A Scheme for Route Optimization in Mobile IP without Compromising Location Privacy

Yang-hua Chu, Jun Gao and Sanjay Rao
{yhchu,jungao,sanjay}@cs.cmu.edu

May 5, 1998

Abstract

Route Optimization extensions to Mobile IP [2] are in fundamental conflict with the need to provide location privacy of mobile nodes. In this project, we propose a scheme for achieving Route Optimization without compromising location privacy. The scheme consists of an addition of two new options to IP, and processing of these options require routers to perform symmetric-key cryptography. The scheme is attractive as it facilitates incremental deployment in the Internet and allows routers to independently choose cryptographic algorithms appropriate to their requirements. We then discuss changes that need to be made to the Route Optimization protocol presented in [2] to support our scheme. We have modified the NetBSD kernel on routers to support processing of the new options we have introduced, and used the RC4 cryptographic scheme in the implementation. We evaluated the overhead associated with our scheme on a test network by measuring the increase in round trip time as well as the reduction in throughput. The increase in round trip time was between 6% and 9%, while the reduction in throughput was between 1% and 10%.

1 Introduction

The growth of mobile computing has brought into focus several security problems. One such problem is **Location Privacy**, where the mobile host wishes to carry on regular communication with other hosts in the Internet, without either the correspondent host or intermediate routers having knowledge of the current physical location of the mobile node. Location privacy is to be contrasted with **Identity Privacy**, where the destination host is not aware of the identity of the node it is communicating with. Today's Internet does not provide identity privacy even to a stationary host. While the Internet does not provide location privacy either to a stationary host, this is unimportant because the IP address of the stationary host gives away its location.

The Mobile IP protocol provides a mechanism by which a mobile node may communicate with the rest of the Internet, even when it is visiting a foreign network. The base IETF Mobile IP protocol requires that when a correspondent host (CH) talks to a mobile host (MH), the packets of CH are routed in the Internet to the home network of MH. Here, they are intercepted by the "home agent" (HA) of MH, are tunneled to a "foreign agent" (FA) in the current foreign network of MH, which then forwards the packets to MH. Clearly, this protocol does not compromise the location privacy of MH. However, the routing employed is suboptimal, in the sense that packets from CH first need to be routed to the home network of the mobile host before reaching the foreign network. In order to remedy this, researchers are working on "Route Optimization" [2]. Here the address of FA is given to CH, so that CH may directly tunnel the packets to FA. However, in this process, CH knows the current location of mobile node MH, and location privacy of MH is compromised. There is thus an inherent conflict between Route Optimization and location privacy.

In this project, we present a scheme for achieving Route Optimization without compromising location privacy. This scheme consists of addition of two new options to IP and requires support from the routers. Processing of these options require routers to perform symmetric-key cryptography. The scheme has the attractive properties that (i) the overhead on intermediate routers is small; (ii) each router can independently make its trade-offs between the need to provide enhanced security features and the need to sustain efficiency and choose appropriate cryptographic algorithms; (iii) legacy routers can be supported and incremental deployment in the internet is feasible; (iv) no special knowledge of topology is required by either the sender or the receiver; and (v) no extra state information needs to be stored in the routers. However, our scheme works only when either the sender or the receiver, (and not both) is mobile. In the case where both the sender and receiver are mobile, we believe that there is no efficient way of providing location privacy and Route Optimization at the same time. Section 2 describes our scheme in detail, elaborates on its advantages, and discusses changes that may have to be made to the Route Optimization protocol presented in [2] to support our scheme.

We have suitably modified the NetBSD kernel on routers, to support the processing of the new options. The symmetric-key cryptographic scheme that we employed is the popular RC4 scheme [5]. Since our focus in this project is to demonstrate that the processing overhead required of routers is limited, and not a complete implementation of the modified Route Optimization protocol, we did not change the end-host kernel. Rather, for testing purposes, we let the end-hosts generate packets with the new IP options using raw IP sockets. The details of our implementation, as well as a specification of the router processing required for each option is presented in Section 3.

