Secure Aggregation Algorithm Based on Infinite Series

Abdolreza Rasouli Kenari
Faculty of Computer Science & Information System
University Technology Malaysia, Johor, Malaysia
(0060) 17 720 1417
rs.reza@gmail.com

Mohd Aizaini Maarof
Faculty of Computer Science & Information System
University Technology Malaysia, Johor, Malaysia
(0060) 7 553 2368
aizaini@utm.my

Mahboubeh Shamsi
Faculty of Computer Science & Information System
University Technology Malaysia, Johor, Malaysia
(0060) 17 720 2930
mahboubehshamsi@yahoo.com

ABSTRACT
DATA MINING HAS OPERATED ON A DATA WAREHOUSING MODEL OF GATHERING ALL DATA INTO A CENTRAL SITE, THEN RUNNING AN ALGORITHM AGAINST THAT DATA. PRIVACY CONSIDERATIONS MAY PREVENT THIS APPROACH. PRIVACY PRESERVING DATA MINING HAS EMERGED TO ADDRESS THIS ISSUE. ONE APPROACH IS TO ALTER THE DATA BEFORE DELIVERING IT TO THE DATA MINER. THE SECOND APPROACH ASSUMES THE DATA IS DISTRIBUTED BETWEEN TWO OR MORE SITES, AND THESE SITES COOPERATE TO LEARN THE GLOBAL DATA MINING RESULTS WITHOUT REVEALING THE DATA AT THEIR INDIVIDUAL SITE. IN THIS PAPER, WE REPRESENT A SECURE SUMMATION ALGORITHM FOR ONLINE TRANSACTIONS, WHERE THE USERS WILL JOIN THE SYSTEM PIECEMEAL. THE ALGORITHM EMERGES THE EXCESSIVELY USEFUL RESPONSE TIME, SO THE EXECUTION TIME OF SUMMATION FOR 1000 USERS' DATA IS ONLY 0.9S.

Categories and Subject Descriptors
H.2.4 [**Database Management**]: Database Applications– *data mining, distributed database, privacy preserving*

General Terms
Algorithms, Management, Security

Keywords
Association rule mining, Frequent Itemset, Distributed Database, Cryptography, Privacy Preserving

1. INTRODUCTION
Data mining has been studied extensively and applied widely [24, 25]. Through the use of data mining techniques, businesses can discover hidden patterns and rules in a database and then employ them to predict features of data items that have not yet arrived. An important scenario in data mining is distributed data mining, in which a database is distributed between two (or more) parties, and each party owns a portion of the data. These parties need to collaborate with each other so that they can jointly mine the data and produce results that are interesting to both of them. Privacy concerns are of great importance in this scenario, because each party may not want to reveal her own portion of the data, although she would like to participate in the mining.

Standard algorithms for association rule mining are based on identification of frequent itemsets. In this research, we study how to maintain privacy in distributed mining of frequent itemsets. That is, we study how two (or more) parties can find frequent itemsets in a distributed database without revealing each party's portion of the data to the other. The existing solution for vertically partitioned data leaks a significant amount of information, while the existing solution for horizontally partitioned data only works for three parties or more. In this research, we propose algorithms for both vertically and horizontally partitioned data, with cryptographically strong privacy.

This paper next describes a brief background and related work, followed by objectives in section 2. The paper then continues on methodology and full description of new algorithm in section 3 and discussion on result in section 4. The paper ends with a conclusion and significance of algorithm in section 5, 6.

1.1 Problem Background
In this research, we address the question of how to maintain privacy in distributed mining of frequent itemsets. That is, we ask how two (or more) parties can find frequent itemsets in a distributed database without revealing each party's portion of the data to the other. We will formally specify what we mean by "privacy," and our definitions of privacy are cryptographically strong. Roughly speaking, for strong privacy, we require that each participant learns nothing about other participants' data except what is implied by the final output. We will also propose solutions for two major types of partitioned data: vertically partitioned and horizontally partitioned, respectively.

Previous work in privacy preserving data mining has addressed two issues. In one, the aim is preserving customer privacy by distorting the data values [2]. The idea is that the distorted data

does not reveal private information and thus is "safe" to use for mining. The key result is that the distorted data, and information on the distribution of the random data used to distort the data, can be used to generate an approximation to the original data distribution, without revealing the original data values. The distribution is used to improve mining results over mining the distorted data directly, primarily through selection of split points to "bin" continuous data. Later refinement of this approach tightened the bounds on what private information is disclosed by showing that the ability to reconstruct the distribution can be used to tighten estimates of original values based on the distorted data [1].

More recently, the data distortion approach has been applied to Boolean association rules [20]. Again, the idea is to modify data values such that reconstruction of the values for any individual transaction is difficult, but the rules learned on the distorted data are still valid. One interesting feature of this work is a flexible definition of privacy, e.g., the ability to correctly guess a value of "1" from the distorted data can be considered a greater threat to privacy than correctly learning a "0." The data distortion approach addresses a different problem from our work. The assumption with distortion is that the values must be kept private from whoever is doing the mining. We instead assume that some parties are allowed to see some of the data, just that no one is allowed to see all the data. In return, we are able to get exact, rather than approximate, results.

The problem of privately computing association rules in vertically partitioned distributed data has also been addressed [22]. The vertically partitioned problem occurs when each transaction is split across multiple sites, with each site having a different set of attributes for the entire set of transactions. With horizontal partitioning, each site has a set of complete transactions. In relational terms, with horizontal partioning, the relation to be mined is the union of the relations at the sites. In vertical partitioning, the relations at the individual sites must be joined to get the relation to be mined. The change in the way the data is distributed makes this a much different problem from the one we address here, resulting in a very different solution.

The research domain is shown in Fig 1.

