Development of Anonymous Networks Based on Symmetric Key Encryptions

Development of Anonymous Networks Based on Symmetric Key Encryptions

Hazim Haddad
University of Fukui, Fukui, Japan
Email: hazimhaddad@yahoo.com

Shinsuke Tamura, Shuji Taniguchi and Tatsuro Yanase
University of Fukui, Fukui, Japan
Email: {tamura, stamigut, yanase}@u-fukui.ac.jp

***Abstract*—Anonymous networks enable entities to send messages without disclosing their identities. Many anonymous networks had been proposed already, such as Mixnet, DC-net, Crowds, etc., however, they still have serious drawbacks. Namely, they require tremendous computation overheads to transmit messages over networks. That is because asymmetric key encryption algorithms are used. This paper proposes ESEBM (Enhanced Symmetric Key Encryption based Mixnet), a new mechanism for anonymous communication that removes drawbacks of existing anonymous networks while exploiting symmetric key encryption algorithms. According to experimentations, throughput of ESEBM is about 1/4.4 of usual non-anonymous networks, and it achieves more than 36 times higher throughput compared with Mixnet. In addition, different from existing anonymous networks, ESEBM can handle reply messages without any additional mechanism, and it can protect itself from various threats, e.g. DOS attacks and message forgeries.**

***Index Terms*—anonymous communication, mixnet, privacy protection, symmetric key encryption algorithm**

I. Introduction

Identities of message senders are sometimes as sensitive as messages themselves. For example, a company may acquire highly confidential information about its rival companies from identities of their customers and suppliers. Therefore, the importance of anonymous communication is increasing as more people are being involved in network based communication. Anonymous networks are ones that enable message senders to send their messages without disclosing their identities, and various anonymous networks had been proposed already, e.g. Mix net [1, 5, 9], DC-net [2], Crowds [4], etc., to protect secrets of entities that communicate through networks. However, they still have serious drawbacks. For example, although Mix net is one of the most promising mechanisms, it requires the tremendous amount of computations to encrypt/decrypt messages that are forwarded from senders to their receivers. That is because asymmetric key encryption/decryption functions are adopted. In this paper, a new anonymous network ESEBM (Enhanced Symmetric Key Encryption based Mix net) is proposed that removes drawbacks of existing anonymous networks by using symmetric key encryption functions.

ESEBM consists of two parts, they are the CP generator (offline) and the anonymous channel (online) each of which is configured as a sequence of servers, and senders obtain secret keys of individual servers in the anonymous channel for encrypting their messages from the CP generator as off-line processes. Then, once encryption keys are shared between senders and servers, servers in the anonymous channel can efficiently transfer messages of senders to their receivers while exploiting symmetric key encryption functions.

According to experimentations, the capacity of ESEBM is more than 36 times higher than that of decryption type Mix net. Different from asymmetric key encryption functions, symmetric key encryption functions also enable message receivers to send reply messages to the anonymous senders in totally the same way as the senders send original messages, and consequently, anyone except the receivers cannot identify even whether messages are replies or not. Also, the CP generator configuration disables unauthorized entities to send messages because only authorized entities that had obtained secret keys from the CP generator can send messages. Therefore, ESEBM is secure against various kinds of attacks including DOS attacks and message forgeries (or modifications) that are difficult to prevent in existing anonymous networks.

II. Requirements for Anonymous Networks

Anonymous networks should satisfy the following requirements, i.e.,

1. no one except senders of messages can know identities of the senders,
2. message senders can confirm their message arrivals at their receivers without disclosing their identities,
3. receivers can send reply messages back to the senders without knowing the senders' identities,

* Graduate School of Engineering, University of Fukui
3-9-1, Bunkyo, Fukui 910-8507, Japan

doi:10.4304/jnw.6.11.1533-1542

4. anonymous networks must be able to protect themselves from accesses from unauthorized entities, and
5. anonymous networks must maintain their performances as same as usual ones.

The 1st requirement is the most important one, and senders of messages must be concealed not only from the receivers but also from network managers, eavesdroppers and any other entities. The 2nd and the 3rd requirements are also important, and especially the 3rd one is essential because information exchanges between entities in many kinds of applications are carried out as conversations between them. To satisfy the 2nd requirement is not so difficult, e. g. senders can confirm deliveries of their messages without disclosing their identities when the receivers put receive signals in public bulletin boards. However, development of practical mechanisms that satisfy the 3rd requirement is not easy as it looks. For example, a receiver, which sends reply message M_R, can identify the sender of the original message by eavesdropping on the communication channel to find out the entity that receives M_R, because it knows M_R. About the 4th requirement, because of anonymity, entities can behave dishonestly much easier than in usual communication systems, therefore, anonymous communication mechanisms must be endowed with the ability to protect them from dishonest events. The important thing here is that dishonest events must be prevented while maintaining anonymities of honest entities. Finally, to use anonymous networks in large scale applications where large volumes of messages are exchanged frequently, they must be efficient enough as usual non-anonymous networks.

III. Related Works

This section summarizes currently available anonymous networks. Although many various kinds of anonymous networks had been proposed already, still they cannot satisfy the requirements in the previous section effectively. Mixnet is an example. It consists of a sequence of mixservers T_1, T_2, ---, T_N, that relay messages from senders to their receivers. Where, senders send their messages while encrypting them repeatedly by public keys of multiple mixservers T_1, T_2, ---, T_N in the sequence. Then, individual mixservers relay their receiving messages to their neighboring servers while decrypting them by their secret decryption keys finally to be sent to their receivers. Namely, sender S encrypts its message M to $E(k_N, E(k_{N-1}, ---, E(k_1, M) ---))$ and each T_j that recieves $E(k_j, E(k_{j-1}, ---, E(k_1, M)---))$ from T_{j+1} decrypts it to $E(k_{j-1}, ---, E(k_1, M)---)$ by its secret decryption key k_j^{-1} to forward it to T_{j-1}, where $E(k_j, M)$ is the encrypted form of M. In this message relaying process, each mixserver stores its incoming messages until pre-defined number of message arrivals, and shuffles decrypted messages before forwarding them to its neighbor. Therefore, each mixserver cannot identify the links between incoming and outgoing messages of other mixservers, and as a consequence, no one except the senders themselves can identify the senders of messages unless all mixservers conspire.

However, Mixnet uses asymmetric key encryption functions, such as RSA or ElGamal, and does not work efficiently in large scale systems where number of senders send large volume of messages. A lot of computation overheads are needed to encrypt and decrypt messages. Asymmetric key encryption functions also make Mixnet require additional mechanisms for sending reply messages to senders of the original messages, therefore, servers can know whether the messages are replies or not [1, 7]. Although Mixnet can protect itself from traffic analysis and replay attacks that are discussed in Sec. VI. A, it cannot prevent DOS attacks or message forgeries (or modifications). Encryption keys are publicly disclosed and servers cannot identify spam or forged messages because they receive messages in their encrypted forms, therefore, anyone can send spam and forged messages.

Crowds [4] also consists of multiple relay servers as same as Mixnet, however, senders send their messages without encrypting them. Instead of encrypting messages, servers randomly decide whether to relay their receiving messages to their receivers or to the other servers in the network. Namely, when a server receives a message from a sender, it forwards it to other server with probability 1-p, and with probability p it sends it to the receiver. Then, it becomes difficult for entities other than the sender to identify the sender, and because no encryption or decryption process is included, Crowds can transfer messages efficiently. However, apparently it cannot disable entities to identify senders by tracing messages from their receivers to their senders. Namely, Crowds cannot satisfy the most important requirement of anonymous networks.