We evaluated the overhead our scheme imposes by measuring the impact on end-to-end delay and bandwidth in a test network. Details of our evaluation may be seen in Section 4. Section 5 presents related work and Section 6 presents conclusions and directions for future work.

2 Design

In Section 2.1 we discuss the central idea of the design: how do the mobile host and the correspondent host communicate with each other using an optimal route, without the mobile host's location privacy being compromised? In Section 2.2 we discuss the modifications that need to be made to Route Optimization to support this scheme.

2.1 Scheme description

We find it convenient to describe the scheme using an example network setup as shown in Figure 1.

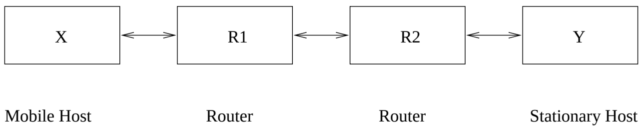

Figure 1: An example network to illustrate our scheme

We use the following notation in our explanation:

```
x       : the IP address of the mobile host X,
y       : the IP address of the stationary host Y,
ri      : the IP address of an intermediate router Ri,
Ki      : a symmetric, private key known only to router Ri,
Ki(n)   : encryption of data n using private key Ki
[a,b,c]: an IP packet containing source IP address a, destination IP
          address b, and IP header option data c.
(a||b) : concatenation of two bit strings a and b
```

The scheme tries to balance two conflicting goals; it tries to establish an efficient route from X to Y (and back) while not revealing the current location of X. In the forward stage where X sends a packet to Y, each router along the way encrypts the path it receives from the previous router, adds its own address, and

forwards it to the next router using normal Internet routing. In the reverse stage, each router first decrypts the path, which reveals the next hop router address to forward to. In addition, the scheme ensures that after decryption, each router knows only the next hop address, but nothing beyond that.
In the forward path, X transmits a packet with the encryption option set, and the following sequence of actions takes place:

1. The mobile host X chooses a nonce n, (a random value), and computes $(n \parallel x)$, a concatenation of the nonce and its IP address. X places $(n \parallel x)$ in the option and sends the packet $[x, y, n \parallel x]$, by normal Internet routing which reaches router R1.
   $X \rightarrow R1 : [x, y, n \parallel x]$

2. When R1 receives an IP packet with header option $(n \parallel x)$, it encrypts $(n \parallel x)$ with its secret key, thereby obtaining s1. Then R1 replaces $(n \parallel x)$ with $(s1 \parallel r1)$ (a concatenation of s1 and its own address r1) and forwards it by normal Internet routing, which reaches R2.
   s1 $= K1(n \parallel x)$. $R1 \rightarrow R2 : [x, y, s1 \parallel r1]$

3. Similarly, R2 computes s2, by encrypting $(s1 \parallel r1)$ and replaces $(s1 \parallel r1)$ with $(s2 \parallel r2)$ (a concatenation of s2 and its own address r2), and forwards the packet which reaches Y by normal Internet routing.
   s2 $= K2(s1 \parallel r2)$. $R2 \rightarrow Y : [x, y, s2 \parallel r2]$

Now $s2 \parallel r2$ encodes an efficient path that a packet from Y should take to reach X (which is the reverse path of a message from X to Y). Y cannot decode the path, and is unaware of the routers along the path except for the immediate first router (r2). A packet from Y is sent to X with the decryption option turned on leading to the following sequence of actions:

1. Y extracts r2 from $(s2 \parallel r2)$ and puts it in destination field of the IP address. s2 is inserted in the IP header option:
   $Y \rightarrow R2 : (y, r2, s2)$

2. R2 decrypts s2 to obtain $(s1 \parallel r1)$. It then extracts r1, the router to which it should forward the packet, and places it in the destination address and replaces the option from $(s2 \parallel r1)$ to s1.
   $s1 \parallel r1 = K2(s2)$. $R2 \rightarrow R1 : (y, r1, s1)$

