Efficiency Optimisation Of Tor Using Diffie-Hellman Chain

Kun Peng
Institute for Infocomm Research, Singapore
dr.kun.peng@gmail.com

Abstract—**Onion routing is the most common anonymous communication channel. Usually onion routing is specified through asymmetric cipher and thus is inefficient. In Tor (the second generation onion router), it is suggested to employ symmetric cipher to encrypt the packets in onion routing. Obviously, symmetric cipher is much more efficient than the asymmetric cipher employed in the original onion routing. However, whether this idea can really work depends on whether an efficient (both in computation and communication) key generation and exchange mechanism can be designed for the symmetric cipher to employ. The suggestion in Tor is simple and it is a direct employment of Diffie-Hellman handshake to generate the secret keys for the routers' symmetric cipher. In this paper we show that direct application of Diffie-Hellman handshake to implement key generation and exchange in onion routing is not efficient in communication as multiple instances of Diffie-Hellman handshake needs a lot of additional communication. Moreover, its efficiency improvement for the sender is not satisfactory. So we design a more advanced application of Diffie-Hellman key exchange technique, Diffie-Hellman chain. This new technique greatly saves a sender's cost and needs very few communication for Diffie-Hellman key exchange. With the efficiency improvement in this paper, Tor can be applied to communication networks with weaker computational capability and smaller communicational bandwidth.**

Index Terms*—*TOR; efficient key exchange; Diffie-Hellman chain

I. Introduction

Anonymous communication channel is a very useful tool in e-commerce, e-government and other cryptographic applications, which often require anonymity and privacy. In an anonymous communication channel, the messages are untraceable, so can be transmitted anonymously. A common method to implement anonymous channels is onion routing [1], [3], [4], which employs multiple nodes to route a message. A node in an onion routing communication network can send a message to any node in the network. The sender can flexibly choose any route from all the connection paths between him and the receiver. Each message is contained in a packet called an onion. In the packet, a message is encrypted layer by layer using the encryption keys of all the routers on its route and the receiver. Each layer of encryption is just like a layer of onion bulb. In onion routing, given a message packet, each router unwraps a layer of encryption by decrypting the message packet using its decryption key, finds out the identity of the next router and forwards the unwrapped message packet to the next router. Unless gaining collusion of all the routers on the routing path of his received message, the receiver cannot trace the message back to the sender, who then obtains anonymity. When a packet is routed together with a large number of other packets, onion routing prevents it from being traced, even if the whole onion network is monitored.

An obvious advantage of onion routing over other specifications of anonymous communication channel (e.g. mix network [5], [6], which sends multiple messages from a unique sender to a unique receiver through a unique path) is that each of multiple senders can send his message to any of multiple receivers and freely choose a dynamic routing path and so higher flexibility and applicability are achieved. Another advantage of onion routing will be illustrated in this paper in our new routing protocols: feasibility to get rid of costly asymmetric encryption and decryption, which are inevitable in mix networks.

The key technique in onion routing is encryption chain, in which a message is successively encrypted with multiple keys. More precisely, multiple keys form a chain and are employed one by one to encrypt a message. Not only the message, the identity of each router on its routing path is encrypted in an encryption chain using the encryption keys of all the routers before it. When an onion packet is routed, each router unwraps it by removing one layer of encryption from each encryption chain. So each router can recover the identity of the next router and forward the partially decrypted packet. In onion routing, the encryption chains are usually implemented through asymmetric cipher. Namely, the message and identities of the routers are encrypted using the routers' public keys and the routers unwrap the onion packet using their private keys. An advantage of using asymmetric cipher is that with the support of PKI or ID-based public key system no special key exchange operation is needed. As there are multiple encryption chains (one for the message and one for each router) and the there are $O(n^2)$ encryption and decryption operations (where n is the number of routers), such an implementation through asymmetric cipher is inefficient.