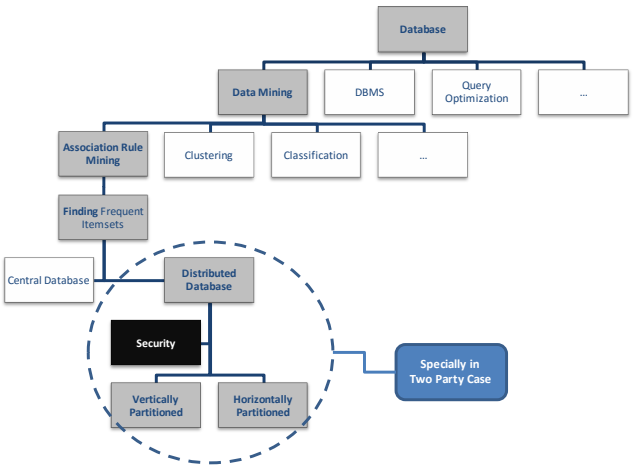


Fig 1. Research Domain

1.2 Related Work

The secure multi-party computation also known as (MPC) is one of the main results of the theory of cryptography. First, Yao's [3] introduced the multi-party computation and nowadays many authors have attend many optimizations and extensions to the basic concept, for two main branch; the two-party (2PC) and the multi-party (MPC) [7, 11, 12, 14, 15, 17, 10]. Most of recently papers on secure multi-party computation area have been focused on theory of multi-party computation and there is no much applicable implementing of MPC, although, in the few last year some practical implementation of multi-party computation has been appeared [18, 5, 23, 8, 4].

There exist many algorithm and techniques for secure multi-party computation. We have focused on more practical and high speed algorithms which have been published.

Secure multi-party computation essentially comes in two flavors [19]. The first approach is typically based upon secret sharing and operates on an arithmetic circuit representation of the computed function, such as in the BGW (Ben-Or, Goldwasser and Wigderson) or CCD (Chaum, Crepeau and Damg°ard) protocols [16, 9]. This approach is usually applied when there is an honest majority among the participants (which can only exist if more than two parties participate in the protocol). An alternative approach represents the function as a binary circuit. This approach was used in the original two-party garbled circuit construction of Yao [3], and in the GMW (Goldreich, Micali and Wigderson) multi-party protocol [12].

Before we introduce our algorithm, we must first make the distinction between semi honest and malicious adversaries in multiparty algorithms. Semi honest adversaries follow the protocol exactly but try to learn additional information by analyzing the messages they received during the execution of the protocol. Such adversaries often model attacks that take place only after the execution of the protocol has completed. Malicious adversaries can always execute some arbitrary, malicious operations which can be very damaging to other parties. The malicious adversaries are much more difficult to defend against when designing the protocol. It is proved that, in the distributed multiparty setting, any probabilistic polynomial time function can be securely computed by assuming a majority of honest parties. Informally, in the semi honest model, a protocol privately computes a function if whatever can be computed by a subset of parties could be computed from their inputs and all intermediate computing messages.

Our proposed algorithm is secure within the semi-honest model. The algorithm is secure under Diffie-Hillman DDH assumption and uses ElGamal encryption for increasing robustness and speed.

1.3 Utilized Cryptographic Method

Some cryptographic tools are used for designing our secured frequency counting contribution. In this section, we introduce our utilized cryptographic fundamental methods.

1.3.1 ElGamal Cryptosystem

The ElGamal cryptosystem is a part of public encryption systems. The public key is (h, G, q, g) where G is a cyclic group of order q with the generator g, $h=g^x$ and x is the private key which is randomly chosen from $[1, q]$. All computation in the ElGamal scheme is done in the group G.

Under the public key (h, G, q, g), the ciphertext of a message m (which is the representation of an element of G) is encrypted as $E(m)=(c_1, c_2)$ where $c_1=m.h^r$, $c_2=g^r$ and r is randomly chosen from [1, q]. To decrypt the ciphertext (c_1, c_2) with the private key x, the plaintext message m can be decrypted as $m=c_1(c_2^x)^{-1}$. It clearly is true because

$$c_1(c_2^x)^{-1}=m.h^r(g^{rx})^{-1}=m.h^r(h^r)^{-1}=m \quad (1)$$

ElGamal encryption is semantically secure under the Decisional Diffie-Hellman (DDH) assumption [6]. One family in which DDH is believed to be intractable is the quadratic residue subgroup Q_p of Z_p* where p, q are two primes and $p=2q+1$.

In the ElGamal encryption scheme, one cleartext has many possible ciphertexts because of the random value r. ElGamal supports rerandomization: a new ciphertext $E'(m)$ of m can be computed from a ciphertext $E(m)=(c_1, c_2)$ as $E'(m)=(c_1.h^{r'}, c_2.g^{r'})$ where r' is randomly chosen from [1, q].

1.3.2 Secret Sharing

Secret sharing is the method of sharing a secret by multiple parties, so that no one and no party know the secret, but the secret could be constructed by combing some parties' shares.

For example, in a two-party case, Alice and Bob share a value x modulo some appropriate value N, in such a way that Alice holds a, Bob holds b, and x is equal to $(a+b)$ *mod* N. This is called additive secret sharing. An important property of this kind of secret sharing is that if Alice and Bob have shares of a and b, then they can each locally add their shares modulo N to obtain shares of $a+b$.

Shamir secret sharing is a threshold scheme [21]. In Shamir secret sharing, there are N parties and a polynomial P of degree k-1 such that $P(0)=x$ where x is a secret. Each of the N parties holds a point in the polynomial P. Because k points (x_i, y_i) ($1 \le i \le k$) uniquely define a polynomial P of degree k-1, a subset of at least k parties can reconstruct the secret x. But, fewer than k parties cannot construct the secret x. This scheme is also called (N, k) *Shamir secret sharing*.