3. Similarly, R1 decrypts s1 to reveal $(n \parallel x)$. It extracts x, and puts it in the destination address and replaces the option from $(s1 \parallel r1)$ to n.
   $n \parallel x = K1(s1)$. $R1 \rightarrow X : (y, x, n)$

We now need to discuss what happens if the mobile host is not the initiator of the conversation. The first message from the correspondent host reaches the mobile host via the home agent of the mobile host. On receipt of a packet without the decryption option set, the mobile host realizes that the packet from the correspondent host has taken an inefficient route. The mobile host then sends an unsolicited message to the correspondent host, with the encryption option turned on, which informs the latter the optimal route to be used.

In the case where both the correspondent host and the mobile host are mobile and need their location information protected from one another, our scheme fails to support Route Optimization. Intuitively, this occurs, because our scheme requires atleast one packet in one direction to be transmitted using an optimized route; neither communicating host can send such a packet unless it knows the current location of the other host. We believe that there exists no efficient solution for two mobile hosts to communicate with each other using an optimized route, and at the same time maintaining their location privacy. One possible solution is that every intermediate router has an explicit routing table entry for each of the mobile hosts. This solution however is fundamentally unscalable as it prevents aggregation of routes, and is only of academic interest.

2.2 Discussion of the scheme

There are a few aspects of the scheme that deserve further discussion:

- *Facilitates incremental deployment* - An advantage of this scheme is that it easily supports existing routers and facilitates incremental deployment in the Internet. Assume for example that the packet is routed along the path $\{X, R1, L1, R2, R3, Y\}$, where, L1 is a legacy router that does not support this option, and R1,R2,R3 are routers that are capable of supporting this option. Now, in the forward stage, L1 merely forwards the packet to its next hop router (R2) as per normal routing conventions, and does not alter any aspect of the packet. On the reverse direction, $R2$ notes that the packet needs to be forwarded to $R1$, and appropriately marks the destination field. Normal Internet routing carries the packet to $R1$ (through a path that possibly does not contain L1). In the forward direction, the packet reaches the destination by normal Internet forwarding, and at the same time a partial list of routers that must be traversed in the reverse direction is constructed. This partial list contains precisely those intermediate routers that participated in the encryption scheme during the forward path of the packet.

  Also, the approach can support end hosts that do not have support for this extension. If in the above example Y does not recognize the special option, then, it replies to X in the normal fashion, which takes an inefficient route (through the home-agent of X).

  Furthermore, the approach above will not lead to any special problems in the handling of ICMP error messages, as the true source of the packet has been listed in each direction.

- *Facilitates efficiency-security tradeoffs to be made by each individual router*- The symmetric encryption/decryption algorithm is deterministic and can be built efficiently in hardware. Each router can independently decide on its own algorithm, depending on the efficiency-security trade off it wishes to make. A very efficient, but possibly not secure algorithm is using a scheme such as fixed permutation of bits. On the other hand, the router could choose to adopt a computationally expensive but more secure algorithm such as Triple-DES. Our implementation uses RC4, which is reasonably secure, and computationally cheap. Further, the location privacy provided is as strong as the most secure router along the path.

- *Encryption packets need be sent only when location changes* - Packets with the encryption option turned on need to be sent by the mobile host (X) only once, and whenever X moves to a new network. This reduces the overhead associated with such a packet (apart from the overhead on routers, as we will see the encryption packets also need to be authenticated). In a sense, a packet with the encryption option turned on is transmitted only when a "binding cache update" message is to be transmitted.

- *No knowledge of Internet routing* - We had considered other possible schemes that required a host to know the route to the destination before hand, and rejected these as they violate the underlying philosophy of Internet routing. Our scheme requires no such knowledge.