Onion routing [3, 8] uses the same principle as Mixnet, i.e. messages travel from senders to receivers through sequences of servers (onion routers) while being encrypted by public keys of multiple onion routers. The difference from Mixnet is that senders in onion routing encrypt not only their messages but also their routes, i.e. servers in onion routing reroute their receiving messages in unpredictable ways. Therefore, onion routers need not wait for large number of messages to shuffle them and can reduce message travelling times. However, onion routing uses asymmetric key encryption functions and shares the same drawbacks with Mixnet. An additional problem of onion routing is that it is vulnerable to timing attacks, i.e. an adversary can embed messages to know the flow times of different paths. Then, while using these message flow times, entities can know senders of messages by observing message sending and receiving times of individual senders and receivers.

Other anonymous networks such as Tor [8], buses for anonymous message delivery [6], Peer to Peer anonymous mechanisms [12], etc. have the same drawbacks as Mixnet or Onion routing.

In DC-net [2], sender S_q constitutes a group $\{S_1, S_2, ---, S_Q\}$ that includes itself, and entities in the group generate their secret numbers $\{N_1, N_2, ---, N_Q\}$ so that the sum of

them becomes 0 in advance. While using its generating secret number, S_q encrypts its message M to $M + N_q$ to send it to its receiver R. At the same time, each S_j in the group also sends its secret number N_j to R. Therefore, R can extract M from messages of $\{S_1, S_2, ---, S_Q\}$, i.e. $N_1 + N_2 + --- + (M + N_q) + N_{q+1} + --- + N_Q = M + 0 = M$. However, no one except S_q can know the sender of M, because each S_j does not know secret numbers of other senders.

As shown above, DC-net provides almost perfect anonymity, however it has fatal drawbacks about its performance, i.e. multiple senders must behave synchronously. Multiple senders must agree with each other about random numbers to encrypt messages, also only one sender can send a message at a time. Therefore, it is applicable only to small and closed networks. Here, it must be noted that each S_j must change random secret number N_j at every message sending. If every S_j uses same random secret number for different messages sent from senders in the group, an entity X that eavesdrops on the communication can easily identify senders of the messages. Namely, when S_j sends same number N_j as its 1st and 2nd messages, X can know that S_j's random secret number is N_j. Also, when S_j sends $(M_j + N_j)$ and N_j as its 1st and 2nd messages, it is easy for X to extract M_j and to identify the sender.

To decrease computation volumes of encryptions and decryptions, SEBM [13] exploits symmetric key encryption functions. SEBM consists of 2 parts, the encryption part and the decryption part, and messages are forwarded to their receivers while being encrypted by servers in the encryption part and decrypted by servers in the decryption part. Here different from other anonymous networks, senders themselves are included as relay servers in both parts to enable the use of symmetric key encryption functions. Therefore, although SEBM can satisfactory reduce the computation overheads caused by asymmetric key encryptions, senders included in the encryption and decryption parts reduce the stability of the communication. For example, when senders, i.e. volunteer servers, stop operations, messages cannot be forwarded. As another drawback, because messages in SEBM must be encrypted and decrypted by servers both in the encryption and the decryption parts, their travelling times increase. Also, it cannot efficiently handle reply messages or prevent accesses from unauthorized entities either.

IV. ESEBM (ENHANCED SYMMETRIC KEY ENCRYPTION BASED MIXNET)

This section proposes ESEBM, a scheme for anonymous networks that efficiently satisfies all the requirements listed in the previous section. ESEBM removes most drawbacks that exist in other anonymous networks, i.e. it can transfer messages without large overheads, it does not require any additional mechanism for forwarding reply messages, and it can protect itself from various attacks.

A. ESEBM Configuration

ESEBM can be considered as a kind of decryption type Mixnet, in which asymmetric key encryption functions are replaced by symmetric ones, where the encryption keys used for sending messages are distributed to senders in advance. At the same time, it is considered as SEBM in which volunteer servers are replaced by permanent ones in order to make the network stable enough [15].

As shown in Fig. 1, ESEBM consists of 2 parts, i.e. the anonymous channel and the concealing pattern generator (CP generator). The anonymous channel is configured as a sequence of N servers as same as Mixnet, and the CP generator consists of Z-groups, where the g-th group is configured by N_g servers, and each server in the anonymous channel is corresponded to a single server in the CP generator and vice versa, therefore $N = N_1 + N_2 + --- + N_Z$. In the remainder, notation $T_g(k)$ that represents the k-th server in the g-th group of the CP generator is used also for representing the p-th server T_p in the anonymous channel that corresponds to $T_g(k)$, and vice versa.

ESEBM adopts onetime pad as the base algorithm to encrypt and decrypt messages, and sender S of message M_S requests servers in the CP generator to issue a bit string called concealing pattern (CP), a pad for encrypting M_S, in advance as an off-line process.

Provided that servers generate their h-th CP at the request of S, each server T_j in the CP generator generates its h-th CP constructor $x_j(h)$, and the h-th concealing pattern X(h) is constructed as XOR of them, i.e. $X(h) = x_1(h) \oplus x_2(h) \oplus --- \oplus x_N(h)$. Then, S sends M_S to the first server T_1 in the anonymous channel while encrypting it to $M_S \oplus X(h)$. Therefore, the length of CPs and CP constructors are defined as L_M, which is the length of messages. When S sends a long message M_S, M_S is divided into multiple frames of length L_M. Here, S uses different CPs for encrypting different messages including different frames of the same message. Also, although notations X(h) and $x_j(h)$ are accompanied by h they do not include any information about h.

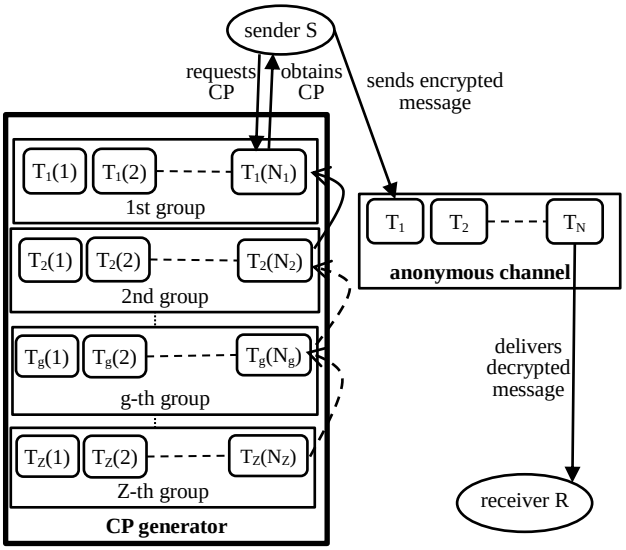


Figure 1. ESEBM configuration

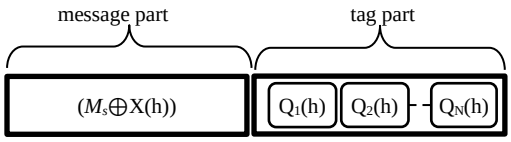


Figure 2. Message structure

As same as usual Mixnet, each server in the anonymous channel stores its receiving messages until it receives the predefined number of messages, and decrypts, shuffles and forwards them to its neighboring server finally to be sent to their receivers. Here, each T_j decrypts its receiving encrypted M_S by simply XORing it by its CP constructor $x_j(h)$ that constitutes X(h), the CP that S had used to encrypt M_S, then, it is apparent that $M_S \oplus X(h)$ is transformed to M_S when all servers decrypt it. On the other hand, because each server knows only its CP constructor $x_j(h)$ in X(h), no one can know the sender of M_S unless all servers conspire with each other as same as in usual Mixnet.

However, different from usual Mixnet where all senders encrypt their messages by using the same single public encryption key of each mixserver, in ESEBM, senders encrypt different messages by using different CPs, Therefore to enable T_j to identify its CP constructor $x_j(h)$ that constitutes X(h) for encrypting M_S, message M_S consists of the message part and the tag part as shown in Fig. 2. The message part maintains encrypted message M_S, i.e. $M_S \oplus X(h)$, and the tag part maintains a sequence of tags, i.e. vector $Q(h) = \{Q_1(h), Q_2(h), ---, Q_N(h)\}$, where server T_j that had generated the CP constructor $x_j(h)$ to construct X(h) can know $x_j(h)$ from $Q_j(h)$. Here, $Q_j(h)$ is constructed so that no one can trace the message by it and no one except T_j can identify $x_j(h)$ from it.