Tor [2] is the second generation of onion routing. It proposes a few optimisations for onion routing. A suggested optimisation in Tor is to replace asymmetric cipher with much more efficient symmetric cipher to improve efficiency of onion routing. It is a commom sense that symmetric cipher is much more efficient than asymmetric cipher. The key point in using symmetric cipher is how to distribute the session keys using public key operations, while a simple solution to the key-exchange problem in application of symmetric cipher is the

Diffie-Hellman key exchange protocol recalled in Section II-B. So it is suggested in Tor [2] to employ "Diffie-Hellman handshake" to implement key changes and generate session keys for the roueters. As the idea is only simply mentioned and not specified in details in [2], it is specified in Section IV in this paper to assess its effect. Our assessment illustrates that although improving computational efficiency of the routers the suggested efficiency improvement in Tor [2] is not very satisfactory. Firstly, it greatly increases communicational cost. Secondly, even using it the sender's computational cost is still high.

The symmetric-cipher-based key chain in Tor [2] is optimised in this paper. Firstly, we optimise the "Diffie-Hellman handshake" and reduce the number of communication rounds in Tor and obtain a simple optimisation. As it is still a direct application of Diffie-Hellman key exchange, its efficiency improvement is still not satisfactory. So Diffie-Hellman key exchange is then extended and adapted for onion routing in a more advanced way such that a sender can efficiently distribute the sysmmetric sessions keys to the routers through the onion packet. The new key exchange technique is called Diffie-Hellman chain, which chain up the Diffie-Hellman handshakes for the routers and receiver such that they are much more efficient then separate Diffie-Hellman handshakes. An efficient onion routing protocol is designed in Section V using Diffie-Hellman chain. It employs Diffie-Hellman chain and block cipher encryption chain to improve computational and communicational efficiency of Tor. The new onion routing protocol is more appliable than most onion routing implementations including Tor. Network with smaller bandwidth and lower-power routers can employ them to achieve anonymity.

II. Preliminaries

Symbol denotions and background knowledge to be used in this paper are introduced and recalled in this section.

A. Parameter Setting and Symbols

The following symbols are used in this paper.

- p and q are large primes and q is a factor of $p-1$. G is the cyclic subgroup with order q in Z_p^*. g is a generator of G.
- Encryption of m using key k is denoted as $E_k(m)$ where block cipher (e.g. AES) is employed.
- Encryption chain of m using block cipher and key $k_1, k_2, \ldots, k_i$ is denoted as $E_{k_1,k_2,\ldots,k_i}(m)$. The encryptions are performed layer by layer. k_1 is the the key used in the most outer layer; k_2 is the the key used in the second most outer layer; $\ldots$; k_i is the the key used in the most inner layer.
- In onion routing, the routers are $P_1, P_2, \ldots, P_n$ and the receiver is denoted as the last router P_{n+1}.
- The private key of P_i is x_i, which is randomly chosen from Z_q. The corresponding public keys are $y_1, y_2, \ldots, y_n$ where $y_i = g^{x_i} \bmod p$ for $i = 1, 2, \ldots, n$.

B. Diffie-Hellman Key Exchange

Symmetric ciphers like block cipher are very efficient. However, unlike asymmetric cipher they depend on key exchange protocols to distribute keys. The most common key exchange protocol is Diffie-Hellman key exchange protocol. Two parties A and B can cooperate to generate a session key as follows.

1) A randomly chooses α from Z_q and sends his key base $\mu = g^{\alpha} \bmod p$ to B.
2) B randomly chooses β from Z_q and sends his key base $\nu = g^{\beta} \bmod p$ to A.
3) A can calculate the key $k = \nu^{\alpha} \bmod p$, while B can calculate the key $k = \mu^{\beta} \bmod p$.

The famous Diffie-Hellman problem is recalled as follows.

Definition 1: (Diffie-Hellman problem) Given μ and ν, it is difficult to calculate k if the discrete logarithm problem is hard.