- *Need for nonces*- Let us assume that X sends Y a message with the encryption option turned on every time that X moves to a new network. Now, while Y may not be able to detect the location of X, it can keep track of X's movement patterns. Our solution is that even while X is in the same location, it sends messages at random intervals with the encryption option turned on, but with a different nonce. Y would then keep updating the route to X, but would be unable to distinguish between whether the updation occurs due to a change of nonce, or a change of location of X.

2.3 Highlights of the modified Route Optimization Protocol

We have by and large tried to retain the original Route Optimization protocol. A few changes are required however to support our scheme and we present these. We have followed the Route Optimization protocol presented in [2]. We understand that the protocol has undergone changes, not all of which have been documented. We believe that some of the changes we require are in fact already supported by newer versions of the protocol.

- *Location Caching*- Route Optimization requires that the correspondent host cache the care-of-address of the mobile host, and tunnel the packets to the cached care-of-address. Our scheme will require correspondent hosts to instead cache the (encrypted) route to the mobile host and appropriately set the decryption option in the reply.

- *Foreign Agent handoff*- When a mobile host moves to a new network, the binding cache entry of a correspondent host becomes out of date. Route Optimization allows the old foreign agent to create a "forwarding pointer" to the new location of the mobile host, and allows the old foreign agent to forward packets tunneled from the correspondent host with out-of-date binding cache entries to the mobile host's new care of address. However, this optimization is not possible in our scheme, causing loss of these packets and requiring retransmission by higher layer protocols. If however, the last router on the path from the correspondent host to the mobile host is the foreign agent, then, a similar optimization is possible. In such a case, the mobile host sends a binding update message to the old foreign agent, except that this message must be sent with the encryption option turned on to prevent the old foreign agent realizing the new location of the mobile host. The foreign agent forwards the packets to the mobile host using the decryption option and the cached route.
- *Binding Cache Updates*- If a home agent intercepts a packet destined to a mobile host, then, Route Optimization requires the home agent to send a binding cache update message to the correspondent host. However, this message has to be completely eliminated as it would give away the location of the mobile host. Rather, we require that the mobile host send a binding cache update to the correspondent host (with the encryption option set), if it receives a packet from the correspondent host without the decryption option set (which indicates the packet came through the home agent). If the correspondent host does not understand the encryption message, then, a binding cache update would be triggered by the mobile host for every packet that the correspondent host transmits. This may be handled in a similar fashion to the Route Optimization protocol, by keeping track of binding updates that were sent, ensuring a minimum time interval between successive binding updates, and doubling the time interval after each update.

  Whenever a mobile host moves to a different network, it needs to send binding cache updates to the correspondent host informing it a new route to communicate with. This requires maintainance of a list of correspondent hosts to which binding cache update messages have been sent in the past.
- *Random, deliberate Binding Cache Updates*- Essentially, the scheme requires a mobile host (X) to send a binding cache update to a correspondent host (Y) each time that it moves to a different network. However, this allows Y to keep track of X's movement patterns, though it may not know the location of X. To prevent Y from inferring such knowledge, X may send random binding cache update messages, with a different nonce. This prevents Y from distinguishing between whether the update was due to a change of location of X, or due to a change of nonce. These update messages must be sent at random intervals, and not periodically to prevent Y from observing any patterns. Another benefit of such messages is that they prevent Y's timer for a Binding Cache Entry from expiring.
- *Binding Cache Authentication*- Just as in Route Optimization, any binding cache update message sent by the mobile host needs to be authenticated.
- *Tunneling of packets from home agent*- Whenever a home agent receives packets destined for a mobile host, it tunnels these packets to the foreign agent. However, an eavesdropper listening to this traffic could correlate the address of the mobile host (present in the inner IP header) with the address of the foreign agent (present in the outer IP header), thereby compromising the location privacy of the mobile host. One solution is for the home agent to encrypt the data to the foreign agent, but this requires sharing of a secret key between the two, which may not be feasible. Another solution is that when the mobile host registers a new care-of address with the home agent, it does so with a packet having the encryption option set. The home agent can then forward packets directly to the mobile host using the encoded path and the decryption option. This solution has the interesting property that the current location of the mobile host (when away from home) is hidden from its own home agent.