B. Behavior of the CP Generator

To disable entities to trace messages forwarded through the anonymous channel, not only correspondences between the message parts of input and output messages of individual servers but also those between their tag parts must be concealed. To achieve this, the CP generator generates 2 kinds of secret encryption keys shared between senders and individual servers, the one is CPs and the other is tag vectors (TVs). The CP generator is a set of server groups, each of which consists of at least 3 servers that generate their secret CP constructors and TV constructors independently of others to construct CPs and TVs jointly with other servers. Here, senders communicate only with servers in the 1st group, i.e. with $T_1(1)$, $T_1(2)$, ---, and $T_1(N_1)$, to disable servers in the other groups to know the senders as shown in Fig. 1.

As discussed already, concealing pattern X(h) is calculated as XOR of CP constructor $x_j(h)$ (j = 1, 2, ---, N) generated by each server T_j, and disables anyone to trace the message parts of a message relayed by the servers. On the other hand, individual elements of N-dimensional tag vector $Q(h) = \{Q_1(h), Q_2(h), ---, Q_N(h)\}$ disable anyone to trace the tag part of a message relayed by the servers, and each $Q_i(h)$ is calculated as XOR of the i-th elements of each N-dimensional TV constructor $q_j(h) = \{0, ---0, q_{j(j+1)}(h), q_{j(j+2)}(h), ---, q_{jN}(h)\}$ generated by T_j (j = 1, ---, N). Here, each $q_{jk}(h)$ in vector $q_j(h)$ is a bit pattern of length L_T as discussed later, 0 represents an all zero bit pattern of length L_T, and a sequence of j-zero patterns precedes before the (N-j)-secret bit patterns $\{q_{j(j+1)}(h), q_{j(j+2)}(h), ---, q_{jN}(h)\}$. By XORing CP constructors and TV constructors of individual serves, concealing pattern X(h) and tag vector Q(h) are calculated as $X(h) = x_1(h) \oplus x_2(h) \oplus --- \oplus x_N(h)$ and $Q(h) = \{0, q_{12}(h), q_{13}(h) \oplus q_{23}(h), ---, q_{1N}(h) \oplus q_{2N}(h) \oplus --- \oplus q_{(N-1)N}(h)\}$. Here, the length of bit pattern $x_j(h)$ is equal to the message frame length L_M as mentioned before, and the last server T_N does not generate its TV constructor.

CPs and TVs above are generated as follows. Provided that $T_1(k)$ in the 1st group of the CP generator corresponds to T_{k*} in the anonymous channel, i.e. $T_1(1) = T_{1*}$, $T_1(2) = T_{2*}$, ---, and $T_1(N_1) = T_{N1*}$, firstly, sender S sends a set of its secret private vectors (PVs) $\{P_1(h), P_2(h), ---, P_{N1}(h)\}$ as a request for a CP to servers T_{1*}, T_{2*}, ---, T_{N1*}, respectively, as shown in Fig. 3 (a). Here, each $P_j(h)$ is vector $\{p_{j0}(h), p_{j1}(h), ---p_{jN}(h)\}$ and except $p_{j0}(h)$, $p_{jk}(h)$ is a bit pattern of the same length as element $q_{jk}(h)$ in TV constructor $q_j(h)$. Bit pattern $p_{j0}(h)$ has the same length as CP constructor $x_j(h)$.

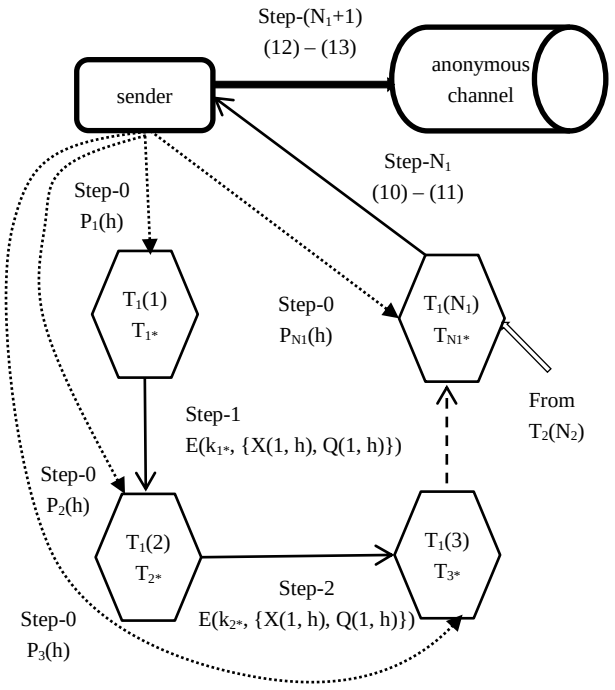
Then, T_{1*} that receives the request with $P_1(h)$, generates its CP constructor $x_{1*}(h)$ and TV constructor $q_{1*}(h) = \{0, ---, 0, q_{1*(1*+1)}(h), q_{1*(1*+2)}(h), ---, q_{1*N}(h)\}$. It also generates $ID_{1*}(x_{1*}(h), q_{1*}(h))$ as an address of CP and TV constructor pair $(x_{1*}(h), q_{1*}(h))$. Here, T_{1*} maintains its CP table, a list of CP and TV constructors that it had generated, and $ID_{1*}(x_{1*}(h), q_{1*}(h))$ represents the address of the constructor pair $\{x_{1*}(h), q_{1*}(h)\}$ in the table. Also, the length of each bit pattern $q_{jk}(h)$ in TV constructor $q_j(h)$ is set as L_T, the length of $ID_j(x_j(h), q_j(h))$.

Then, X(1, h) and Q(1, h), the h-th CP and TV that the 1st group generates, are constructed by 1st server T_1^* as $X(1, h) = p_{10}(h) \oplus x_{1*}(h)$ and $Q(1, h) = \{p_{11}(h), p_{12}(h), ---, p_{11*}(h) \oplus ID_{1*}(x_{1*}(h), q_{1*}(h)), p_{1(1*+1)}(h) \oplus q_{1*(1*+1)}(h), p_{1(1*+2)}(h) \oplus q_{1*(1*+2)}(h), ---, p_{1N}(h) \oplus q_{1*N}(h)\}$, respectively. X(1, h) and Q(1, h) are then forwarded to T_{2*}. However, to protect them from eavesdropping, they are encrypted by the secret key k_{1*} that is shared between T_{1*} and T_{2*}, i.e. X(1, h) and Q(1, h) are sent to T_{2*} in the form $E(k_{1*}, X(1, h), Q(1, h))$, where, $E(k_{1*}, x)$ represents x encrypted by key k_{1*}. It is also possible that T_{1*} encrypts X(1, h) and Q(1, h) by using a public key of T_{2*}, however to decrease encryption overheads, a symmetric key encryption function is adopted here.