III. Specifying and Assessing the Suggested Efficiency Improvement in Tor

The suggestion to employ symmetric cipher in Tor [2] is quite simple. To precisely assessing its cost and comparing it with our new design of key exchange, we need to specify it in details. For simplicity of description, our specification focuses on efficiency improvement through symmetric cipher as it is the focus of this paper, while the other optimisations of onion routing in Tor are ignored. The suggested efficiency improvement in Tor is specified in details as follows where a message m is sent by a sender through n routers $P_1, P_2, \ldots, P_n$ to a receiver P_{n+1}.

1) For the receiver and each router P_i where $1 \leq i \leq n+1$, the sender randomly chooses an integer s_i from Z_q and calculates $\hat{k}_i = g^{s_i} \bmod p$.
2) The sender sends $\hat{k}_1$ to P_1, which returns $\hat{k}'_1 = g^{s'_1} \bmod p$ where s'_1 is randomly chosen from Z_q. Both the sender and P_1 obtains their session key $k_1 = g^{s_1 s'_1} \bmod p$.
3) The sender sends $E_{k_1}(P_2)$ and $E_{k_1}(\hat{k}_2)$ to P_1, who decrypts the two ciphertexts using his session key and then sends $\hat{k}_2$ and $\hat{k}'_1$ to P_2.
4) P_2 randomly chooses s'_2 from Z_q and obtains his session key with the sender $k_2 = \hat{k}_2^{s'_2} = g^{s_2 s'_2}$ and his session key with P_1, $K_{1,2} =$. He sends E
5) The sender encrypts the message m, the key base list $g^{s_1}, g^{s_2}, \ldots, g^{s_{n+1}}$ and the route list $p_1, p_2, \ldots, p_{n+1}$ as follows.
   a) He calculates $e = E_{k_1,k_2,\ldots,k_{n+1}}(m)$.
   b) He calculates $K_i = E_{k_1,k_2,\ldots,k_{i-1}}(g^{s_i})$ for $i = 1, 2, \ldots, n+1$.
   c) He calculates $p_i = E_{k_1,k_2,\ldots,k_i}(P_{i+1})$ for $i = 1, 2, \ldots, n+1$ where $P_{n+2} = P_{n+1}$.
   d) He sends out the initial onion

$$O_1 = (a_1, b_{1,1}, b_{1,2}, \ldots, b_{1,n+1},$$
$$c_{1,1}, c_{1,2}, \ldots, c_{1,n+1})$$
$$= (e, K_1, K_2, \ldots, K_{n+1}, p_1, p_2, \ldots, p_{n+1})$$

to P_1.

6) Each router P_i routes the onion as follows where the onion is in the form $O_i = (a_i, b_{i,1}, b_{i,2}, \ldots, b_{i,n+1}, c_{i,1}, c_{i,2}, \ldots, c_{i,n+1})$ when it is sent to P_i.
   a) P_i generates his session key $k_i = b_{i,1}^{x_i} \bmod p$.
   b) P_i uses k_i to decrypt $c_{i,j}$ for $j = 1, 2, \ldots, n+1$ and obtains $P_{i+1} = D_{k_i}(c_{i,1})$.
   c) P_i uses k_i to decrypt a_i and obtains $a_{i+1} = D_{k_i}(a_i)$.
   d) Finally, P_i sends

$$O_{i+1} = (a_{i+1}, b_{i+1,1}, b_{i+1,2}, \ldots, \\ b_{i+1,n+1}, c_{i+1,1}, c_{i+1,2}, \ldots, c_{i+1,n+1})$$

   to P_{i+1} where $b_{i+1,j} = D_{k_i}(b_{i,j+1})$ and $c_{i+1,j} = D_{k_i}(c_{i,j+1})$ for $j = 1, 2, \ldots, n$ and $b_{i+1,n+1}$ and $c_{i+1,n+1}$ are two random integers in the ciphertext space of the employed symmetric encryption algorithm.