3 Implementation

The purpose of our implementation was to demonstrate that the overhead associated with the routers if they must process the new options using symmetric-key cryptography is small. We did not attempt to implement the modifications to the Route Optimization protocol.

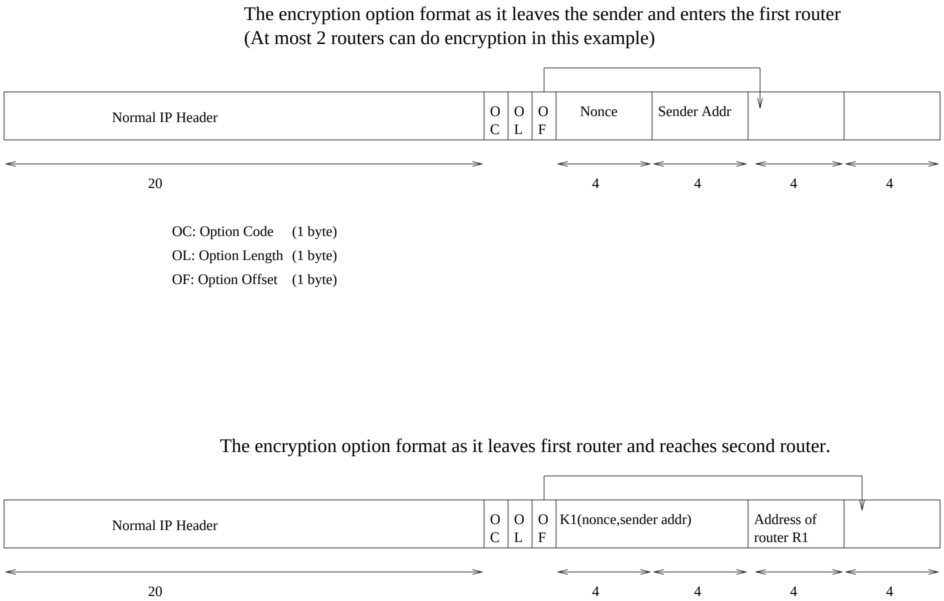

Figure 2: Structure of encryption option, and how a router modifies it

3.1 User-level implementation

We first implemented our scheme at the user level as a test of correctness. This implementation is built on top of TCP and it simulates relevant functions of the IP layer.

In order to test the design, a "virtual" network is constructed. It contains some end-host machines connected to each other by "virtual" links, which are simulated by bidirectional TCP sockets. The "interfaces" on each machine in this network are the corresponding socket descriptors. In this "virtual" network, writing and reading of data to a particular socket simulates transmitting and receiving data from an actual interface of a real network. The data exchanged in this context are TCP stream bytes, but for our purpose we assign each data message a special format analogous to the format of a real IP packet - containing an IP-like header at the beginning followed by the payload portion. During initialization, the simulation code (which runs as a daemon) on each machine reads in a configuration file that contains the topology of the virtual network, the "neighbours" of the particular machine and the "routing table" that this machine must use. Once the "virtual" network is set up, the simulation daemon simulates IP layer input, output and routing functionalities. It waits for data from any of the "interfaces" or from a higher level layer (simulated by a process). The daemon processes each received packet the same way as IP does - it reads the IP header, processes the IP options and makes appropriate forwarding decisions if it is not the intended receiver of the message.

Our design modifies the IP protocol by adding two new option types, encryption and decryption. The format of the IP options and the corresponding processing actions are the same for both user-level and kernel-level implementations, which will be discussed in detail in next subsection.