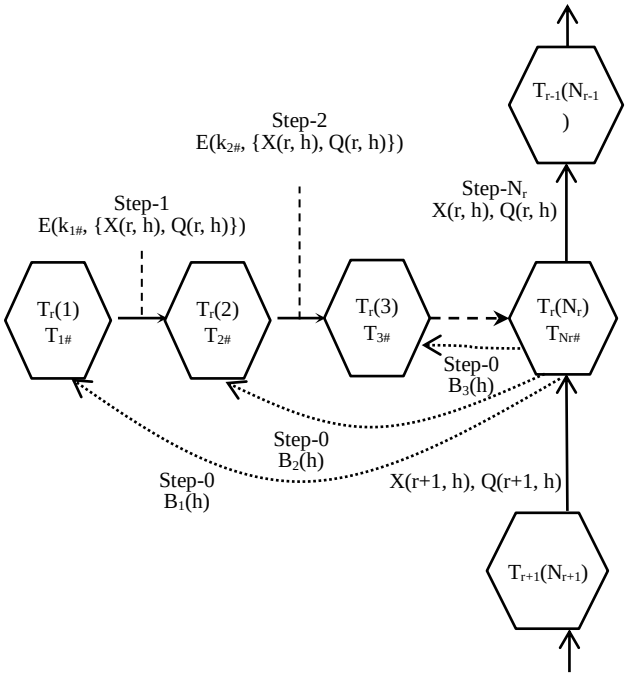
T_{2*} that receives $E(k_{1*}, \{X(1, h), Q(1, h)\})$ decrypts it to $\{X(1, h), Q(1, h)\}$, and generates its CP constructor $x_{2*}(h)$ to modify X(1, h) to $X(1, h) = p_{10}(h) \oplus p_{20}(h) \oplus x_{1*}(h) \oplus x_{2*}(h)$. T_{2*} also generates TV constructor $q_{2*}(h) = (0, ---0, q_{2*(2*+1)}(h), q_{2*(2*+2)}(h), ---, q_{2*N}(h))$ to modify Q(1, h) to $\{p_{11}(h) \oplus p_{21}(h), p_{12}(h) \oplus p_{22}(h), ---, p_{11*}(h) \oplus p_{21*}(h) \oplus ID_{1*}(x_{1*}(h), q_{1*}(h)), p_{1(1*+1)}(h) \oplus p_{2(1*+1)}(h) \oplus q_{1*(1*+1)}(h), ---, p_{12*}(h) \oplus p_{22*}(h) \oplus q_{1*2*}(h) \oplus ID_{2*}(x_{2*}(h), q_{2*}(h)), p_{1(2*+1)}(h) \oplus p_{2(2*+1)}(h) \oplus q_{1*(2*+1)}(h) \oplus q_{2*(2*+1)}(h), ---, p_{1N}(h) \oplus p_{2N}(h) \oplus q_{1*N}(h) \oplus q_{2*N}(h)\}$.

Here as same as T_{1*}, T_{2*} also maintains its CP table, and $ID_{2*}(x_{2*}(h), q_{2*}(h))$ represents the address where

$\{x_{2*}(h), q_{2*}(h)\}$ is located in it. Also, it is not necessary but to simplify the descriptions, it is assumed that servers in the anonymous channel are arranged so that $T_j(g)$ is placed at the earlier position in the anonymous channel than $T_j(h)$ when $g < h$, for every j-th group. Then, X(1, h) and Q(1, h) are sent to T_{3*} while being encrypted by k_{2*}, a secret encryption key shared between T_{2*} and T_{3*}, and this process continues until T_{N1*} calculates X(1, h) and Q(1, h). Therefore, X(1, h) and Q(1, h) = $\{Q_1(1, h), Q_2(1, h), ---, Q_N(1, h)\}$, the CP and the TV pair generated by the 1st group becomes as shown in equations (1) – (3).



(a) 1st group



(b) r-th group

Figure 3. Behaviour of the CP generator

$$X(1, h) = p_{10} \oplus p_{20} \oplus --- \oplus p_{(N1)0} \oplus x_{1*}(h) \oplus x_{2*}(h) \oplus --- \oplus x_{(N1)*}(h) \quad (1)$$

for g* included in the 1st group

$$Q_{g*}(1, h) = p_{1g*}(h) \oplus p_{2g*}(h) \oplus --- \oplus p_{(N1)g*}(h) \oplus q_{1*g*}(h) \oplus q_{2*g*}(h) \oplus --- \oplus q_{(g-1)*g*}(h) \oplus ID_{g*}(x_{g*}(h), q_{g*}(h)), \text{ where } q_{0*g*}(h) = 0 \quad (2)$$

for i not included in the 1st group,

$$Q_i(1, h) = p_{1i}(h) \oplus p_{2i}(h) \oplus --- \oplus p_{(N1)i}(h) \oplus q_{1*i}(h) \oplus q_{2*i}(h) \oplus --- \oplus q_{(gj*)i}(h), \text{ where } g_{j*} < i < g_{(j+1)*} \quad (3)$$

Severs in the r-th group (r > 1) behave in the same way as the 1st group as shown in Fig. 3 (b), where server $T_r(k)$, the k-th server in the r-th group, corresponds to $T_{k\#}$ in the anonymous channel. However, different from the 1st group where senders generate PVs and sends them as a request for a CP to severs T_{1*}, T_{2*}, ---, T_{N1*}, servers $T_{1\#}$, $T_{2\#}$, ---, $T_{Nr\#}$ in the r-th group generate CP and TV pairs spontaneously without requests from senders, also the last server $T_{Nr\#}$ in the r-th group generates group blinding vector B(h) = $\{B_1(h), B_2(h), ---, B_{Nr}(h)\}$. Then, the r-th group calculates X(r, h) and Q(r, h) = $\{Q_1(r, h), Q_2(r, h), ---, Q_N(r, h)\}$ as its h-th CP and TV values as shown in equations (4) – (6). In the equations, the j-th element $B_j(h)$ of B(h) = $\{B_1(h), B_2(h), ---, B_{Nr}(h)\}$ is a vector of patterns $\{b_{j0}(h), b_{j1}(h), ---, b_{jN}(h)\}$, where the length of $b_{j0}(h)$ is L_M and the length of $b_{jk}(h)$ is L_T for each k.

$$X(r, h) = b_{10} \oplus b_{20} \oplus --- \oplus b_{(Nr)0} \oplus x_{1\#}(h) \oplus x_{2\#}(h) \oplus --- \oplus x_{Nr\#}(h) \quad (4)$$

for g# included in the r-th group

$$Q_{g\#}(r, h) = b_{1g\#}(h) \oplus b_{2g\#}(h) \oplus --- \oplus b_{(Nr)g\#}(h) \oplus q_{1\#g\#}(h) \oplus q_{2\#g\#}(h) \oplus --- \oplus q_{(g-1)\#g\#}(h) \oplus ID_{g\#}(x_{g\#}(h), q_{g\#}(h)), \text{ where } q_{0\#g\#}(h) = 0 \quad (5)$$

for i not included in the r-th group,

$$Q_i(r, h) = b_{1i}(h) \oplus b_{2i}(h) \oplus --- \oplus b_{(Nr)i}(h) \oplus q_{1\#i}(h) \oplus q_{2\#i}(h) \oplus --- \oplus q_{(gj\#)i}(h), \text{ where } g_{j\#} < i < g_{(j+1)\#} \quad (6)$$

After calculating X(r, h) and Q(r, h) as equations (4) – (6), $T_{Nr\#}$ removes group blinding vector B(h) by XORing them by B(h). Namely, they are transformed as shown in equations (7) – (9).

$$X(r, h) = x_{1\#}(h) \oplus x_{2\#}(h) \oplus --- \oplus x_{Nr\#}(h) \quad (7)$$

for g# included in the r-th group

$$Q_{g\#}(r, h) = q_{1\#g\#}(h) \oplus q_{2\#g\#}(h) \oplus --- \oplus q_{(g-1)\#g\#}(h) \oplus ID_{g\#}(x_{g\#}(h), q_{g\#}(h)), \text{ where } q_{0\#g\#}(h) = 0 \quad (8)$$

for i not included in the r-th group,

$$Q_i(r, h) = q_{1\#i}(h) \oplus q_{2\#i}(h) \oplus --- \oplus q_{(gj\#)i}(h), \text{ where } g_{j\#} < i < g_{(j+1)\#} \quad (9)$$

The last server $T_r(N_r) = T_{Nr\#}$ in the r-th group also receives X(r+1, h) and Q(r+1, h), the CP and TV values generated by the (r+1)-th group, from $T_{r+1}(N_{r+1})$, the last server in the (r+1)-th group, and it calculates X(r, h)⊕X(r+1, h), and Q(r, h)⊕Q(r+1, h) to combine CPs and

