Anonymous return route information for onion based mix-nets

Yoshifumi Manabe
NTT Communication Science Laboratories
NTT Corporation
Atsugi, Kanagawa 239-0198 Japan
manabe.yoshifumi@lab.ntt.co.jp

Tatsuaki Okamoto
NTT Information Sharing Platform Laboratories
NTT Corporation
Musashino, Tokyo 180-8585 Japan
okamoto.tatsuaki@lab.ntt.co.jp

ABSTRACT

This paper proposes a return route information encryption scheme for onion-based e-mail systems and mix-nets. Our scheme has the following two properties. (1) It allows any node on the message route to send reply messages to the sender of the message. This property is necessary for sending error replies. (2) It allows the replying node to send multiple reply messages from one piece of return route information. This property is necessary when responding with large amounts of data using multiple messages. In order to construct a return route information scheme, we must consider a new type of attack, namely the replace attack. A malicious node obtains information about the route by replacing secret information that only the node can read. This paper describes the new type of attack and shows that previous schemes are vulnerable to it. Our scheme prevents replace attacks. In addition, we show that by slightly modifying our scheme malicious nodes cannot distinguish whether a message is a forward message or a reply message, thus improving the security of the routing scheme.

Categories and Subject Descriptors

E.3 [Data]: Data Encryption—Public key cryptosystems; C.2.2 [Computer-Communication Networks]: Network Protocols—Routing Protocols

General Terms

Security, Theory

Keywords

anonymous communication, route information, return address, mix-nets

1. INTRODUCTION

E-mail systems are widely used in business and daily life. The names of the sender and receiver of each e-mail are provided in plain text in the e-mail header. Thus, every node in the network can obtain the names of the sender and the receiver when an e-mail is relayed (This paper does not discuss spam e-mails that forge the name of the sender).

We sometimes want to send e-mail anonymously, for example, when whistle-blowing. If we are to achieve anonymity, no node in the network should be able to obtain information about the sender or receiver of an e-mail, although the nodes must be able to obtain the route information needed to relay the e-mail. Onion-based mix-nets[1][3][8][10][11][12][13][15][19] is a mechanism for sending e-mails anonymously, which is one of the techniques of anonymous communication [6]. In onion-based mix-net schemes, route information is appended to each message, thus multiple messages with the same pair of the sender and the receiver might be sent by different routes. Note that as an alternative mechanism, many works, for example, Onion-routing[9] and Tor [7], have been carried out to create a virtual circuit that can be used for bidirectional anonymous communication[6].

The problem of sending reply messages using onion-based mix-net has not been carefully discussed. Most of previous works considered sending one reply message from the destination node. Two important properties, multiple reply senders and multiple reply messages, must be considered.

Let us consider the first problem, the sender of the reply messages. There are two cases where a reply is sent. The first is where the reply is sent by the receiver. This type of reply is necessary, for example, when responding to whistle-blowers. The second case involves sending error messages from the nodes on the route. This type of reply message is necessary when the address used by the sender is incorrect or the node/link for relaying the message is down. Without an error reply mechanism, the sender cannot know that the message was not sent correctly. This type of error message is necessary even if the network is robust and errors rarely occur. Since it is not realistic for the entire e-mail system to support onion-based e-mail routing, the easiest implementation for the current Internet may be a mix-net operated by a number of volunteer servers. Messages between servers are sent via the Internet using encryption between the servers. A message sender randomly selects a sequence of servers and appends the sequence before the final destination. Since the servers are provided voluntarily, some servers might suddenly stop providing the service. Thus, such mix-nets need a mechanism for sending error replies to inform the sender that a server is not operating.

The second problem is sending multiple reply messages from one replying node. Every onion-based mix-net scheme assumes that the size of the message content is fixed, in order to prevent malicious nodes obtaining the route of a message from its size. Thus the reply needs to be sent by multiple messages if its size is large (e.g. image data). However, many schemes such as those in [1][2], the replying node cannot randomize the obtained return route information, thus relaying nodes can detect that the same route information is used multiple times. (Note that in [1], the relaying nodes discard messages that have same route information to prevent a replay attack. Thus it is impossible to send multiple reply messages with the same route information). Some mechanisms that allows multiple replies were proposed [4][14]. Our scheme allows any relaying node to send multiple reply messages with the same route information.

For the return route information a malicious node can execute a new type of attack, the replace attack. A malicious node obtains information about the route by replacing secret information that only the node can read. Our scheme prevents the replace attack.

In addition, with a slight modification, the scheme described in this paper can become symmetric with respect to forward and reply messages with the result that malicious nodes cannot distinguish these messages.

2. DEFINITION OF PROBLEM

The definition in [1] does not consider the ability to send reply messages from any node on the route, thus we need to redefine the problem.

The network topology is represented by an undirected graph $G = (V, E)$ where V is the set of nodes and E is the set of edges. Each node $v \in V$ acts as a sender, a receiver, and a message relayer. Edge $e = (v_i, v_j)$ is a bidirectional communication channel between v_i and v_j. G is public information. Note that G is a complete graph when any node can directly send messages to any other node. Each node v_i encodes a special character $\perp_i$ (meaning v_i is the destination of the message) and edges connected to v_i, to a message space M_i. In the rest of the paper, when an edge e appears in encrypted route information, e is not a pair of nodes but an encoded message.

A message M consists of message content m and route header h. When a node S sends message content m to a node R, S first obtains a path on G, $p = v_0(= S), e_1, v_1, \ldots, v_{n-1}, e_n, v_n(= R)$ from S to R, where $e_i = (v_{i-1}, v_i) \in E(1 \le i \le n)$. Let $p(S, R)$ denote a path from S to R. Throughout this paper, we use this path for explanation. There are two ways to obtain a route header for a certain path. The first consists of S selecting a path $p(S, R)$ on G from S to R and encrypting it. The second consists of obtaining a route header while relaying or receiving a message. We call the former type of route information forward route information and the latter type of route information return route information. A message sent using forward (return) route information is called as a forward (return) message.

When node v receives message M and relays it to the next edge, it obtains new route header h' from v to M's original sender, that is, h' is an encryption of $p(v, S)$. In addition, when R receives a message M whose destination is R, R obtains new route header h'' from R to M's original sender, that is, h'' is an encryption of $p(R, S)$. We denote $h(M)$ as the header of message M and $pa(M)$ as the path represented by $h(M)$.

Each node generates a pair consisting of a public key and a private key. Let $\mathcal{A}$ be an adversary. $\mathcal{A}$ can corrupt any node and modify the incoming messages at the corrupted nodes. Assume that $\mathcal{A}$ never refuses to relay a message. It never discards messages or rewrites the headers that prevent messages being sent to the correct receiver. Note that this assumption does not exclude a message denial attack. This paper discusses the adversary's action before the day of a message denial attack. Before that day, the adversary does not want to be detected as an adversary, thus it is honest about relaying messages but it tries to collect route information from the relaying messages. On the attack day, using the collected route information, the adversary stops relaying messages whose destination is specified nodes (or executes some other type of attack). Without collecting such information, the adversary can only execute a trivial message denial attack, for example, it can only stop sending all messages or randomly selected messages.

Adversary might study the timings of messages moving through the system to find correlations. To prevent timing analysis, we assume that there is a sufficient volume of background message traffic on every edge by employing a method such as those described in [16][17][18].

The anonymity of route information demands that no message be linked to any other message by anyone. We describe an unlinkability experiment, $Exp_{\mathcal{A}}$, as follows. There is a route information oracle O. When O receives a route header from $\mathcal{A}$, O decrypts it and responds to $\mathcal{A}$.

Experiment $Exp_{\mathcal{A}}$

(1) $\mathcal{A}$ first selects any path $p(S, R)$ and forces S to send a message M to R by $p(S, R)$. By sending M, every node v_i on $p(S, R)$ obtains a return route header h_i for path $p(v_i, S)$. $\mathcal{A}$ can force any (non-corrupted or corrupted) node v_i to send a message using h_i. In addition, $\mathcal{A}$ can ask route information oracle O to decrypt the route header h of any message on any edge. $\mathcal{A}$ can execute this procedure as many times as $\mathcal{A}$ wants.

(2) $\mathcal{A}$ outputs either (2-1) a non-trivial relation Rel between two messages or (2-2) a non-trivial relation Rel' between a message and a node.

