Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

A Mechanism For Enhanced Symmetric Key Encryption Based Mixnet

Hazim Anas Haddad
Graduate School of Engineering
University of Fukui, Fukui, Japan
h7hazim@radio.fuis.fukui-u.ac.jp

Hiroya Tsurugi
Graduate School of Engineering
University of Fukui, Fukui, Japan

Shinsuke Tamura
Graduate School of Engineering
University Of Fukui, Fukui, Japan
tamura@fuis.fuis.fukui-u.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 a new mechanism for anonymous communication that removes these drawbacks while exploiting symmetric key encryption algorithms.**

***Keywords*—anonymous network, mixnet, privacy protection, symmetric key encryption algorithm.**

I. INTRODUCTION

This paper proposes a new mechanism for anonymous communication. 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. Although, many mechanisms had been established already, e.g. Mixnet [1], DC-net [2], Crowds [3], etc., they still have serious drawbacks. For example, Mixnet is one of the most promising mechanisms, however, it requires tremendous amount of computations to encrypt/decrypt messages that are forwarded from senders to their receivers. That is because asymmetric key encryption/decryption mechanisms are adopted. In this paper, a new anonymous network ESEBM (Enhanced Symmetric key Encryption Based Mixnet) is proposed that removes above drawbacks, while exploiting symmetric key encryption algorithms.

II. EXISTING ANONYMOUS NETWORKS

A. Mixnet

Mixnet is one of the most practical solutions for concealing identities of message senders. It consists of a sequence of mixservers that relay messages from senders to their receivers. The behavior of Mixnet is shown in Fig. 1. In the figure, 3 senders are sending their messages, Msg1, Msg2 and Msg3. Here, senders send their messages while encrypting them repeatedly by public keys of multiple mix-servers in the sequence, and individual mixservers relay their receiving messages to their neighboring servers finally to be sent to the receivers. In this message relaying process, each mixserver stores its incoming messages until pre-defined number of message arrivals, and executes the following two operations before forwarding them to its neighboring server:

1) decrypts incoming messages by its secret key, and
2) shuffles decrypted incoming messages.

Therefore, no entity except the mixserver itself can identify the links between incoming and outgoing messages of the mixserver, and as a consequence, no one except the senders themselves can identify the senders of messages unless all mixservers conspire. However, because Mixnet is using asymmetric key encryption algorithms, such as RSA [4] or ElGamal [5] encryption, a lot of computation overheads are needed to deliver these messages to their receivers. As a result, it becomes difficult to use Mixnet for large scale communication.

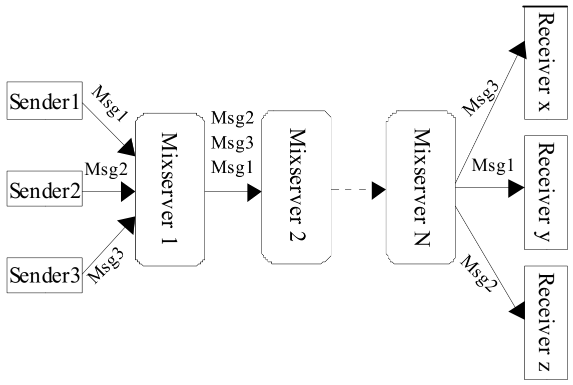


Figure 1. Behovior of Mixnet

B. SEBM (Symmetric Key Encryption Based Mixnet)

To remove the drawbacks of Mixnet, SEBM exploits symmetric key encryption and decryption algorithms [6]. SEBM consists of multiple relay servers as same as Mixnet, however different from Mixnet, it consists of two parts as shown in Fig. 2, i.e. SEBM is configured as follows:

1) servers are divided into 2 parts, i.e. encryption part and decryption part, the former is for message encryption and the latter is for decryption. In the following, servers in the encryption and decryption parts are denoted as encryption and decryption servers, respectively,

2) individual encryption servers are corresponded to single decryption servers, and

3) every possible sender is included in the network as one of servers.

