A comparison between symmetric and asymmetric key encryption algorithm based decryption mixnets

Article · February 2015

DOI: 10.1109/NSysS.2015.7043532

Sahan Ahmad
University of Louisiana at Lafayette

Kazi Md. Rokibul Alam
Khulna University of Engineering and Technology

A Comparison between Symmetric and Asymmetric Key Encryption Algorithm based Decryption Mixnets

Sahan Ahmad[+], Kazi Md. Rokibul Alam[+], Habibur Rahman[+], and Shinsuke Tamura[*]
[+]Department of Computer Science and Engineering, Khulna University of Engineering and Technology
Khulna-9203, Bangladesh
[*]Graduate School of Engineering, University of Fukui,
Fukui 910-8507, Japan
Email: sahan.ahmad@gmail.com[+], rokibcse@yahoo.com[+], habib090722@gmail.com[+], tamura@dance.plala.or.jp

***Abstract*— This paper presents a comparison between symmetric and asymmetric key encryption algorithm based decryption mixnets through simulation. Mix-servers involved in a decryption mixnet receive independently and repeatedly encrypted messages as their input, then successively decrypt and shuffle them to generate a new altered output from which finally the messages are regained. Thus mixnets confirm unlinkability and anonymity between senders and the receiver of messages. Both symmetric (*e.g.* onetime pad, AES) and asymmetric (*e.g.* RSA and ElGamal cryptosystems) key encryption algorithms can be exploited to accomplish decryption mixnets. This paper evaluates both symmetric (*e.g.* ESEBM: enhanced symmetric key encryption based mixnet) and asymmetric (*e.g.* RSA and ElGamal based) key encryption algorithm based decryption mixnets. Here they are evaluated based on several criteria such as: the number of messages traversing through the mixnet, the number of mix-servers involved in the mixnet and the key length of the underlying cryptosystem. Finally mixnets are compared on the basis of the computation time requirement for the above mentioned criteria while sending messages anonymously.**

Keywords*— *Anonymity, Mixnet, Privacy, Protocol, RSA, ElGamal, Symmetric key encryption algorithm

I. Introduction

In cryptographic social applications such as electronic voting system [1], location privacy in wireless network [2] and so on, concealing identities of message senders are as important as concealing messages themselves. Anonymous networks [3, 4] provide frameworks that enable message senders to send their messages while concealing their identities *i.e.* anonymously, to the receiver and to any observer of the communication. Thus finally received messages at the receiver end can never be traced back to their original senders unless all mix-servers conspire. Nowadays gradually more people are involving in network based communication; therefore the demands of anonymous networks are increasing [5]. Practically anonymous networks are implemented through mixnets [6].

In decryption mixnets, senders are required to encrypt messages with the keys of mix-servers. Hence, mix-servers can change the appearance of their input batch by decrypting with their keys and shuffling. As described in [4], a decryption mixnet consists of l mix-servers works as follows. Let K_i be the public key of ith mix-server. A sender merges a message m_j with a secret string r_j as $(m_j \| r_j)$, then encrypts as, $EK_1(EK_2(\ldots(EK_l(m_j \| r_j))\ldots))$ and broadcasts it. Mix-server i with private key K_i^{-1}, receives input batch as, $EK_i(EK_{i+1}(\ldots(EK_l(m_j \| r_j))\ldots))$ from mix-server$_{i-1}$.

Input: $EK_1(EK_2(\ldots(EK_l(m_j \| r_j))\ldots)); j = 1,\ldots,n$.
For $i = 1,\ldots,l$. For $j = 1,\ldots,n$. Here n is the number of message senders and l is the number of mix-servers involved in the mixnet.

Step 1: Decrypt as $DK_i\, EK_i(EK_{i+1}(\ldots(EK_l(m_j \| r_j))\ldots)) = EK_{i+1}(\ldots(EK_l(m_j \| r_j))\ldots)$.
Step 2: Lexicographically order all decrypted quantities obtained in Step 1.
Output: $\{m_j\}_R$, a batch of mixed messages that cannot be traced back to senders.

This paper performs the simulation of a symmetric key encryption algorithm based and 2 asymmetric key encryption algorithm based decryption mixnets *i.e.* ESEBM [5], RSA and ElGamal based decryption mixnets [3, 7]. Here mixnets are evaluated employing both 1024 bit and 2048 bit encryption while varying the number of messages and the number of mix-servers. Finally the evaluation provides some guidelines such as:

- Which decryption mixnet implementation is faster?
- Does the performance of mixnet depend on the number of messages?
- Does the key length of the underlying cryptosystem affect the speed?
- Does the number of mix-servers involved in a decryption mixnet influence the time of total transmission of the messages from the sender to the receiver?