(3) $\mathcal{A}$ then executes step (1) again except for the limitation that $\mathcal{A}$ cannot use oracle O.

(4) $\mathcal{A}$ outputs two messages M_1 and M_2 or a pair consisting of message M_1 and a node v according to whether Rel or Rel' is output in (2). These messages are those that $\mathcal{A}$ observes during step (3).

The advantage of $\mathcal{A}$, $Adv_{ul}(\mathcal{A})$, is defined as
$Prob[Rel(M_1, M_2)]$ (or $Prob[Rel'(M_1, v)]$). ■

Adversary $\mathcal{A}$ (t, ϵ)-breaks the unlinkability problem if it runs in time at most t and $Adv_{ul}(\mathcal{A})$ is at least ϵ. The route information system is (t, ϵ)-secure if no adversary $\mathcal{A}$ (t, ϵ)-breaks the unlinkability problem.

The trivial relation between two messages, $TRel(M_1, M_2)$ is as follows. $M_i(i = 1, 2)$ are observed at different edges $e_i(i = 1, 2)$, M_1's route $pa(M_1)$ contains e_1 and e_2, and all nodes on $pa(M_1)$ between e_1 and e_2 are corrupted by $\mathcal{A}$. It is easy to detect whether M_1 and M_2 are the same message from the above corrupt condition.

Similarly, the trivial relation between a message and a node, $TRel'(M, v)$ is as follows. M is observed at edge e, M's route $pa(M)$ contains v, and all nodes on $pa(M)$ between e and v are corrupted by $\mathcal{A}$.

We consider the following non-trivial relations when not every node on the path is corrupted.

(1) M_1 observed at e_1 and M_2 observed at e_2 are the same message.

(2) M_1 and M_2 use the same route information.

(3) M_2 is a reply message to M_1.

(4) $pa(M_1)$ contains v_i.

This paper provides some assumptions to solve the problem.

Assumption 1. At step (4) of $Exp_{\mathcal{A}}$, $\mathcal{A}$ must not output message M whose sender or receiver is corrupted.

Assumption 2. $\mathcal{A}$ must not corrupt a pair of adjacent nodes.

The reason of these assumptions are shown later. Finding a way to eliminate these assumptions requires further study.

3. PREVIOUS RESULTS

A simple implementation for sending reply messages is as follows. Each relaying node records a tuple (M_0, M_1, e_1) consisting of an incoming message M_0, an outgoing message M_1, and the edge e_1 from which M_0 arrived in its local log. When a node wants to send reply message M' to message M_k, the initiator of M' sends the pair (M', M_k) to the edge from which M_k arrived. The node that receives reply message (M', M_k) searches the log and finds a tuple (M_{k-1}, M_k, e_k), then it relays (M', M_{k-1}) to e_k. By repeating this procedure, M' will arrive at the original sender of M. This mechanism is not secure because a malicious node can obtain the relationship between a message and its reply.

Three solutions have already been reported that enable the receiver node to send a reply message. Chaum [2] proposed a return address as follows (we omit the message content part):

$K_n(e_n, K_{n-1}(e_{n-1}, \ldots, K_1(e_1, K_0(\perp_0)) \cdots)$, where K_i is the public key of v_i.

$v_i (i = n, n-1, \ldots, 1)$ decrypts the information encrypted by K_i and obtains e_i. For v_i to obtain e_i, this route information must be decrypted by $v_n, v_{n-1}, \ldots, v_{i+1}$ in advance. When v_i needs to send an error message because v_{i+1} (or e_{i+1}) is down, v_i cannot obtain e_i because the route information is encrypted by the key of v_{i+1}. In order to use this type of return route information, v_0 needs to prepare one piece of return route information originating at each intermediate node,

$K_n(e_n, K_{n-1}(e_{n-1}, \ldots, K_1(e_1, K_0(\perp_0)) \cdots)$

$K_{n-1}(e_{n-1}, K_{n-2}(e_{n-2}, \ldots, K_1(e_1, K_0(\perp_0)) \cdots)$

$K_{n-2}(e_{n-2}, K_{n-3}(e_{n-3}, \ldots, K_1(e_1, K_0(\perp_0)) \cdots)$

and so on. Thus the total size of the return route information is $O(n^2)$. We need a mechanism that does not increase the size of the route information.

Camenisch et al. [1] proposed an onion-based message routing scheme and its reply mechanism. However, in their return route information scheme, the procedure at each node is deterministic, that is, when a message arrived at v the output from v is uniquely defined. Replay attack is avoided by destroying messages that use the same header twice. This means that the receiver node cannot send multiple reply messages to a given message. In addition, their scheme needs a non-standard assumption on the pseudorandom permutations they use.

Toriyama et al. [19] proposed a mechanism for obtaining return route information while messages are forwarded by onion-based message routing. v_j adds return edge information e_j to message M while sending M. The following replace attack can be performed on their scheme. A malicious node v_j adds e'_j instead of e_j. v_j can detect that message M' is a reply to M by obtaining edge e'_j instead of e_j (and continues sending messages correctly by relaying this reply message to e_j). v_j can thus obtain the relationship between a message and its reply. To prevent the replay attack, the return route information must be set by the original sender S.

4. FORWARD ROUTE INFORMATION

First, we outline the ElGamal type route information randomization mechanism described in [19], which is modified from that in [10], which is proved to be insecure in [5]. Our scheme uses the following mechanism for the forward route information.

The main idea is exactly the same as standard onion routing, namely that the route information that v_j must use, e_{j+1}, is encrypted as $Enc(pk_1, Enc(pk_2, \ldots, Enc(pk_j, e_{j+1}) \ldots))$. Every node decrypts it by using its secret key and v_j obtains e_{j+1}.

Let g be a generator of cyclic group $\mathcal{G}$ whose order is a large prime. Node v_j selects $x_j \overset{R}{\leftarrow} Z_p^*$ and sets $y_j \leftarrow g^{x_j}$. The public key of v_j is y_j and the private key of v_j is x_j.

The basic ElGamal encryption scheme is as follows. For a given plaintext (encoded edge information) e_{j+1}, randomly selects $r \overset{R}{\leftarrow} Z_p^*$ and the ciphertext (X, Y) is $(g^r, e_{j+1} \cdot y_j^r)$. For a ciphertext (X, Y), obtain the plaintext by Y/X^{x_j}. The scheme in [19] modifies the scheme such that encryption is performed by $(X, Y) \leftarrow (g^r, y_j^{r \cdot e_{j+1}})$ and decryption is performed by searching for the value e_{j+1} that satisfies $Y = X^{x_j \cdot e_{j+1}}$. Since in the route information scheme, the set of edges (the message space) v_j is connected to is a relatively small fixed set, this type of encryption/decryption is possible.

The main idea of [19] is as follows. The route information that v_j must obtain, e_{j+1}, is encrypted at the sender node as $(g^r, y_1^r \cdot y_2^r \cdot \ldots \cdot y_{j-1}^r \cdot y_j^{r \cdot e_{j+1}})$. When v_1 receives this information, v_1 partially decrypts it and obtains $(g^r, y_2^r \cdot \ldots \cdot y_{j-1}^r \cdot y_j^{r \cdot e_{j+1}})$. Then v_2, $v_3 \ldots$ decrypts it and when this message comes to v_j, the route information becomes $(g^r, y_j^{r \cdot e_{j+1}})$ and v_j can obtain e_{j+1}. To hide the relationship between incoming messages and outgoing messages, this route information is randomized at each node. The details are provided below.

(Initialization) Let us consider a case where a node S sends a message m to a node R by the route $p(S, R)$ defined above. From the public keys, S generates message $(\beta, \alpha_1, \alpha_2, \ldots, \alpha_N, \gamma_1, \gamma_2, \gamma_3, \gamma_4)$ as follows.

S chooses $r \overset{R}{\leftarrow} Z_p^*$ and sets $\beta \leftarrow g^r$,
$\alpha_i \leftarrow (y_1 \cdot y_2 \cdot \ldots \cdot y_{i-1})^r \cdot y_i^{e_{i+1} \cdot r} (1 \leq i \leq n-1)$, and
$\alpha_n \leftarrow (y_1 \cdot y_2 \cdot \ldots \cdot y_{n-1})^r \cdot y_n^{\perp_{n+1} \cdot r}$.

