions and secret keys of the honest members. The protocol is anonymous if the adversary can win the game with only a negligible probability. Prior to the game, the adversary chooses plaintext messages for all honest members and gives them to the oracle, who then participates in the anonymous message submission protocol using those messages for the honest members. The adversary may repeat this process polynomial times. Formally, the anonymization game is defined as follows.

1) The adversary chooses two honest members m_α and m_β, and two plaintext messages d_0 and d_1. He also chooses a plaintext message d_{m_i} for each other honest member m_i, and gives these messages to the oracle.
2) The oracle selects a bit $b \in \{0,1\}$ uniformly at random, and sets $d_{m_\alpha} = d_b$ and $d_{m_\beta} = d_{\bar{b}}$, where $\bar{b}$ denotes the negation of b.
3) The oracle participates in the anonymous message submission protocol with the response of honest group member m_i as d_{m_i}. Note that the oracle plays the roles of all the honest group members. The adversary plays the roles of the collector and all dishonest members, and he may deviate arbitrarily from the protocol specification.
4) After observing the protocol process, the adversary guesses whether $b = 0$ or $b = 1$.

Let $\mathcal{D}$ be a probabilistic polynomial time adversary. Then

$$Pr[\mathcal{D}(1^\lambda, m_\alpha, m_\beta, d_0, d_1, 0) = 1]$$

is the probability that $\mathcal{D}$ outputs 1 when $b = 0$, and

$$Pr[\mathcal{D}(1^\lambda, m_\alpha, m_\beta, d_0, d_1, 1) = 1]$$

is the probability that $\mathcal{D}$ outputs 1 when $b = 1$. The adversary's advantage is

$$\begin{aligned} Adv_{\mathcal{D}} = &Pr[\mathcal{D}(1^\lambda, m_\alpha, m_\beta, d_0, d_1, 1) = 1] \\ &- Pr[\mathcal{D}(1^\lambda, m_\alpha, m_\beta, d_0, d_1, 0) = 1]. \end{aligned}$$

Definition 1. *A message submission protocol is anonymous if, for all probabilistic polynomial time adversaries $\mathcal{D}$, the advantage in the anonymization game is a negligible function in λ.* □

Note that this definition is valid only when there are at least two honest group members, since it is impossible for any anonymity to exist when there is only one honest member.

III. Related Work

In this section, we review the most recently works for the anonymous message submission and the related topic of secure multi-party computation(SMC).

The anonymous message submission problem was first introduced by Yang *et al.* [27], who proposed a shuffle technique to solve it. In their protocol, t leaders are chosen out of the N group members. Each member's message is encrypted with all leaders' public keys. Each leader randomly shuffles the messages and proves that the shuffle is correct using a zero-knowledge proof. However, if the last leader colludes with the collector, the anonymity will be violated. In [5], Brickell and Shmatikov proposed a collusion resistant anonymous data collection protocol. Each member in their protocol generates his primary and secondary public/private key pairs. They first encrypt their messages using the collector's public key, then encrypt the ciphertexts using each member's secondary public key, and finally encrypt the ciphertexts using each member's primary public key. Next, the ciphertexts are randomly shuffled by each member who also strips off one layer of the encryption during the shuffle. Finally, the ciphertexts are sent to the collector in random order. The collector can then decrypt the ciphertexts using the secondary private keys sent by the members and his private key. In [9], Corrigan-Gibbs and Ford extended the shuffle protocol and proposed an accountable anonymous message submission protocol, called Dissent. Dissent consists of two protocols: shuffle and bulk. They still use the shuffle technique in [5] in their shuffle protocol. In addition, each member in [9] creates a log file and updates its state during the protocol execution. If at some point the protocol terminates abnormally, all members execute the protocol again according to their log files so that they can find the member who is responsible for the abnormality. The bulk protocol enables the members to send variable length messages. However, there are $2N + 7$ communication rounds and $O(N^2)$ total computations. In our work, we improve the performance of the shuffle protocol.

Other anonymous data submission schemes, such as Mix-networks[7] [24] and DC-nets [8] [15] [22] [25], also achieve a strong anonymity. However, they seem to be a poor match for our scenarios, since the collector may need to know who the group members are. In this case, complicated techniques are needed to enable a well-defined group to submit their messages to the collector, and one or more nodes may be compromised, which breaks the anonymity. Our protocol can provide a strong anonymity even when k $(k \leq N-2)$ out of the N group members are compromised.

SMC was first proposed by Yao [28] and focused on a limited set of problems [11], such as privacy-preserving database query [26] [6], privacy-preserving geometric computation [18] and privacy-preserving data mining [1] [17]. While these problems are interesting, they are different from what we focus on in this paper. Generally speaking, the definition of SMC is that several parties each has an input and then collaborate to compute the function on their inputs without revealing their inputs to the other parties. While in our problem, the group members do want the collector to learn their inputs. As far as we know, the secret sharing scheme is extensively used in SMC [19] [11] [16]. We are the first to use the secret sharing scheme in anonymous message submission.

IV. Technical Preliminaries

A. The Secret Sharing Scheme

The (t, N)-secret sharing ((t, N)-SS) scheme [21] states that if we want a secret s to be shared among N parties, we can divide the secret into N shares and give the i^{th} share $[s]_i^{(t,N)}$ to the i^{th} party. At least t parties are required to reconstruct s. This is a polynomial time algorithm. However, when $t = N$, there is a simplified secret sharing ((N, N)-SS) scheme which can achieve linear time complexity [23]. Suppose the secret $s \in \mathbb{Z}_m$, s is to be shared among N parties. First we secretly choose (independently at random) $N-1$ elements $s_1, s_2, \cdots, s_{N-1}$ from $\mathbb{Z}_m$, compute $s_N = s - s_1 - \cdots - s_{N-1}$, then give s_i to party i. In order to reconstruct s, N parties expose their shares, and compute $s = s_1 + s_2 + \cdots + s_N$. In our paper, we use the simplified (N, N)-SS.

The simplified (N, N)-SS is additive homomorphic [3]. Suppose a_0 and a_1 are two secrets to be shared, then $[a_0 + a_1]_i = [a_0]_i + [a_1]_i$, where $[a_0 + a_1]_i$ is one share of $a_0 + a_1$, $[a_0]_i$ is one share of a_0, and $[a_1]_i$ is one share of a_1.

We require that the (N, N)-SS be *indistinguishable*, i.e., it is impossible to learn any information about a secret from any $N-1$ shares under (N, N)-SS. The indistinguishability of the (N, N)-SS is defined by the following distinguishing game, which is played between an adversary and an oracle.