7) At last, P_{n+1} receives

$$O_{n+1} = (a_{n+1}, b_{n+1,1}, b_{n+1,2}, \ldots, \\ b_{n+1,n+1}, c_{n+1,1}, c_{n+1,2}, \ldots, c_{n+1,n+1})$$

   and operates as follows.
   a) P_{n+1} generates his session key $k_{n+1} = b_{n+1,1}^{x_{n+1}} \bmod p$.
   b) P_{n+1} uses k_{n+1} to decrypt $c_{n+1,j}$ and obtains $P_{n+1} = D_{k_{n+1}}(c_{n+1,1})$.
   c) P_{n+1} knows that itself is the receiver as P_{n+1} is its own identity.
   d) P_{n+1} uses k_{n+1} to decrypt a_{n+1} and obtains $m = D_{k_{n+1}}(a_{n+1})$.

IV. A Simple Optimisation of Tor and its Drawback: Simpler but still Direct Application of Diffie-Hellman Key Exchange

A simple optimisation of Tor is proposed in this section. Like in the original onion routing (and many other cryptographia protocols), it assumes that every router and the receiver have discrete-logarithm-based public key encryption algorithms (e.g. ElGamal encryption) and already set up their public keys so that half of the preparation work in Diffie-Hellman key exchange can be saved. Moreover, multiple rounds of communication between each pair of participants are combined to improve communication efficiency. It still employ Diffie-Hellman handshakes in the staightforward way and is described as follows.

1) For the receiver and each router P_i where $1 \leq i \leq n+1$, the sender randomly chooses an integer s_i from Z_q and generates a session key $k_i = y_i^{s_i}$.
2) The sender encrypts the message m, the key base list $g^{s_1}, g^{s_2}, \ldots, g^{s_{n+1}}$ and the route list $p_1, p_2, \ldots, p_{n+1}$ as follows.
   a) He calculates $e = E_{k_1,k_2,\ldots,k_{n+1}}(m)$.
   b) He calculates $K_i = E_{k_1,k_2,\ldots,k_{i-1}}(g^{s_i})$ for $i = 1, 2, \ldots, n+1$.
   c) He calculates $p_i = E_{k_1,k_2,\ldots,k_i}(P_{i+1})$ for $i = 1, 2, \ldots, n+1$ where $P_{n+2} = P_{n+1}$.
   d) He sends out the initial onion

$$O_1 = (a_1, b_{1,1}, b_{1,2}, \ldots, b_{1,n+1}, \\ c_{1,1}, c_{1,2}, \ldots, c_{1,n+1}) \\ = (e, K_1, K_2, \ldots, K_{n+1}, p_1, p_2, \ldots, p_{n+1})$$

   to P_1.

3) Each router P_i routes the onion as follows where the onion is in the form $O_i = (a_i, b_{i,1}, b_{i,2}, \ldots, b_{i,n+1}, c_{i,1}, c_{i,2}, \ldots, c_{i,n+1})$ when it is sent to P_i.
   a) P_i generates his session key $k_i = b_{i,1}^{x_i} \bmod p$.
   b) P_i uses k_i to decrypt $c_{i,j}$ for $j = 1, 2, \ldots, n+1$ and obtains $P_{i+1} = D_{k_i}(c_{i,1})$.
   c) P_i uses k_i to decrypt a_i and obtains $a_{i+1} = D_{k_i}(a_i)$.
   d) Finally, P_i sends

$$O_{i+1} = (a_{i+1}, b_{i+1,1}, b_{i+1,2}, \ldots, \\ b_{i+1,n+1}, c_{i+1,1}, c_{i+1,2}, \ldots, c_{i+1,n+1})$$

   to P_{i+1} where $b_{i+1,j} = D_{k_i}(b_{i,j+1})$ and $c_{i+1,j} = D_{k_i}(c_{i,j+1})$ for $j = 1, 2, \ldots, n$ and $b_{i+1,n+1}$ and $c_{i+1,n+1}$ are two random integers in the ciphertext space of the employed symmetric encryption algorithm.