$\perp_{n+1}$ indicates that v_n is the destination of the message, that is, $\perp_{n+1}$ is a special value that differs from all the values for edges connected to v_n. $\alpha_{n+1}, \alpha_{n+2}, \ldots, \alpha_N$ are dummy values to hide the length of the actual route.

$\gamma_i (i = 1, 2, 3, 4)$ carries the message content m. S selects $r_1, r_2 \overset{R}{\leftarrow} Z_p^*$ and sets
$\gamma_1 \leftarrow m \cdot (y_1 \cdot y_2 \cdot \ldots \cdot y_{n-1} \cdot y_n)^{r_1}$, $\gamma_2 \leftarrow g^{r_1}$,
$\gamma_3 \leftarrow (y_1 \cdot y_2 \cdot \ldots \cdot y_{n-1} \cdot y_n)^{r_2}$, and $\gamma_4 \leftarrow g^{r_2}$.

S permutes $\alpha_i (1 \leq i \leq N)$ and sends the message to e_1.

(Decryption) When a message $(\beta, \alpha_1, \ldots, \alpha_N, \gamma_1, \gamma_2, \gamma_3, \gamma_4)$ arrives at node v_j, it decrypts the route header and obtains the next edge to relay this message. v_j then re-encrypts all the values and sends them to the next edge. The procedure is shown below.

v_j searches for $\alpha_i (1 \leq i \leq N)$ that satisfies $\alpha_i = \beta^{e_{j+1} \cdot x_j}$ for some e_{j+1} or $\alpha_i = \beta^{\perp_{j+1} \cdot x_j}$. Let α_k be the element that satisfies the above condition.

If the latter condition is satisfied, v_j is the destination of this message. v_j obtains the message content m by calculating $\gamma_1 / {\gamma_2}^{x_j}$.

If the former condition is satisfied, e_{j+1} is the edge to which v_j must relay this message. v_j decrypts every element of the header $\alpha_i (1 \leq i \leq N, i \neq k)$ by calculating $\bar{\alpha_i} \leftarrow \alpha_i / \beta^{x_j}$.

α_k is no longer necessary, thus v_j sets $\bar{\alpha_k}$ as a random dummy value.

For the elements $\gamma_i (i = 1, 2, 3, 4)$ that carry the message content, v_j calculates
$\bar{\gamma_1} \leftarrow \gamma_1 / \gamma_2^{x_j}$ and $\bar{\gamma_3} \leftarrow \gamma_3 / \gamma_4^{x_j}$.

(Re-randomization) To prevent a replay attack, every block must be randomized at v_j. Let $(\beta, \bar{\alpha_1}, \bar{\alpha_2}, \ldots, \bar{\alpha_N}, \bar{\gamma_1}, \gamma_2, \bar{\gamma_3}, \gamma_4)$ be the message after the above decryption is performed. v_j selects $r_0, r_1, r_2 \overset{R}{\leftarrow} Z_p^*$ and calculates
$\widetilde{\alpha_i} \leftarrow \bar{\alpha_i}^{r_0} (1 \leq i \leq N)$,
$\widetilde{\beta} \leftarrow \beta^{r_0}$,
$\widetilde{\gamma_1} \leftarrow \bar{\gamma_1} \cdot \bar{\gamma_3}^{r_1}$,
$\widetilde{\gamma_2} \leftarrow \gamma_2 \cdot \gamma_4^{r_1}$,
$\widetilde{\gamma_3} \leftarrow \bar{\gamma_3}^{r_2}$, and
$\widetilde{\gamma_4} \leftarrow \gamma_4^{r_2}$.

$(\widetilde{\beta}, \widetilde{\alpha_1}, \widetilde{\alpha_2}, \ldots, \widetilde{\alpha_N}, \widetilde{\gamma_1}, \widetilde{\gamma_2}, \widetilde{\gamma_3}, \widetilde{\gamma_4})$ is the new message.

(Permutation) After the above re-randomization, v_j permutes $\widetilde{\alpha_1}, \widetilde{\alpha_2}, \ldots, \widetilde{\alpha_N}$ at random. ■

In the scheme in [10], each edge value e_i is stored in an independent block, that is, e_i is stored in $(\alpha_0, \beta_0, \alpha_1, \beta_1)$, where $\beta_0 \leftarrow g^{r_0}$, $\alpha_0 \leftarrow e_{i+1} \cdot (y_1 \cdot y_2 \cdot \ldots \cdot y_{i-1} \cdot y_i)^{r_0}$, $\beta_1 \leftarrow g^{r_1}$, and $\alpha_1 \leftarrow (y_1 \cdot y_2 \cdot \ldots \cdot y_{i-1} \cdot y_i)^{r_1}$.

The decryption at v_j is $\bar{\alpha_0} \leftarrow \alpha_0 / \beta_0^{x_j}$.

An attack on this scheme is described in [5]. Since (α_1, β_1) is an encryption of plaintext '1', the relaying node can add a false route to the malicious node, e_x, by executing $\beta'_0 \leftarrow \beta_1$, $\alpha'_0 \leftarrow e_x \alpha_1$, $\beta'_1 \leftarrow \beta_1^{r'}$, and $\alpha'_1 \leftarrow \alpha_1^{r'}$.

The above attack cannot be applied to the scheme described by Toriyama et al., because there is no encryption of '1' in the route information. In their scheme, every block uses the common value β. v_j cannot replace α_k with valid route information because v_j does not know $r \in Z_p^*$ that satisfies $\beta = g^r$.

On the other hand, the message block is just the same as that in [10], thus the following replay attack [5] is possible if some node on the route and the receiver R are malicious. Message m is stored in (γ_1, γ_2), where $\gamma_1 = m \cdot y_i^{r_0} \cdot y_{i+1}^{r_0} \cdot \ldots \cdot y_n^{r_0}$ and $\gamma_2 = g^{r_0}$ for some r_0. The malicious node v_i sends a new message with (γ_1^t, γ_2^t), which is a ciphertext of m^t. When receiver R obtains m and m^t, it detects that the route contains v_i. Thus, this paper assumes that the receiver of the message is not corrupted in Assumption 1.

Note that this scheme is not secure if $\mathcal{A}$ corrupts the sender S and some node v_j on the route. S uses a special value e'_{j+1} instead of e_{j+1} in α_j. When v_j decrypts the data and obtains the special value e'_{j+1}, v_j can detect that this message was sent by S. v_j can continue relaying this message to e_{j+1} by hearing the next edge from S using some method. Using the information, $\mathcal{A}$ can discard all messages whose sender is not $\mathcal{A}$. This type of attack cannot be prevented in schemes those described in [1][2][19]. One way to prevent this type of attack is for S to ask a trusted third party to obtain the route information. However, this might not be practical for all users to ask for the route information from a trusted third party. Thus, Assumption 1 excludes the case where the sender of a message helps to detect the route information. Finding a way to eliminate this assumption will require further study.

5. PROPOSED RETURN ROUTE INFORMATION SCHEME

A natural implementation of the return route information may be exactly the same as for the forward route information in the previous section.

Let us consider a case where the return route information (θ, δ_j) is encrypted so that v_j can obtain e_j, that is, $(\theta, \delta_j) = (g^r, y_1^r \cdot y_2^r \cdot \ldots \cdot y_{j-1}^r \cdot y_j^{r \cdot e_j})$.

To decrypt this route information, each node executes $\bar{\delta_j} \leftarrow \delta_j / \theta^{x_i}$. Thus δ_j becomes $y_j^{r \cdot e_j}$ at v_j and v_j can obtain return route information e_j.

When this message goes beyond v_j without replacing with a dummy value, δ_j is decrypted using the secret keys x_{j+1}, $x_{j+2} \ldots$ that were not encrypted at all as $\bar{\delta_j} \leftarrow \delta_j / \theta^{x_i}$. Thus δ_j becomes $y_j^{r \cdot e_j} / (y_{j+1}^r \cdot y_{j+2}^r \cdot \ldots \cdot y_{j'}^r)$ at $v_{j'}$.

When v'_j sends a reply message, this unnecessary decryption can be cancelled out by executing $\bar{\delta_j} \leftarrow \delta_j \cdot \theta^{x_i}$ while relaying the reply at v_i. Thus, when the reply message arrives at v_j, δ_j becomes $y_j^{r \cdot e_j}$ again and v_j can obtain e_j. Although this mechanism seems to work, we must consider the replace attack described below.