2. OBJECTIVES

In this research, we specify the problems and security requirements of privacy preserving aggregation protocol. The goal of research is giving algorithms for summation of partitioned data. These algorithms should be very efficient than the existing solutions.

2.1 Problem Statement

In our problem there are several parties that have their owned data. All parties are interest to share their data to a miner for gaining advantage of data mining. The basic idea is that a mining is secure if at the end of the mining, no party knows anything except its own input and the results. One way to view this is to imagine a trusted third party. Everyone gives their input to the trusted party, who performs the mining and sends the results to the participants. Now imagine that we can achieve the same result without having a trusted party. Obviously, some communication between the parties is required for any interesting mining. How do we ensure that this communication doesn't disclose anything?

Let us consider a real life motivating example where privacy is important issues. Credit card transactions are taking a large share of the payment systems worldwide and have led to a higher rate of stolen account numbers and subsequent losses by banks. Each bank has a database which stores both legitimate and fraudulent credit card transactions, so let us consider distributed data mining for credit card fraud detection. The objective is to build a data mining model which will be used by automated fraud detector systems to prevent fraudulent transactions. This data mining model must be very general and accurate, because a mistake means great loss of money. In order to build accurate models of credit card fraud, a data mining system must have access to information about the fraudulent patterns of all banks (i.e., some types of fraud can have occurred only in one bank). The problem is that banks are restricted by law (and also by competitive reasons) from sharing data about their customers with other banks. However, they may share "black-box" models; that is, they can share knowledge about fraudulent transactions, but not their data.

The main statements of this proposal are summarized below:

1. How to count a frequency of an interest Boolean data in multi party database without knowing each party frequency?
2. How to sum values of an interest data in multi party database without knowing each party data?
3. How to adopt algorithm for online cases?

The main objectives of this research are:

- To developing a secure distributed frequency counting protocol
- To developing a secure aggregation algorithm on partitioned database
- To developing a online secure algorithm for Multi Party databases

3. METHODOLOGY

We assume a scenario in which there are possibly large numbers of respondents with their private votes and a Teller System who counts their votes. All respondents' private votes need to be protected. The Teller system should only know the poll result, without revealing the exact vote of respondents.

3.1 Problem Explanation

In our scenario, there are n respondents $U_1, \ldots, U_n$. Each respondent U_i has a vote d_i. This value could be a Boolean value or Numerical value. The Boolean value denotes as such case same as electing a people in charge of a responsibility, like president or senate election. On the other hand, the numerical integer vote represents the value of an option that respondents could select such as questionary forms. The Teller System would like to find out the sum $d = \sum_{i=1}^{n} d_i$ $\mathrm{d} = \sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{d_i}$ without revealing each d_i.

Our solution also satisfies a few important restrictions:

- Each respondent only sends one message to the Teller System.
- No respondent communicates with other respondents.

This model is highly practical, because they do not need communication channels between different respondents or multi round interaction between any respondent and the Teller System.

3.2 E-Voting Algorithm

The implemented algorithm is based on the homomorphism property of mentioned ElGamal encryption [13]. The DDH assumption and the ElGamal cryptosystem ensure the privacy of the algorithm. The algorithm also uses the exponentiation's mathematical properties for converting multiplication to desired sums. We also use modular arithmetic operation to speed up the computing time of big prime numbers. It is surely affect on algorithm time.

Let G be a group where ($|G|=q$ for a large prime q), and let g be a generator of G. The group G is assumed for all computations in this paper. Suppose that each respondent U_i has two pairs of keys: $((x_i \bmod_q), (X_i \bmod_q=(g^{x}_i \bmod_q)), ((y_i \bmod_q), (Y_i \bmod_q=(g^{y}_i \bmod_q))$. We also define

$$(X \bmod_q) = \prod_{i=1}^{n}(X_i \bmod_q) \quad (2)$$

$$(Y \bmod_q) = \prod_{i=1}^{n}(Y_i \bmod_q) \quad (3)$$

The values x_i and y_i are private keys; X_i and Y_i are public keys. All respondents need to calculate the values of X and Y from public keys.

As we mentioned before, each respondent U_i holds the vote d_i and the Teller's goal is to learn $d = \sum_{i=1}^{n} d_i$. The system architecture is shown in Fig 2.

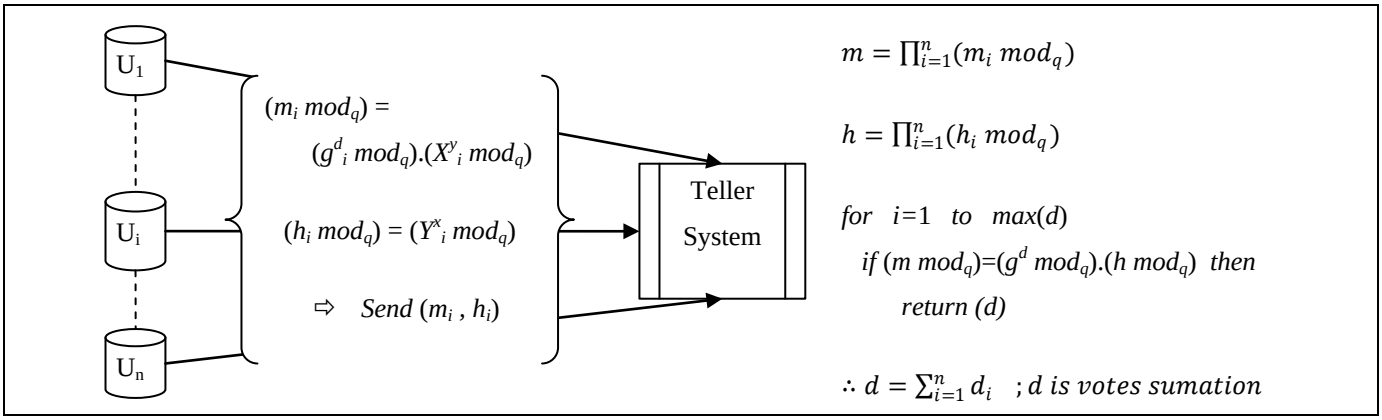


Fig 2. E-Voting System Architecture

First, each respondent U_i try to encrypt his value d_i using his private key x_i , y_i and shared public key X , Y in ElGamal encryption system as described below

$$E(d_i) = \begin{Bmatrix} (m_i \bmod_q) = (g^{d_i} \bmod_q).(X^{y_i} \bmod_q) \\ (h_i \bmod_q) = (Y^{x_i} \bmod_q) \end{Bmatrix} \quad (4)$$