3.2 Kernel Implementation

Our implementation consisted of changes to the NetBSD kernel to support the new encryption and decryption options. We describe the structure of each option along with the sequence of operations that a router has to perform for each operation. We needed to change just one file (*ip_input.c*), and only one function, *ip_doptions*(). We also present code that needs to be executed by the end hosts, but did not implement these at the kernel.

Encryption:
The structure of the encryption option is shown in Figure 2. We present the code that must be executed by a sender (end host) below. As the maximum possible length of an IP option is 40 bytes, and $(11+4m)$ bytes

are consumed if m routers on the path were to perform encryption (Step 2), at most 7 routers participate in the encryption process along any path.

```
1. Set OptionCode <- Encryption
2. Let n be the maximum number of routers in the path that must perform encryption.
   Set Option Length to (3+4+4+4*m). This is to provide for 4 bytes of nonce, 4 bytes
   of source address and m router addresses. Note, $m \leq 7$.
3. Generate a random nonce, and copy it to the appropriate position (bytes 4-7)
4. Copy source address (location dependent) to the appropriate position. (bytes 8-11).
5. Set Option Offset to point to the position after the source address (12).
```

The code executed by a recipient (router or end host receiver) of a packet which has the encryption option set is presented below. It consist of a check as to whether the node is the intended recipient of the packet (Step 1), in which case a binding cache entry is made for the sender and the packet is handed over to higher level protocols. In the case where the node is not the intended recipient, appropriate router code is executed and the packet forwarded.

```
1. if (destination address = myaddress){
        /* This is end host */
        Let source address be SA. Create Binding Cache Entry as follows:
        Set BCE(SA).lastrouter= bytes from offset-4 to offset-1
        Set BCE(SA).route= bytes from 4 to offset-5.
        Hand over data to appropriate higher level protocol.
}
else{
        /* This is a router */
        2. if (offset > length) merely forward.
        3. Encrypt bytes in position 4 upto offset-1.
        4. Add the router's address in position (offset,offset+3)
        5. Increment offset by 4 and forward.
}
```

Decryption:
The structure of the decryption option is shown in Figure 3.

The code executed by the correspondent host when it sends a packet to the mobile host with the decryption option set is presented below.

```
1. Set OptionCode <- Decryption.
2. Set sender address to the binding cache entry's lastrouter address. That is,
   SA <- BCE(SA).lastrouter
3. Set Option Length <- 3+length(BCE(SA).route)
4. Set Option Offset <- 3+length(BCE(SA).route)+1
5. Copy BCE(SA).route to the appropriate place in the option.
```

The code executed by a recipient of a packet with Decryption option set (mobile host or intermediate router) is presented below. First, a check is made as to whether the node is the intended recipient of a packet (Step 1). If this is not the case, the node must be an intermediate router not involved in encryption, and the packet is merely forwarded along. However, if this is the case, it is determined whether the node is the final destination (by verifying the offset in Step 2), when the packet is handed over to the appropriate higher level protocol. Otherwise, the node is an intermediate router which is involved in encryption/decryption, and hence it needs to take appropriate action and forward the packet to the next router.