1) The adversary can split any secret using the (N, N)-SS and perform any number of operations.
2) The adversary submits two distinct secrets a_0 and a_1 to the oracle.
3) The oracle selects a bit $b \in \{0, 1\}$ uniformly at random, then splits $a_b, a_{\bar{b}}$ under (N, N)-SS, and returns any $N-1$ shares of a_b and $a_{\bar{b}}$ specified by the adversary. Without loss of generality, we assume the returned shares are $[a_b]_1, \cdots, [a_b]_{N-1}$ and $[a_{\bar{b}}]_1, \cdots, [a_{\bar{b}}]_{N-1}$, respectively.
4) The adversary is free to perform any number of operations, and, finally, outputs a guess for the value of b.

Let $\mathcal{A}$ be an adversary with unlimited computing power. Then

$$Pr[\mathcal{A}(a_0, a_1, [a_b]_1, \cdots, [a_b]_{N-1}, [a_{\bar{b}}]_1, \cdots, [a_{\bar{b}}]_{N-1}, 0) = 1]$$

is the probability that $\mathcal{A}$ outputs 1 when $b = 0$, and

$$Pr[\mathcal{A}(a_0, a_1, [a_b]_1, \cdots, [a_b]_{N-1}, [a_{\bar{b}}]_1, \cdots, [a_{\bar{b}}]_{N-1}, 1) = 1]$$

is the probability that $\mathcal{A}$ outputs 1 when $b = 1$. The adversary's advantage is

$$Adv_{\mathcal{A}} = Pr[\mathcal{A}(a_0, a_1, [a_b]_1, \cdots, [a_b]_{N-1}, [a_{\bar{b}}]_1, \cdots, [a_{\bar{b}}]_{N-1}, 1) = 1] - Pr[\mathcal{A}(a_0, a_1, [a_b]_1, \cdots, [a_b]_{N-1}, [a_{\bar{b}}]_1, \cdots, [a_{\bar{b}}]_{N-1}, 0) = 1].$$

Definition 2. *A (N, N)-secret sharing scheme is said to be unconditionally indistinguishable if for any two secrets, a_0, a_1 which are shared among N parties under the (N, N)-secret sharing scheme, the adversary $\mathcal{A}$'s advantage in the distinguishing game is 0.* □

Theorem 1. *The simplified (N, N)-secret sharing scheme is unconditionally indistinguishable.* □

TABLE I
MAIN NOTATIONS

$\mathcal{M}; m_1, \cdots, m_N$	group member set; N members in $\mathcal{M}$
C; $\|\|$	the data collector; concatenation
$\mathbb{Z}_m$	the underlying group in the secret sharing scheme
$d_i(1 \le i \le N)$; l	m_i's message; message length
$e_i(1 \le i \le N)$; r	m_i's ciphertext; ciphertext length
$k_i(1 \le i \le N)$; h	m_i's secret key; the key length
$\mathbf{p}'$; M	the auxiliary vector; the vector size
$p'_i(1 \le i \le M)$	the element in $\mathbf{p}'$
$\mathbf{p_i}(1 \le \mathbf{i} \le N)$	m_i's individual position vector
$[p_{ij}]_l(1 \le l \le N)$	m_i's share of p_{ij} under (N, N)-SS
$\mathbf{e}; \mathbf{k}$	the message vector; the key vector
$[\mathbf{e}]_i(1 \le i \le N)$	m_i's share of the vector $\mathbf{e}$
$\mathbf{e_i}; \mathbf{k_i}(1 \le \mathbf{i} \le N)$	m_i's individual message vector and key vector

Proof: Suppose $\mathcal{A}$ gets $N-1$ shares $[a_0]_1, \cdots, [a_0]_{N-1}$ of a secret $a_0 \in \mathbb{Z}_m$. Then for each $a'_0 \in \mathbb{Z}_m$, there is a unique $[a'_0]_N \in \mathbb{Z}_m$ such that $[a_0]_1 + \cdots + [a_0]_{N-1} + [a'_0]_N = a'_0$. Since a'_0 is distributed uniformly over $\mathbb{Z}_m$, the construction of a'_0 is equally likely. Thus, $[a_0]_1, \cdots, [a_0]_{N-1}$ could be $N-1$ shares of any number $a'_0 \in \mathbb{Z}_m$ with equal probability. It's the same when the adversary gets any $N-1$ shares of a secret $a_1 \in \mathbb{Z}_m$. We conclude that the distribution of the $N-1$ shares of a_0 and a_1 are identical. Thus $Adv_{\mathcal{A}} = 0$. ■

B. The Symmetric Key Cryptosystem

In this paper, rather than using the public key cryptosystem, we use a symmetric key cryptosystem [23] for our encryption and decryption. First, for the same level of security, the symmetric key cryptosystem requires a smaller secret key length. Furthermore, the encryption and decryption of the symmetric key cryptosystem is much faster. We require that the cryptosystem be semantically secure [14] in our paper. Fortunately, AES-128 under the cipher block chaining (CBC) counter (CTR) mode meets this requirement [2].

V. CONSTRUCTION OF OUR PROTOCOL

There are five phases in our protocol, which are the application, encryption, anonymization, verification and decryption. Note that before the protocol execution, each member m_i should generate his secret key k_i. Please refer to Table I for the main notations in our paper .

A. Protocol Overview

- **Application:** In the application phase, there is a position vector $\mathbf{p}$ containing N elements. Each element in $\mathbf{p}$ has a length of $\lceil \log N \rceil$ bits. Every member m_i applies his position L_i in the position vector $\mathbf{p}$, such that L_i is only known to m_i. For any two positions L_i and L_j, they are not equal if and only if $i \neq j$.
- **Encryption:** Every member m_i encrypts his message d_i and gets $e_i = E_{k_i}(d_i)$ with a length of r bits. In our protocol, we only deal with messages of the same length. In order to submit variable length messages, we can use the bulk protocol proposed in [9].
- **Anonymization:** All members secretly construct N shares of a data vector $\mathbf{e}$ with N elements such that e_i $(1 \le i \le N)$ appears in L_i $(1 \le i \le N)$, and send them to the collector C.
- **Verification:** The collector C reconstructs $\mathbf{e}$, and sends $\mathbf{e}$ back to all members. All members check whether their ciphertexts appear in $\mathbf{e}$ or not. If any member's message has been modified or substituted, he broadcasts an alarm message, and then all members abort the protocol. Otherwise, all members send the secret keys k_i to C by secretly constructing a key vector $\mathbf{k}$ with k_i $(1 \le i \le N)$ appearing in L_i $(1 \le i \le N)$.
- **Decryption:** C retrieves the secret keys from $\mathbf{k}$ and decrypts each ciphertext e_i.