4) At last, P_{n+1} receives

$$O_{n+1} = (a_{n+1}, b_{n+1,1}, b_{n+1,2}, \ldots, \\ b_{n+1,n+1}, c_{n+1,1}, c_{n+1,2}, \ldots, c_{n+1,n+1})$$

   and operates as follows.
   a) P_{n+1} generates his session key $k_{n+1} = b_{n+1,1}^{x_{n+1}} \bmod p$.
   b) P_{n+1} uses k_{n+1} to decrypt $c_{n+1,j}$ and obtains $P_{n+1} = D_{k_{n+1}}(c_{n+1,1})$.
   c) P_{n+1} knows that itself is the receiver as P_{n+1} is its own identity.
   d) P_{n+1} uses k_{n+1} to decrypt a_{n+1} and obtains $m = D_{k_{n+1}}(a_{n+1})$.

This modified Tor protocol only employs symmetric cipher in encryption and decryption operations. The only public key operations in it are $n+1$ instances of Diffie-Hellman key exchange. So although more encryption and decryption operations are needed than in traditional onion routing, it is still more efficient in computation. However, it is less efficient in communication than traditional onion routing as its onion packet contains additional encrypted key bases $b_{i,1}, b_{i,2}, \ldots, b_{i,n+1}$. So its advantage in efficiency is not obvious. Therefore, it is only a prototype, while our final proposal is based on it but has higher requirements on efficiency: only using symmetric cipher in encryption and decryption while in comparison with traditional onion routing

- very little additional communication (e.g. one more integer) is needed;
- no more additional encryption or decryption operation is needed.

V. A NEW AND MORE ADVANCED TECHNIQUE: DIFFIE-HELLMAN CHAIN

The simple optimisation protocol in Section IV has demonstrated that direct application of Diffie-Hellman key exchange to onion routing (including original onion routing and Tor) cannot achieve satisfiactory advantage in efficiency. To reduce additional communication and encryption and decryption operations, a novel technique, Diffie-Hellman chain, is designed. The Diffie-Hellman key bases for all the routers and the receiver are sealed in the Diffie-Hellman chain, which appears in each onion packet in the form of a single integer. For each router, to generate his session key, he needs his private key and a key base initially sealed in the Diffie-Hellman chain by the sender and then recovered by cooperation of all the previous routers in the course of routing. As only one single integer is needed in each onion packet to represent the Diffie-Hellman chain and commit to all the Diffie-Hellman key bases, a very small amount of additional communication is employed and no more encryption (decryption) operation is needed in comparison with traditional onion routing.

A new onion routing protocol, called compressed onion routing, is proposed. In compressed onion routing, a packet (onion) consists of three parts: message, route list and key base. Route list contains the identities of all the nodes on the route. Key base is the base to generate the session keys (symmetric keys) distributed to the nodes. The message part in compressed onion routing is similar to that in most onion routing schemes. The message is encrypted in a encryption chain using the sessions keys of all the nodes. The readers only need to note that efficient block cipher is employed in the encryption chain. In compressed onion routing, the route list is the same as in other onion routing schemes. It consists of all the routers' identities. One encryption chain is used to seal each router's identity using the session keys of the all the routers before it. The readers only need to note that efficient block cipher is employed in the encryption chains for the route list.

The most important novel technique is generation and update of the key base, which enables key exchange. Each router builds his session key on the base of the key base using his private key and update the key base for the next router. The key generation function is similar to Diffie-Hellman key generation, but we do not employ separate Diffie-Hellman key exchange protocols to distribute the session keys to the routers. Instead the key base updating mechanism actually generates a key base chain and so all the session keys and their generation functions are linked in a chain structure. So the key exchange technique is called Diffie-Hellman chain. After obtaining his session key, each router can extracts the identity of the next router from the route list using his session key, removes one layer of encryption from the message using his session key and then forwards the onion to the next router. The Diffie-Hellman chain only needs the bandwidth of one integer, and thus is much more efficient than separate key distribution in communication. Novelty of the new compressed onion routing protocol is that distribution of the sessions keys and encryption of the routers' identities are compressed such that fewer computationally-costly public key operations and communicationally-costly encryption chains are needed.

