A New Multi Party Aggregation Algorithm Using Infinite Product Series

ABDOLREZA RASOULI, MOHD AIZAINI MAAROOF, MAHBOUBEH SHAMSI
Department of Computer Science and Information Systems
University of Technology, Malaysia
UTM, Skudai, Johor, Malaysia
rs.reza@gmail.com, aizaini@utm.my, mh.shamsi@yahoo.com

Abstract: - The worthwhile data mining tools encourage the companies to share their data to be mined. Whereas, the companies are avoided passing their data to the miner directly because of their privacy and confidentially roles. Multi Party Computation (MPC) is a cryptographic tool which consummates aggregation on distributed data with ensuring the privacy preserving of sensitive data. 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.

Key-Words: - Multi Party Computation, Privacy Preserving, Secure Counting Algorithm, Distributed Data

1 Introduction

Today, the large amount of data will be used among the large number of companies. There exist a lot of valuable patterns and roles which are hided between wooded data. Data mining tools has been developed to discover these worth facts. It is sophisticated where the data has been distributed among number of parties because of privacy of their sensitive data. Each company leans to gain the advantage of data mining tools, but he does not propend to share his sensitive data. Multi Party Computation (MPS) has been solved this conflict. MPC ensures the privacy preserving of data using cryptographic tools as well as precious aggregation over the distributed data.

In this paper, we propose a cryptographic method that allows a Miner system to compute aggregation of respondents' data. As we mentioned before, it is important that content of data should not be revealed. In this algorithm, each client only sends a message to Miner system and there is no need to any inner interaction between clients. The algorithm ensures that no extra information is revealed to Miner System except the summation of respondents' data.

Our algorithm could be used as basic fundamental for data mining tools. It is also useful for implementing an E-Voting system or calculation of web pages poll result.

First, we will briefly introduce some related work and basic cryptographic method in section 2, respectively and then we will fully explain the base algorithm in section 3. We explain the problem and proposed algorithm in section 4 and we represent our result in section 5. The paper will ends with a discussion about the earned result and precise conclusion in section 6.

2 Background and 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 [1] 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) [2-8]. 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 [9-13].

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 [14]. 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 [15, 16]. 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 [1], and in the GMW (Goldreich, Micali and Wigderson) multi-party protocol [4].

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.

2.1 ElGammel 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

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

ElGamal encryption is semantically secure under the Decisional Diffie-Hellman (DDH) assumption [17]. 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].

2.2 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 [18]. 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) ($1\leq i\leq k$) 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*.

3 System Architecture

The system includes possibly large numbers of clients who own their private data and a Miner who counts or sums their data. Although the Miner should be able to count or sum the data, but the system need to protect privacy of clients' data. It means that the miner compute the summation of data without revealing the client's exact data to neither himself nor any other clients. So the clients used an encryption system to send their data to Miner and the miner will compute the summation of their data without decrypting them.
In our scenario, there are n clients, so we call them $C_1, \ldots, C_n$; respectively. Each client owns his private data d_i. The aim of the Miner System is to calculate the sum $d = \sum_{i=1}^{n} d_i$ with ensuring the privacy of d_i.
In our model, because of its practicability, the clients do not need to know about other clients and they never communicate themselves. So there is no communication channel between different clients. Moreover, each client only sends one encrypted message to the Miner. So, they do not need multi round interaction between clients and the Miner System.

3.1 Aggregation Algorithm

The implemented algorithm [19] is based on the homomorphism property of mentioned ElGamal encryption [20]. 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 C_i has two pairs of keys: (($x_i \ mod_q$), ($X_i \ mod_q=(g^{x}{}_i \ mod_q)$), (($y_i \ mod_q$), ($Y_i \ mod_q=(g^{y}{}_i \ mod_q)$). We also define

$$(X \ mod_q) = \prod_{i=1}^{n}(X_i mod_q)$$
$$(Y \ mod_q) = \prod_{i=1}^{n}(Y_i \ mod_q)$$

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.