B. The Data Aggregation Primitive

The main technique in our paper is to secretly aggregate elements held by the members in a vector $\mathbf{v}$ such that each member has one share of $\mathbf{v}$ and the identity of each member is not revealed during the message submission process. Suppose $\mathbf{v} = [v_{L_1}, v_{L_2}, \cdots, v_{L_N}]$. Each member m_i has an element v_{L_i} such that the position L_i is only known to m_i. Here, $L_1, L_2, \cdots, L_N$ is a permutation of $\{1, 2, \cdots, N\}$. The group members collaboratively compute their shares of $\mathbf{v}$ without revealing their positions. In order to do this, each member m_i constructs an individual vector $\mathbf{v_i}$ with v_{L_i} in L_i and 0 otherwise, and then shares each element of $\mathbf{v_i}$ with all the other members using the (N, N)-SS. In this way, each member m_i receives $N-1$ shares from the other members. Each member m_i sums up the received $N-1$ shares and the share kept by himself. Since the (N, N)-SS is additive homomorphic, the summation is a share of $\mathbf{v}$.

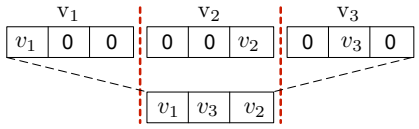


Fig. 1. Data aggregation

Fig. 1 shows an example of the data aggregation. There are three members m_1, m_2, and m_3. Each member m_i has an element v_i and knows v_i's position L_i in $\mathbf{v}$. And the position information is private. For example, m_1's position L_1 is equal to 1 and is only known to m_1. Each member m_i constructs an individual vector $\mathbf{v_i}$ and put v_i in L_i and 0 otherwise. After that, each member m_i shares each element of $\mathbf{v_i}$ with all other members, using $(3, 3)$-SS. In this way, each member gets two shares from the other members. Each member sums up the two received shares and the share kept by himself, and gets a share of $\mathbf{v}$. In this scenario, 3 members are needed to reconstruct $\mathbf{v}$.

C. Construction of Our Protocol

In this section, we present the details of each phase in our protocol.

There are two rounds in the application phase. In Phase 1-Round 1, each member randomly selects his position in the position vector $\mathbf{p}$ and keeps the position secret. Since each member does not know his peer-members' choices, there most likely exist collisions among the members. In order to solve this problem, we introduce an auxiliary vector $\mathbf{p}'$ with M elements. To guarantee no collisions (with a high probability) among N members, we define M be the larger of the two values $361 + N$ and $2N^2 - 2N$, and the success (finding a non-collision $\mathbf{p}$) probability at the end of the application phase is more than 95% (refer to the proof in section VI-B).

Phase 1: Application
Round 1:
Each member $m_i (1 \leq i \leq N)$ executes the following:

1) Initially, m_i has an individual vector $\mathbf{p_i}$ with M elements, where $M = max\{361 + N, 2N^2 - 2N\}$.
2) m_i randomly choose an element p_{ij} in $\mathbf{p_i}$, sets $p_{ij} = 1$, and shares each element $p_{ij} (1 \leq j \leq M)$ among all the members by giving $[p_{i1}]_l || [p_{i2}]_l || \cdots || [p_{iM}]_l$ to m_l, under (N, N)-SS.
3) Upon receiving $[p_{11}]_i || \cdots || [p_{1M}]_i$, $\cdots$, $[p_{(i-1)1}]_i || \cdots || [p_{(i-1)M}]_i$, $[p_{(i+1)1}]_i || \cdots || [p_{(i+1)M}]_i$, $\cdots$, $[p_{N1}]_i || \cdots || [p_{NM}]_i$ from the other members, m_i locally computes $[\mathbf{p'}]_i = \sum_{a=1}^{N} [p_{a1}]_i || \sum_{a=1}^{N} [p_{a2}]_i || \cdots || \sum_{a=1}^{N} [p_{aM}]_i$, and sends $[\mathbf{p'}]_i$ to all other $N-1$ members.
4) Upon receiving $[\mathbf{p'}]_1, \cdots, [\mathbf{p'}]_{i-1}, [\mathbf{p'}]_{i+1}, \cdots, [\mathbf{p'}]_N$ from all the other group members, m_i locally reconstructs $\mathbf{p'} = \sum_{b=1}^{N} \sum_{a=1}^{N} [p_{a1}]_b || \cdots || \sum_{b=1}^{N} \sum_{a=1}^{N} [p_{aM}]_b$.
5) m_i checks the number of collisions in $\mathbf{p'}$. If there are some elements in $\mathbf{p'}$ such that $\sum_{p'_i > 1} p'_i > 6$, they repeat round 1 again. If there are elements in $\mathbf{p'}$ such that $\sum_{p'_i > 1} p'_i \leq 6$, they go to round 2. Otherwise, they go to Step 10.

Round 2:
Each member $m_i (1 \leq i \leq N)$ executes the following:

6) In vector $\mathbf{p'}$, if the value on position L_i is 1, then m_i does not change his position in his individual vector $\mathbf{p_i}$, otherwise m_i randomly chooses another position except those that have not been occupied, and sets $p_{ij} = 1$.
7) m_i shares each element in $\mathbf{p_i}$ among all other group members, under (N, N)-SS.
8) Repeating Step 3 to Step 4 in Round 1, m_i can reconstruct the vector $\mathbf{p'}$.
9) m_i checks the number of collisions in $\mathbf{p'}$. If there is any element in $\mathbf{p'}$ such that $p'_i \neq 0, 1$, they repeat Phase 1 again, otherwise they go to the next step.
10) m_i removes all 0 elements in $\mathbf{p'}$ and gets the position vector $\mathbf{p}$. m_i's position in $\mathbf{p}$ is $L_i = \sum_{c=1}^{j} p'_c$.

Fig. 2. Application Phase

Fig. 2 shows the application phase. Initially, every member m_i has an individual vector $\mathbf{p_i}$ initialized to be 0. The size of $\mathbf{p_i}$ is the same as that of $\mathbf{p'}$. Every member m_i randomly picks an element p_{ij} in $\mathbf{p_i}$ and sets this element to be 1. Thus, the position m_i picks is $L'_i = j$. Then each member m_i shares each element in $\mathbf{p'}$ with the other $N-1$ members using the (N, N)-SS. Utilizing the additive homomorphic property of the (N, N)-SS, m_i gets one share of $\mathbf{p'}$ by computing $[\mathbf{p'}]_i = \sum_{a=1}^{N} [p_{a1}]_i || \cdots || \sum_{a=1}^{N} [p_{aM}]_i$. Then each member m_i sends his share to the other group members. Since each member m_i has N shares of $\mathbf{p'}$, he can reconstruct $\mathbf{p'}$. In $\mathbf{p'}$, any element p'_i with a value greater than 1 indicates that there must exist at least a two members collision at position i with the value of p'_i determining the number of collisions. Thus, each member immediately knows if somebody conflicts with himself. There are three cases for the number of collisions:

- If $\sum_{p'_i > 1} p'_i = 0$, all members directly go to Step 10.
- If $\sum_{p'_i > 1} p'_i > 6$, all members repeat round 1 again.
- If $\sum_{p'_i > 1} p'_i \leq 6$, all members go to round 2.