Then all respondents send their encrypted message to Teller. The Teller System gathers all encrypted votes together and computes m, h as:

$$m = \prod_{i=1}^{n}(m_i \bmod_q) \ , h = \prod_{i=1}^{n}(h_i \bmod_q) \quad (5)$$

The Teller will use m, h for decrypting the d as sum of d_i. Then the Teller tries to find correct d between all possible values. It means that the miner tries to calculate ($g^d \bmod_p$) for all possible of d values. This stage is more time consuming step and will continue until there exist any d as

$$(m \bmod_q) = (g^d \bmod_q) \ . \ (h \bmod_q) \quad (6)$$

This value is the desired summation of respondents' votes. The Teller cannot take discrete logarithms; the Teller must use trial and error to learn d. The time consumption parameter of algorithm returns to the range of possible values of d. In case of Boolean votes the range of d is the number of respondent's n. The miner algorithm is shown below.

```
for   i=1   to   max(d)
   if (m mod_q)=(g^d mod_q).(h mod_q)  then
         return (d)
```

We note that the security of ElGamal encryption depends on new random values being used for each encryption. In our setting, this means that the x_i and y_i values, and associated X and Y, cannot be reused in different uses of the running algorithm. However, since these parameters do not depend on the actual votes values, they can in general be precomputed offline before the poll starts.

3.3 Algorithm Demonstration

In this section, we prove that the algorithm represented in Fig 2 correctly computes the summation of respondents' votes. Suppose that the Teller finds a d so

$$(m \bmod_q) = (g^d \bmod_q) \ . \ (h \bmod_q) \quad (6^*)$$

We will show that $d = \sum_{i=1}^{n} d_i$.

$$(m \bmod_q) = (g^d \bmod_q).(h \bmod_q) \Rightarrow$$

$$\prod_{i=1}^{n}(m_i \bmod_q) = (g^d \bmod_q).\prod_{i=1}^{n}(h_i \bmod_q)$$

$$\Rightarrow (g^d \bmod_q) = \frac{\prod_{i=1}^{n}(m_i \bmod_q)}{\prod_{i=1}^{n}(h_i \bmod_q)}$$

$$= \frac{\prod_{i=1}^{n}(g^{d_i} \bmod_q).(X^{y_i} \bmod_q)}{\prod_{i=1}^{n}(Y^{x_i} \bmod_q)}$$

$$= \frac{\prod_{i=1}^{n}(g^{d_i} \bmod_q).\prod_{i=1}^{n}(X^{y_i} \bmod_q)}{\prod_{i=1}^{n}(Y^{x_i} \bmod_q)}$$

$$= \prod_{i=1}^{n}(g^{d_i} \bmod_q).\frac{\prod_{i=1}^{n}(X^{y_i} \bmod_q)}{\prod_{i=1}^{n}(Y^{x_i} \bmod_q)}$$

$$= g^{\sum_{i=1}^{n}(d_i \bmod_q)}.\prod_{i=1}^{n}\frac{((\prod_{j=1}^{n}(X_j \bmod_q)^{y_i} \bmod_q)}{((\prod_{j=1}^{n}(Y_j \bmod_q))^{x_i} \bmod_q)}$$

$$= g^{\sum_{i=1}^{n}(d_i \bmod_q)}.\prod_{i=1}^{n}\frac{((g^{\sum_{i=1}^{n}(x_j \bmod_q)})^{y_i} \bmod_q)}{((g^{\sum_{i=1}^{n}(y_j \bmod_q)})^{x_i} \bmod_q)}$$

$$= g^{\sum_{i=1}^{n}(d_i \bmod_q)}.\frac{g^{\sum_{i=1}^{n}\sum_{j=1}^{n}(x_j y_i \bmod_q)}}{g^{\sum_{i=1}^{n}\sum_{j=1}^{n}(y_j x_i \bmod_q)}} = g^{\sum_{i=1}^{n}(d_i \bmod_q)}$$

$$\Rightarrow (g^{d} \bmod_q) = g^{\sum_{i=1}^{n}(d_i \bmod_q)} \therefore d = \sum_{i=1}^{n} d_i \quad \blacksquare \qquad (7)$$

4. PROBLEM EXPLANATION

Base on the mentioned algorithm, the first and mostly important assumption is that all clients are ready to send their data and the Miner knows the number of clients. It means that the system is semi-offline system. This condition avoids us to use the algorithm in real world online problems like as E-Voting or web page's poll result. Because in this cases, the clients are not ready at the beginning time and the time of user's appearance is optional. The miner does not have any knowledge which how many clients could join to system and when they will appear. So the Miner could not able to compute the secret shared between the clients. As we mentioned before, the secret shared between the users is the product of their public keys as we called *X*, *Y*. All clients should be present their public keys to compute the shared secret key *X*, *Y* before aggregation started. Moreover, all users should send their data; therefore the Miner could compute the summation.

So, in most online cases the algorithm will fail and is inapplicable. Due to the Miner, likewise, clients cannot assess the product of user's public keys that do not join yet.