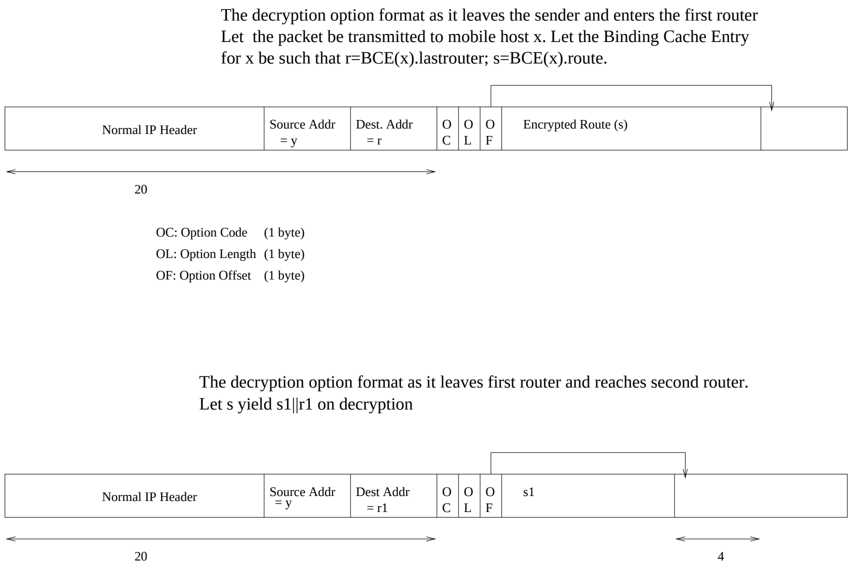

Figure 3: Structure of decryption option, and how a router modifies it

```
1. if (address != myaddress){
    This node was not involved in the encryption process. Merely forward
}
else{
        2. if (offset = 8) {
                /* This is the end host.*/
                Hand packet over to appropriate higher level protocol layer.
        }
        else {
                  /* This is a router */
                3. Decrypt bytes from position 4 to position offset-1 in the packet.
                4. Copy bytes from position offset-4 to offset-1 to the Destination Address
                   field and zero these bytes out in the option.
                5. Decrement offset by 4.
                6. Forward to the new destination.
           }
}
```

Choice of Encryption Function:
A primary requirement of the symmetric-key cryptographic algorithm employed is that the size of the input and output must be same. In our implementation, we pick a stream cipher called RC4, which gives an output that has the same size as the input. Conventional block ciphers, such as DES (8 byte block), have block size too large that it requires sophisticated padding (such as ciphertext stealing) to keep the output size the same as the input. Further, RC4 is about ten times faster than DES in software, and can be even further optimized in hardware [5].

4 Evaluation

4.1 Testbed

The topology of the testbed is shown in Figure 4. The two routers, R1 and R2, are Pentium II 266 MHz PCs running NetBSD 1.2D. The end systems, M1 and M2 are Digital Alpha 21064A 300 MHz workstations running Digital Unix 4.0. All links are full-duplex point-to-point Ethernet links configured as 100 Mbps.

Figure 4: The Test Network

The testbed used is very simple and has several limitations (eg. only 2 hops), however the resources at our disposal were limited. We demonstrate that the increase in the end-to-end delay from M1 to M2 is small and the reduction in the bandwidth that M2 receives when M1 transmits at full blast is small, when the routers are configured to run our kernel. We did not touch the kernel on the end hosts. Rather, the user-level test programs on the end hosts used raw IP sockets to send and receive raw IP packets.

4.2 Delay Measurements

In this experiment, we measure the additional delay incurred on a message with encryption/decryption option set as opposed to an ordinary IP packet without any IP option. Our metric of delay is the round trip time (RTT) observed by the end hosts. RTT is defined as follows: the timer starts when the sender sends a packet (ping) to the receiver; upon receiving the packet, the receiver immediately sends a reply (pong) to the sender, and the timer stops when the sender receives the reply. As our testbed consists of only two links of low-delay, we measure the time taken over several (10,000) ping-pong exchanges to get the aggregate RTT, from which we compute the average RTT. In the table below, the Round Trip Time with option is obtained by sending all pings with the encryption option set, and sending pongs with the decryption option set.

data size (bytes)	Average RTT (without option) (microseconds)	Average RTT (with option) (microseconds)	% increase in delay
350	884.68	971.20	9.77
700	1142.30	1237.03	8.29
1400	1576.68	1673.26	6.12