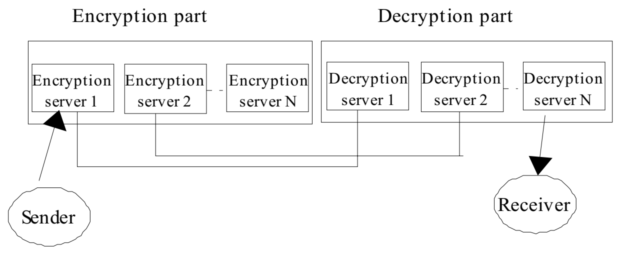


Figure 2. SEBM configuration

A sender of the message encrypts its message by using its secret key and then sends the result to its selecting encryption server. At the same time, HTL (hopes to live) value is assigned to the message by the sender in order to specify that the message will be encrypted HTL times by HTL number of encryption servers. Here, each encryption server encrypts its incoming messages by its secret key and relays the results to its selecting encryption servers while inserting dummy messages and decreasing HTL values by 1. Then the message, which is repeatedly encrypted by HTL number of encryption servers, is sent to a server in the decryption part. At the same time, DHTL (decryption hops to live) value which is sufficiently large compared with the HTL number, is attached by the last server in the encryption part. Decryption servers are arrayed in a sequence and repeatedly encrypted incoming messages are relayed while being decrypted through this sequence by secret keys of individual decryption servers. At the same time, DHTL values are decreased by 1 every time the messages are decrypted by decryption servers, and when DHTL values become 0 or when messages are successfully decrypted, the addresses of receivers are revealed and decryption servers deliver the messages to their receivers. Here, to enable a decryption server to determine whether the incoming repeatedly encrypted message is encrypted by the encryption server corresponding to it or not, and to identify the encryption key used when its corresponding encryption server encrypted the message, a tag list consists of a set of anonymous tags is attached to the message. Namely, when a decryption server finds its anonymous tag in the tag list attached to its receiving message, the message has been encrypted by its corresponding encryption server, and the decryption server can identify the key for decrypting the message based on this anonymous tag.

Here, an anonymous tag in the tag list satisfies the following properties:

1) no one can identify the encryption key from the anonymous tag except the one who attaches it to the message,

2) the server that attaches the anonymous tag can identify its secret key even if the tag is encrypted and decrypted repeatedly by other severs, and

3) no one can trace messages in the network by using anonymous tags.

Although SEBM can reduce the computation overheads caused by asymmetric key encryption algorithms in Mixnet, it includes senders as encryption and decryption servers, which may reduce the performance and the reliability of the communication. For example, when senders, i.e. volunteer servers, stop operations, messages must be re-sent.

III. ESEBM (Enhanced symmetric key encryption based Mixnet)

In order to remove the drawback of SEBM, ESEBM replaces volunteer servers in SEBM by permanent ones. ESEBM is, on one hand, the SEBM, in which volunteer servers are replaced by permanent ones, and on the other hand, can be considered as a kind of decryption type Mixnet, in which asymmetric key encryption is replaced by symmetric one, and messages put into the network are encrypted through the collaboration between the senders and mixservers, not by the senders alone.

A. ESEBM configuration

ESEBM consists of two parts, i.e. the anonymous channel and the concealing pattern generator (CP generator) as shown in Fig. 3. In the figure, the CP generator is configured as M groups, each of which consists of N_j servers, and server $S_j(k)$ in the CP generator corresponds to some server S_p in the anonymous channel; therefore $N = N_1 + N_2 + \cdots + N_M$. In the remainder of this paper, $S_j(k)$ is used to represent S_p, its corresponding server in the anonymous channel, and vise versa.

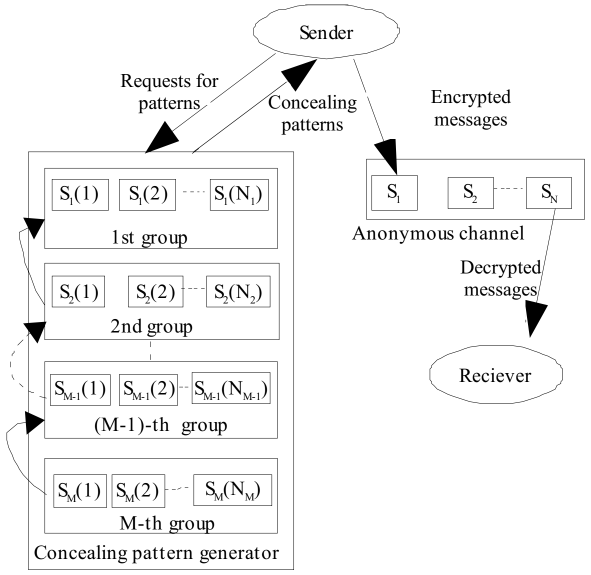


Figure 3. ESEBM configuration

1) *Concealing Pattern Generator* (CP generator) To conceal senders of messages, messages sent from the senders must have different forms from those arriving at their receivers, i.e. if they are same, anyone can identify the senders easily by comparing messages