In round 2, the members who do not collide with others do not change their positions. While other members randomly pick their positions again except those that have been occupied. Then all the members execute the secret sharing again in Steps 7 and 8, and get the auxiliary vector $\mathbf{p'}$. If no collisions occur, all elements in $\mathbf{p'}$ should be either 1 or 0. Each member removes the 0 elements in $\mathbf{p'}$, getting the position vector $\mathbf{p}$, and computes his position in $\mathbf{p}$, $L_i = \sum_{d=1}^{j} p'_d$, where p'_d is the element in the non-collision vector $\mathbf{p'}$ and j is m_i's position in $\mathbf{p_i}$. Note that in Phase 1-Round 1, m is an integer of $\lceil \log N \rceil$ bits. While in Phase 1-Round 2, m is an integer of 3 bits.

Now, one might think of a jamming attack, in which an adversary always fills every element of $\mathbf{p'}$ such that Phase 1 cannot be ended with a non-collision vector. Such a jamming attack will be detected by the members if they run Phase 1 for an unreasonable amount of time. Using the standard audit trail technique [9], the adversary can be easily traced. However, we will not further discuss this in our work because we focus on the anonymity issues. The adversary in our paper is assumed to be more interested in the members' identities than jamming the protocol.

Phase 2: Encryption
Each member $m_i (1 \leq i \leq N)$ executes the following:

1) m_i encrypts d_i using a symmetric key encryption scheme, and gets $e_i = E_{k_i}(d_i)$.

Fig. 3. Encryption phase

Fig. 3 shows the encryption phase. In the encryption phase, each member m_i encrypts his message d_i using a symmetric key cryptosystem. The ciphertext $e_i = E_{k_i}(d_i)$.

Phase 3: Anonymization
Each member $m_i (1 \leq i \leq N)$ executes the following:

1) m_i constructs an individual vector $\mathbf{e_i}$ with the e_i in L_i and 0 otherwise.
2) m_i shares each element e_{ij} in $\mathbf{e_i}$ among all other group members, by giving $[e_{i1}]_l || \cdots || [e_{iN}]_l$ to m_l, under (N, N)-SS.
3) Upon receiving $[e_{11}]_i || \cdots || [e_{1N}]_i$, $\cdots$, $[e_{(i-1)1}]_i || \cdots || [e_{(i-1)N}]_i$, $[e_{(i+1)1}]_i || \cdots || [e_{(i+1)N}]_i$, $\cdots$, $[e_{N1}]_i || \cdots || [e_{NN}]_i$ from the other members, m_i locally computes $[\mathbf{e}]_i = \sum_{a=1}^{N} [e_{a1}]_i || \sum_{a=1}^{N} [e_{a2}]_i || \cdots || \sum_{a=1}^{N} [e_{aN}]_i$, and sends $[\mathbf{e}]_i$ to the data collector C.

Fig. 4. Anonymization phase

Fig. 4 shows the anonymization phase. In this phase, all members securely construct shares of the message vector $\mathbf{e}$ with e_i in L_i. To do this, first every member m_i locally constructs an individual vector $\mathbf{e_i}$ with e_i in L_i and 0 otherwise. Then every member m_i shares each element in $\mathbf{e_i}$ with all the other members using (N, N)-SS. Thus, each member m_i receives $N-1$ shares from the other members. Every member m_i sums up the $N-1$ received shares together with the share kept by himself, and gets one share of $\mathbf{e}$. Then all members send their shares $[\mathbf{e}]_1, [\mathbf{e}]_2, \cdots, [\mathbf{e}]_N$ to the collector C. Note that in this phase, m is an integer of r bits.

Fig. 5 shows the verification phase. When C receives all the shares of $\mathbf{e}$, he can reconstruct $\mathbf{e}$. Then C sends $\mathbf{e}$ to all the members. All members check whether their ciphertexts are in $\mathbf{e}$ or not, by simply comparing their ciphertexts e_i with the one in the position L_i. If some ciphertext is not in $\mathbf{e}$, this member broadcasts an alarm message $b = \perp$. If there is a $b = \perp$ message, the protocol is aborted. Otherwise, the members secretly construct the key vector $\mathbf{k}$ with N elements such that k_i is in the position L_i. For this purpose, each member m_i constructs an individual vector $\mathbf{k_i}$ with k_i in L_i and 0 otherwise. The size of $\mathbf{k_i}$ is the same as that $\mathbf{k}$. Then each member m_i shares each element in $\mathbf{k_i}$ with all other members using the (N, N)-SS. Each member adds up the $N-1$ received shares together with the share kept by himself, and gets one

Phase 4: Verification
The collector C executes the following:

1) Upon receiving $[\mathbf{e}]_1, [\mathbf{e}]_2, \cdots, [\mathbf{e}]_N$ from all group members, C locally reconstructs $\mathbf{e} = \sum_{b=1}^{N}\sum_{a=1}^{N}[e_{a1}]_b || \cdots || \sum_{b=1}^{N}\sum_{a=1}^{N}[e_{aN}]_b$.
2) C sends $\mathbf{e}$ to all the group members.

Each member $m_i (1 \leq i \leq N)$ executes the following:

3) m_i checks if his ciphertext is in $\mathbf{e}$. If e_i does not appear in $\mathbf{e}$, m_i broadcasts a message $b = \perp$, the protocol is aborted if there is any $b = \perp$ message, otherwise, m_i constructs an individual vector $\mathbf{k_i}$ with the one on position L_i to be k_i and the other elements to be 0,
4) m_i shares each element among all other group members, by giving $[k_{i1}]_l || [k_{i2}]_l || \cdots || [k_{iN}]_l$ to m_l, under (N, N)-SS.
5) Upon receiving $[k_{11}]_i || \cdots || [k_{1N}]_i$, $\cdots$, $[k_{(i-1)1}]_i || \cdots || [k_{(i-1)N}]_i$, $[k_{(i+1)1}]_i || \cdots || [k_{(i+1)N}]_i$, $\cdots$, $[k_{N1}]_i || \cdots || [k_{NN}]_i$ from all other group members, m_i locally computes $[\mathbf{k}]_i = \sum_{a=1}^{N}[k_{a1}]_i || \sum_{a=1}^{N}[k_{a2}]_i || \cdots || \sum_{a=1}^{N}[k_{aN}]_i$, and sends $[\mathbf{k}]_i$ to C.