The outline of this paper is as follows: Section 2 describes the chosen mixnets. Section 3 presents the experimental analysis including experimental results and discussions among the mixnets. Finally, concluding remarks are explained in Section 4.

II. Decryption Mixnets

A. Enhanced symmetric key encryption based mixnet (ESEBM)

As described in [5], ESEBM is one kind of decryption mixnet in which asymmetric key encryption functions are

978–1–4799–8162–7/15/$31.00 ©2015 IEEE

replaced by symmetric ones. Here the encryption keys used for sending messages are distributed to senders in advance in order to make the network stable enough for encryption and decryption purposes of mix-servers. Therefore there is no need to add key or additional information as an overhead with the message. Thus the process of ESBM is very simple and it saves additional computation time for calculating overhead.

Fig. 1 shows the relationship among the components of ESEBM in which onetime pad algorithm is used for encryption and decryption purposes. Therefore the length of the message must be equal to the key length of ESEBM. It consists of two parts, *i.e.* the anonymous channel and the concealing pattern (CP) generator. The anonymous channel is configured as a sequence of N mix-servers as same as a decryption mixnet, and the CP generator consists of Z-groups, where the j-th group is configured by N_j mix-server, and each mix-server in the anonymous channel is corresponded to a single mix-server in the *CP* generator and vice versa, therefore $N = N_1 + N_2 + \text{---} + N_Z$. Here each mix-server contains a table known as CP table that holds at most G number of various decryption keys and from the tag part it determines which key should be used for decryption. Any sender S sends her message m_S to the first mix-server T_1 in the anonymous channel while encrypting it to $m_S \oplus X(h)$ where $X(h) = x_1(h) \oplus x_2(h) \oplus\text{------}\oplus x_N(h)$. If necessary, message m_S can be divided into multiple frames of length L_m.

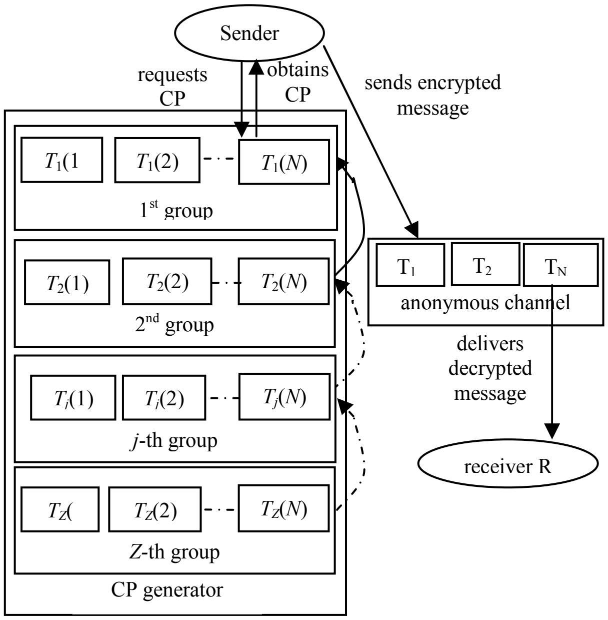

Fig. 1. Relationships among the components of ESEBM.

As same as usual decryption mixnet, each mix-server in the anonymous channel stores its receiving messages until it receives the predefined number of messages. Then after decryption and shuffling, each mix-server forwards the messages to its neighboring mix-server finally to be sent to the receiver. Here any mix-server T_j decrypts its receiving encrypted message m_S by simply XORing it by its CP constructor $x_j(h)$ that constitutes $X(h)$, the CP that sender S had used to encrypt m_S. Here it is apparent that $m_S \oplus X(h)$ is transformed to m_S when all mix-servers decrypt it. On the other hand, because each mix-server knows only its CP constructor x_j (h) in $X(h)$, no one can know the sender of message m_S unless all mix-servers conspire with each other as same as in usual decryption mixnets.

The message part maintains encrypted message m_S *i.e.* $m_S \oplus X(h)$, and the tag part maintains a sequence of tag vector (TV), *i.e.* $Q(h) = \{Q_1(h), Q_2(h), \text{---}, Q_N(h)\}$, where server T_j that had generated the CP constructor $x_j(h)$ to construct $X(h)$ can know $x_j(h)$ from $Q_j(h)$ through CP table as shown within anonymous channel of Fig. 1. Here $Q_j(h)$ is constructed so that no one can trace the message by it and no one except mix-server T_j can identify $x_j(h)$ from it. Fig. 2 shows the message structure of ESEBM.