Suppose a message m is sent by a sender through n routers $P_1, P_2, \ldots, P_n$ to the receiver P_{n+1}. Firstly, the sender generates the session keys $k_1, k_2, \ldots, k_{n+1}$ respectively for $P_1, P_2, \ldots, P_{n+1}$ as follows.

1) The sender randomly chooses an integer s_1 from Z_q.
2) The sender calculates P_1's session key $k_1 = y_1^{s_1} \bmod p$.
3) The sender calculates $s_2 = s_1 + k_1 \bmod q$.
4) The sender calculates P_2's session key $k_2 = y_2^{s_2} \bmod p$.
5) ......
6) ......
7) The sender calculates $s_{n+1} = s_n + k_n \bmod q$.
8) The sender calculates P_{n+1}'s session key $k_{n+1} = y_{n+1}^{s_{n+1}} \bmod p$.

Generally speaking, for $i = 1, 2, \ldots, n+1$, the sender

1) if $i > 1$ then calculates $s_i = s_{i-1} + k_{i-1} \bmod q$ as his secret seed in the Diffie-Hellman chain for generation of k_i
2) calculates $k_i = y_i^{s_i}$

where s_1 is randomly chosen from Z_q. In summary, the sender uses the sum of the previous node's session key and his secret seed in the Diffie-Hellman generation of the previous node's session key as his secret seed to generate a node's Diffie-Hellman session key. The other secret seed to generate the node's session key is the node's private key.

The route list consists of $p_1, p_2, \ldots, p_{n+1}$ where $p_i = E_{k_1,k_2,\ldots,k_i}(P_{i+1})$ and $P_{n+2} = P_{n+1}$. The message is encrypted into $e = E_{k_1,k_2,\ldots,k_{n+1}}(m)$. The onion is in the form of $O_i = (a_i, b_i, c_{i,1}, c_{i,2}, \ldots, c_{i,n+1})$ when it reaches P_i where a_i is the encrypted message, b_i is the key base and $c_{i,1}, c_{i,2}, \ldots, c_{i,n+1}$ is the encrypted route list. Note that although the encryption chain for the next router's identity is completely decrypted and discarded by each router, the length of the encrypted route list is kept unchanged for the sake of untraceability. If an onion packet becomes shorter after each router's routing, its change in length can be observed and exploited to trace it. So we keep the length of each encrypted route list constant to maintain the size of onion packets. This can be implemented by inserting a random tag into the onion packets after they discard an encryption chain. The initial onion $O_1 = (a_1, b_1, c_{1,1}, c_{1,2}, \ldots, c_{1,n+1}) = (e, g^{s_1}, p_1, p_2, \ldots, p_{n+1})$. Note that e may actually contain multiple symmetric ciphertext blocks as the message may be long and is divided into multiple blocks when being encrypted. For convenience of description encryption of the message is still denoted as a single variable and the readers should be aware that it is the encryption of the whole message and may contain multiple blocks.

P_1 receives $O_1 = (a_1, b_1, c_{1,1}, c_{1,2}, \dots, c_{1,n+1})$ from the sender and then operates as follows.

1) P_1 generates his session key $k_1 = b_1^{x_1} \bmod p$.
2) P_1 uses k_1 to decrypt $c_{1,j}$ for $j = 1, 2, \dots, n+1$ and obtains $P_2 = D_{k_1}(c_{1,1})$.
3) P_1 uses k_1 to decrypt a_1 and obtains $a_2 = D_{k_1}(a_1)$.
4) P_1 calculates the new key base $b_2 = b_1 g^{k_1} \bmod p$.