Fig. 5. Verification phase

share of $\mathbf{k}$ in Step 5. All members send their shares to C. Note that in this phase, m is an integer of h bits.

Phase 5: Decryption
The collector C executes the following:

1) Upon receiving $[\mathbf{k}]_1, [\mathbf{k}]_2, \cdots, [\mathbf{k}]_N$ from all the group members, C locally reconstructs $\mathbf{k} = \sum_{b=1}^{N}\sum_{a=1}^{N}[k_{a1}]_b || \cdots || \sum_{b=1}^{N}\sum_{a=1}^{N}[k_{aN}]_b$.
2) C retrieves each secret key k_i from $\mathbf{k}$ to decrypt each ciphertext e_i.

Fig. 6. Decryption phase

Fig. 6 shows the decryption phase. When C receives all shares $[\mathbf{k}]_1, [\mathbf{k}]_2, \cdots, [\mathbf{k}]_N$ of $\mathbf{k}$, he locally reconstructs $\mathbf{k}$. Then he retrieves each secret key k_i from $\mathbf{k}$ and decrypts each ciphertext e_i.

VI. Performance Analysis

In this section, we analyze the performance of our protocol from two aspects: security and efficiency.

A. Security

In this section, we prove that the security properties defined in section II-C can be preserved under malicious attacks.

1) Anonymity: We prove the anonymity of our protocol in two aspects. First, we prove that if the collector and some members behave dishonestly and learn some associations between the identities and the ciphertexts, they cannot pass the verification phase and thus they cannot learn the plaintexts. Second, we prove that if the collector and the dishonest members behave honestly, they pass the verification phase and learn the final decrypted plaintexts, but they will not learn the associations between the identities and the plaintexts.

Theorem 2. *In the anonymous message submission protocol, if the collector colludes with k ($k \leq N-2$) group members, the collector has only a negligible probability to get the associations between the messages and the identities of the group members.* □

Proof: Our proof is done in two parts. First, we show that in the verification phase, either there is exactly one copy of the ciphertext for each honest member, or the deviation from the protocol could be detected by the members before the collector gets the secret keys. Second, we show that an adversary who can win the anonymization game while maintaining the security properties can also win the distinguishing game, which is a contradiction because the (N, N)-SS is unconditionally indistinguishable and the underlying encryption scheme is semantically secure.

Part 1: The honest members' messages cannot appear more than once due to the indistinguishable property of the (N, N)-SS. If the adversary can reproduce the honest members' messages, it means that the adversary can break the (N, N)-SS scheme from less than N shares, which is a contradiction to the security of the (N, N)-SS.

Now we show that if the adversary modifies the honest members' messages, this modification will be detected in the verification phase, since the honest members do not find their ciphertexts in the message vector. Hence, the protocol is aborted and the adversary cannot get the secret keys of the ciphertexts. In this case it is infeasible for the adversary to learn the plaintexts without the secret keys since the underlying encryption scheme is semantically secure.

Part 2: Now we suppose that the adversary honestly handles all ciphertexts belonging to the honest members. If there is a probabilistic polynomial time algorithm $\mathcal{D}$ that allows this adversary to win the anonymization game with a non-negligible probability, we show how to use $\mathcal{D}$ as a subroutine to the algorithm $\mathcal{A}$ that wins the distinguishing game with a non-negligible probability. Because the underlying (N, N)-SS is unconditionally indistinguishable, this is a contradiction, and we conclude that no such $\mathcal{D}$ exists.

Let the set of $N-k$ honest group members in the anonymization game be $\mathcal{K} = m_1, \cdots, m_{N-k}$. Let $\mathcal{D}$ be an algorithm that allows the adversary to win the anonymization game with a non-negligible probability. Then there exist honest group members m_α and m_β such that for a negligible function in λ, $\varepsilon(\lambda)$, the advantage

$$Adv_{\mathcal{D}} > \varepsilon(\lambda).$$

To apply $\mathcal{D}$, $\mathcal{A}$ must simulate the oracle in the anonymization game to reproduce the view of the adversary. We show how $\mathcal{A}$ is able to do this.

Algorithm $\mathcal{A}$ begins by applying $\mathcal{D}$ to learn its choice in step 1 of the anonymization game. $\mathcal{A}$ therefore learns the following:

- Two honest participates m_α and m_β, and two plaintext messages d_0 and d_1.
- A message d_{m_i} for each other honest member m_i.

Then, for each honest member m_i, $\mathcal{A}$ chooses

- k_i, a secret key.

$\mathcal{A}$ selects plaintexts d_0 and d_1 to be the plaintext messages for m_α and m_β, respectively.

$\mathcal{A}$ is now ready to play the role of the oracle in the anonymization game by simulating the messages of the honest members in the protocol execution. For each phase of the protocol, we explain how $\mathcal{A}$ is able to reproduce the messages sent in that phase.

Phase 1: $\mathcal{A}$ has all the necessary information to reproduce this phase exactly.

Phase 2: For all honest members other than m_α and m_β, $\mathcal{A}$ encrypts d_{m_i} using k_i which results in the ciphertext e_{m_i}. $\mathcal{A}$

then encrypts d_0 and d_1 using the keys k_0 and k_1, respectively, getting $e_0 = E_{k_0}(d_0)$ and $e_1 = E_{k_1}(d_1)$.

Phase 3: $\mathcal{A}$ gives e_0 and e_1 to the distinguishing game oracle, getting back $[e_b]_1, \cdots, [e_b]_{N-1}$ and $[e_{\bar{b}}]_1, \cdots, [e_{\bar{b}}]_{N-1}$.

$\mathcal{A}$ now constructs $N-1$ shares for m_α and m_β. Suppose in the application phase, m_α and m_β obtain positions L_α and L_β respectively. $\mathcal{A}$ then constructs $N-1$ shares of the vector $\mathbf{e}_\alpha$ (resp. $\mathbf{e}_\beta$) with the shares of e_b (resp. $e_{\bar{b}}$) in L_α (resp. L_β) and the shares of 0 in the other positions, and sends out these shares on behalf of m_α (resp. m_β).