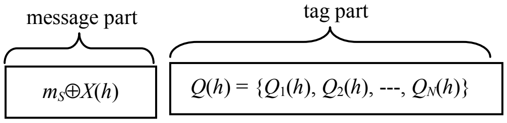

Fig. 2. Message structure of ESEBM.

The first mix-server T_1 that receives $\{x_1(h) \oplus x_2(h) \oplus\text{---}\oplus x_N(h) \oplus m_S, Q_1(h), Q_2(h),\text{---},Q_N(h)\}$ retrieves CP constructor $x_1(h)$ and TV constructor $q_1(h)$ from its CP table based on $ID_1(x_1(h), q_1(h))$ in $Q_1(h)$. Now for decryption, mix-server T_1 calculates XOR of $x_1(h)$ and m_S, to discard $q_1(h)$ as well as $Q_1(h)$. Therefore $\{x_1(h) \oplus x_2(h) \oplus\text{---}\oplus x_N(h) \oplus m_S, Q_1(h), Q_2(h),\text{---},Q_N(h)\}$ becomes $m_S = x_1(h) \oplus (x_1(h) \oplus x_2(h) \oplus\text{---}\oplus x_N(h) \oplus m_S) = x_2(h) \oplus x_3(h) \oplus\text{--}\oplus x_N(h) \oplus m_S$, $Q_j(h) = q_{1j}(h) \oplus (q_{1j}(h) \oplus q_{2j}(h) \oplus\text{---}\oplus q_{G\mathrm{j}}(\mathrm{h}) \oplus ID_{1j}(x_j(h), q_j(h))) = q_{2j}(h) \oplus q_{3j}(h) \oplus\text{---}\oplus q_{Gj}(h) \oplus ID_{1j}(x_j(h), q_j(h))$. Thus T_1 removes $Q_1(h)$ from the tag part, waits for the arrival of predefined number of messages, and shuffles them to forward each result to mix-server T_2. Here all mix-servers involved in ESEBM perform their operations in the same way which is shown in Fig. 3.

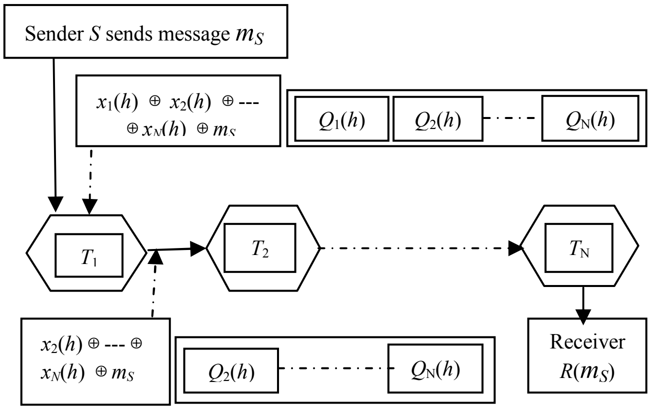

Fig. 3. Working Procedure of ESEBM.

B. *RSA based Decryption Mixnet*

The mechanism proposed in [8] introduced RSA based decryption mixnet. Herein using RSA cryptosystem, senders encrypt their messages with the public keys of the mix-servers involved in a mixnet. Then mix-servers decrypt their input batch with corresponding private keys that change the appearance of the messages and randomly shuffles that ensure unlinkability between input and output batch. As shown in [6], Fig. 4 graphically shows the decryption operation of sender's messages by a decryption mixnet. A RSA based decryption mixnet as described in [3] proceeds as follows.

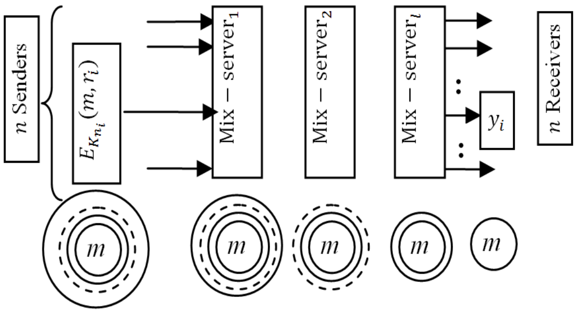

Fig. 4. Message sending through a decryption mixnet. Each mix-server decrypts and shuffles messages.