Finally, P_1 sends $O_2 = (a_2, b_2, c_{2,1}, c_{2,2}, \dots, c_{2,n+1})$ to P_2 where $c_{2,i} = D_{k_1}(c_{1,i+1})$ for $i = 1, 2, \dots, n$ and $c_{2,n+1}$ is a random integer in the ciphertext space of the employed block encryption algorithm.

More generally, for $i = 1, 2, \dots, n$ each P_i receives $O_i = (a_i, b_i, c_{i,1}, c_{i,2}, \dots, c_{i,n+1})$ and operates as follows.

1) P_i generates his session key $k_i = b_i^{x_i} \bmod p$.
2) P_i uses k_i to decrypt $c_{i,j}$ for $j = 1, 2, \dots, n+1$ and obtains $P_{i+1} = D_{k_i}(c_{i,1})$.
3) P_i uses k_i to decrypt a_i and obtains $a_{i+1} = D_{k_i}(a_i)$.
4) P_i calculates the new key base $b_{i+1} = b_i g^{k_i} \bmod p$.

Finally, P_i sends $O_{i+1} = (a_{i+1}, b_{i+1}, c_{i+1,1}, c_{i+1,2}, \dots, c_{i+1,n+1})$ to P_{i+1} where $c_{i+1,j} = D_{k_i}(c_{i,j+1})$ for $j = 1, 2, \dots, n$ and $c_{i+1,n+1}$ is a random integer in the ciphertext space of the employed symmetric encryption algorithm.

At last, P_{n+1} receives $O_{n+1} = (a_{n+1}, b_{n+1}, c_{n+1,1}, c_{n+1,2}, \dots, c_{n+1,n+1})$ and operates as follows.

1) P_{n+1} generates his session key $k_{n+1} = b_{n+1}^{x_{n+1}} \bmod p$.
2) P_{n+1} uses k_{n+1} to decrypt $c_{n+1,j}$ and obtains $P_{n+1} = D_{k_{n+1}}(c_{n+1,1})$.
3) P_{n+1} knows that itself is the receiver as P_{n+1} is its own identity.
4) P_{n+1} uses k_{n+1} to decrypt a_{n+1} and obtains $m = D_{k_{n+1}}(a_{n+1})$.

VI. Analysis and Comparison

Security of the compressed onion routing scheme depends on hardness of Diffie-Hellman problem as its key exchange mechanism is an extension of Diffie-Hellman key exchange. Its main trick is combining key exchange with encryption chain such that every router can obtain his session key with the help the previous router. As security of Diffie-Hellman key exchange has been formally proved and hardness of Diffie-Hellman problem is widely accepted, no further proof of security is needed except for Theorem 1, which shows that the session keys can be correctly exchanged.

Theorem 1: For $j = 1, 2, \dots, n+1$, the same session key k_i is generated, respectively by the sender as $k_i = y_i^{s_i} \bmod p$ and by P_i as $k_i = b_i^{x_i} \bmod p$.

To prove Theorem 1, a lemma has to be proved first.

Lemma 1: For $j = 1, 2, \dots, n+1$, $b_i = g^{s_i} \bmod p$.

Proof: Mathematical induction is used.

1) When $i = 1$, $b_1 = g^{s_1} \bmod p$
2) When $i = j$, suppose $b_j = g^{s_j} \bmod p$. Then a deduction can be made in next step.
3) When $i = j+1$, $b_{j+1} = b_j g^{k_j} = g^{s_j} g^{k_j} \bmod p$ as it is supposed in last step that $b_i = g^{s_i}$ when $i = j$. So

$$b_{j+1} = g^{s_j} g^{k_j} = g^{s_j + k_j} = g^{s_{j+1}} \bmod p$$

Therefore, $b_i = g^{s_i} \bmod p$ for $j = 1, 2, \dots, n+1$ as a result of mathematical induction. □