In the above procedure, v_j sees the same edge information e_j twice: once while relaying a forward message and once while relaying its reply message. Thus the following replace attack is possible. When $v_j \in \mathcal{A}$ decrypts e_j from δ_j while relaying a forward message M, it replaces e_j with a special value e'_j by calculating $\delta'_j \leftarrow {\delta_j}^{e'_j / e_j}$. When v_j receives a reply message M', it can detect that M' is a reply message for M if it obtains e'_j. Therefore, the relationship between these two messages can be detected.

The idea for preventing the replace attack is as follows. When v_j wants to send a reply message M' to the received message M, it is unnecessary for v_j to calculate the next edge to send M'. It should simply return M' to the edge M arrived from. This mechanism does not require a log that must be kept for a long time by intermediate nodes, because each intermediate node can detect instantly whether or not the next node or edge is down. At the destination node R, the edge information needed to send a reply to message M can be kept by R together with M.

On the other hand, when a reply message is relayed to v_j from some other node, it is necessary for v_j to obtain the next edge. Thus, return route information e_j is unnecessary until M is forwarded to v_{j+1}, which is the next node on the forward route. e_j is encrypted using $x_1, x_2, \ldots, x_j$ and x_{j+1} so that v_j cannot decrypt e_j when v_j receives M from v_{j-1}.

e_j is encrypted as follows. $(\theta, \delta_j) = (g^r, y_1^r \cdot y_2^r \cdot \ldots \cdot y_{j-1}^r \cdot y_j^{r \cdot (e_j + 1)} \cdot y_{j+1}^r)$, where y_{j+1}^r is the extra term. At v_j, this route information becomes $(\theta, \delta_j) = (g^r, y_j^{r\cdot(e_j+1)} \cdot y_{j+1}^r)$ as a result of the decryption at $v_1, v_2, \dots v_{j-1}$. v_j cannot obtain e_j from this route information because of the extra term y_{j+1}^r. Since v_j cannot know that this is the return route information for v_j, v_j executes δ_j/θ^{x_j}, which is the procedure for the other return route information. Then this route information becomes $(g^r, y_j^{r\cdot e_j} \cdot y_{j+1}^r)$. When this route information is sent to v_{j+1}, v_{j+1} converts it by a special extra decryption rule described later, to $(g^r, y_j^{r\cdot e_j}/y_{j+1}^r)$, which is the normal form of the return route information at v_{j+1}. When this route information is sent to $v_{j+2}, \dots$ and then returned to v_{j+1}, the return route information becomes $(g^r, y_j^{r\cdot e_j}/y_{j+1}^r)$ again. Then, v_{j+1} executes $\delta_j \cdot \theta^{x_{j+1}}$ and the route information becomes $(g^r, y_j^{r\cdot e_j})$ and v_j can obtain e_j. By employing this mechanism, v_j sees e_j just once and a replace attack becomes impossible.

The above mechanism needs v_{j+1} to execute an extra decryption for the block δ_j that encrypts e_j. To inform v_{j+1} of the necessity of an extra decryption, pairs of $(\alpha_{i+1}, \delta_i)(0 \leq i \leq n-1)$ are made, where α_{i+1} is the encryption of e_{i+2} in the forward route information. Since edge information e_{j+2} is obtained from α_{j+1} only at v_{j+1}, it can be used to inform v_{j+1} that it needs to execute an extra decryption to δ_j.

The following describes the proposed system in detail. Consider the case when a node S sends a message m to node R via $p(S, R)$.

(Initialization) Note that the forward route information $\beta, \alpha_1, \alpha_2, \dots, \alpha_N, \gamma_1, \gamma_2, \gamma_3$, and γ_4 are calculated in the same way as in the previous section.

From the public keys, S generates the combined route information $(\beta, \theta, B_0, B_1, \dots, B_{N-1}, \gamma_1, \gamma_2, \gamma_3, \gamma_4, \lambda_1, \lambda_2)$, where $B_i = (\alpha_{i+1}, \delta_i)(0 \leq i \leq N-1)$ as follows.

S chooses $r \overset{R}{\leftarrow} Z_p^*$ (note that the random numbers must be different from those for forward route information) and sets $\theta \leftarrow g^r$,
$\delta_0 \leftarrow y_0^{\perp_0 \cdot r} \cdot y_1^r$, and
$\delta_i \leftarrow (y_1 \cdot y_2 \cdot \dots \cdot y_{i-1})^r \cdot y_i^{(e_i+1)\cdot r} \cdot y_{i+1}^r (1 \leq i \leq n-1)$.

$B_n, B_{n+1} \dots, B_{N-1}$ are dummy values.

λ_1, λ_2 are used to carry the message content of the reply. S generates $r' \overset{R}{\leftarrow} Z_p^*$ and sets
$\lambda_1 \leftarrow y_0^{r'}$, and
$\lambda_2 \leftarrow g^{r'}$.

S permutes $B_i(0 \leq i \leq N-1)$ randomly. The message M is $(\beta, \theta, B_0, B_1, \dots, B_{N-1}, \gamma_1, \gamma_2, \gamma_3, \gamma_4, \lambda_1, \lambda_2)$. v_0 sends M to e_1.

(Decryption) When a message arrives at node v_j, v_j searches for a block $B_i = (\alpha_{i+1}, \delta_i)(0 \leq i \leq N-1)$ that satisfies $\alpha_{i+1} = \beta^{e_{j+1} \cdot x_j}$ for some e_{j+1} or $\alpha_{i+1} = \beta^{\perp_{j+1} \cdot x_j}$. Let $\alpha_k = (\alpha_{k+1}, \delta_k)$ be the block that satisfies the above condition.

If the latter condition is satisfied, v_j is the destination of this message, Then v_j obtains the message content m by calculating $\gamma_1/{\gamma_2}^{x_j}$.

If the former condition is satisfied, e_{j+1} is the edge to which v_j must relay this message.

If v_j wants to send a reply message to S, the procedure for initialization on return should be employed.

Otherwise, v_j decrypts every block $B_i = (\alpha_{i+1}, \delta_i)(i \neq k)$ by calculating
$\bar{\alpha_{i+1}} \leftarrow \alpha_{i+1}/\beta^{x_j}$ and
$\bar{\delta_i} \leftarrow \delta_i/\theta^{x_j}$.

v_j calculates $\bar{\delta_k} \leftarrow \delta_k/\theta^{2\cdot x_j}$, which is the extra decryption. α_{k+1} is no longer necessary, thus v_j sets $\bar{\alpha_{k+1}}$ as a random dummy value.

For the elements $\gamma_i(i = 1, 2, 3, 4)$ and $\lambda_i(i = 1, 2)$ that carry the message content, v_j calculates
$\bar{\gamma_1} \leftarrow \gamma_1/\gamma_2^{x_j}$,
$\bar{\gamma_3} \leftarrow \gamma_3/\gamma_4^{x_j}$, and
$\bar{\lambda_1} \leftarrow \lambda_1 \cdot \lambda_2^{x_j}$.

(Re-randomization) Let $(\beta, \theta, \bar{B_0}, \bar{B_1}, \dots, \bar{B_{N-1}}, \bar{\gamma_1}, \gamma_2, \bar{\gamma_3}, \gamma_4, \bar{\lambda_1}, \lambda_2)$, where $\bar{B_i} = (\bar{\alpha_{i+1}}, \bar{\delta_i})$ is the message after the above decryption has been performed. v_i re-randomizes the message with the following procedure.

v_j selects $r_0, r_1, r_2, r_3, r_4 \overset{R}{\leftarrow} Z_p^*$ and calculates
$\widetilde{\alpha_i} \leftarrow \bar{\alpha_i}^{r_0}(1 \leq i \leq N)$,
$\widetilde{\beta} \leftarrow \beta^{r_0}$,
$\widetilde{\delta_i} \leftarrow \bar{\delta_i}^{r_1}(0 \leq i \leq N-1)$,
$\widetilde{\theta} \leftarrow \theta^{r_1}$,
$\widetilde{\gamma_1} \leftarrow \bar{\gamma_1} \cdot \bar{\gamma_3}^{r_2}$,
$\widetilde{\gamma_2} \leftarrow \gamma_2 \cdot \gamma_4^{r_2}$,
$\widetilde{\gamma_3} \leftarrow \bar{\gamma_3}^{r_3}$,
$\widetilde{\gamma_4} \leftarrow \gamma_4^{r_3}$,
$\widetilde{\lambda_1} \leftarrow \bar{\lambda_1}^{r_4}$, and
$\widetilde{\lambda_2} \leftarrow \lambda_2^{r_4}$.