In this paper, infinite product series have been used as shared secret key instead of *X*, *Y*. The final amounts of these series do not change with increasing the number of elements. An infinite product series define as

$$A = \prod_{i=1}^{\infty}\alpha_i \ \ like\ as\ \prod_{i=1}^{\infty}\frac{1}{e}\left(\frac{1}{3i+1}\right)^{3i+\frac{1}{2}} = 1.0123785 \qquad (8)$$

4.1 Proposed Algorithm

The proposed algorithm also is based on the homomorphism property of mentioned ElGamal encryption. We note that the security of ElGamal encryption depends on new random values being used for each encryption. Same as before, each client C_i has two pairs of keys: $((x_i \bmod_q), (X_i \bmod_q=(g^{x_i} \bmod_q))$, $((y_i \bmod_q), (Y_i \bmod_q=(g^{y_i} \bmod_q))$ as private and public keys, respectively. Remember that the x_i and y_i values cannot be reused in different uses of the running algorithm. Here, we are not able to compute the *X*, *Y*. So we use two infinite series Ω, Ψ instead of *X*, *Y*.

$$(\Omega_i \bmod_q) = (g^{\omega_i} \bmod_q),\ (\Omega \bmod_q) = \prod_{i=1}^{n}(\Omega_i \bmod_q) \qquad (9)$$

$$(\Psi_i \bmod_q) = (g^{\psi_i} \bmod_q),\ (\Psi \bmod_q) = \prod_{i=1}^{n}(\Psi_i \bmod_q) \qquad (10)$$

The client C_i encrypt his value d_i in ElGamal encryption system using Ω, Ψ as described below

$$E(d_i) = \begin{Bmatrix}(m_i \bmod_q) = (g^{d_i} \bmod_q).(\Omega^{y_i} \bmod_q)\\ (h_i \bmod_q) = (\Psi^{x_i} \bmod_q)\end{Bmatrix} \qquad (11)$$

and send his encrypted message in assition to his public keys X_i, Y_i to the Miner. The Miner System computes *m*, *h*, *X*, *Y* as before:

$$m = \prod_{i=1}^{n}(m_i \bmod_q)\ , h = \prod_{i=1}^{n}(h_i \bmod_q)$$

$$X = \prod_{i=1}^{n}(X_i \bmod_q)\ , Y = \prod_{i=1}^{n}(Y_i \bmod_q) \qquad (12)$$

The Miner try to find desired *d* to satisfy below

$$(m \bmod_q).(X^{\sum_{i=1}^{n}\psi_i}) = (g^{d} \bmod_q).(h \bmod_q).(Y^{\sum_{i=1}^{n}\omega_i}) \qquad (13)$$

The desired *d* is the summation of clients' data.

4.2 Algorithm Demonstration

In this section, we prove that the algorithm represented in 4.1 correctly computes the summation of respondents' data. Suppose that the Miner finds a *d* so

$$(m \bmod_q).(X^{\sum_{i=1}^{n}\psi_i}) = (g^{d} \bmod_q).(h \bmod_q).(Y^{\sum_{i=1}^{n}\omega_i}) \qquad (13^*)$$

We will show that $d = \sum_{i=1}^{n} d_i$.

$$(m \bmod_q).(X^{\sum_{i=1}^{n}\psi_i}) = (g^{d} \bmod_q).(h \bmod_q).(Y^{\sum_{i=1}^{n}\omega_i})$$

$$\Rightarrow (\prod_{i=1}^{n}(m_i \bmod_q)).(X^{\sum_{i=1}^{n}\psi_i})$$

$$= (g^{d} \bmod_q).(\prod_{i=1}^{n}(h_i \bmod_q)).(Y^{\sum_{i=1}^{n}\omega_i})$$

$$\Rightarrow (g^{d} \bmod_q) = \frac{(\prod_{i=1}^{n}(m_i \bmod_q)).(X^{\sum_{i=1}^{n}\psi_i})}{(\prod_{i=1}^{n}(h_i \bmod_q)).(Y^{\sum_{i=1}^{n}\omega_i})}$$

$$= \frac{(\prod_{i=1}^{n}(g^{d_i} \bmod_q).(\Omega^{y_i} \bmod_q)).(X^{\sum_{i=1}^{n}\psi_i})}{(\prod_{i=1}^{n}(\Psi^{x_i} \bmod_q)).(Y^{\sum_{i=1}^{n}\omega_i})}$$

$$= \frac{(\prod_{i=1}^{n}(g^{d_i} \bmod_q).\prod_{i=1}^{n}(\Omega^{y_i} \bmod_q)).(X^{\sum_{i=1}^{n}\psi_i})}{(\prod_{i=1}^{n}(\Psi^{x_i} \bmod_q)).(Y^{\sum_{i=1}^{n}\omega_i})}$$

$$= \prod_{i=1}^{n}(g^{d_i} \bmod_q).\frac{\prod_{i=1}^{n}(\Omega^{y_i} \bmod_q)}{\prod_{i=1}^{n}(\Psi^{x_i} \bmod_q)}.\frac{(X^{\sum_{i=1}^{n}\psi_i})}{(Y^{\sum_{i=1}^{n}\omega_i})}$$

$$= g^{\sum_{i=1}^{n}(d_i \bmod_q)}.\frac{\Omega^{\sum_{j=1}^{n}(y_j \bmod_q)}}{\Psi^{\sum_{j=1}^{n}(x_j \bmod_q)}}.\frac{(X^{\sum_{i=1}^{n}\psi_i})}{(Y^{\sum_{i=1}^{n}\omega_i})}$$

