DOI: 10.1109/ICCAE.2010.5451877

A Robust and High Speed E-Voting Algorithm Using ElGammel CryptoSystem

Abdolreza Rasouli Kenari[1], Javad Hosseinkhani[2], Mahboubeh Shamsi[3], Majid Harouni[4]
Department of Computer Science and Information System
Islamic Azad University, Bardsir Branch, Iran1,3 - University of Technology, Malaysia (UTM)2,4
rs.reza@gmail.com, jhkhani@gmail.com, mahboubehshamsi@yahoo.com, majid.harouni@gmail.com

***Abstract*—Nowadays Multi-Party computation (MPC) is a big challenge in cryptography theory that has a potential power to solve the real-life problems. But there is no much practical implementation of MPC. In this paper, we have represented a new robust algorithm for counting the respondent's votes on an electronic poll with ensuring the confidentially of their votes. The algorithm shows the extremely good response time, so the time consuming for gathering votes of 1000 respondents is only 0.5 milliseconds. The mathematic demonstration ensures the fully privacy preserving of the algorithm. The experiments show that the algorithm is highly applicable for real world problems.**

Keywords-component; e-voting; robustness; elGammel; Cryptography, Multi-Party Computation.

I. INTRODUCTION

You as a citizen may join the elections during your life, like as president or senate election. The classic process of election and counting all votes is a so time and cost consuming process. Today, all developed countries leans to modern E-Voting systems. On the other hand, almost all sites use polls to know their users' opinion. You can see polls in all around World Wide Web. The most important criteria in counting votes is voters' privacy preserving. It means that, even tough, the teller system should be able to count the summation of votes, but he shouldn't able to know the exact private opinion of a voter. Because of the votes are distributed in many parties and ensuring votes' confidentially, we need an algorithm to send the encrypted votes to Teller System and computes the poll result over multi parties.

In this paper, we propose a cryptographic method that allows a teller system to compute frequencies of respondents' votes. As we mentioned before, it is important that content of votes shouldn't be revealed. In this algorithm, each respondent only sends a message to teller system and there is no need to any inner interaction between respondents. The algorithm ensures that no extra information is revealed to Teller System except the frequency of respondents' votes.

Our algorithm could be used as basic fundamental for Electronic Voting systems. It is also useful for government that wants to pass from classic voting systems to new network-based poll formally known as E-Voting.

First, we will briefly introduce some related work and basic cryptographic method in section II.A and II.B, respectively and then we will fully explain the algorithm in section III. We represent our result in section IV and the paper will ends with a discussion about the earned result and precise conclusion in section V.

II. BACKGROUND

A. 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) [5, 8, 9, 10, 12, 13, 15]. 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 [2, 3, 6, 16, 19].

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 [17]. 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 [7, 14]. 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 [10].

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

978-1-4244-5586-7/10/$26.00 ©2010 IEEE

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. We have focused on a big real life challenge on Electronic Voting Systems.

B. 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 [4]. 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 $[1, q]$.

C. 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.*

III. Proposed Algorithm

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.

A. 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 a option that respondents could select such as questionary forms. The Teller System would like to find out the sum $d = \sum_{i=1}^{n} d_i$ 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.

B. E-Voting Algorithm

The implemented algorithm is based on the homomorphism property of mentioned ElGamal encryption [11]. 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: $((x_i \ mod_q)$, $(X_i \ mod_q=(g^x{}_i \ mod_q))$, $((y_i \ mod_q)$, $(Y_i \ mod_q=(g^y{}_i \ mod_q))$. We also define

$$\left(X \ mod_q\right) = \prod_{i=1}^{n}\left(X_i mod_q\right)$$

$$\left(Y \ mod_q\right) = \prod_{i=1}^{n}\left(Y_i \ mod_q\right)$$

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 figure 1.

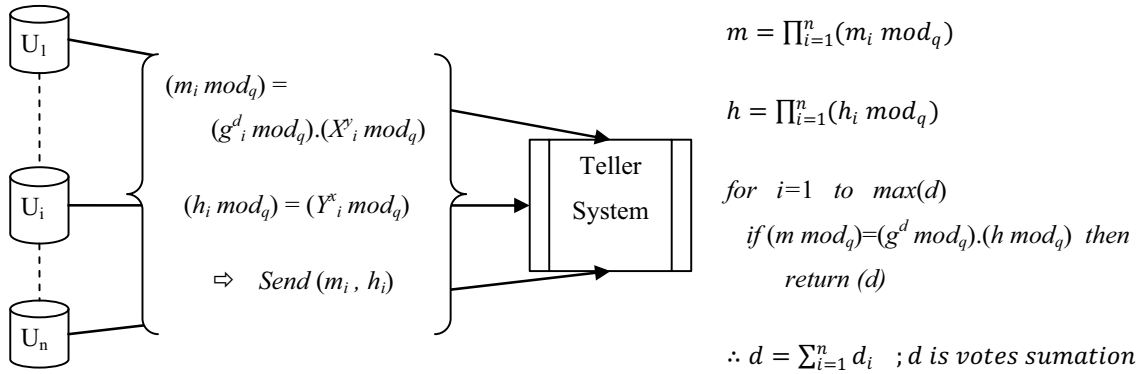


Figure 1. E-Voting System Architecture

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

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

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

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

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

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

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.