The message after re-randomization is
$(\widetilde{\beta}, \widetilde{\theta}, \widetilde{B_0}, \widetilde{B_1}, \dots, \widetilde{B_{N-1}}, \widetilde{\gamma_1}, \widetilde{\gamma_2}, \widetilde{\gamma_3}, \widetilde{\gamma_4}, \widetilde{\lambda_1}, \widetilde{\lambda_2})$, where
$\widetilde{B_i} = (\widetilde{\alpha_{i+1}}, \widetilde{\delta_i})(0 \leq i \leq N-1)$.

(Permutation) After the re-randomization, v_j permutes the blocks $B_i(0 \leq i \leq N-1)$ at random. After the permutation, v_j sends the message to e_{j+1}.

(Initialization on return) When v_j decides to send a reply message, the forward route information is no longer used thus it is discarded. The following procedure is executed to send a reply message to S.

$\delta_k \leftarrow \delta_k/\theta^{x_j}$ to decrypt the extra encryption.

Re-randomize the return route information by choosing $r \overset{R}{\leftarrow} Z_p^*$ and setting
$\widetilde{\delta_i} \leftarrow \delta_i^r(0 \leq i \leq N-1)$ and
$\widetilde{\theta} \leftarrow \theta^r$.

In order to append the reply message content m', v_j executes the following. v_j chooses $r_0, r_1 \overset{R}{\leftarrow} Z_p^*$ and calculates
$\widetilde{\lambda_1} \leftarrow m' \cdot \lambda_1^{r_0}$,
$\widetilde{\lambda_2} \leftarrow \lambda_2^{r_0}$,
$\widetilde{\lambda_3} \leftarrow \lambda_1^{r_1}$, and
$\widetilde{\lambda_4} \leftarrow \lambda_2^{r_1}$,

Now the reply message is $(\widetilde{\theta}, \widetilde{\delta_0}, \widetilde{\delta_1}, \dots, \widetilde{\delta_{N-1}}, \widetilde{\lambda_1}, \widetilde{\lambda_2}, \widetilde{\lambda_3}, \widetilde{\lambda_4})$

v_j permutes $\widetilde{\delta_i}(0 \leq i \leq N-1)$ at random. v_j sends the reply message to e_j, from which the relaying message arrived.

When v_j receives a reply message $(\theta, \delta_0, \delta_1, \dots, \delta_{N-1}, \lambda_1, \lambda_2, \lambda_3, \lambda_4)$, it executes the following procedure.

(Decryption on return) v_j searches for an element $\delta_i(0 \leq i \leq N-1)$ such that $\delta_i = \theta^{x_j \cdot e_j}$ for some e_j or $\delta_i = \theta^{x_j \cdot \perp_j}$. Let δ_k be the element that satisfies the above condition.

If the latter condition is satisfied, v_j is the destination of the reply message. v_j then decrypts the message m' by $\lambda_1/\lambda_2^{x_j}$.

If the former condition is satisfied, e_j is the edge to which v_j must relay this reply message. Then v_j decrypts every element by calculating
$\bar{\delta_i} \leftarrow \delta_i \cdot \theta^{x_j}$.

For the message content, v_j decrypts the elements by cal-

culating
$\bar{\lambda}_1 \leftarrow \lambda_1/\lambda_2^{x_j}$, and
$\bar{\lambda}_3 \leftarrow \lambda_3/\lambda_4^{x_j}$.

(Re-randomization on return) For the message $(\theta, \bar{\delta}_0, \bar{\delta}_1, \ldots, \bar{\delta}_{N-1}, \bar{\lambda}_1, \lambda_2, \bar{\lambda}_3, \lambda_4)$ after the above decryption, v_j executes the following re-randomization.

v_j selects $r \overset{R}{\leftarrow} Z_p^*$ and calculates
$\widetilde{\delta_i} \leftarrow \bar{\delta_i}^r (1 \leq i \leq N-1)$, and
$\widetilde{\theta} \leftarrow \theta^r$.

v_j then selects $r_0, r_1 \overset{R}{\leftarrow} Z_p^*$ and re-randomize them by
$\widetilde{\lambda_1} \leftarrow \bar{\lambda}_1 \cdot \bar{\lambda_3}^{r_0}$,
$\widetilde{\lambda_2} \leftarrow \lambda_2 \cdot \lambda_4^{r_0}$,
$\widetilde{\lambda_3} \leftarrow \bar{\lambda_3}^{r_1}$, and
$\widetilde{\lambda_4} \leftarrow \lambda_4^{r_1}$.

$(\widetilde{\theta}, \widetilde{\delta_0}, \widetilde{\delta_1}, \ldots, \widetilde{\delta_{N-1}}, \widetilde{\lambda_1}, \widetilde{\lambda_2}, \widetilde{\lambda_3}, \widetilde{\lambda_4})$ is the randomized message.

(Permutation on return) After the re-randomization, v_j permutes $\widetilde{\delta_i}$ at random.

6. PROOF OF SECURITY

This section provides proof of the security of the proposed scheme.

We recall the decisional Diffie-Hellman assumption on which the security of our routing scheme is based.

Definition 1. Let q be a large prime. Let $\mathcal{G}$ be a cyclic group of order q and g be a generator of $\mathcal{G}$. The decisional Diffie-Hellman (DDH) problem is defined as follows.

Define $\mathcal{D}$ and $\mathcal{R}$ as follows:
$\mathcal{D} = \{(g, g^x, g^y, g^{xy}) \in \mathcal{G}^4 | x, y \in Z_q\}$
$\mathcal{R} = \{(g, g^x, g^y, g^z) \in \mathcal{G}^4 | x, y, z \in Z_q\}$

Adversary $\mathcal{A}$ outputs 0 or 1 on input $(g, A, B, R) \in \mathcal{G}^4$. $\mathcal{A}$'s advantage $Adv_{DDH}(\mathcal{A})$ is defined as
$Adv_{DDH}(\mathcal{A}) = |Prob[\mathcal{A}(g, A, B, R) = 1 | (g, A, B, R) \in \mathcal{D}] - Prob[\mathcal{A}(g, A, B, R) = 1 | (g, A, B, R) \in \mathcal{R}]|$.

Adversary $\mathcal{A}$ (t, ϵ)-breaks the DDH problem if $\mathcal{A}$ runs in time at most t and $Adv_{DDH}(\mathcal{A})$ is at least ϵ. The (t, ϵ)-DDH assumption holds if no adversary $\mathcal{A}$ (t, ϵ)-breaks the DDH problem. ■

Theorem 1. The proposed route information scheme is secure under the DDH assumption if no node is corrupted.

(Sketch of proof) Since $\mathcal{A}$ corrupts no node, it cannot rewrite messages using private keys for some nodes. Assume that there is an adversary $\mathcal{A}$ that can break unlinkability. Using $\mathcal{A}$ we can obtain an adversary $\mathcal{B}$ that solves the DDH problem. For a given instance $(g, A, B, R) \in \mathcal{G}^4$ of the DDH problem, $\mathcal{B}$ randomly selects a node v_k and sets B as the public key of v_k. $\mathcal{B}$ generates private key x_j for $v_j (j \neq k)$ and sets $y_j \leftarrow g^{x_j}$.

Now consider relation (1): M_1 at e_1 and M_2 at e_2 are the same message. The proof for the other cases can also be shown similarly.