At the end of Phase 3, $\mathcal{A}$ needs to compute a share of the message vector $\mathbf{e}$. $\mathcal{A}$ does not have the N^{th} share of e_b and $e_{\bar{b}}$. But he has the original ciphertexts e_0 and e_1. With any $N-1$ shares and e_0 (resp. e_1), $\mathcal{A}$ could compute the last share such that all shares are reconstructed as e_0 (resp. e_1). At this moment, $\mathcal{A}$ computes the last share for m_α (resp. m_β) such that either e_0 or e_1 appears in the position L_α (resp. L_β). Because all honest group members' positions are chosen randomly and the position sequence can be viewed as a random permutation on the set $\{1, \cdots, N\}$, so e_0 appears in L_α and e_1 appears in L_β or e_1 appears in L_α and e_0 appears in L_β does not change $\mathcal{D}$'s view. Therefore, $\mathcal{A}$ computes two last shares to let either e_0 (resp. e_1) appear in L_α (resp. L_β) or e_1 (resp. e_0) appear in L_α (resp. L_β) and sends them to the collector.

Phase 4 and Phase 5: $\mathcal{A}$ has all the necessary information to complete it.

Now $\mathcal{A}$ simulates the view of the adversary and applies $\mathcal{D}$ to the view. If $\mathcal{D}$ outputs 1, then $\mathcal{A}$ outputs 1, and if $\mathcal{D}$ outputs 0, $\mathcal{A}$ outputs 0.

We now analyze the probability of $\mathcal{A}$ outputting 1 if the distinguishing oracle chose $b = 0$ and if the distinguishing oracle chose $b = 1$. If $b = 0$, then the view of the adversary is $(m_\alpha, m_\beta, d_0, d_1, 0)$. If $b = 1$, then the view of the adversary is $(m_\alpha, m_\beta, d_0, d_1, 1)$. Based on our assumption that $\mathcal{D}$ wins the anonymization game, we have that $Adv_{\mathcal{D}} > \varepsilon(\lambda)$. Now we make a simple substitution,

$$Adv_{\mathcal{A}} = Adv_{\mathcal{D}} > \varepsilon(\lambda).$$

We conclude that $\mathcal{A}$ can win the distinguishing game with a non-negligible probability, which contradicts with the unconditionally indistinguishable property of the (N, N)-SS. ■

2) Authentication: The honest members verify whether their ciphertexts appear in the message vector $\mathbf{e}$ in the verification phase. If the verification fails, in which case the honest members' ciphertexts have been modified or substituted, the protocol is aborted. So the adversary cannot get the secret keys. If the verification succeeds, the honest members send their secret keys to the collector who can then decrypt the ciphertexts. We state that the authentication defined in section II-C can be preserved.

3) Confidentiality: In Phase 3, all members send out $N-1$ shares of the individual vector to the other members except for one share. Because the (N, N)-SS is unconditionally indistinguishable, even if $N-1$ group members collude, they will not get any useful information about the honest messages. Thus, we conclude that the confidentiality defined in section II-C can be preserved.

B. Efficiency

In this section, we analyze the success probability of finding a non-collision vector in Phase 1, and the computational overhead and communication rounds compared to existing works.

1) The Success Probability in Phase 1:

Theorem 3. *In the application phase, at the end of round 2, the success probability of finding a non-collision vector* $\mathbf{p}$ *is greater than 95%.* □

Proof: Let η_{ij} denote the event that member m_i and m_j choose the same position, which is a collision. Let ϕ_i^w denote the event that member m_i chooses the position $w (1 \le w \le M)$. Because all members choose their positions independently at random, the probability $p(\phi_i^w) = 1/M$. Using Bayes' theorem, we have

$$p[\eta_{ij}] = \sum_{w=1}^{M} p[\phi_i^w | \phi_j^w] p[\phi_j^w] = \sum_{w=1}^{M} \frac{1}{M} p[\phi_j^w] = \frac{1}{M}.$$

We define an indicator random variable X_{ij} such that $X_{ij} = 1$ means the event η_{ij} happens and $X_{ij} = 0$ means η_{ij} does not happen. Note that since X_{ij} only takes the value 0 or 1, it is a Bernoulli variable. We also define X to be the number of collisions, i.e. $X = \sum_{i<j} X_{ij}$. Then we compute

$$E[X] = E[\sum_{i<j} X_{ij}] = \sum_{i<j} E[X_{ij}]$$
$$= \sum_{i<j} p[X_{ij} = 1] = \sum_{i<j} \frac{1}{M} = \frac{1}{M} \binom{N}{2}.$$

Assume that $E[X] = \mu$, from the above equation, we get $M = N(N-1)/2\mu$. Let $\mu = 1/4$, we have $M = 2N^2 - 2N$. Utilizing the Chernoff bound, we have

$$p[X \ge (1+\delta)\mu] \le (\frac{e^\delta}{(1+\delta)^{1+\delta}})^\mu$$

for any $\delta > 0$. Let $\delta = 11$, we get $(1+\delta)\mu = 3$ and $p[X \ge 3] \le (\frac{e^{11}}{(1+11)^{1+11}})^{0.25} \approx 0.009$. Hence, at the end of Phase 1-Round 1, the probability that there exist no less than 6 collisions is not greater than 0.9%. In Phase 1-Round 2, since there are at most 6 members who collide with each other, we need to have enough positions for the colliding members to choose from. Suppose the positions left at the end of round 1 is M_l, the probability that 6 members' positions are not collide is $p = \frac{M_l(M_l-1)(M_l-2)\cdots(M_l-5)}{M_l^6}$. We let $p > 0.96$, and get the smallest integer which is 361. So the lower bound of M is $361 + N$. We define ϱ to be the event that the members find a non-collision vector in Phase 1. $p[\varrho] \ge p[$succeed in round $1 \wedge$ succeed in round $2] \ge (1 - 0.009) \times 0.96 = 0.95136$. The statement is proved. ■

We define the total number of execution times of Phase 1 to be T. Then the expectation $E[T] = \sum_{i=1}^{\infty} i(1-p[\varrho])^{i-1} p[\varrho] = 1/p[\varrho]$. Since $p[\varrho] > 95\%$, so the average execution times are $\lceil 1/0.95 \rceil = 2$.

2) Communication Rounds: All our phases are parallelizable. In Phase 1, the number of communication rounds is 4. There is no communication in Phase 2. Phase 3 needs 2 communication rounds. Phase 4 needs 4 communication rounds. Finally, there is no communication in Phase 5. The expected total communication rounds is $4/p[\varrho] + 6$, which is approximately 10 and independent of the group size. While in Brickell and Shmatikov's [5] protocol, the total communication rounds is $2N + 7$. We recall that our protocol is probabilistic, while the protocol in [5] is deterministic.