$$= g^{\sum_{i=1}^{n}(d_i \bmod_q)}.\frac{(\prod_{i=1}^{n}(\Omega_i \bmod_q))^{\sum_{j=1}^{n}(y_j \bmod_q)}}{(\prod_{i=1}^{n}(\Psi_i \bmod_q))^{\sum_{j=1}^{n}(x_j \bmod_q)}}.\frac{(X^{\sum_{i=1}^{n}\psi_i})}{(Y^{\sum_{i=1}^{n}\omega_i})}$$

$$= g^{\sum_{i=1}^{n}(d_i \bmod_q)}.\frac{(\prod_{i=1}^{n}(g^{\omega_i} \bmod_q))^{\sum_{j=1}^{n}(y_j \bmod_q)}}{(\prod_{i=1}^{n}(g^{\psi_i} \bmod_q))^{\sum_{j=1}^{n}(x_j \bmod_q)}}.\frac{(X^{\sum_{i=1}^{n}\psi_i})}{(Y^{\sum_{i=1}^{n}\omega_i})}$$

$$= g^{\sum_{i=1}^{n}(d_i \bmod_q)}.\frac{g^{\sum_{i=1}^{n}(\omega_i \bmod_q).\sum_{j=1}^{n}(y_j \bmod_q)}}{g^{\sum_{i=1}^{n}(\psi_i \bmod_q).\sum_{j=1}^{n}(x_j \bmod_q)}}.\frac{(X^{\sum_{i=1}^{n}\psi_i})}{(Y^{\sum_{i=1}^{n}\omega_i})}$$

$$= g^{\sum_{i=1}^{n}(d_i \bmod_q)} \cdot \frac{g^{\sum_{j=1}^{n}(y_j \bmod_q)\sum_{i=1}^{n}(\omega_i \bmod_q)}}{g^{\sum_{j=1}^{n}(x_j \bmod_q)\sum_{i=1}^{n}(\psi_i \bmod_q)}} \cdot \frac{(X^{\sum_{i=1}^{n}\psi_i})}{(Y^{\sum_{i=1}^{n}\omega_i})}$$

$$= g^{\sum_{i=1}^{n}(d_i \bmod_q)} \cdot \frac{\left(\prod_{i=1}^{n}(g^{y_i} \bmod_q)\right)^{\sum_{i=1}^{n}(\omega_i \bmod_q)}}{\left(\prod_{i=1}^{n}(g^{x_i} \bmod_q)\right)^{\sum_{i=1}^{n}(\psi_i \bmod_q)}} \cdot \frac{(X^{\sum_{i=1}^{n}\psi_i})}{(Y^{\sum_{i=1}^{n}\omega_i})}$$

$$= g^{\sum_{i=1}^{n}(d_i \bmod_q)} \cdot \frac{\left(\prod_{i=1}^{n}(Y_i \bmod_q)\right)^{\sum_{i=1}^{n}(\omega_i \bmod_q)}}{\left(\prod_{i=1}^{n}(X_i \bmod_q)\right)^{\sum_{i=1}^{n}(\psi_i \bmod_q)}} \cdot \frac{(X^{\sum_{i=1}^{n}\psi_i})}{(Y^{\sum_{i=1}^{n}\omega_i})}$$

$$= g^{\sum_{i=1}^{n}(d_i \bmod_q)} \cdot \frac{Y^{\sum_{i=1}^{n}(\omega_i \bmod_q)}}{X^{\sum_{i=1}^{n}(\psi_i \bmod_q)}} \cdot \frac{(X^{\sum_{i=1}^{n}\psi_i})}{(Y^{\sum_{i=1}^{n}\omega_i})}$$

$$\Rightarrow \left(g^{d} \bmod_q\right) = g^{\sum_{i=1}^{n}(d_i \bmod_q)} \therefore d = \sum_{i=1}^{n} d_i \quad \blacksquare \qquad (14)$$

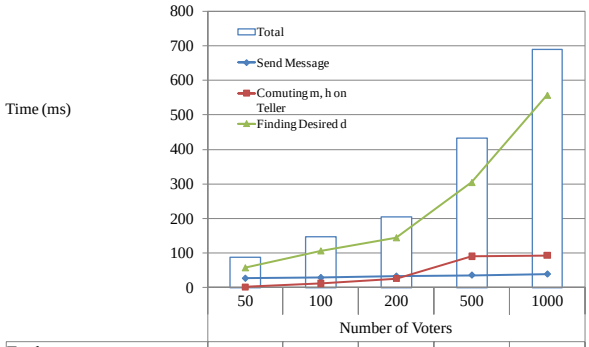
5. RESULT AND DISCUSSION

We implemented our algorithm in Delphi. All cryptographic operations use the OpenSSL and FGBigInt libraries. The OpenSSL library is accessible at www.openssl.org website. The IIS HTTP Server has been used for network simulation under windows Vista on a PC with a 2.4GHz processor and 2GB memory. We choose the 512 bits as the length of each cryptographic key. The main parameters of experiment are: Number of Clients and the range of d. Moreover, the time of computing d_i in each respondent will affect on final time. The key generation also is time consuming process, but these parameters are ignored in our experimental result. Because, as we mentioned before, these values can be precomputed offline before the protocol starts.

In our first algorithm result, we propose the time consumption of algorithm as the main factor. After offline precomputation steps, total process of algorithm includes some stages as you can see in Fig 2. These stages are:

- Computing $E(d_i)$ on respondent U_i
- Sending Encrypted message to Teller
- Computing *m*, *h* on Miner
- Finding desired *d* across its' possible values

The first step is independent from our parameters number of voters and range of *d*, because this step is local and self sufficient in each respondent. Therefore, we ignore this step in our final time result. Next two factors are depending to number of voters. The final result with enforce on these factors is shown in Fig 3. In this experiment, the respondents send the Boolean vote to Teller. We use this model to exterminate the last parameter, because in this condition, the range of *d* is equal to number of voters. We use 50, 100, 200, 500 and 1000 voters in our experiment and the earned time is base on the average of five algorithm runs. As you can see, the time offers a linear behavior related to number of voters. The total time of algorithm is shown by bars. For example, Teller computation takes 690 milliseconds for 1000 voters.