If v_k is not between e_1 and e_2, $\mathcal{B}$ terminates. Otherwise, $\mathcal{B}$ generates message M seen at the incoming edge of v_k as follows. $\mathcal{B}$ randomly selects $R_0, R_1, R_2, R_3, R_4 \overset{R}{\leftarrow} Z_p^*$ and $M \leftarrow (\beta, \theta, B_0, B_1, \ldots, B_{N-1}, \gamma_1, \gamma_2, \gamma_3, \gamma_4, \lambda_1, \lambda_2)$, where $B_i = (\alpha_{i+1}, \delta_i)$,
$\beta \leftarrow A^{R_0}$,
$\alpha_i \leftarrow d(\text{random value}), (1 \leq i \leq k-1)$
$\alpha_k \leftarrow R^{R_0 \cdot e_{k+1}}$,
$\alpha_i \leftarrow R^{R_0} \cdot A^{x_{k+1} \cdot R_0} \cdot \ldots \cdot A^{x_{i-1} \cdot R_0} \cdot A^{x_i \cdot e_{i+1} \cdot R_0} (k+1 \leq i \leq n-1))$,
$\alpha_n \leftarrow R^{R_0} \cdot A^{x_{k+1} \cdot R_0} \cdot \ldots \cdot A^{x_{n-1} \cdot R_0} \cdot A^{x_n \cdot \perp_{n+1} \cdot R_0}$,
$\theta \leftarrow A^{R_1}$,
$\delta_0 \leftarrow A^{R_1 \cdot x_0 \cdot \perp_0} / (A^{x_1 \cdot R_1} \cdot A^{x_2 \cdot R_1} \cdot \ldots \cdot A^{x_{k-1} \cdot R_1})$,
$\delta_i \leftarrow A^{R_1 \cdot x_i \cdot e_i} / (A^{x_{i+1} \cdot R_1} \cdot A^{x_{i+2} \cdot R_1} \cdot \ldots \cdot A^{x_{k-1} \cdot R_1}) (1 \leq i \leq k-2)$,
$\delta_{k-1} \leftarrow A^{R_1 \cdot x_{k-1} \cdot e_{k-1}} \cdot R^{R_1}$
$\delta_k \leftarrow R^{R_1 \cdot (e_k + 1)} \cdot A^{R_1 \cdot x_{k+1}}$,
$\delta_i \leftarrow R^{R_1} \cdot A^{R_1 \cdot x_{k+1}} \cdot \ldots \cdot A^{R_1 \cdot x_{i-1}} \cdot A^{R_1 \cdot x_i \cdot (e_i + 1)} \cdot A^{R_1 \cdot x_{i+1}} (k+1 \leq i \leq n-1)$,
$\gamma_1 \leftarrow m \cdot R^{R_2} \cdot A^{x_{k+1} \cdot R_2} \cdot \ldots \cdot A^{x_n \cdot R_2}$,
$\gamma_2 \leftarrow A^{R_2}$,
$\gamma_3 \leftarrow R^{R_3} \cdot A^{x_{k+1} \cdot R_3} \cdot \ldots \cdot A^{x_n \cdot R_3}$,
$\gamma_4 \leftarrow A^{R_3}$,
$\lambda_1 \leftarrow A^{x_0 \cdot R_4} \cdot A^{x_1 \cdot R_4} \cdot \ldots \cdot A^{x_{k-1} \cdot R_4}$, and
$\lambda_2 \leftarrow A^{R_4}$.

Note that $\mathcal{B}$ knows the values R_0, R_1, R_2, R_3, R_4 and is able to generate a message observed at any node before v_k on the route that is consistent with this message.

$\mathcal{B}$ generates the message after decryption at v_k $(\beta, \theta, \bar{B}_0, \bar{B}_1, \ldots, \bar{B}_{N-1}, \bar{\gamma}_1, \gamma_2, \bar{\gamma}_3, \gamma_4, \bar{\lambda}_1, \lambda_2)$, where $\bar{B}_i = (\bar{\alpha}_{i+1}, \bar{\delta}_i)$ as follows.
$\bar{\alpha}_k \leftarrow d(\text{random value})$,
$\bar{\alpha}_i \leftarrow A^{x_{k+1} \cdot R_0} \cdot \ldots \cdot A^{x_{i-1} \cdot R_0} \cdot A^{x_i \cdot e_{i+1} \cdot R_0} (k+1 \leq i \leq n-1)$,
$\bar{\alpha}_n \leftarrow A^{x_{k+1} \cdot R_0} \cdot \ldots \cdot A^{x_{n-1} \cdot R_0} \cdot A^{x_n \cdot \perp_{n+1} \cdot R_0}$,
$\bar{\delta}_0 \leftarrow A^{R_1 \cdot x_0 \cdot \perp_0} / (A^{x_1 \cdot R_1} \cdot A^{x_2 \cdot R_1} \cdot \ldots \cdot A^{x_{k-1} \cdot R_1} \cdot R^{R_1})$,
$\bar{\delta}_i \leftarrow A^{R_1 \cdot x_i \cdot e_i} / (A^{x_{i+1} \cdot R_1} \cdot A^{x_{i+2} \cdot R_1} \cdot \ldots \cdot A^{x_{k-1} \cdot R_1} \cdot R^{R_1}) (1 \leq i \leq k-2)$,
$\bar{\delta}_{k-1} \leftarrow A^{R_1 \cdot x_{k-1} \cdot e_{k-1}} / R^{R_1}$,
$\bar{\delta}_k \leftarrow R^{R_1 \cdot e_k} \cdot A^{R_1 \cdot x_{k+1}}$,
$\bar{\delta}_i \leftarrow A^{R_1 \cdot x_{k+1}} \cdot \ldots \cdot A^{R_1 \cdot x_{i-1}} \cdot A^{R_1 \cdot x_i \cdot (e_i + 1)} A^{R_1 \cdot x_{i+1}} (k+1 \leq i \leq n-1)$,
$\bar{\gamma}_1 \leftarrow m \cdot A^{x_{k+1} \cdot R_2} \cdot \ldots \cdot A^{x_n \cdot R_2}$,
$\bar{\gamma}_3 \leftarrow A^{x_{k+1} \cdot R_3} \cdot \ldots \cdot A^{x_n \cdot R_3}$, and
$\bar{\lambda}_1 \leftarrow A^{x_0 \cdot R_4} \cdot A^{x_1 \cdot R_4} \cdot \ldots \cdot A^{x_{k-1} \cdot R_4} \cdot R^{R_4}$.

$\mathcal{B}$ then re-randomizes and continues relaying this message. The procedure for relaying the return message can be performed similarly.

If $(g, A, B, R) \in \mathcal{D}$, then the simulation is perfect, that is, the route information is the real information. Thus $\mathcal{A}$ can detect whether $M_1 = M_2$ with non-negligible advantage ϵ. If $(g, A, B, R) \in \mathcal{R}$, then these blocks are random blocks, and $\mathcal{A}$'s advantage is 0.

Thus, if the proposed return route information scheme is not secure, the DDH problem can be solved with non-negligible probability. ■

If two neighboring nodes v_j and v_{j+1} are corrupted, the following attack is possible. When v_{j+1} receives message M from v_j and detects $\alpha_{i+1} = \beta^{e_{j+2} \cdot x_{j+1}}$ from $B_i = (\alpha_{i+1}, \delta_i)$, it knows that $\delta_i = \theta^{x_{j+1}} \cdot \theta^{x_j \cdot e_j}$. Thus, x_{j+1} replaces δ_i with $\delta_i' = \theta^{x_{j+1}} \cdot \theta^{x_j \cdot e_j'}$. and continues the decryption procedure. When v_j decrypts e_j', it can detect that this message is a reply to M. Thus Assumption 2 is introduced for corruption.

7. SYMMETRIC SCHEME FOR FORWARD AND REPLY MESSAGES

The above scheme is asymmetric, that is, the procedures for forward messages and reply messages differ. Thus, any node can detect whether a message is a forward message or a reply message. This section discusses a modification to prevent this type of detection. We need the additional assumption that the replying node is not corrupted.

The main idea is as follows. The ElGamal encryption is symmetric, that is, encryption and decryption can be exchanged as follows:

(Public-key and private-key) $x_i \overset{R}{\leftarrow} Z_p^*$. $y_i \leftarrow g^{x_i}$. x_i is the private key and y_i is the public key.

(Scheme 1) Encryption: $r \overset{R}{\leftarrow} Z_p^*$. $(g^r, m \cdot y_i^r)$ is the ciphertext.

Decryption: Given (X, Y), execute Y/X^{x_i} and obtain m.

(Scheme 2) Encryption: $r \overset{R}{\leftarrow} Z_p^*$. $(g^r, m/y_i^r)$ is the ciphertext.

Decryption: Given (X, Y), execute $Y \cdot X^{x_i}$ and obtain m.

When encryption is executed by "$\cdot$", the decryption can be performed by "/" and vice versa.