TVs generated by the r-th and the (r+1)-th groups into the single CP and TV, respectively. Then, $T_r(N_r)$ waits for the arrivals of predefined number of CP and TV pairs, and shuffles them to sends the results to the last server $T_{r-1}(N_{r-1})$ of the (r-1)-th group. As the result of the behaviors of all groups, the last server of the 1st group, i.e. $T_1(N_1)$, generates the CP and TV as equations (10) and (11).

$$X(h) = p_{10}(h)\oplus p_{20}(h)\oplus\text{---}\oplus p_{(N1)0}(h)\oplus x_1(h)\oplus x_2(h)\oplus\text{---}\oplus x_N(h) \quad (10)$$

$$Q_g(h) = p_{1g}(h)\oplus\text{---}\oplus p_{(N1)g}(h)\oplus q_{1g}(h)\oplus\text{---}\oplus q_{(g-1)g}(h)\oplus ID_g(x_g(h), q_g(h)), \text{ where } q_{0g}(h) = 0 \quad (11)$$

Then, $T_1(N_1)$ sends X(h) and Q(h) = {$Q_1(h)$, $Q_2(h)$, ---, $Q_N(h)$} to sender S, and S removes private vectors PVs from X(h) and Q(h) by XORing them by PVs. As the result, finally CP and TV values become as (12) and (13).

$$X(h) = x_1(h)\oplus x_2(h)\oplus\text{---}\oplus x_{(N-1)}(h)\oplus x_N(h) \quad (12)$$

$$Q_g(h) = q_{1g}(h)\oplus\text{---}\oplus q_{(g-1)g}(h)\oplus ID_g(x_g(h), q_g(h)), \text{ where } q_{0g}(h) = 0 \quad (13)$$

It must be noted that because PVs and group blinding vectors are secrets of sender S and last server of each group (except the 1st group), respectively, and each server T_j does not disclose $x_j(h)$ or $q_j(h)$ to others, any server cannot know CP or TV constructors of other servers. No server can know X(h) or Q(h) either unless all servers conspire with each other.

C. Behavior of the Anonymous Channel

Fig. 4 shows the behavior of the anonymous channel. Firstly, sender S encrypts its message M_S by XORing it by concealing pattern X(h) that it had acquired from $T_1(N_1)$. S also attaches tag vector Q(h) = {$Q_1(h)$, $Q_2(h)$, ---, $Q_N(h)$} corresponding to X(h), to the message, and sends {$\underline{M_S} = x_1(h)\oplus x_2(h)\oplus\text{---}\oplus x_N(h)\oplus M_S$, $Q_1(h)$, $Q_2(h)$, ---, $Q_N(h)$} to the 1st server T_1 in the anonymous channel. Here, $Q_1(h)$ has the form $ID_1(x_1(h), q_1(h))$.

Then, T_1 that receives {$x_1(h)\oplus x_2(h)\oplus\text{---}\oplus x_N(h)\oplus M_S$, $Q_1(h)$, $Q_2(h)$, ---, $Q_N(h)$} 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 $Q_1(h)$, calculates XOR of $x_1(h)$ and $\underline{M_S}$, and $q_{1j}(h)$ and $Q_j(h)$ for each j as new values of $\underline{M_S}$ and $Q_j(h)$. Therefore, $\underline{M_S}$ and $Q_j(h)$ become $\underline{M_S} = x_1(h)\oplus(x_1(h)\oplus x_2(h)\oplus\text{---}\oplus x_N(h)\oplus M_S) = x_2(h)\oplus x_3(h)\oplus\text{---}\oplus x_N(h)\oplus M_S$ and $Q_j(h) = q_{1j}(h)\oplus(q_{1j}(h)\oplus q_{2j}(h)\oplus\text{---}\oplus q_{(j-1)j}(h)\oplus ID_j(x_j(h), q_j(h))) = q_{2j}(h)\oplus q_{3j}(h)\oplus\text{---}\oplus q_{(j-1)j}(h)\oplus ID_j(x_j(h), q_j(h))$. After that, T_1 removes $Q_1(h)$ from the tag part, waits for the predefined number of message arrivals, and shuffles them to forward each result to server T_2.

All servers in the anonymous channel perform in the same way, i.e. each T_j converts its incoming message to {$\underline{M_S} = x_{j+1}(h)\oplus x_{j+2}(h)\oplus\text{---}\oplus x_N(h)\oplus M_S$, $Q_{j+1}(h)$, $Q_{j+2}(h)$, ---, $Q_N(h)$}, where $Q_g(h) = q_{(j+1)g}(h)\oplus\text{---}\oplus q_{(g-1)g}(h)\oplus ID_g(x_g(h), q_g(h))$. Consequently, when T_N, the last server in the anonymous channel, completes its operations on the message, the message is converted into M_S, and T_N can deliver M_S to its receiver while extracting the address of the receiver from M_S.

The anonymous channel together with the CP generator protects identities of message senders from various threats as follows. Firstly, each server T_j transforms the message part while XORing it by CP constructor $x_j(h)$ which is not known to other servers and also T_j assigns different values as CP constructors for encrypting different messages. Therefore no one including other server T_i can identify the input and output pair of T_j that corresponding to M_S by comparing message parts of T_j's receiving and forwarding messages. For T_i, 2 input and output pairs of T_j, e. g. {$x_j(h)\oplus x_{j+1}(h)\oplus\text{---}\oplus x_N(h)\oplus M_S$, $x_{j+1}(h_1)\oplus\text{---}\oplus x_N(h_1)\oplus M_1$} and {$x_j(h)\oplus x_{j+1}(h)\oplus\text{---}\oplus x_N(h)\oplus M_S$, $x_{j+1}(h_2)\oplus\text{---}\oplus x_N(h_2)\oplus M_2$}, have equal possibilities that they are encrypted form pairs of M_S. As a consequence, it is impossible for entities including servers to identify the sender of message M_S by tracing the message parts of messages unless all servers conspire.

Any entity cannot trace M_S by examining the tag parts of messages either. Because each T_j generates different secret TV constructors for different messages and assigns different bit patterns to individual elements {$q_{j(j+1)}(h)$, ---, $q_{jN}(h)$} in TV constructor $q_j(h)$, it is impossible for other entities to identify links between incoming messages of T_j and its outgoing messages by examining pattern transitions in individual tags made by T_j. Namely, individual tags change their forms within T_j in different ways, and entities except T_j cannot extract any relation between transitions of different tags in the tag part to identify input and output pairs of same messages.

Also, although, each server T_{j*} in the 1st group in the CP generator can know the senders of encrypted messages from their CP and TV constructors, because T_{j*} generates them at requests of the senders, when T_{j*} is placed at the earlier position of the anonymous channel, its tags disappear in the later positions, i.e. the tag parts of messages that are received by servers at later positions of the anonymous channel do not include tags of any server in the 1st group, therefore even if T_{j*} conspires with servers at the later positions, it is not possible to identify senders.