sent from senders and received by receivers. ESEBM achieves this by XORing the messages by Concealing bit Patterns (CPs). When the length of a message is longer than a CP, the message is divided into multiple frames of which lengths are less than or equal to that of CP and these frames are sent while being XORed by different CPs. The CP generator generates these CPs at the requests of senders in advance, i.e. in off-line processes before actual communication. Here, a CP itself is a XOR of multiple secret random bit patterns $\{x_1, x_2, x_3, ---, x_N\}$ generated by multiple independent servers in the CP generator. Because the behavior of the CP generator is a set of independent activities of multiple servers, no entity can know all x_j unless all servers conspire.

2) *Anonymous Channel* The anonymous channel consists of a sequence of servers, i.e. $\{S_1, S_2, S_3, ---, S_N\}$. As same as usual Mixnets, senders send their messages to the first server S_1 in the anonymous channel, and each server 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 forwarded to their receivers. Different from usual Mixnets, in ESEBM, senders send their messages to the first server while XORing them by CPs, which they acquire from the CP generator in advance (different CPs are used for different messages), and server S_j decrypts its receiving message by simply XORing it by its secret bit pattern x_j that constitutes the CP of the message. To enable S_j to identify the secret bit pattern x_j that constitutes the CP of the message, each message consists of the concealed message part and the tag list part, as shown in Fig. 4.

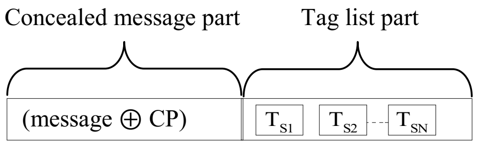


Figure 4. Message structure

In the figure, a concealed (encrypted) message is the XOR of the message and the CP = $(x_1 \oplus x_2 \oplus --- \oplus x_N)$, and the tag list is a sequence of tags $\{T_{S1}, T_{S2}, T_{S3}, ---, T_{SN}\}$, where T_{Sj} is the identifier of the secret bit pattern x_j generated by the server in the CP generator that corresponds to the j-th server S_j in the anonymous channel for constructing the CP. It should be noticed that T_{Sj} must be constructed, so that no one can trace messages by it and no one except S_j can identify x_j from it.

B. Behavior of the CP generator

The CP generator is a set of server groups, each of which consists of at least 3 servers that generate their secret random bit patterns independently of others, as shown in Fig. 3, and senders communicate only with servers in the 1st group, i.e. with $S_1(1)$, $S_1(2)$, ---, and $S_1(N_1)$, to disable servers in the other groups to know the senders. The CP generator generates 2 kinds of random bit patterns, CPs (concealing bit patterns) and TVs (tag concealing vectors). To conceal a message, a CP is constituted as XOR of multiple CP constructors x_j (j = 1, 2, ---, N), and to conceal the tags, individual elements of a TV are constituted as XOR of individual elements in TV constructors $Q_j = \{0, ---0, q_{j(j+1)}, q_{j(j+2)}, ---, q_{jN}\}$, which is generated by S_j (j = 1, ---, N). Here, q_{jk} in Q_j is a bit pattern of length T, which is the length of $ID(x_k, Q_k)$, the tag of S_k, 0 represents an all zero bit pattern of the length T, and a sequence of j zero patterns precedes before the (N-j) random bit pattern sequence. The length of bit pattern x_j is equal to the message frame length, of course. By XORing CP constructors and TV constructors of individual serves, CP and TV are calculated as CP = $(x_{11} \oplus x_{12} \oplus --- \oplus x_{15} \oplus --- \oplus x_N)$, and TV = $\{0, q_{12}, q_{13} \oplus q_{23}, ---, q_{1N} \oplus q_{2N} \oplus --- \oplus q_{(N-1)N}\}$, here, S_N does not generate its TV constructor.