In Phase 1-Round 1, the communication bits are $2N(N-1)M\lceil \log N \rceil$. In Phase 1-Round 2, the communication bits are $6N(N-1)M$. In Phase 3, the total communication bits are $2N^2(N-1)r$. In Phase 4, the collector sends out N^2r bits. There are N bits alarm messages, and $2N^2(N-1)h$ bits for the (N, N)-SS. The expected total communication bits are $N(N-1)M(2.1 * \lceil \log N \rceil + 6.32) + 2N^2(N-1)r + N^2r + N + 2N^2(N-1)h$. The total communication bits in [5] are $3SN^2+SN+(Q+T)N^2+(Q+T)N+QN^2$, where S is the ciphertext length using the IND-CCA2 secure cryptosystem, T is key length, and Q is the signature length.

3) Computational Efficiency: In Phase 1-Round 1, there are $3(N-1)M\ \lceil \log N \rceil$-bit additions for each member. In Phase 1-Round 2, there are approximately $3(N-1)M$ 3-bit additions for each member. In Phase 2, each member does 1 encryption. In Phase 3, there are $2(N-1)N$ r-bit additions for each member. In Phase 4, there are $(N-1)N$ r-bit additions for the collector and $2(N-1)N$ h-bit additions for each member. In Phase 5, there are $(N-1)N$ h-bit additions and N decryption for the collector. The expected total bit operations for each member are $(3(N-1)M\lceil \log N \rceil + 9(N-1)M) * 1.05 + 2N(N-1)(r+h) + \mathsf{Enc}$, where Enc is the bit operations for the symmetric key encryption. If we use the Cramer-Shoup cryptosystem (the most efficient IND-CCA2 secure cryptosystem in the standard model known to us), the computational overhead of the shuffle protocol in [5] on each member is $(10 + 15N)\mathsf{Exp}_k + (12N + 5)\mathsf{Mult}_k + (3N + 1)\mathsf{Hash} + 2\mathsf{Sign} + 2(N-1)\mathsf{Veri}$, where Exp_k, Mult_k, Hash, Sign and Veri are the computations of k-bit exponentiation, k-bit multiplication, the cryptographic hash, the signing and verifying function, respectively. k is the bit length of the underlying field size used in the Cramer-Shoup cryptosystem. We use Toom-3 fast multiplication, which has computational complexity of $O(k^{1.46})$ and known to be the best choice for integers ranging from 360-bit to 8000-bit [4] and make a rough comparison to show that our protocol has a better performance for a practical group size. We assume a k-bit multiplication is $k^{0.46}$ times slower than a k-bit addition, and omit the $3N + 1$ hashing, 2 signing and $2(N-1)$ verifying operations in [5] and one symmetric key encryption operation in our protocol. We fix the message length to 3072 bits and select the symmetric key encryption with 128-bit key size for our protocol and 3072-bit field size for the Crame-shoup cryptosystem to achieve the same security level as that of the 128-bit encryption. Note that this comparison actually favours the protocol in [5] because we omit a large constant factor in $O(k^{1.46})$ and the computation of $3N+1$ hashing, 2 signing and $2(N-1)$ verifying functions. In this setting, the computational overhead of our protocol is always better than that of [5] as long as the group size is smaller than 12, 517 through calculation.

VII. Simulation

In this section, we made a proof-of-concept implementation of our protocol compared to the protocols in [5] and [9]. All the experiments were carried out on an Intel(R) Core(TM)2 Duo CPU P8600 @ 2.40GHz under Windows 7 Professional in 32-bit mode computer. We used Crypto++ [10] as our underlying cryptographic algorithms library.

Dissent [9] uses the same technique as the one in [5] in its shuffle protocol, which requires a IND-CCA2 secure cryptosystem. Since Crypto++ does not have any IND-CCA2 secure algorithm in the standard model, we implemented the Cramer-Shoup cryptosystem by using SHA-1 and the group of quadratic residues modulo a safe prime. The symmetric key cryptosystem in our protocol was implemented by using AES-128 in CBC CTR mode. All the algorithms were implemented in C++ and compiled by g++ 4.4.3 with O3 level optimization. Ns-2 [20] was used to simulate the network environment. Since our purpose is to compare the performance of our protocol with that of the existing protocols, we carried out our implementations on a network with a star topology which is the same as that in [9]. We note that this is not a typical representation of all possible networks, but we are not aware of any other topologies that were used to test anonymous message submission protocols. All the links in the simulated network were 100Mbps with a 50ms latency. We used the TCP to deliver the messages in the protocols. We simplified the Dissent protocol by using the shuffle protocol in [5] instead of the original shuffle parts because we did not consider the audit and focused on the basic anonymous submission functionality.

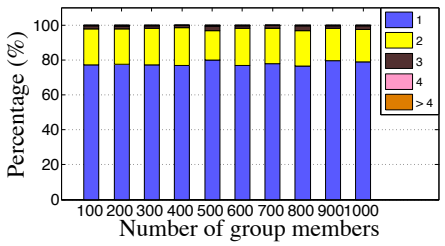
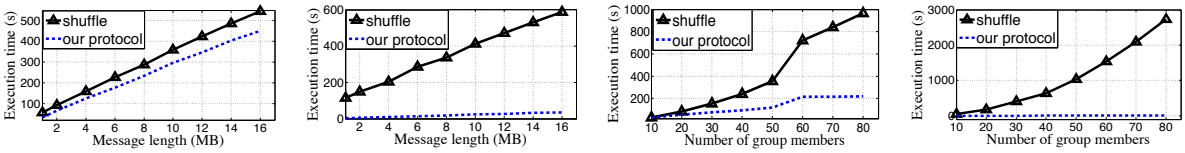


Fig. 7. The distribution of the number of rounds needed in Phase 1

First we tested the number of rounds needed in Phase 1. We increased N from 100 to 1000 using an increment of 100. For each N, we run Phase 1 for 1000 times and counted the number of rounds needed to find a non-collision vector $\mathbf{p}$. Fig. 7 is the simulation result. We can see that at lease 77% of the tests are completed in 1 round no matter what N is. Actually, 97% of them are completed within 2 rounds finding a non-collision vector $\mathbf{p}$.