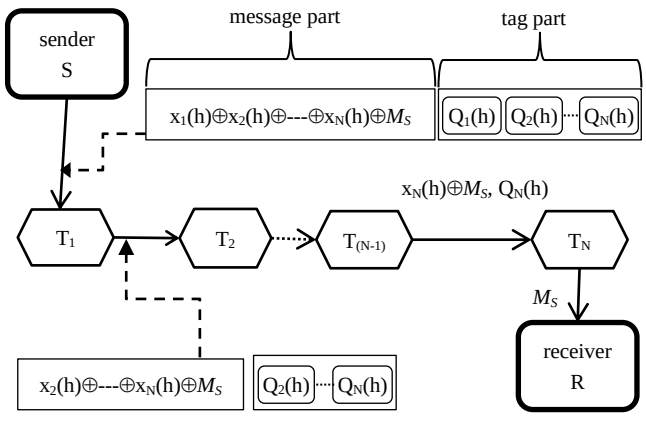


Figure 4. Behavior of the anonymous channel

V. REPLIES TO ANONYMOUS SENDERS

Different from other existing mechanisms [2, 7], in ESEBM, individual servers can handle reply messages to anonymous senders without any additional mechanism. This means that servers cannot decide even whether a message is the reply or not. Sender S can receive reply messages as follows. Firstly, S obtains 2 CP and TV pairs $\{X(h_1), Q(h_1)\}$, $\{X(h_2), Q(h_2)\}$, and constructs its message while attaching tag vector $Q(h_2)$ and its encrypted address A_S to its sending message M_S as shown in Fig. 5 (a). Namely, S constructs $M_S \| Q(h_2) \| (X_U(h_2) \oplus A_S)$, concatenation of M_S, $Q(h_2)$, and $X_U(h_2) \oplus A_S$. Where bit strings $X_U(h_2)$ and $X_L(h_2)$ are upper and lower parts of bit string $X(h_2)$, in other words, $X(h_2) = X_U(h_2) \| X_L(h_2)$. Also it is assumed that message M_S includes its destination address at its left most bit positions.

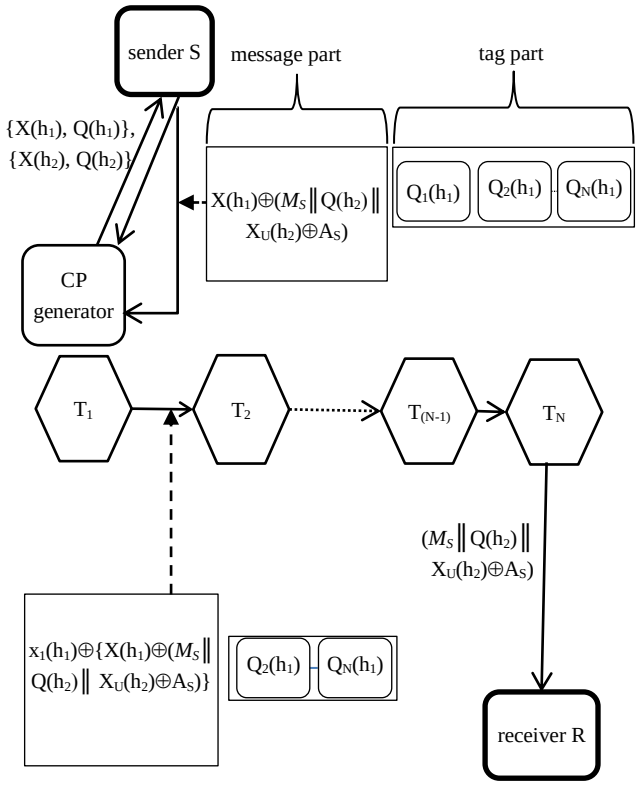
After that, S encrypts $M_S \| Q(h_2) \| X_U(h_2) \oplus A_S$ to $X(h_1) \oplus (M_S \| Q(h_2) \| X_U(h_2) \oplus A_S)$, and sends $\{X(h_1) \oplus (M_S \| Q(h_2) \| X_U(h_2) \oplus A_S)$, $Q_1(h_1)$, $Q_2(h_1)$, ---, $Q_N(h_1)\}$ to the 1st server T_1 in the anonymous channel. Then, T_1 decrypts it by $x_1(h_1)$, CP constructor of T_1. As a result, the message becomes $\{x_1(h_1) \oplus X(h_1) \oplus (M_S \| Q(h_2) \| X_U(h_2) \oplus A_S)$, $Q_2(h_1)$, ---, $Q_N(h_1)\} = \{x_2(h_1) \oplus \text{---} \oplus x_N(h_1) \oplus (M_S \| Q(h_2) \| X_U(h_2) \oplus A_S)$, $Q_2(h_1)$,---, $Q_N(h_1)\}$. Each server T_j in the anonymous channel carries out the same procedure until receiver R receives $M_S \| Q(h_2) \| X_U(h_2) \oplus A_S$. Then R can extract message M_S, encrypted address $X_U(h_2) \oplus A_S$ of S and tag vector $Q(h_2)$ to construct its reply message as $\{(X_U(h_2) \oplus A_S) \| M_R, Q_1(h_2), \text{---}, Q_N(h_2)\}$ to be encrypted to $X(h_2) \oplus \{X_U(h_2) \oplus A_S \| M_R\} = \{A_S \| X_L(h_2) \oplus M_R\}$, by the anonymous channel as shown in Fig. 5 (b). Therefore, T_N can deliver $X_L(h_2) \oplus M_R$ to S and finally S that knows $X_L(h_2)$ decrypts $X_L(h_2) \oplus M_R$ to $X_L(h_2) \oplus X_L(h_2) \oplus M_R = M_R$.

In the above, R receives $M_S \| Q(h_2) \| X_U(h_2) \oplus A_S$, and it cannot know A_S because $X_U(h_2)$ is known only to S. Also, message $X_U(h_2) \oplus A_S \| M_R$ sent by R is transformed to $A_S \| X_L(h_2) \oplus M_R$ in the anonymous channel, therefore, no one except S can know that $X_L(h_2) \oplus M_R$ corresponds to M_R, and consequently even receiver R that knows M_R cannot identify the original sender of M_S. In this way, servers in ESEBM can handle original and reply messages totally in the same way, different from usual Mixnets where each mixserver adds extra operations on reply messages.

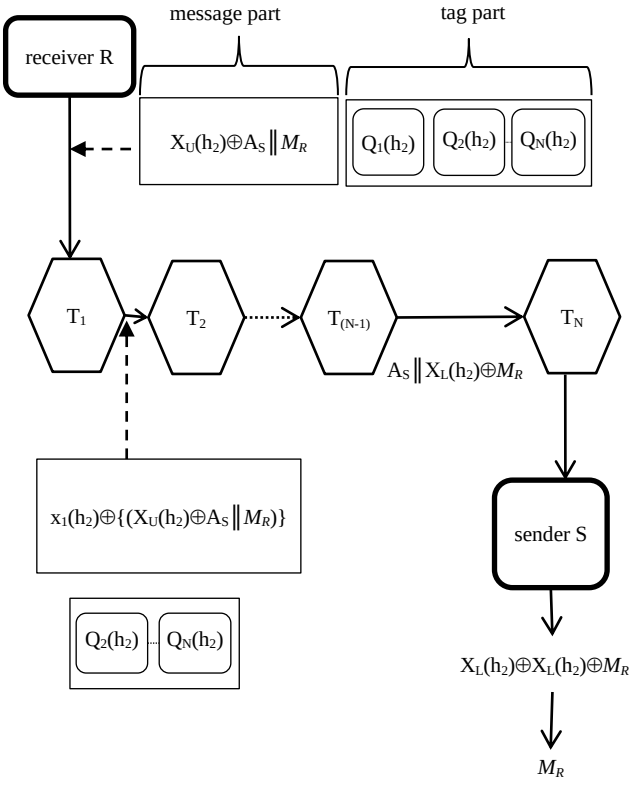
VI. EVALUATION OF ESEBM

A. Analysis of ESEBM Behavior

ESEBM satisfies the requirements for anonymous networks listed in Sec. II as follows. Firstly as discussed in Sec IV. C, no one except senders themselves can trace messages from senders to receivers. Secondly, the message reply mechanism discussed in Sec. V enables receivers to send replies to senders of original messages without knowing identities of the senders. Also by this reply mechanism, senders can confirm the deliveries of their messages. In addition the reply mechanism of ESEBM does not require additional operations on reply messages, therefore different from other existing anonymous networks, servers cannot know even whether their handling messages are replies or not.