CPs and TVs are generated as follows. Firstly, sender M sends its secret private vectors (PVs) = P_1, P_2, P_3, ---, and P_{N1} as a request for a CP to servers in the 1st group, $S_{1*}, S_{2*}, ---, S_{N1*}$, respectively, as shown in Fig. 5 (a). Here, $P_1 = \{p_{10}, p_{11}, ---p_{1N}\}$, $P_2 = \{p_{20}, p_{21}, ---p_{2N}\}$, ---, and $P_{N1} = \{p_{(N1)0}, p_{(N1)1}, ---p_{(N1)N}\}$, and each p_{jk} except p_{j0} is a bit pattern of the same length as q_{jk}, p_{j0} is a bit pattern of the same length as x_j. Also, it is assumed that $S_1(k)$ corresponds to S_{k*} in the anonymous channel, i.e. $S_1(1) = S_{1*}$, $S_1(2) = S_{2*}$, ---, and $S_1(N_1) = S_{N1*}$. S_{1*} that receives the request with P_1, generates its CP constructor x_{1*} and TV constructor $Q_{1*} = \{0, ---, 0, q_{1*(1*+1)}, q_{1*(1*+2)}, ---q_{1*N}\}$. It also generates $ID_{1*}(x_{1*}, Q_{1*})$ as T_{S1*}, a tag to identify x_{1*} and Q_{1*}. Here, S_{1*} maintains its CP table, a list of CP and TV constructors that it has generated, and $ID_{1*}(x_{1*}, Q_{1*})$ represents the index of the constructor pair $\{x_{1*}, Q_{1*}\}$ in this list. Then, CP_{1*} and TL_{1*}, the concealing pattern and the tag list that S_{1*} generates, are constructed as $CP_{1*} = p_{10} \oplus x_{1*}$ and $TL_{1*} = \{p_{11}, p_{12}, ---, p_{11*} \oplus ID_{1*}(x_{1*}, Q_{1*}), p_{1(1*+1)} \oplus q_{1*(1*+1)}, p_{1(1*+2)} \oplus q_{1*(1*+2)}, ---, p_{1N} \oplus q_{1*N}\}$, respectively. The CP_{1*} and TL_{1*} is then forwarded to S_{2*} (=$S_1(2)$). However, to protect them from eavesdropping, they are encrypted by the secret key K_{1*} that is shared between S_{1*} and S_{2*}, i.e. CP_{1*} and TL_{1*} are sent to S_{2*} in the form $E_{k1*}(CP_{1*}, TL_{1*})$. It is also possible that S_{1*} encrypts CP_{1*} and TL_{1*} by using a public key of S_{2*}, however to decrease encryption overheads, a symmetric key encryption algorithm is adopted here.

S_{2*} that receives $E_{k1*}(CP_{1*}, TL_{1*})$ decrypts it by using the shared secret key K_{1*} to $\{CP_{1*}, TL_{1*}\}$. After that, it generates its CP constructor x_{2*} and TV constructor $Q_{2*} = (0, ---0, q_{2*(2*+1)}, q_{2*(2*+2)}, ---, q_{2*N})$, and modifies CP_{1*} to $CP_{2*} = \{p_{10} \oplus p_{20} \oplus x_{1*} \oplus x_{2*}\}$. S_{2*} also modifies the tag list TL_{1*} to $TL_{2*} = \{p_{11} \oplus p_{21}, p_{12} \oplus p_{22}, ---, p_{11*} \oplus p_{21*} \oplus ID_{1*}(x_{1*}, Q_{1*}), p_{1(1*+1)} \oplus p_{2(1*+1)} \oplus q_{1*(1*+1)}, ---, p_{12*} \oplus p_{22*} \oplus q_{1*2*} \oplus ID_{2*}(x_{2*}, Q_{2*}), p_{1(2*+1)} \oplus p_{2(2*+1)} \oplus q_{1*(2*+1)} \oplus q_{2*(2*+1)}, ---, p_{1N} \oplus p_{2N} \oplus q_{1*N} \oplus q_{2*N}\}$. Here, it is not necessary but to simplify the notations, it is assumed that servers in the anonymous channel are arranged so that $S_j(g)$ comes earlier than $S_j(h)$ when $g < h$, for every j-th group.