First, each respondent C_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

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

Then all clients send their encrypted message to Miner. The Miner System gathers all encrypted data together and computes m, h as:

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

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

$$(m \ mod_q) = (g^d \ mod_q).(h \ mod_q)$$
$$\Rightarrow \prod_{i=1}^{n}(m_i \ mod_q) = (g^d \ mod_q).\prod_{i=1}^{n}(h_i \ mod_q)$$
$$\Rightarrow (g^d \ mod_q) = \frac{\prod_{i=1}^{n}(m_i \ mod_q)}{\prod_{i=1}^{n}(h_i \ mod_q)}$$
$$= \frac{\prod_{i=1}^{n}(g^{d_i} \ mod_q).(X^{y_i} \ mod_q)}{\prod_{i=1}^{n}(Y^{x_i} \ mod_q)}$$
$$= \frac{\prod_{i=1}^{n}(g^{d_i} \ mod_q).\prod_{i=1}^{n}(X^{y_i} \ mod_q)}{\prod_{i=1}^{n}(Y^{x_i} \ mod_q)}$$
$$= \prod_{i=1}^{n}(g^{d_i} \ mod_q).\frac{\prod_{i=1}^{n}(X^{y_i} \ mod_q)}{\prod_{i=1}^{n}(Y^{x_i} \ mod_q)}$$
$$= g^{\sum_{i=1}^{n}(d_i \ mod_q)}.\prod_{i=1}^{n}\frac{((\prod_{j=1}^{n}(X_j \ mod_q)^{y_i} \ mod_q)}{((\prod_{j=1}^{n}(Y_j \ mod_q))^{x_i} \ mod_q)}$$
$$= g^{\sum_{i=1}^{n}(d_i \ mod_q)}.\prod_{i=1}^{n}\frac{((g^{\sum_{i=1}^{n}(x_j \ mod_q)})^{y_i} \ mod_q)}{((g^{\sum_{i=1}^{n}(y_j \ mod_q)})^{x_i} \ mod_q)}$$
$$= g^{\sum_{i=1}^{n}(d_i \ mod_q)}.\frac{g^{\sum_{i=1}^{n}\sum_{j=1}^{n}(x_j y_i \ mod_q)}}{g^{\sum_{i=1}^{n}\sum_{j=1}^{n}(y_j x_i \ mod_q)}} = g^{\sum_{i=1}^{n}(d_i \ mod_q)}$$
$$\Rightarrow (g^d \ mod_q) = g^{\sum_{i=1}^{n}(d_i \ mod_q)} \therefore d = \sum_{i=1}^{n} d_i \quad \blacksquare$$