Proof of Theorem 1:
According to Lemma 1,

$$y_i^{s_i} = g^{x_i s_i} = b_i^{x_i} \bmod p$$

for $j = 1, 2, \dots, n+1$. □

Efficiency comparison between our new onion routing protocol and the existing anonymous communication channels is given in Table I, Table II, Table III and Table IV where AOR stands for asymmetric cipher based onion routing and COR stands for compressed onion routing. The first table shows the advantage of our new technique over the existing anonymous communication channels including onion routing and mix network. The last three tables show our optimisation of onion routing. It is assumed that the employed block cipher is 256-bit AES. For simplicity, it is assumed that the message is one block long, while the size of one block of the employed block cipher should be large enough for a router's identity. So all the ciphertexts are one block long in our analysis, which does not lose generality and can be extended to long message cases in a straightforward way. As for asymmetric cipher in AOR, it is supposed that ElGamal encryption, which is the most popular with onion routing, is employed. More precisely, it is assumed that the ElGamal encryption algorithm uses 1024-bit integers. Comparison in the four tables illustrates that great efficiency improvement is achieved in the two compressed onion routing protocols.

TABLE I
Comparison of the anonymous communication channels

Scheme	public key exponentiation	flexibility and applicability
Mix network	$\geq 6n + 4$	No
AOS	$2(n+1)(n+4)$	Yes
Tor	$2(2n-1)$	Yes
COR	$3(n+1)$	Yes

TABLE II
Computational Efficiency Comparison for the Sender

Scheme	public key exponentiation	block cipher encryption
AOS	$(n+1)(n+4)$	0
Tor	$n+1$	$(n+1)(1+(3n+2)/2)$
COR	$n+1$	$(n+1)(1+(n+2)/2)$

VII. Conclusion

The new onion routing scheme proposed in this paper greatly improves efficiency of onion routing by using symmetric cipher and Diffie-Hellman chain. It needs smaller packet

TABLE III
COMPUTATIONAL EFFICIENCY COMPARISON FOR A ROUTER (RECEIVER)

Scheme	average public key exponentiation	average block cipher decryption
AOS	$n+4$	0
Tor	3	$2(n+1)$
COR	2	$(n+4)/2$

TABLE IV
COMMUNICATIONAL EFFICIENCY COMPARISON

Scheme	number of bits in an onion packet	rounds
AOS	$2048(n+2)$	$n+1$
Tor	$256(n+2)$	$(n+1)(n+3)$
COR	$256(n+2)+1024$	$n+1$

size and less computation than the existing onion routing schemes including TOR.

An open question in the future work is how to further compress the size of onion packets. The route list chains occupy most room in an onion packet. Can they be compressed to further improve communication efficiency?

REFERENCES

[1] J. Camenisch and A. Mityagin. A formal treatment of onion routing. In *CRYPTO '05*, volume 3089 of *Lecture Notes in Computer Science*, pages 169–187, Berlin, 2005. Springer-Verlag.

[2] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *USENIX Security Symposium*, pages 303–320, 2004.

[3] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, STOC 1987*, pages 218–229, 1987.

[4] D. Goldschlag, M. Reed, and P. Syverson. Onion routing for anonymous and private internet connections. *Comm. of the ACM, 42(2)*, page 84–88, 1999.

[5] K. Peng, C. Boyd, and E. Dawson. Simple and efficient shuffling with provable correctness and ZK privacy. In *PKC '04*, volume 2947 of *Lecture Notes in Computer Science*, pages 439–454, Berlin, 2004. Springer-Verlag.

[6] K. Peng, C. Boyd, and E. Dawson. Simple and efficient shuffling with provable correctness and ZK privacy. In *CRYPTO '05*, volume 3089 of *Lecture Notes in Computer Science*, pages 188–204, Berlin, 2005. Springer-Verlag.