(a) From sender to receiver



(b) From receiver to sender

Figure 5. Anonymous reply mechanism

About the efficiency, the configuration of ESEBM where senders must obtain CPs before sending their individual messages is obviously a disadvantage, e. g. message travelling times increase when durations required for obtaining CPs are counted. However because senders can obtain CPs as offline processes, actual message traveling times can be suppressed at values comparable to Mixnet. Also, when each server is configured by 2 independent CPUs, tasks for generating CPs and forwarding messages can be assigned to different CPUs so that the anonymous channel can forward messages without being interrupted by tasks for the CP generator. Then, despite of the disadvantages of the CP obtaining process, ESEBM configuration enables anonymous networks to adopt symmetric key encryption functions that make ESEBM efficient enough as usual non-anonymous networks to handle messages in practical scale applications as demonstrated in the next subsection.

ESEBM configuration brings advantageous features not only about the efficiency but also about security as follows. Among various threats to networks, DOS attacks [10], in which meaningless or spam messages are sent to decrease availabilities of networks, and illegitimate message forgeries (modifications), in which malicious entities forge (modify) messages sent from anonymous senders, are especially serious in anonymous networks. Different from in usual networks where all entities that send messages can be identified if costs and efforts are not considered, in anonymous networks where identities of senders are completely hidden, entities can behave dishonestly more easily. In addition, about message forgeries (modifications), in many cases receivers cannot notice even if their receiving messages are forged (modified) because their senders are anonymous.

The CP generator in ESEBM reduces the occurrence of DOS attacks substantially and makes forged (modified) messages detectable. Namely, senders must attach consistent TVs to their messages to let servers transfer the messages; however, the CP generator gives CPs and TVs only to authorized entities. Therefore, unauthorized entities must send their messages while attaching nonregistered TVs, and servers in ESEBM that cannot find CPs and TVs from their CP tables discard the messages immediately, as the consequence, messages from unauthorized entities do not decrease the availability of the network. About the malicious message forgeries (modification), provided that the malicious entity X does not know the original message *M*, X cannot forge (modify) encrypted *M* consistently because no one except the sender of *M* knows the CP used for encrypting *M*, then the receiver of *M* can notice the forgeries (modification) because its receiving message is meaningless.

In the same way, ESEBM disables entities to carry out traffic analysis attacks and replay attacks. A traffic analysis attack is a way to identify the sender S of a message *M* by sending multiple replies to it [7, 14]. Namely, when receiver R of *M* sends many replies at a time or periodically to S, R can identify S by observing entities that receives many messages at a time or periodically. However, in ESEBM every message must have different CPs and TVs, and this means that every server discards CP and TV constructors in its CP table once they are used. Therefore, provided that at least one of the servers is honest, even when R sends multiple replies only one of them is delivered to S, and R cannot identify S. It must be noted that, it is also possible to enable receivers to send up to predefined number of replies. If each server T_j maintains F(h), the number of messages allowed to send by using tag vector Q(h), in its CP table in addition to $\{x_j(h), q_j(h)\}$, T_j does not invalidate $\{x_j(h), q_j(h)\}$ until it receives F(h)-messages attached by Q(h).

In a replay attack [11], an entity X identifies sender S of message *M* by eavesdropping on the network to pick M_*, encrypted form of *M*, just sent from S, and putting M_* to the network repeatedly. Then, because *M* is delivered to the same receiver R many times, X can easily identify the correspondence between S and *M* received by R. Apparently ESEBM can disable replay attacks in the same way as disabling traffic analysis attacks.

B. Message Processing Performance

Performance of ESEBM has been compared with that of the usual non-anonymous networks and Mixnet each of which consisted of multiple PCs that worked as relay servers. Where individual PCs were equipped with 1.6GHz CPUs and 1GB of RAM and they were connected by 100Mbits/sec Ethernet. Because delays of message arrivals depend on the number of relay servers and the time that individual servers must wait for shuffling messages, only the throughput were compared while changing the sizes of messages. For evaluating ESEBM, 16 tags each of which consisted of 64 bits were attached to individual messages, therefore for ESEBM, the actual length of a 10 Kbits message is 11 Kbits for example. For Mixnet, RSA with 1K bits length key was adopted as the encryption function. In real applications, a sender must combine its message *M* with random secret numbers to make the encryption function probabilistic. Also to maintain strengths of encryption keys, different servers must use different modulo arithmetic. However in this evaluation, random bit strings were not attached to messages, and all servers used the same modulo arithmetic.

Table 1 shows the computation times required by each server in non-anonymous network, ESEBM and Mixnet to transfer different sizes of messages, and Fig. 6 graphically represents them. For example, while ESEBM needs less than 6 seconds to transfer a 20Mbits message, Mixnet needs more than 3 minutes to transfer the same message. Fig. 7 shows the volume of messages that usual non-anonymous networks, ESEBM and Mixnet can send within 1 second. These results show that, although the throughput of ESEBM is 1/4.4 of that of non-anonymous networks, it is more than 36 times higher than that of Mixnet. According to statistics [16], e-mail message size is 59KB on average, therefore, even in the environments used for evaluations, ESEBM can handle 7 clients at a time that send usual e-mail messages while the non-anonymous network can handle 33 clients at a time. On the other hand, Mixnet can handle only 0.2 clients. The beneficial thing is that, when multiple processors are

available, volume of messages can be processed almost in parallel. Therefore, ESEBM can transfer the same volume of messages as usual non-anonymous networks do when each server is constituted by multiple processors and memories with 4.4 times of costs. Here, although it depends on individual applications, value 4.4 can be considered acceptable. On the other hand, to improve the performance of Mixnet as non-anonymous networks, 158 times of costs are necessary. Namely, ESEBM can be used for large scale networks, in which number of clients exchange usual sizes of messages at less extra costs.

TABLE I. COMPUTATION TIME FOR TRANSFERRING DIFFERENT SIZES OF MESSAGES

Message size Mbits	Non-anonymous (msec)	ESEBM (msec)	Mixnet (msec)
10	625	2780	105255
20	1230	5510	207440
30	1924	8556	310986
40	2520	11225	412679
50	3125	14127	528276
60	3745	17342	---
70	4325	19710	---
80	4995	22862	---
90	5643	25595	---
100	6246	28344	---

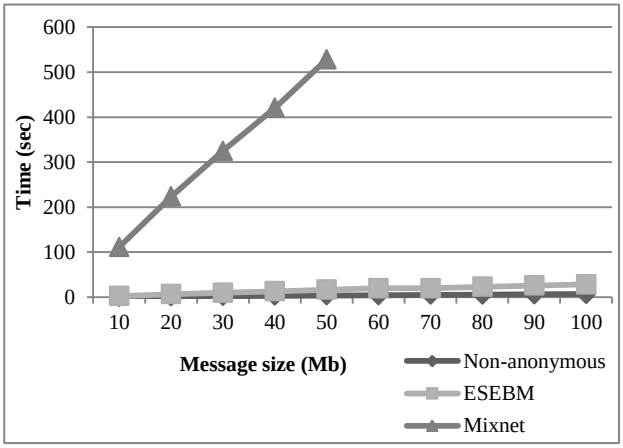


Figure 6. Comparison of computation time for transferring different sizes of messages

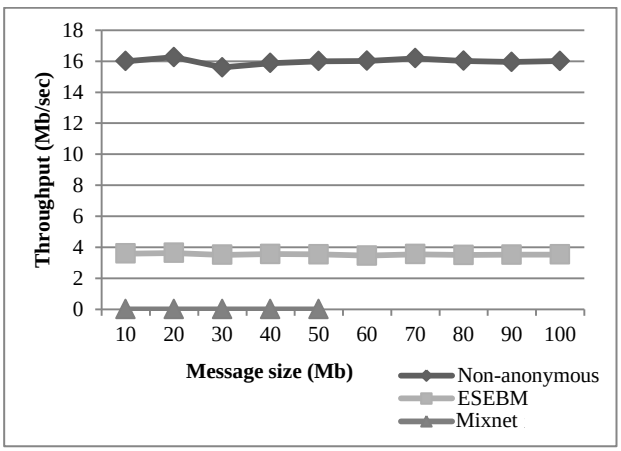


Figure 7. Comparison of throughputs for transferring different sizes of messages

About the breakdown of message processing time of each server in ESEBM, it consists of shuffling (31%), message decryption (26%), and others (43%). On the other hand, message processing time of each server in Mixnet consists of shuffling (0.8%), message decryption (98.6%), and others (0.6%). As shown above, different from Mixnet in which message decryptions require 123 times of message shuffling time, in ESEBM, message decryptions require less than 0.84 times of the shuffling time. When the fact that both ESEBM and Mixnet shuffle same number of messages is considered, this means that message decryption process in Mixnet degrades its overall performance seriously. In other words, symmetric key encryption functions used in ESEBM had successfully reduced decryption times. Namely, while RSA used in Mixnet requires the number of multiplications that is proportional to $\log_2(n)$, onetime pad used in ESEBM requires only a single XOR operation, where n is the size of encryption keys.