As same as S_{1*}, S_{2*} also maintains its CP table, and $ID_{2*}(x_{2*}, Q_{2*})$ represents the index of (x_{2*}, Q_{2*}) in it. Calculated CP_{2*} and TL_{2*} are sent to S_{3*} (=$S_1(3)$) while being encrypted by K_{2*}, a shared key between S_{2*} and S_{3*}, and this process continues until S_{N1*} calculates the CP_{N1*} and TL_{N1*}. Therefore, CP_{N1*} and

TL_{N1*}, the CP and the tag list pair generated by the 1st group become as shown in (1) - (4).

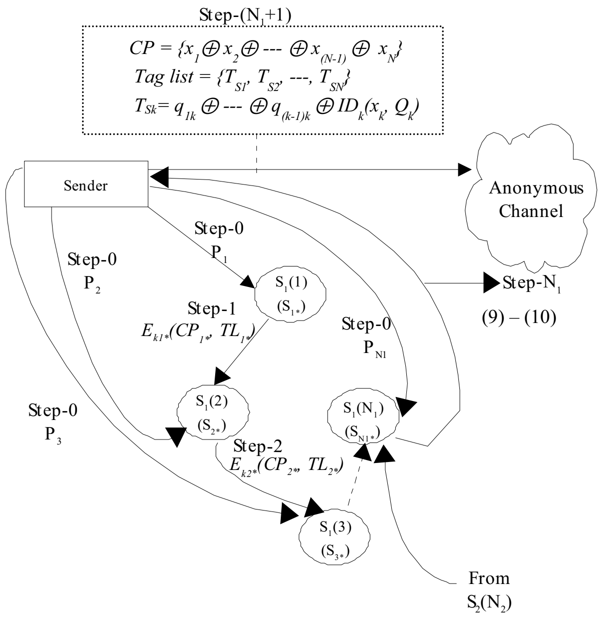
$$CP_{(N1)*} = p_{10} \oplus p_{20} \oplus \text{---} \oplus P_{(N1)0} \oplus x_{1*} \oplus x_{2*} \oplus \text{---} \oplus x_{(N1)*} \quad (1)$$

$$TL_{N1*} = \{T_{S1}, T_{S2}, \text{---}, T_{SN}\} \quad (2)$$

$$T_{Sk*} = p_{1k*} \oplus p_{2k*} \oplus \text{---} p_{(N1)k*} \oplus q_{1*k*} \oplus q_{2*k*} \oplus \text{---} \oplus q_{(k-1)*k*} \oplus ID_{k*}(x_{k*}, Q_{k*}) \quad (3)$$

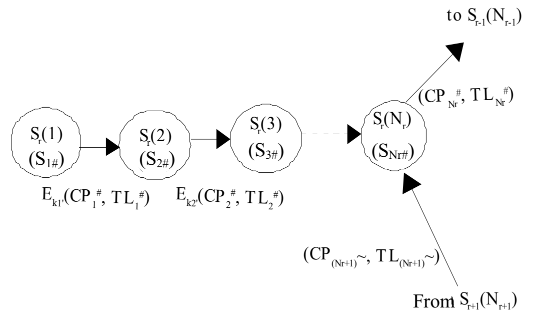
$$T_{Sh} = p_{1h} \oplus p_{2h} \oplus \text{---} p_{(N1)h} \oplus q_{1*h} \oplus \text{---} \oplus q_{(kj)*h} \quad (4)$$

for S_h not included in the 1st group ($k_{j*} < h < k_{(j+1)*}$)



Step-(N_1+1)
$CP = \{x_1 \oplus x_2 \oplus \text{---} \oplus x_{(N-1)} \oplus x_N\}$
$Tag\ list = \{T_{S1}, T_{S2}, \text{---}, T_{SN}\}$
$T_{Sk} = q_{1k} \oplus \text{---} \oplus q_{(k-1)k} \oplus ID_k(x_k, Q_k)$
Sender
Anonymous Channel
Step-0 P_1
Step-0 P_2
Step-0 P_3
Step-0 P_{N1}
Step-N_1
(9) – (10)
$S_1(1)$ (S_{1*})
Step-1 $E_{k1*}(CP_{1*}, TL_{1*})$
$S_1(2)$ (S_{2*})
$S_1(N_1)$ (S_{N1*})
Step-2 $E_{k2*}(CP_{2**}, TL_{2*})$
$S_1(3)$ (S_{3*})
From $S_2(N_2)$