	50	100	200	500	1000
Total	87	148	204	432	690
Send Message	27	29	33	36	39
Comuting m, h on Teller	2	12	26	91	93
Finding Desired d	58	107	145	305	558

Fig 3. Total time of Frequency Counting Protocol related to Number of respondents

On the other hand, observing the finding desired summation loop on the Teller shows that in non Boolean cases, the time is highly related to range of *d*. The range of *d* is not a simple parameter and return to many conditions. It also is impressible from the domain of vote's options. The Fig 4 shows the time of algorithm execution related to number of voters and range of *d*. We earn the algorithm takes less than 0.5 second for 1000 voters with *d* in range of 1 to 1,000,000.

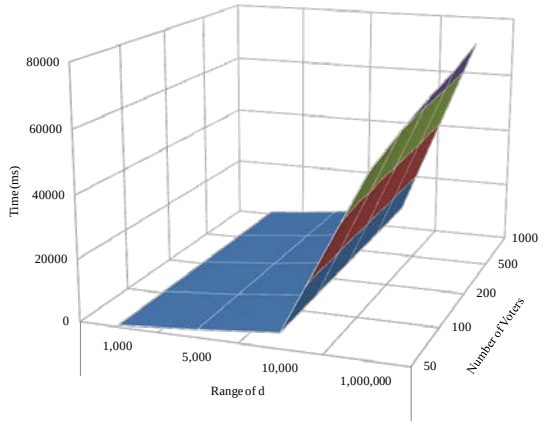


Fig 4. Total time related to Number of Voters and Range of d

In our second algorithm result, we propose the time consumption of algorithm as the main factor. The algorithm's stages are:

- Sending Encrypted message to Miner
- Computing *m*, *h*, *X*, *Y* on Miner
- Finding desired *d* across its' possible values

We define two different phase of result. In first phase, each client owns a Boolean data. In sooth, we are counting the users' data. This case is similar to E-Voting systems that each client is free to select an option and the Miner aims to know how many users do vote the idea. The final result of phase 1 is shown in Fig 5. In this experiment, the respondents send the Boolean data to Miner. In this condition, the range of d is equal to number of clients. We use 50, 100, 200, 500 and 1000 users in our experiment and the

earned time is base on the average of five algorithm runs. As you can see, the time offers a linear behavior related to number of users. For example, Miner computation takes 952 milliseconds for 1000 voters.

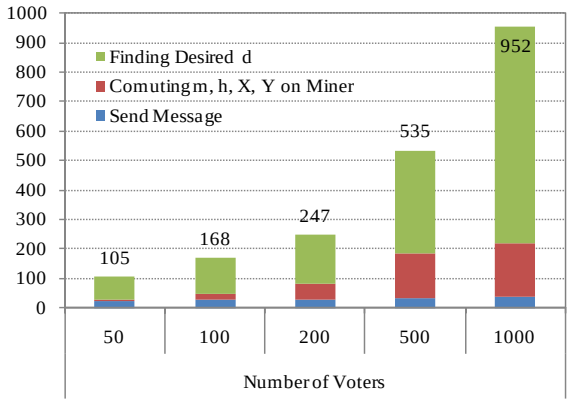


Fig 5. Total time of Frequency Counting Protocol related to Number of respondents

In second phase, we have focused on non Boolean cases. In this case, the time is highly related to range of *d*. The range of *d* is not a simple parameter and return to many conditions. The Fig 6, 7 show the time of algorithm execution related to number of clients and range of *d*. We earn the algorithm takes less than 0.5 second for 1000 voters with *d* in range of 1 to 1,000,000.

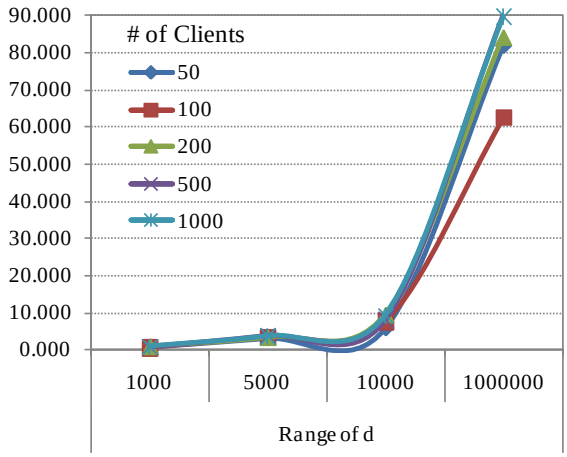


Fig 6. Total time related to Number of clients

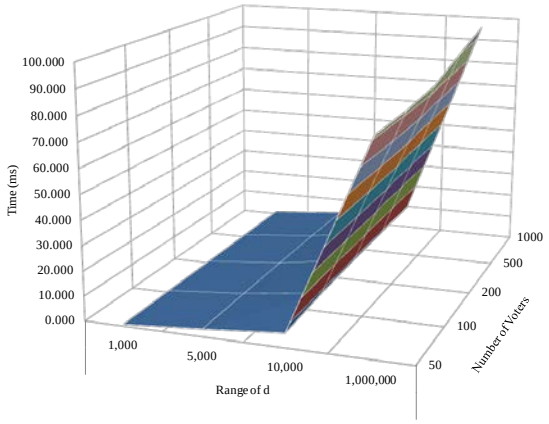


Fig 7. Total time related to Number of Clients and Range of d – 3d view

6. CONCLUSION

In this paper, we proposed a secure aggregation algorithm. Our proposed algorithm ensures the confidentially of sensitive respondents' data. Furthermore, it does not need communication channels between different respondents or multi round interaction between any respondent and the Miner Systems.