The procedure in the previous section performs δ/θ^{x_i} for forward messages and $\delta \cdot \theta^{x_i}$ for reply messages. We can make another scheme that uses $\delta \cdot \theta^{x_i}$ for forward messages and δ/θ^{x_i} for reply messages. The sender randomly chooses these two schemes, then it becomes impossible for a node to detect whether a given message is a forward message or a reply message. The details are as follows.

(Initialization) First, sender node S chooses a random bit $b \overset{R}{\leftarrow} \{0, 1\}$.

S chooses $r, r', r_1, r_2.r'_1 \overset{R}{\leftarrow} Z_p^*$ and generates the message $M = (b, \beta, \theta, B_0, B_1, \ldots, B_{N-1}, \gamma_1, \gamma_2, \gamma_3, \gamma_4, \lambda_1, \lambda_2)$, where $B_i = (\alpha_{i+1}, \delta_i)$ as follows.

(Case 1) when $b = 0$:
$\beta \leftarrow g^r$,
$\alpha_i \leftarrow (y_1 \cdot y_2 \cdot \ldots \cdot y_{i-1})^r \cdot y_i^{e_{i+1} \cdot r} (1 \leq i \leq n-1)$,
$\alpha_n \leftarrow (y_1 \cdot y_2 \cdot \ldots \cdot y_{n-1})^r \cdot y_n^{\perp_{n+1} \cdot r}$,
$\theta \leftarrow g^{r'}$,
$\delta_0 \leftarrow y_0^{\perp_0 \cdot r'} \cdot y_1^{r'}$,
$\delta_i \leftarrow (y_1 \cdot y_2 \cdot \ldots \cdot y_{i-1})^{r'} \cdot y_i^{(e_i+1) \cdot r'} \cdot y_{i+1}^{r'} (1 \leq i \leq n-1)$.
$\gamma_1 \leftarrow m \cdot (y_1 \cdot y_2 \cdot \ldots \cdot y_{n-1} \cdot y_n)^{r_1}$,
$\gamma_2 \leftarrow g^{r_1}$,
$\gamma_3 \leftarrow (y_1 \cdot y_2 \cdot \ldots \cdot y_{n-1} \cdot y_n)^{r_2}$,
$\gamma_4 \leftarrow g^{r_2}$,
$\lambda_1 \leftarrow y_0^{r'_1}$, and
$\lambda_2 \leftarrow g^{r'_1}$.

(case 2) when $b = 1$:
$\beta \leftarrow g^r$,
$\alpha_i \leftarrow (y_1 \cdot y_2 \cdot \ldots \cdot y_{i-1})^{-r} \cdot y_i^{e_{i+1} \cdot r} (1 \leq i \leq n-1)$,
$\alpha_n \leftarrow (y_1 \cdot y_2 \cdot \ldots \cdot y_{n-1})^{-r} \cdot y_n^{\perp_{n+1} \cdot r}$,
$\theta \leftarrow g^{r'}$,
$\delta_0 \leftarrow y_0^{\perp_0 \cdot r'} \cdot y_1^{-r'}$,
$\delta_i \leftarrow (y_1 \cdot y_2 \cdot \ldots \cdot y_{i-1})^{-r'} \cdot y_i^{(e_i-1) \cdot r'} \cdot y_{i+1}^{-r'} (1 \leq i \leq n-1)$,
$\gamma_1 \leftarrow m \cdot (y_1 \cdot y_2 \cdot \ldots \cdot y_{n-1} \cdot y_n)^{r_1}$,
$\gamma_2 \leftarrow g^{r_1}$,
$\gamma_3 \leftarrow (y_1 \cdot y_2 \cdot \ldots \cdot y_{n-1} \cdot y_n)^{r_2}$,
$\gamma_4 \leftarrow g^{r_2}$,
$\lambda_1 \leftarrow y_0^{r'_1}$, and
$\lambda_2 \leftarrow g^{r'_1}$.

$B_n, B_{n+1} \ldots, B_{N-1}$ are dummy values. S permutes $B_i (0 \leq i \leq N-1)$ randomly. v_0 sends M to e_1.

(Decryption) When a message $(b, \beta, \theta, B_0, B_1, \ldots, B_{N-1}, \gamma_1, \gamma_2, \gamma_3, \gamma_4, \lambda_1, \lambda_2)$. arrives at node v_j, v_j searches for a block $B_i = (\alpha_{i+1}, \delta_i)(0 \leq i \leq N-1)$ that satisfies $\alpha_{i+1} = \beta^{e_{j+1} \cdot x_j}$ for some e_{j+1} or $\alpha_{i+1} = \beta^{\perp_{j+1} \cdot x_j}$. Let $B_k = (\alpha_{k+1}, \delta_k)$ be the block that satisfies the above condition.

If the latter condition is satisfied, v_j is the destination of this message, Then v_j obtains the message content m by calculating $\gamma_1/{\gamma_2}^{x_j}$.

If the former condition is satisfied, e_{j+1} is the edge to which v_j must relay this message.

If v_j wants to send a reply message to S, the procedure for the initialization on return should be employed.

Otherwise, execute the following procedure.

For the elements $\gamma_i (i = 1, 2, 3, 4)$, and $\lambda_i (i = 1, 2)$, v_j calculates
$\bar{\gamma_1} \leftarrow \gamma_1/\gamma_2^{x_j}$,
$\bar{\gamma_3} \leftarrow \gamma_3/\gamma_4^{x_j}$, and
$\bar{\lambda_1} \leftarrow \lambda_1 \cdot \lambda_2^{x_j}$.

For B_i, v_j executes the following procedure according to the value of b.

(Case 1) when $b = 0$:

v_j decrypts every block $B_i = (\alpha_{i+1}, \delta_i)(i \neq k)$ by calculating
$\bar{\alpha_{i+1}} \leftarrow \alpha_{i+1}/\beta^{x_j}$ and
$\bar{\delta_i} \leftarrow \delta_i/\theta^{x_j}$.

v_j calculates
$\bar{\delta_k} \leftarrow \delta_k/\theta^{2 \cdot x_j}$,
which is the extra decryption.

(Case 2) when $b = 1$:

v_j decrypts every block $B_i = (\alpha_{i+1}, \delta_i)(i \neq k)$ by calculating
$\bar{\alpha_{i+1}} \leftarrow \alpha_{i+1} \cdot \beta^{x_j}$ and
$\bar{\delta_i} \leftarrow \delta_i \cdot \theta^{x_j}$.

v_j calculates
$\bar{\delta_k} \leftarrow \delta_k \cdot \theta^{2 \cdot x_j}$,
which is the extra decryption.

(Re-randomization) (the same for $b = 0$ and $b = 1$)

Let $(b, \beta, \theta, \bar{B_0}, \bar{B_1}, \ldots, \bar{B_{N-1}}, \bar{\gamma_1}, \gamma_2, \bar{\gamma_3}, \gamma_4, \bar{\lambda_1}, \lambda_2)$, where $\bar{B_i} = (\bar{\alpha_{i+1}}, \bar{\delta_i})$ is the message after the above decryption is performed. v_i re-randomizes the message with the following procedure.

v_j selects $r_0, r_1, r_2, r_3, r_4 \overset{R}{\leftarrow} Z_p^*$ and calculates
$\widetilde{\alpha_i} \leftarrow \bar{\alpha_i}^{r_0} (1 \leq i \leq N)$,
$\widetilde{\beta} \leftarrow \beta^{r_0}$,
$\widetilde{\delta_i} \leftarrow \bar{\delta_i}^{r_1} (0 \leq i \leq N-1)$,
$\widetilde{\theta} \leftarrow \theta^{r_1}$,
$\widetilde{\gamma_1} \leftarrow \bar{\gamma_1} \cdot \bar{\gamma_3}^{r_2}$,
$\widetilde{\gamma_2} \leftarrow \gamma_2 \cdot \gamma_4^{r_2}$,
$\widetilde{\gamma_3} \leftarrow \bar{\gamma_3}^{r_3}$,
$\widetilde{\gamma_4} \leftarrow \gamma_4^{r_3}$,
$\widetilde{\lambda_1} \leftarrow \bar{\lambda_1}^{r_4}$, and
$\widetilde{\lambda_2} \leftarrow \lambda_2^{r_4}$.