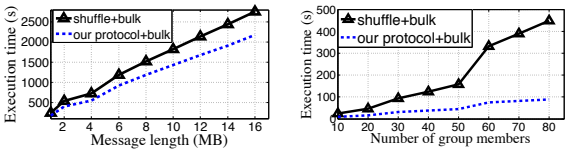
Next, we compare the execution time of our protocol to that of the shuffle protocol in [5]. Fig. 8 shows the result. There are two variables, the group size N and the message length l. First we fix $N = 16$, change l from 1MB to 16MB, and compare the execution time of each member and the collector in the two protocols. Note that in Fig. 8 (b), the run time of the collector in our protocol is much faster than that



(a) change l, fix $N = 16$, execution time of each m_i (b) change l, fix $N = 16$, execution time of C (c) change N, fix $l = 1MB$, execution time of each m_i (d) change N, fix $l = 1MB$, execution time of C

Fig. 8. protocol execution time of each member and the collector

of shuffle. This is because the decryption time of an IND-CCA2 secure cryptosystem is much slower than that of a symmetric key decryption algorithm. In addition, shuffle needs $N^2 + N$ decryptions on the collector. While the collector in our protocol only needs to decrypt N ciphertexts. In Fig. 8 (c) and (d), we fix l to be 1MB, and change N from 10 to 80 on an increment of 10. Since the number of the communication rounds depends on N in the shuffle protocol, the execution time of each member increases with N, while the execution time of each member in our protocol increases slowly because the the number of the communication rounds is independent of N. In Fig 8 (d), for the shuffle protocol, the execution time increases polynomially in N. However, the execution time of the collector in our protocol nearly did not change, which shows that the addition and symmetric key decryption is so fast that we can not tell the time difference when the group size is small.



(a) change l, fix $N = 25$, execution time of each m_i (b) change N, fix $l = 1Kb$, execution time of each m_i

Fig. 9. execution time of each m_i

Since the dissent protocol in [9] uses the same shuffle technique in [5]. We compare the execution time when using our protocol and the shuffle protocol prior to the bulk protocol in [9]. Fig. 9 shows the result. We can easily see from Fig. 9 that the execution time of our protocol and the bulk protocol is much less than that using the shuffle and bulk protocols. Especially, we note that when the number of members increases, the run time of our protocol and the bulk protocol grows much slower than that of the shuffle and bulk protocols.

VIII. Conclusion

In this paper, we have proposed an efficient online anonymous message protocol. Utilizing the simplified secret sharing scheme, we have presented a novel position application technique, in which all members secretly select their positions in a position vector, such that a member knows nothing about the other members' message positions. We have introduced a data aggregation technique, in which all members aggregate their messages into a message vector and submit it to the collector without exposing their identities. We have theoretically proved that our protocol is anonymous under malicious attacks. Our efficiency analysis and simulation have shown that our protocol is computationally more efficient than existing works and requires a constant communication rounds on average.

References

[1] R. Agrawal and R. Srikant, "Privacy-preserving data mining," in *SIGMOD*, vol. 29, no. 2. ACM, 2000, pp. 439–450.

[2] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway, "A concrete security treatment of symmetric encryption," in *focs*. Published by the IEEE Computer Society, 1997, p. 394.

[3] J. Benaloh, "Secret sharing homomorphisms: Keeping shares of a secret secret (extended abstract)," in *Advances in Cryptology?CRYPTO?86*. Springer, 1987, pp. 251–260.

[4] M. Bodrato, "Towards optimal toom-cook multiplication for univariate and multivariate polynomials in characteristic 2 and 0," *Arithmetic of Finite Fields*, pp. 116–133, 2007.

[5] J. Brickell and V. Shmatikov, "Efficient anonymity-preserving data collection," in *Proceedings of the 12th ACM SIGKDD*. ACM, 2006, pp. 76–85.

[6] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 829–837.

[7] D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM*, vol. 24, no. 2, pp. 84–90, 1981.

[8] D. Chaum, "The dining cryptographers problem: Unconditional sender and recipient untraceability," *Journal of cryptology*, vol. 1, no. 1, pp. 65–75, 1988.

[9] H. Corrigan-Gibbs and B. Ford, "Dissent: accountable anonymous group messaging," in *ACM CCS*. ACM, 2010, pp. 340–350.

[10] Crypto++, http://www.cryptopp.com/, 2010.

[11] W. Du and M. Atallah, "Secure multi-party computation problems and their applications: a review and open problems," in *Proceedings of the 2001 workshop on New security paradigms*. ACM, 2001, pp. 13–22.

[12] M. Freedman, K. Nissim, and B. Pinkas, "Efficient private matching and set intersection," in *Advances in Cryptology-EUROCRYPT 2004*. Springer, 2004, pp. 1–19.

[13] O. Goldreich, "Secure multi-party computation," *Working Draft*, 2000.

[14] S. Goldwasser and S. Micali, "Probabilistic encryption* 1," *Journal of computer and system sciences*, vol. 28, no. 2, pp. 270–299, 1984.

[15] P. Golle and A. Juels, "Dining cryptographers revisited," in *Advances in Cryptology-Eurocrypt 2004*. Springer, 2004, pp. 456–473.

[16] M. Li, N. Cao, S. Yu, and W. Lou, "Findu: Privacy-preserving personal profile matching in mobile social networks," in *Proc. of Infocom*, 2011.

[17] Y. Lindell and B. Pinkas, "Privacy preserving data mining," in *Advances in Cryptology?CRYPTO 2000*. Springer, 2000, pp. 36–54.

[18] A. Narayanan, N. Thiagarajan, M. Lakhani, M. Hamburg, and D. Boneh, "Location privacy via private proximity testing," in *Proc. of NDSS*, 2011.

[19] T. Nishide and K. Ohta, "Multiparty computation for interval, equality, and comparison without bit-decomposition protocol," *Public Key Cryptography–PKC 2007*, pp. 343–360, 2007.

[20] ns 2, http://www.isi.edu/nsnam/ns/.

[21] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[22] E. Sirer, S. Goel, M. Robson, and D. Engin, "Eluding carnivores: File sharing with strong anonymity," in *Proceedings of the 11th workshop on ACM SIGOPS European workshop*. ACM, 2004, pp. 19–es.

[23] D. Stinson, *Cryptography: theory and practice*. CRC press, 2006.

[24] P. Syverson, D. Goldschlag, and M. Reed, "Onion routing for anonymous and private internet connections," 1999.

[25] M. Waidner, B. Pfitzmann *et al.*, "The dining cryptographers in the disco: Unconditional sender and recipient untraceability with computationally secure serviceability."

[26] W. Wong, D. Cheung, B. Kao, and N. Mamoulis, "Secure knn computation on encrypted databases," in *SIGMOD*. ACM, 2009, pp. 139–152.

[27] Z. Yang, S. Zhong, and R. Wright, "Anonymity-preserving data collection," in *ACM SIGKDD*. ACM, 2005, pp. 334–343.

[28] A. Yao, "How to generate and exchange secrets," in *FOCS*. IEEE, 1986, pp. 162–167.