A sender i concatenates its message with a random string r and encrypts it with the public keys of l mix-servers in the anonymous path as-

forwarding onion$_l = E_{Kx} \| r_l$
forwarding onion$_{l-1} = E_{Kl}$ (*forwarding onion*$_l$) $\| r_{l-1}$
..
...
forwarding onion$_j = E_{Kj+1}$ (*forwarding onion*$_{j+1}$) $\| r_j$
forwarding onion$_0 = E_{K1}$ (*forwarding onion*$_1$) $\| r_0$
$E_K(m, r) =$ *forwarding onion*$_0$

Here $K = (K_1, K_2, K_3, \ldots, K_l)$ are the public keys of l mix-servers and $A_1, \mathrm{A}_2,\ldots,A_l$ are the addresses of l mix- servers. $r = r_1, r_2, \ldots, r_l$ are the random strings used to randomize the encryption at each mix-server. K_X is the public key of the receiver. So the sender broadcasts an l layer onion which is given as $E_K(m, r) =$

$$E_{K1}(E_{K2}(\ldots E_{Kl}(E_{KX}(m\|r_l)) \| r_{l-1} \ldots) \| r_1) \| r_0 \qquad (1)$$

Each mix-server j on the path peels off a layer from the onion, *i.e.*, decrypts using the private key $D_{Kj = Kj}$ as

$$D_{Kj}(E_{Kj}(\textit{forwarding onion}_{j+1})) \qquad (2)$$

Thus mix-server j decrypts other onions of its input batch, received from the mix server j–1. After the completion of the decryption operation, mix-server j shuffles all the onions of the batch using a random permutation $\pi_{j:\, n \to n}$ where n is the batch size. This concludes the mixing operation of mix-server j. The resulting quantities are the forwarding onions which have been reduced in size and constitute the mixed output batch of mix-server j. Theses onions are forwarded simultaneously to the next mix-server j+1. All the remaining mix-servers, on the path, repeat the same operations until the last mix-server l outputs the decrypted quantity $E_{KX}(m)$, which is send to the receiver.

C. Elgamal based Decryption Mixnet

In ElGamal based decryption mixnet as described in [7], a sender i only needs to perform a single encryption for all the l mix-servers as

$$E_K(m, r) = (g^r \| mK^r) \qquad (3)$$

where g is a generator, r is a random string and K is the public key of the mixnet, computed from the public keys of the mix-servers as-

$$K = \prod\nolimits^l_{j=1} K_j = \prod\nolimits^l_{j=1} g^d{}_j = g^{\Sigma l}{}_{\mathrm{j}=1}d_j \qquad (4)$$

where $K_j = g^d{}_j$ and d_j, are the public and private key, respectively, of mix-server j. Any mix-server j can randomly decrypt the sender i's input, using its private key d_j and random string r_j as-

$$D_{Kj}(E_K(m, r)) = g^r g^{rj} \| mK^r (g^r)^{-d}{}_j (\prod\nolimits^l_{\mathrm{a}=1,\, \mathrm{a} \neq j} g^d{}_a)^{rj}$$
$$= g^{r+rj} \| m (\mathrm{g}^{\Sigma \mathrm{l}}{}_{a=1,\, a \neq j}(d_a{}^r g^{d\,r}_{\;j}))(g-^{d\,r}_{\;j})(\prod\nolimits^l_{\mathrm{a}=1,\, \mathrm{a} \neq j}(g^d{}_a r_j))$$
$$= g^{r+rj} \| m (\prod\nolimits^l_{\mathrm{a}=1,\, \mathrm{a} \neq j} g^d{}_a r_j) \qquad (5)$$

The first step of (5), the mix-server j uses the first component of its input in (3) to obtain $(g^r)^{-dj}$ and uses the product of public keys of the mix-servers that are yet to decrypt to obtain $(\prod^l_{\mathrm{a}=1,\, \mathrm{a} \neq j}(g^d{}_a))^{rj}$. After the decryption of more inputs to form a batch, the mix-server j broadcasts the mixed batch to the remaining l–1 mix-servers. The process repeats, with another mix-server performing the decryption, until finally all l mix-servers have decrypted using their private keys to obtain $gr+\Sigma^l_{j=1} r_j \| m$.

III. Experimental Analysis

This section evaluates the performance of the considered mixnets based on varying the number of messages, the number of mix-servers involved in mixnets and the key length of the underlying cryptosystem.