(a) 1st group



(b) r-th group

Figure 5. Behavior of the CP generator

Severs in the r-th group (r > 1) behave in the same way as the 1st group as shown in Fig. 5 (b). However, different from the 1st group, the r-th group generates CP and TV pairs spontaneously without requests from senders, and they do not use private vectors PVs. Therefore, the r-th group calculates CP and the tag list as shown in (5) - (8). Here, it is assumed that $S_r(k)$ corresponds to $S_{k\#}$ in the anonymous channel.

$$CP_{Nr\#} = x_{1\#} \oplus x_{2\#} \oplus \text{---} \oplus x_{Nr\#} \quad (5)$$

$$TL_{Nr} = \{T_{S1}, T_{S2}, \text{---}, T_{SN}\} \quad (6)$$

$$T_{Sk\#} = q_{1\#k\#} \oplus q_{2\#k\#} \oplus \text{---} \oplus q_{(k-1)\#k\#} \oplus ID_{k\#}(x_{k\#}, Q_{k\#}) \quad (7)$$

$$T_{Sh} = q_{1\#h} \oplus q_{2\#h} \oplus \text{---} \oplus q_{(kj)\#h} \quad (8)$$

for S_h not included in the r-th group ($k_{j\#} < h < k_{(j+1)\#}$)

The last server $S_{Nr\#}$ in the r-th group also receives $CP_{(Nr+1)\sim}$ and $TL_{(Nr+1)\sim}$, the CP and tag list pair generated by the (r+1)-th group, from $S_{(Nr+1)\sim}$ ($S_{(Nr+1)\sim}$ is the server in the anonymous channel corresponding to $S_{r+1}(N_{r+1})$), and it calculates XOR of $CP_{(Nr+1)\sim}$ and $CP_{Nr\#}$, and $TL_{(Nr+1)\sim}$ and $TL_{Nr\#}$, to combine CPs and tag lists of the (r+1)-th and r-th groups into the single CP and tag list, respectively. After that, $S_r(N_r)$ waits for the arrivals of predefined number of (CP,TL) pairs, and then shuffles them and sends the combining results to the last server $S_{r-1}(N_{r-1})$ of the (r-1)-th group. As the result of the behaviors of all groups, the last server of the 1st group, i.e. $S_1(N_1) = S_{N1*}$ generates the CP and tag list as (9) – (11).

$$CP_{1*} = \{p_{10} \oplus p_{20} \oplus \text{---} \oplus p_{(N1)0} \oplus x_1 \oplus x_2 \oplus \text{---} \oplus x_N\} \quad (9)$$

$$TL_{1*} = \{T_{S1}, T_{S2}, \text{---}, T_{SN}\} \quad (10)$$

$$T_{Sk} = p_{1k} \oplus \text{---} \oplus p_{(N1)k} \oplus q_{1k} \oplus \text{---} \oplus q_{(k-1)k} \oplus ID_k(x_k, Q_k) \quad (11)$$

Then, S_{N1*} sends the CP and tag list to the sender M, and M removes private vectors PVs from CP and the tag list by XORing them by PVs. As the result, CP and the tag list become as (12) - (14).

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

$$TL_{1*} = \{T_{S1}, T_{S2}, \text{---}, T_{SN}\} \quad (13)$$

$$T_{Sk} = q_{1k} \oplus \text{---} \oplus q_{(k-1)k} \oplus ID_k(x_k, Q_k) \quad (14)$$

It should be noticed that because of PVs, no entity except senders themselves can know CPs and TVs that the senders obtain, unless all server in the 1st group conspire.

C. Behavior of the Anonymous Channel

Fig. 6 shows the behavior of the anonymous channel. Firstly, sender M sends its message m, while XORing it by an arbitrary CP that M acquired from $S_1(N_1)$ and attaching the tag list corresponding to the CP, i.e. it sends m in the form $\{x_1 \oplus x_2 \oplus \text{---} \oplus x_N \oplus m, T_{S1}, T_{S2}, \text{---}, T_{SN}\}$ to the 1st server S_1 in the anonymous channel. Here, T_{S1} has the form $ID_1(x_1, Q_1)$.