The mathematical demonstration ensures the accuracy of data's aggregation. On the other hand, The DDH assumption and the ElGamal cryptosystem ensure the privacy of the algorithm and respondents' data, so the Miner system can not reveal respondents' data. Our experimental results also show significantly desirable response time. The time increasing is derived from high security of algorithm, because the Miner could not able to decrypt the message, so the Miner should use trial and error to find the result.

The algorithm can also be used for any model enabled by counting values. Our both theoretical analysis and proof in addition to experimental results show that the algorithm is very efficient and runs in desirable time.

7. SIGNIFICANCE

Mining frequent itemsets is a fundamental and essential task in many data mining applications. These applications include the discovery of association rules, strong rules, correlations, sequential rules, episodes, multi dimensional patterns, and several other important discovery tasks. Association rules are widely used in various areas such as telecommunication networks, market and risk management, and inventory control. Association rules also are employed today in many application areas including Web usage mining, intrusion detection and bioinformatics.

8. ACKNOWLEDGMENTS

This project is supported by Ministry of Science, Technology and Innovation (MOSTI) through the E-Science Fund grant titled Development of Web Content Filtering System Based on Image Understanding Algorithm.

9. REFERENCES

[1] Agrawal, D. and Agrawal, C. 2001 On the design and quantification of privacy preserving data mining algorithms. Proceedings of 12th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems: pp 247–255.

[2] Agrawal, R., Imielinski, T. and Swami, A. 1993 Mining Association Rules Between Sets of Items in Large Databases. SIGMOD Conference: pp 207-216.

[3] Andrew Chi-Chih, Y. 1986 In Proceedings of the 27th Annual Symposium on Foundations of Computer ScienceIEEE Computer Society.

[4] Assaf, B.-D., Noam, N. and Benny, P. 2008 In Proceedings of the 15th ACM conference on Computer and communications securityACM, Alexandria, Virginia, USA.

[5] Bogetoft, P., Damgård, I., Jakobsen, T., Nielsen, K., Pagter, J. and Toft, T. 2006 In Financial Cryptography and Data Security, pp. 142-147.

[6] Boneh, D. 1998 The decision Diffie-Hellman problem. Lecture Notes in Computer Science.

[7] Canetti, R. 2000 Security and Composition of Multiparty Cryptographic Protocols. Journal of Cryptology, 13(1): 143-202.

[8] Dahlia, M., Noam, N., Benny, P. and Yaron, S. 2004 In Proceedings of the 13th conference on USENIX Security Symposium - Volume 13USENIX Association, San Diego, CA.

[9] David, C., Claude, C., peau and Ivan, D. 1988 In Proceedings of the twentieth annual ACM symposium on Theory of computingACM, Chicago, Illinois, United States.

[10] David, P. W. 2007 In Proceedings of the 26th annual international conference on Advances in CryptologySpringer-Verlag, Barcelona, Spain.

[11] Goldreich, O. 2004 Foundations of Cryptography, Cambridge Univ. Press.

[12] Goldreich, O., Micali, S. and Wigderson, A. 1987 In Proceedings of the nineteenth annual ACM symposium on Theory of computingACM, New York, New York, United States.

[13] Hirt, M. and Sako, K. 2000 Efficient receipt-free voting based on homomorphic encryption. In Advances in Cryptology - Proceedings of EUROCRYPT volume 1807 of Lecture Notes in Computer Science.

[14] Jarecki, S. and Shmatikov, V. 2007 Efficient Two-Party Secure Computation on Committed Inputs. EUROCRYPT.

[15] Lindell, Y. and Pinkas, B. 2007 An Efficient Protocol for Secure Two-Party Computation in the Presence of Malicious Adversaries. EUROCRYPT 52-79.

[16] Michael, B.-O., Shafi, G. and Avi, W. 1988 In Proceedings of the twentieth annual ACM symposium on Theory of computingACM, Chicago, Illinois, United States.

[17] Mohassel, P. and Franklin, M. 2006 In Public Key Cryptography - PKC 2006, pp. 458-473.

[18] Peter, B., Dan Lund, C., Ivan, D., rd, Martin, G., Thomas, J., Mikkel, K., igaard, Janus Dam, N., Jesper Buus, N., Kurt, N., Jakob, P., Michael, S. and Tomas, T. 2009 In Financial Cryptography and Data Security: 13th International Conference, FC 2009, Accra Beach, Barbados, February 23-26, 2009. Revised Selected PapersSpringer-Verlag, pp. 325-343.

[19] Pinkas, B., Schneider, T., Smart, N. and Williams, S. 2009 In Advances in Cryptology – ASIACRYPT 2009, pp. 250-267.

[20] Rizvi, S. J. and Haritsa, J. R. 2002 Maintaining data privacy in association rule mining. Proceedings of 28th International Conference on Very Large Data Bases: pp 682–693.

[21] Shamir, A. 1979 How to share a secret. Communications of the ACM: 22(11) 612–613.

[22] Vaidya, J. and Clifton, C. 2004 Privacy preserving naive Bayes classifier on vertically partitioned data. In Proceedings of 2004 SIAM International Conference on Data Mining.

[23] Yehuda, L., Benny, P. and Nigel, P. S. 2008 In Proceedings of the 6th international conference on Security and Cryptography for NetworksSpringer-Verlag, Amalfi, Italy.

[24] Yu, J. X., Chong, Z., Lu, H., Zhang, Z. and Zhou, A. 2006 A false negative approach to mining frequent itemsets from high speed transactional data streams. Inform. Sci, 176: 1986-2015.

[25] Zhang, L. and Zhang, B. 2005 Fuzzy reasoning model under quotient space structure. Inform. Sci: 353-364.