The message after re-randomization is $(b, \widetilde{\beta}, \widetilde{\theta}, \widetilde{B_0}, \widetilde{B_1}, \ldots, \widetilde{B_{N-1}}, \widetilde{\gamma_1}, \widetilde{\gamma_2}, \widetilde{\gamma_3}, \widetilde{\gamma_4}, \widetilde{\lambda_1}, \widetilde{\lambda_2})$, where $\widetilde{B_i} = (\widetilde{\alpha_{i+1}}, \widetilde{\delta_i})$.

(Permutation) After the re-randomization, v_j permutes the blocks $B_i (0 \leq i \leq N-1)$ at random. After the permutation, v_j sends the message to e_{j+1}.

(Initialization on return) When v_j decides to send a reply message m' to S, the following procedure is executed.

(Case 1) when $b = 0$: v_j sets $\bar{\alpha_{k+1}} \leftarrow \delta_k/\theta^{x_j}$.

(Case 2) when $b = 1$: v_j sets $\bar{\alpha_{k+1}} \leftarrow \delta_k \cdot \theta^{x_j}$.

The following is the common procedure for $b = 0$ and $b = 1$.

For each $B_i = (\alpha_{i+1}, \delta_i)(i \neq k)$, v_j sets $\bar{\alpha_{i+1}} \leftarrow \delta_i$.

v_j re-randomizes them by choosing $r \overset{R}{\leftarrow} Z_p^*$ and setting
$\widetilde{\beta} \leftarrow \theta^r$, and
$\widetilde{\alpha_i} \leftarrow \bar{\alpha_i}^r (0 \leq i \leq N-1)$.

The forward route information is replaced with dummy values as follows:

v_j sets $\widetilde{\delta_i} \leftarrow g_i (0 \leq i \leq N-1)$ and $\theta \leftarrow g_N$ where $g_i \overset{R}{\leftarrow} \mathcal{G} (0 \leq i \leq N)$ are randomly selected elements.

In order to append the reply message content m', v_j executes the following. v_j chooses $r_0, r_1 \overset{R}{\leftarrow} Z_p^*$ and calculates
$\widetilde{\gamma_1} \leftarrow m' \cdot \lambda_1^{r_0}$,
$\widetilde{\gamma_2} \leftarrow \lambda_2^{r_0}$,
$\widetilde{\gamma_3} \leftarrow \lambda_1^{r_1}$, and
$\widetilde{\gamma_4} \leftarrow \lambda_2^{r_1}$.

The original message is also replaced with a dummy value.
$\widetilde{\lambda_1} \leftarrow g_{N+1}$,
$\widetilde{\lambda_1} \leftarrow g_{N+2}$, where
$g_{N+1}, g_{N+2} \overset{R}{\leftarrow} \mathcal{G}$.

Now the return message is $(1-b, \widetilde{\beta}, \widetilde{\theta}, \widetilde{B_0}, \widetilde{B_1}, \ldots, \widetilde{B_{N-1}}, \widetilde{\gamma_1}, \widetilde{\gamma_2}, \widetilde{\gamma_3}, \widetilde{\gamma_4}, \widetilde{\lambda_1}, \widetilde{\lambda_2})$, where $\widetilde{B_i} = (\widetilde{\alpha_{i+1}}, \widetilde{\delta_i})(0 \leq i \leq N-1)$.

v_j permutes $\widetilde{B_i}(0 \leq i \leq N-1)$ at random. v_j sends the reply message to e_j, from which the relaying message arrived. ■

Note that the value of b is also replaced with $1-b$ to indicate whether "·" or "/" is applied. The procedure for receiving reply messages is the same as that for forward messages.

When returning a message, the forward route information is not removed but replaced with a dummy value. If a malicious node v_i replaces $\alpha_i = g^{r \cdot x_i e_i}$ with $\alpha_i' = g^{r \cdot x_i e_i'}$ and v_j does not replace it with a dummy value, v_i can decrypt e_i' when this message is returned to v_i from δ_{i-1} (which was originally α_i). Thus it is necessary for the return node v_j to honestly replace forward route information with a dummy value.

8. CONCLUSION

This paper presented a return route information encryption scheme for onion-based mix-nets. The proposed scheme is the first one that (1) allows any node to send a reply message to the original sender and (2) allows multiple reply messages. The remaining problem includes finding a way to eliminate the corruption assumptions.

9. REFERENCES

[1] J. Camenisch and A. Lysyanskaya: "A Formal Treatment of Onion Routing," Proc. of Crypto 2005, LNCS Vol. 3621, pp. 169-187 (Aug. 2005).

[2] D.L. Chaum: "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms," Communications of the ACM, Vol. 24, No. 2, pp. 84-88 (1981).

[3] R. Clayton: "Improving Onion Notation," 3rd Workshop on Privacy Enhancing Technologies LNCS Vol. 2760 pp. 81-87 (Mar. 2003).

[4] G. Danezis, R. Dingledine, and N. Mathewson: "Mixminion: Design of a Type III Anonymous Remailer Protocol," Proc. of IEEE Symposium on Security and Privacy, pp.2-15 (May 2003).

[5] G. Danezis: "Breaking Four Mix-related Schemes Based on Universal Re-encryption," Proc. of 9th Information Security Conference LNCS Vol. 4176, pp. 46-59 (Aug. 2006).

[6] G. Danezis and C. Diaz: "A Survey of Anonymous Communication Channels," Microsoft Research Technical Report MSR-TR-2008-35 (Feb. 2008).

[7] R. Dingledine, N. Mathewson, and P. Syverson: "Tor: The Second-Generation Onion Router," Proc. of 13th USENIX Security Symp. p.21 (Aug. 2004).

[8] P. Golle, M. Jakobson, A. Juels, and P. Syverson: "Universal Re-encryption for Mixnets," CT-RSA 2004, LNCS Vol. 2964, pp. 163-178 (2004).

[9] D. M. Goldschlag, M. G. Reed, and P. F. Syverson: "Onion Routing for Anonymous and Private Internet Connections," Communications of the ACM, Vol. 42, No. 2, pp.39-41 (Feb. 1999).

[10] M. Gomułkiewicz, M. Klonowski, and M. Kutyłowski; "Onions Based on Universal Re-encryption - Anonymous Communication Immune Against Repetitive Attack," WISA 2004, LNCS Vol. 3325, pp. 400-410 (2004).

[11] C. Gülcü and G. Tsudik: "Mixing E-mail with BABEL," Proc. of IEEE Symp. on Network and Distributed System Security, pp. 2-16 (Feb. 1996).

[12] M. Klonowski, M. Kutylowski, and F. Zagorski: "Anonymous Communication with On-line and Off-line Onion Encoding," Proc. of Current Trends in Theory and Practice of Informatics (SOFSEM 2005), LNCS Vol. 3381, pp. 229-238 (Jan. 2005).

[13] T. Lu, B. Fang, Y. Sun, and L. Guo: "Some Remarks on Universal Re-encryption and A Novel Practical Anonymous Tunnel," Proc. of ICCNMC, LNCS Vol. 3619, pp. 853-862 (2005).

[14] C. A. Melchor and Y. Deswarte: "From DC-Nets to pMIXes: Multiple Variants for Anonymous Communications," Proc. of 5th Symp. on Network Computing and Applications(NCA), pp.163-172 (July 2006).

[15] K. Peng, J. M. Nieto, Y. Desmedt, and E. Dawson: "Klein Bottle Routing: An Alternative to Onion Routing and Mix Network," ICISC 2006, LNCS Vol. 4296, pp. 296-309 (2006).

[16] V. Shmatikov and M.-H. Wang: "Timing Analysis in Low-Latency Mix Networks: Attacks and Defenses," ESORICS 2006, LNCS Vol. 4189, pp. 18-33 (Sep. 2006).

[17] B. Timmerman: "A Security Model for Dynamic Adaptive Traffic Masking," Proc. of 1997 workshop on New security paradigms, pp. 10 -116 (1997).

[18] B. Timmerman: "Secure Dynamic Adaptive Traffic Masking," Proc. of 1999 workshop on New security paradigms, pp. 13-24 (1999).

[19] H. Toriyama, N. Kunihiro, and K. Ohta: "Return Message-Receivable Anonymous Routing Scheme without Reveal of Sender ID," SCIS2007, 3B4-6 (Jan. 2007).