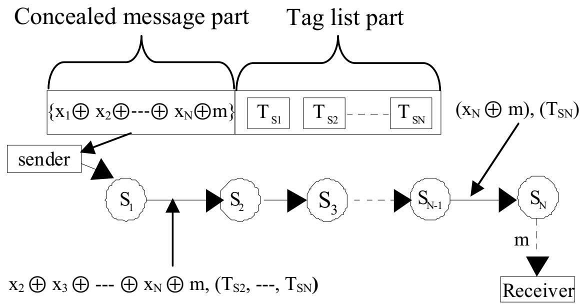


Figure 6. Behavior of Anonymous Channel

Then, S_1 retrieves the CP constructor x_1 and the TV constructor Q_1 from its CP table, based on $ID_1(x_1, Q_1)$ in T_{S1}, and calculates XOR of x_1 and $\{x_1 \oplus x_2 \oplus --- \oplus x_N \oplus m\}$ and XOR of $q_{1j} \oplus T_{Sj}$ for each j ($1 \leq j \leq N$). Therefore, the concealed message is converted into $(x_2 \oplus x_3 \oplus --- \oplus m)$, and T_{Sj} in the tag list is converted into $(q_{2j} \oplus q_{3j} \oplus --- \oplus q_{(j-1)j} \oplus ID_j(x_j, Q_j))$. Then, S_1 removes T_{S1} from the tag list attached to the message, and forwards the results to S_2. Here, S_1 waits for the predefined number of message arrivals, and shuffles them before forwarding them to S_2, to conceal the links between its incoming and outgoing messages.

All servers in the anonymous channel perform the same operations, therefore, S_j converts its incoming concealed message and the tag list to $\{x_{j+1} \oplus --- \oplus x_N \oplus m\}$ and $\{Ts_{j+1}, Ts_{j+2}, ---, T_{SN}\}$, where $T_{Sk} = \{q_{(j+1)k} \oplus --- \oplus q_{(k-1)k} \oplus ID_k(x_k, Q_k)\}$. Consequently when S_N, the last server in the anonymous channel, completes its operation on the message, it is converted into m. Then S_N can deliver m to its receiver while extracting its receiver's address from m.

IV. Analysis of ESEBM Behavior

This section discusses behavior of ESEBM against various threats. Firstly, concealed messages themselves are secure enough, i.e. it is impossible to identify senders of messages by tracing concealed message parts of individual messages unless all servers conspire. Since, each server S_j knows only single CP constructor x_j included in a CP, it cannot link concealed message parts of incoming messages of S_h to its outgoing messages when $j \neq h$, therefore, no single entity can link concealed message parts of messages sent from senders to that received by receivers.

Any entity cannot identify message senders by tracing tags of the messages either. Because $ID_j(x_j, Q_j)$ has the meaning only to S_j ($ID_j(x_j, Q_j)$ is an index for CP table of S_j), no one can trace messages by it. Also, because elements $\{q_{j(j+1)}, ---, q_{jN}\}$ of Q_j are mutually independent, it is impossible for S_h to link incoming messages of server S_j ($h \neq j$) to its outgoing messages by comparing the pattern transitions of individual tags made by S_j. Although, each server S_{j*} in the 1st group can know the sender from its tag, because S_{j*} generates its tag as the response to the sender's request, when S_{j*} is placed at the early position of the anonymous channel, its tag disappears in the later positions. Namely, tag lists 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 of the CP generator, therefore even S_{j*} conspires with servers at the later positions, it is not possible to identify senders.

About, well known, replay attacks, an entity can identify the receiver of message m_S by eavesdropping m_S from the network, and putting m_S to the network repeatedly. Because the same m_{S*}, the decrypted form of m_S, are delivered to the same receiver R many times, the entity can easily identify the sender of m_{S*} by linking m_s to the frequently arriving message m_{S*} at R. Protection of replay attacks is easy for ESEBM, i.e. it is enough for S_j to remove CP constructor and TV constructor corresponding to index $ID_j(x_j, Q_j)$ from its CP table, when it receives a message that includes $ID_j(x_j, Q_j)$ as its tag. When S_j receives a replayed message, S_j simply discards it from the network, because S_j cannot get consistent $ID_j(x_j, Q_j)$ from S_{j-1} for finding CP and TV constructors in its CP table.

Another possible threat is the message modification caused by servers. Because message senders are not known, modification may cause more serious effects in anonymous networks, e.g. receivers cannot confirm eligibilities of their receiving messages. ESEBM also removes this kind of threats easily. Because the senders receives CPs that are XORed by its secret PVs, no server can know the exact CP values applied to individual messages; therefore modified messages cannot have consistent CPs, and they will be decrypted into meaningless messages.