A. Experimental Setup

To perform the experiments, an environment consists of a 64 bit windows machine (windows 8.1), Dual-core 2.60 GHz CPU with 2.0 GB of RAM is used. Language Java, Android developer tool (eclipse) with jdk 1.6.0 [9] is used for coding purpose. While simulating, the address of sender-receiver is considered as a static member function of Java, and router to router travelling time is assumed to be negligible *i.e.* all computation times do not include the communication time. Moreover different operations that are not related to cryptography are not considered. RSA and ElGamal based decryption mixnets are simulated using the key length of 1024 bit and 2048 bit while keeping the message length 40 digit. For ESEBM, both the message length and the key length are kept 300 and 600 digit. To prevent trace back from output to input, Fisher Yates [10] algorithm is used for shuffling. For RSA encryption key generation, Euclidian algorithm [11] is an efficient method and is used to compute the greatest common divisor (GCD) of two integers. Extended Euclidian algorithm [11] is also used to find the modular multiplicative inverse of encryption key *i.e.* to generate decryption key.

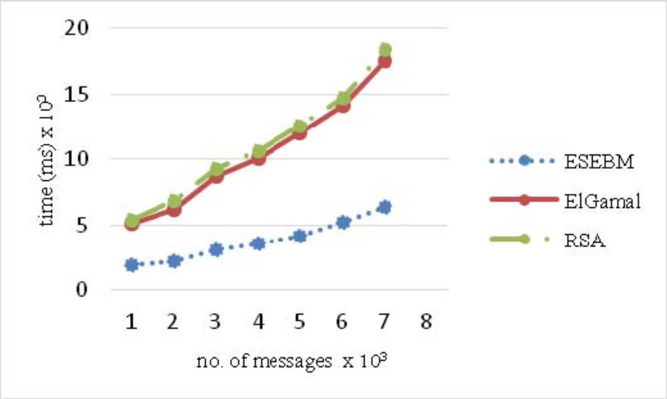

Fig. 5. Decryption time for ESEBM, RSA and ElGamal based decryption mixnets for 1024 bit key length where messages vary from 1000 to 7000.

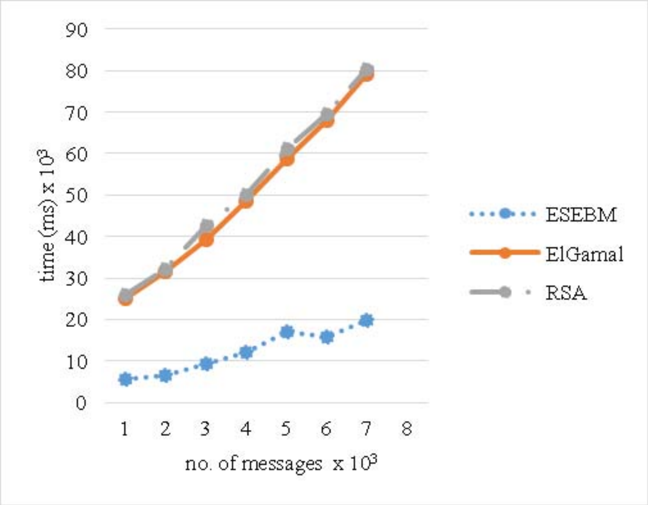

Fig. 6. Decryption time for ESEBM, RSA and ElGamal based decryption mixnets for 2048 bit key length where messages vary from 1000 to 7000.

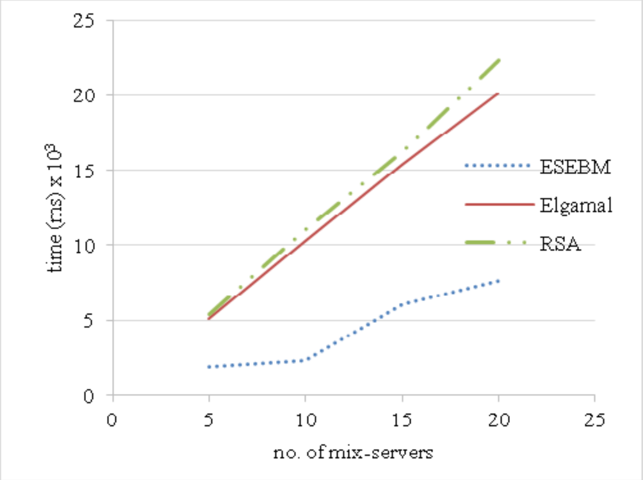

Fig. 7. Decryption time for ESEBM, RSA and ElGamal based decryption mixnets for 1024 bit key length where 1000 messages pass throuh 5 to 20 mix-servers.