C. Algorithm Demonstration

In this section, we prove that the algorithm represented in Figure.1 correctly computes the summation of respondents' votes. Suppose that the Teller finds a *d* so

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

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

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

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

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

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

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

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

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

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

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

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

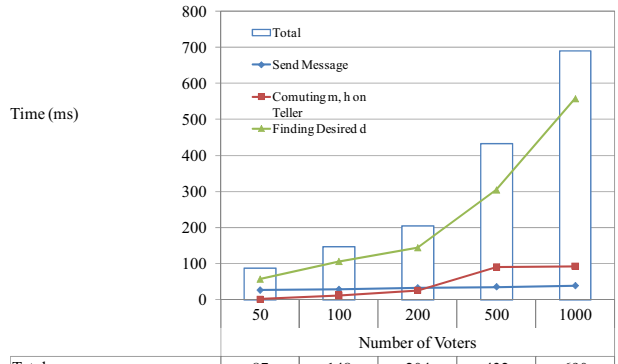
IV. EXPRIMENTAL RESULT

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 respondents (Voters) and the range of d. Moreover, the time of computing vote d_i in each respondent will affect on final time. The key generation time and computing the algorithm parameters X and Y for each respondent also is time consuming process, but these two last 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. After offline precomputation steps, total process of algorithm includes some stages as you can see in Figure.1. 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 Figure.2. 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.



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

Figure 2. 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 Figure.3 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.

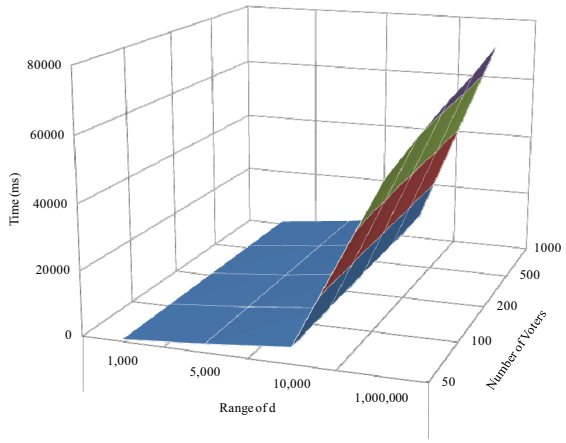


Figure 3. Total time related to Number of Voters and Range of d

V. DISCUSSION AND CONCLUSION

In this paper, we proposed a Secure E-Voting algorithm as fundamental algorithm for election systems. Our proposed algorithm ensures the confidentially of sensitive respondents' votes. Furthermore, it doesn't need communication channels between different respondents or multi round interaction between any respondent and the Teller Systems.

The mathematical demonstration ensures the accuracy of vote's counting. On the other hand, The DDH assumption and the ElGamal cryptosystem ensure the privacy of the algorithm and respondents' votes, so the Teller system can't reveal respondents' votes. Our experimental results also show significantly desirable response time for election system where votes are Boolean. In case of valuable votes, the response time is coveted. The time increasing is derived from high security of algorithm, because the Teller couldn't able to decrypt the message, so the Teller should use trial and error to find poll 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*: IEEE Computer Society, 1986.

[2] 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*. Alexandria, Virginia, USA: ACM, 2008.

[3] Bogetoft, P., I. Damgård, T. Jakobsen, K. Nielsen, J. Pagter, and T. Toft. "A Practical Implementation of Secure Auctions Based on Multiparty Integer Computation." In *Financial Cryptography and Data Security*, 142-47, 2006.

[4] Boneh, D. "The Decision Diffie-Hellman Problem." *Lecture Notes in Computer Science* 1998.

[5] Canetti, R. "Security and Composition of Multiparty Cryptographic Protocols." *Journal of Cryptology* 13, no. 1 2000: 143-202.

[6] Dahlia, M., N. Noam, P. Benny, and S. Yaron. "Fairplay\&\#8212;a Secure Two-Party Computation System." In *Proceedings of the 13th conference on USENIX Security Symposium - Volume 13*. San Diego, CA: USENIX Association, 2004.

[7] David, C., C. Claude, peau, and D. Ivan. "Multiparty Unconditionally Secure Protocols." In *Proceedings of the twentieth annual ACM symposium on Theory of computing*. Chicago, Illinois, United States: ACM, 1988.

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

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

[10] 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*. New York, New York, United States: ACM, 1987.

[11] Hirt, M., and K. Sako. "Efficient Receipt-Free Voting Based on Homomorphic Encryption." *In Advances in Cryptology - Proceedings of EUROCRYPT* 2000: volume 1807 of Lecture Notes in Computer Science.

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

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

[14] 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*. Chicago, Illinois, United States: ACM, 1988.

[15] Mohassel, P., and M. Franklin. "Efficiency Tradeoffs for Malicious Two-Party Computation." In *Public Key Cryptography - Pkc 2006*, 458-73, 2006.

[16] Peter, B., C. Dan Lund, D. Ivan, rd, G. Martin, J. Thomas, K. Mikkel, igaard, N. Janus Dam, N. Jesper Buus, N. Kurt, P. Jakob, S. Michael, and T. Tomas. "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*, 325-43: Springer-Verlag, 2009.

[17] Pinkas, B., T. Schneider, N. Smart, and S. Williams. "Secure Two-Party Computation Is Practical." In *Advances in Cryptology – Asiacrypt 2009*, 250-67, 2009.

[18] Shamir, A. "How to Share a Secret." *Communications of the ACM* 1979: 22(11) 612–13.

[19] 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*. Amalfi, Italy: Springer-Verlag, 2008.