V. Conclusion

In this paper, a new anonymous network has been proposed based on symmetric key encryption algorithms. As a drawback of ESEBM, a sender must acquire a concealing pattern from the CP generator in advance to send its every message. However because of ESEBM configuration, i.e. by dividing the network into the CP generator and the anonymous channel parts, every time-consuming task is removed from the anonymous channel part and ESEBM enables highly efficient communication. According to the preliminary evaluation, different from usual mixnets which cannot relay more than 1KB length messages in every 300 msec, ESEBM successfully relays messages even their length become more than 10KB, i.e. computation load of each server in the anonymous channel remains less than few percent.

As future works, firstly, efficient mechanisms that enable message senders to confirm delivery of their messages and that enable message receivers to send their response messages must be developed. About the response messages, the simplest solution is to attach encrypted return addresses to individual messages. Namely, the message sender attaches its own address and tag list, to its message while XORing them by an unused CP that the sender acquired. Then, the receiver can reply to its receiving message without knowing its sender by XORing its message by the encrypted return address, and attaching the received tag list. Here, the receiver generates its reply message as follows. Firstly, the encrypted return address generated by the sender has the form shown is Fig. 7 (a), i.e. the sender generates it by putting its address and zero patterns in the address and message parts, respectively, and XORing the result by the CP. Then, the receiver generates its reply message, by putting zero patterns and its message in the address and the message parts in the reply message as shown in Fig. 7 (b), and by XORing the result by its receiving encrypted return address. However, the sender must attach its encrypted reply address and tag list as additional frames, because the size of the encrypted reply address is just as same as that of CPs. Also, this solution does not allow a receiver to send multiple replies to the sender. Moreover, when the receiver puts a copy of its receiving message in the message part of the reply message, the sender of the original message will be revealed, if the last server and the first server in the anonymous channel conspire, because the original and reply messages are the same. It becomes more serious when the receiver and the last server conspire; because the receiver knows its response message, the last server can identify the senders of the original message by tracking where the known response message reaches. These threats can be removed by a mechanism that changes the response message to the forms known only to the original senders. Although it is not a problem usually, the sender of the original message can trace the reply message of course, because the sender knows the receiver from the beginning.

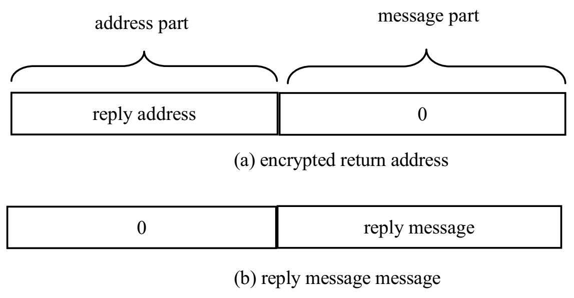


Figure 7. Structure of Response

As another issues, mechanisms that enhance the reliability of ESEBM are also 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.

ESEBM shares common threats such as Denial of service (DOS) attacks [7] with not only other anonymous networks but also usual non-anonymous networks, and mechanisms to mitigate effects of these attacks also must be developed. Effects of DOS attacks can be reduced when different messages use different servers as the 1st server in the anonymous channel. Namely, the computation load for discarding invalid messages does not concentrate on the fixed 1st server, i.e. it can be distributed over different servers. However, in this case tag finding processes of individual servers become complicated.

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] David Chaum, “The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability, ” Journal of Cryptology, vol. 1, pp. 65-75, 1988.

[3] Reiter, M. K. and Rubin, “Crowds: anonymity for Web transactions,” ACM Trans. Inf. Syst. Secur,vol. 1, no. 1, pp. 66-92, Nov 1998.

[4] Rivest, R., A. Shamir, L. Adleman “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems,” Communications of the ACM, vol. 21, no.2, pp. 120–126, 1978.

[5] Taher ElGamal, “A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms,” IEEE Transactions on Information Theory, vol. IT-31, no. 4, pp. 469–472 or CRYPTO, vol. 84, pp. 10–18, 1985.

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

[7] Znati, T., Amadei, J., Pazehoski, D. R., and 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, Mar 2007.