This value is the desired summation of respondents' votes. The Miner cannot take discrete logarithms; the Miner 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)
```

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

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

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: (($x_i \ mod_q$), ($X_i \ mod_q=(g^{x}_i \ mod_q)$), (($y_i \ mod_q$), ($Y_i \ mod_q=(g^{y}_i \ mod_q)$) 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*.

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

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

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

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

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:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$= g^{\sum_{i=1}^{n}(d_i \ mod_q)}.\frac{(\prod_{i=1}^{n}(Y_i \ mod_q))^{\sum_{i=1}^{n}(\omega_i \ mod_q)}}{(\prod_{i=1}^{n}(X_i \ mod_q))^{\sum_{i=1}^{n}(\psi_i \ mod_q)}}.\frac{(X^{\sum_{i=1}^{n}\psi_i})}{(Y^{\sum_{i=1}^{n}\omega_i})}$$

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

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

5 Experimental Result

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 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.1. 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.

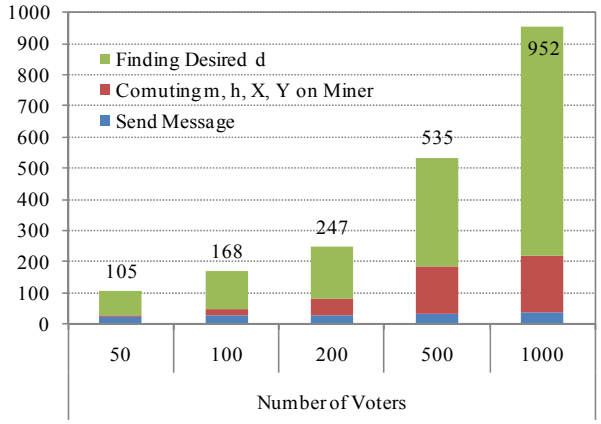

Fig 1. Total time of Frequency Counting Protocol related to Number of respondents

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.2 shows 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.

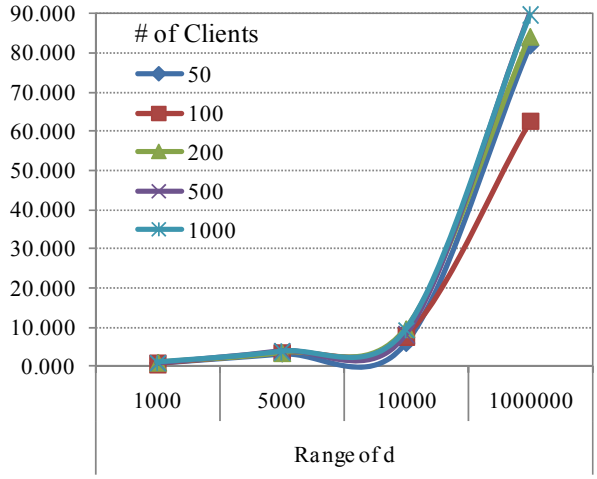

Fig 2. Total time related to Number of Voters and Range of d – 3d view

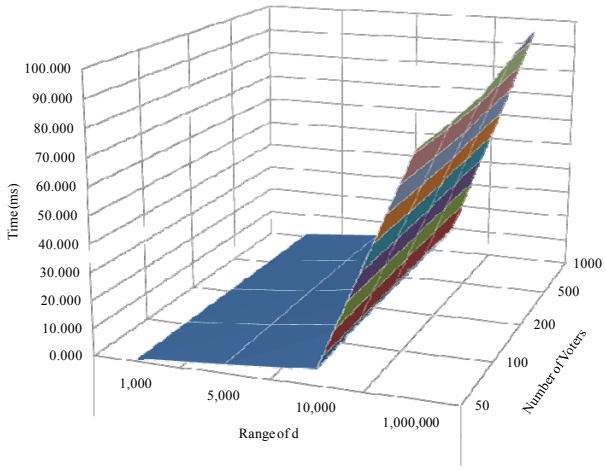

Fig 3. Total time related to Number of Voters and Range of d – 3d view

6 Discussion and 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.

References

[1]. Andrew Chi-Chih, Y., *How to generate and exchange secrets*, in *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*. 1986, IEEE Computer Society.

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

[3]. Goldreich, O., *Foundations of Cryptography*. Basic Applications. Vol. 2. 2004: Cambridge Univ. Press.

[4]. Goldreich, O., S. Micali, and A. Wigderson, *How to play ANY mental game*, in *Proceedings*

of the nineteenth annual ACM symposium on Theory of computing. 1987, ACM: New York, New York, United States.

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

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

[7]. Mohassel, P. and M. Franklin, *Efficiency Tradeoffs for Malicious Two-Party Computation*, in *Public Key Cryptography - PKC 2006*. 2006. p. 458-473.

[8]. David, P.W., *Revisiting the Efficiency of Malicious Two-Party Computation*, in *Proceedings of the 26th annual international conference on Advances in Cryptology*. 2007, Springer-Verlag: Barcelona, Spain.

[9]. Peter, B., et al., *Secure Multiparty Computation Goes Live*, in *Financial Cryptography and Data Security: 13th International Conference, FC 2009, Accra Beach, Barbados, February 23-26, 2009. Revised Selected Papers*. 2009, Springer-Verlag. p. 325-343.

[10]. Bogetoft, P., et al., *A Practical Implementation of Secure Auctions Based on Multiparty Integer Computation*, in *Financial Cryptography and Data Security*. 2006. p. 142-147.

[11]. Yehuda, L., P. Benny, and P.S. Nigel, *Implementing Two-Party Computation Efficiently with Security Against Malicious Adversaries*, in *Proceedings of the 6th international conference on Security and Cryptography for Networks*. 2008, Springer-Verlag: Amalfi, Italy.

[12]. Dahlia, M., et al., *Fairplay\&\#8212;a secure two-party computation system*, in *Proceedings of the 13th conference on USENIX Security Symposium - Volume 13*. 2004, USENIX Association: San Diego, CA.

[13]. Assaf, B.-D., N. Noam, and P. Benny, *FairplayMP: a system for secure multi-party computation*, in *Proceedings of the 15th ACM conference on Computer and communications security*. 2008, ACM: Alexandria, Virginia, USA.

[14]. Pinkas, B., et al., *Secure Two-Party Computation Is Practical*, in *Advances in Cryptology – ASIACRYPT 2009*. 2009. p. 250-267.

[15]. Michael, B.-O., G. Shafi, and W. Avi, *Completeness theorems for non-cryptographic fault-tolerant distributed computation*, in *Proceedings of the twentieth annual ACM symposium on Theory of computing*. 1988, ACM: Chicago, Illinois, United States.

[16]. David, C., et al., *Multiparty unconditionally secure protocols*, in *Proceedings of the twentieth annual ACM symposium on Theory of computing*. 1988, ACM: Chicago, Illinois, United States.

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

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

[19]. Rasouli, A., et al. *A Robust and High Speed E-Voting Algorithm Using ElGammel CryptoSystem*. in *The 2nd International Conference on Computer and Automation Engineering (ISI Indexed)*. 2010. Singapore.

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