SEBM also uses symmetric key encryption functions [13], and as ESEBM, it can achieve the higher throughput than other anonymous networks such as Mixnet. However, when compared with ESEBM, in SEBM, more servers must be involved in forwarding messages, because it consists of encryption and decryption servers. Therefore, message traveling times in SEBM become longer than that of ESEBM, i.e. different from in ESEBM where messages are encrypted by their senders, in SEBM, they are encrypted by a sequence of encryption servers. As other advantages of ESEBM over SEBM, ESEBM works more stably because all servers in ESEBM are permanent servers different from SEBM where senders are included as servers. Also a mechanism for reply messages is not straightforward in SEBM.

VII. CONCLUSION

Enhanced symmetric key encryption based Mixnet has been proposed that removes the drawbacks of many existing anonymous networks such as Mixnet, DC-net, etc. It satisfies all the requirements of anonymous networks. Most importantly, while being supported by concealing patterns, those requirements are satisfied in a simple and efficient way. Unlike complicated Mixnet based systems, the simplified computational requirements of individual entities make the scheme practical and scalable.

As a drawback of ESEBM, a sender must acquire a concealing pattern from the CP generator in advance to send its every message as an offline process. However because of ESEBM configuration, i.e. by dividing the network into the CP generator (off-line) and the anonymous channel (on-line) parts, every time-consuming task is removed from the anonymous channel part and highly efficient communication becomes possible. Moreover, concealing patterns enable receivers not only to send replies to the original anonymous message senders but also to receive messages without disclosing their identities. Namely, when concealing patterns are publicly disclosed with the receivers' interests, the receivers can receive messages from senders without disclosing their identities.

As a future work, mechanisms that enhance the

reliability of ESEBM are necessary. When senders or receivers claim that some server is dishonest, ESEBM must prove all servers are honest or detect dishonest servers if exist. Also, ESEBM must continue its operations even some of servers are out of their services.

REFERENCES

[1] D. Chaum, "Untraceable electronic mail, return address and digital pseudonyms," *Communications of the ACM*, vol. 24, no. 2, pp. 84-88, 1981.

[2] D. Chaum, "The dining cryptographers problem: unconditional sender and recipient untraceability," *Journal of Cryptology*, vol. 1, pp. 65-75, 1988.

[3] M. G. Reed, P. F. Syverson and D. M. Goldschlag, "Anonymous connections and onion routing," *Selected Areas in Communications*, vol. 16, no. 4, pp. 482-494, May 1998.

[4] M. K. Reiter and A. D. Rubin, "Crowds: anonymity for Web transactions," *ACM Transactions on Information and System Security*, vol. 1, no. 1, pp. 66-92, Nov 1998.

[5] R. Ingledine, M. J. Freedman, D. Hopwood and D. Molnar, "A reputation system to increase MIX-Net reliability," *Proc. of the 4th international Workshop on information Hiding. I. S. Moskowitz, Ed. Lecture Notes In Computer Science, Springer-Verlag*, vol. 2137, London, pp. 126-141, April 2001.

[6] A. Beimel and S. Dolev, "Buses for anonymous message delivery," *Proc. of the Second International Conference on FUN with Algorithms*, Elba, Italy, pp. 1-13, May 2001.

[7] P. Golle and M. Jakobsson, "Reusable anonymous return channels," *Proc. of the 2003 ACM Workshop on Privacy in the Electronic Society*, (Washington, DC), WPES '03, ACM, New York, NY, pp. 94-100, 2003.

[8] R. Dingledine and N. Mathewson, "Tor: The second-generation onion router," *Proc. of the 13th USENIX Security Symposium*, San Diego, CA, USA, pp. 303-320, August 2004.

[9] P. Golle, M. Jakobsson, A. Juels and P. Syverson, "Universal re-encryption for Mixnets," *RSA Conference Cryptographers' Track '04, Springer-Verlag*, pp. 163-178, 2004.

[10] T. Znati, J. Amadei, D. R. Pazehoski and S. Sweeny, "On the design and performance of an adaptive, global Strategy for detecting and mitigating distributed DOS attacks in GRID and collaborative workflow environments," *Simulation*, vol. 83, pp. 291-303, March 2007.

[11] S. Y. Kang, J. S. Park and I. Y. Lee, "A study on authentication protocol in offering identification synchronization and position detection in RFID system," *Proc. of The 2007 International Conference on Intelligent Pervasive Computing (IPC 2007)*, pp. 150-154, 2007.

[12] X. Wang and J. Luo, "A collaboration scheme for making peer-to-peer anonymous routing resilient," *Computer Supported Cooperative Work in Design, 2008, CSCWD 2008*, pp. 70-75, April 2008.

[13] S. Tamura, K. Kouro, M. Sasatani, K. M. Alam and H. A. Haddad, "An information system platform for anonymous product recycling," *Journal of Software*, vol. 3, no. 6, pp. 46-56, 2008.

[14] L. Li, S. Fu and X. Che, "Active attacks on reputable Mix Networks," *ispa, 2009 IEEE International Symposium on Parallel and Distributed Processing with Applications*, pp. 447-450, 2009.

[15] H. Haddad, H. Tsurugi and S. Tamura, "A mechanism for enhanced symmetric key encryption based Mixnet," *SMC 2009 IEEE International Conference on Systems, Man and Cybernetics*, San Antonio, TX, USA, pp. 4541–4546, 11-14 Oct 2009, doi: 10.1109/ ICSMC.2009.5346788.

[16] http://www.idc.com/

Hazim A. Haddad received the B.E. degree in Computer science and Engineering from Ittihad University, from UAE (United Arab Emirates) in 2003, M.S. degree in nuclear and safety engineering, from the University of Fukui in 2008. He is currently a doctor course student of University of Fukui.

Shinsuke Tamura was born in Hyogo, Japan on Jan. 16, 1948, and received the B.S., M.S. and Dr. (Eng.) degrees in Control Engineering from Osaka University in 1970, 1972 and 1991, respectively. During 1972 to 2001, he worked for Toshiba Corporation. He is currently a professor of Graduate School of Engineering, University of Fukui, Japan. Prof. Tamura is a member of IEEJ, SICE and JSST.

Shuji Taniguchi received the B.E. and Ph.D. degrees in electronics engineering from University of Fukui, Fukui, Japan, in 1973, 1996, respectively. In 1973-1978, he was with the Hitachi co. Ltd. He is currently an associate professor of Graduate School of Engineering in University of Fukui.

Tatsuro Yanase received the Dr. (Eng.) degrees in Electric & Electronic Engineering from Nagoya University in 1977. During 1967 to 1969 he worked for Nippon Calculating Machine Corporation. He is now an associate professor of Graduate School of Engineering, University of Fukui.