B. *Experimental Results and Discussions*

In any decryption mixnet, usually the number of decryptions is equal to its number of mix-servers. Fig. 5 shows the decryption time requirement of the chosen mixnets employing 5 mix-servers while varying messages from 1000 to 7000 where the key length is 1024 bit. Similarly Fig. 6 shows the same scenario where the key length is 2048 bit. Here among the three chosen mixnets, ESEBM requires lowest time to send messages from the sender to the receiver because of the simplicity of encryption and decryption function. It uses onetime pad in which the execution time of bit-wise XOR operation occurs very fast. On the other hand RSA and ElGamal based decryption mixnets takes huge time because of large mathematical computation and the extra overhead of encrypted messages. Fig. 7 shows the decryption time requirement of 1000 messages passing through 5 to 20 mix-servers where the key length is 1024 bit. When the key length and the number of mix-servers increase, it takes more time for decryption (but at the same time the security also increase) *i.e.* the time requirement is proportional with number of messages, number of mix-servers as well as the key length.

In case of ElGamal based decryption mixnet, it require less time than RSA based decryption mixnet because there is only a single encryption on sender's side. The experimental results presented in Fig. 5 and Fig. 6 show that while message traversing (*i.e.* decryption), ESEBM is almost 3.1 times faster than RSA and ElGamal based decryption mixnets.

IV. Conclusions

In this paper the experimental analysis of symmetric key encryption algorithm based ESEBM and asymmetric key encryption algorithm *i.e.*, RSA and ElGamal based decryption mixnets have been presented. Nowadays due to the popularity of numerous cryptographic social applications, the importance of decryption mixnets are raising, therefore their performances have been evaluated. According to the considered criteria the experimental results show that for every case, ESEBM requires the lowest time among the chosen mixnets. The reason is that it uses optimal cryptographic technique 'onetime pad' for encryption and decryption purposes and does not require large number of overhead for message while passing. RSA based decryption mixnet is very slow because of repeated encryptions by the sender and requires more time than other 2 mixnets. ElGamal based decryption mixnet requires less time than RSA based decryption mixnet because a sender only needs to perform a single encryption for all mix-servers.

References

[1] A. K. M. Rokibul, S. Tamura, S. Taniguchi, and T. Yanase, "An Anonymous Voting Scheme based on Confirmation Numbers," IEEJ Transactions EIS, Vol. 130, No. 11, pp. 2065-2073, November 2010.

[2] L. Huang, K. Matsuura, H. Yamane, and K. Sezaki, "Enhancing wireless location privacy using silent period," in Proc. IEEE Wireless Communications Networking Conference, 2005.

[3] K. Sampigethaya, R. Poovendran, "A Survey on Mix Networks and Their Secure Applications" in Proc. IEEE, Vol. 94, Issue.12, pp. 2142–2181, Dec. 2006.

[4] K. Sampigethaya; R. Poovendran, "A Framework and Taxonomy for Comparison of Electronic Voting Schemes" in Elsevier Computers and Security, Vol. 25, pp. 137-153, 2006.

[5] H. Haddad, S. Tamura, S. Taniguchi, and T. Yanase, “Development of anonymous networks based on symmetric key encryptions,” in Journal of Networks, Vol. 6, No. 11, pp.1533–1542, November, 2011.

[6] N. Islam, A. K. M. Rokibul, and A. Rahman,“The effectiveness of mixnets – an empirical study,” in Elsevier Computer Fraud & Security, Vol. 2013, Issue 12, pp. 9–14, December, 2013.

[7] C. Park, K. Itoh, and K. Kurosawa, “Efficient anonymous channel and all/nothing election scheme,” in Advances in Cryptology-Eurocrypt. New York: Springer-Verlag, pp. 248–259, 1994.

[8] D. Chaum, “Untraceable electronic mail, return addresses, and digital pseudonyms,” Commun. ACM, Vol. 24, No. 2, pp. 84–88, 1981.

[9] Java software available at http://www.eclipse.org/ on May, 2014.

[10] R.A Fisher, F. Yates, “Statistical tables for biological, agricultural and medical research (3rd ed.)”. London: Oliver & Boyd. pp. 26–27, 1948.

[11] A. J. Menezes, P. C. V. Oorschot and S. A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1996.