An increase in RTT in the case where packets have the option set for the same data size is due to two reasons (i) it takes longer to transmit a larger packet (the packet with the option set is 20 bytes more in our experiments); and (ii) the overhead incurred in IP option processing, particularly with regard to encryption and decryption. We find that in our experiments, the absolute increase in delay when the option is present is about 100 microseconds, independent of the size of the data packet. This is reasonable because the routers operate only on the IP header portion, whose size is fixed independent of the data size. Adding the option incurs only a 6% overhead for a packet of 1400 bytes. We expect a smaller percentage increase in delay if the data packet is larger, or if the distance between two routers is longer (leading to a larger propagation delay). On the other hand , our routers are software based and we can expect worse delay for commercial routers where processing a packet with an IP option is much slower than processing an IP packet without option. Further, delay could also increase because of many more routers along the path performing encryption (as opposed to two we have considered).

4.3 Throughput Measurements

We compare the throughput obtained, on transmitting packets without any option set, with encryption option set and with decryption option set. Our experiment was performed as follows: The sender sends a start packet to the receiver, followed by 100,000 data packets of a given size, followed by several stop packets. The receiver starts a timer on receiving a start packet, and stops on receiving a stop packet. The throughput is the number of bytes in the packets that the receiver receives when the timer is active divided by the time taken. For the decryption case, the receiver sends the first packet (encrypted) so that the sender is aware of the route to the receiver. The sender then sends as usual a start packet, 100,000 data packets and several stop packets, all with the decryption option set. The results obtained are summarized in the Table below (each value is an average of 3 iterations). The throughput measurement does not include the IP header, but only the useful data.

Data Size	No Option		Encryption			Decryption		
(bytes)	Sender (Mbps)	Rcv (Mbps)	Sender (Mbps)	Rcv (Mbps)	%Drop in Rcv throughput relative to rcv throughput without option	Sender (Mbps)	Receiver (Mbps)	% Drop in Rcv throughput relative to rcv throughput without option
600	50.64	48.42	49.98	47.90	1.07%	49.15	46.96	3.01 %
1440	94.12	53.23	93.02	49.45	7.10%	92.91	47.78	10.23 %

We find that the average receiver throughput decreases by about 10% for datasize of 1440 bytes with either option set. The throughput with the decryption option set is slightly less than the throughput with encryption option set. This is probably because of extra work needed during processing of decryption that involves changing the destination address of every packet.

Some caution is required while choosing the data size. For a given data size, the total packet size differs depending on whether there is an option or not. Now, the sender and receiver throughput depends on the raw size of the packet transmitted. For example, memory management policy changes avoiding memory-to-memory copy for packet sizes larger than 1024 bytes leading to drastic increase in sender throughput. Hence, a choice of data size of 1010 bytes would be misleading as it would imply a much higher sender throughput for packets with the option set. Similarly, for total packet sizes greater than 1500 bytes, fragmentation sets in reducing throughput. Hence, data sizes above 1440 bytes could potentially lead to unfairly poorer performance for packets with the option set.

5 Related Work

Chaum [1] has proposed a scheme for untraceable email. The scheme involves routing of data through a cascade of intermediate nodes called mixes. The sender of the message encrypts a path through this sequence of mixes using the public keys of these mixes in such a manner that each mix knows the next mix to which it must forward the data and the mix from which it received the data and no other mix in the cascade. Onion Routing [3] extends Chaum's scheme by deploying application-level "onion-routers" that handle not only email, but other applications such as anonymous web-browsing. Crowds [4] is another scheme to protect anonymity on the World Wide Web. It is a set of geographically diverse hosts (crowd) on the Internet, linked by encrypted tunnels. An HTTP request is initiated from one crowd member, and sent to another randomly selected crowd member. The crowd member flips a biased coin, which either forwards the request to another selected crowd member, or sends the request to the HTTP server. The scheme deters local observers from knowing who the original sender is. These solution if directly extended to our problem, would require support for computationally expensive public key cryptography in the routers, and would require the sender of the message to know the route to the destination apriori

6 Conclusions and Future Work

In this project, we have proposed a scheme that can ensure location privacy without compromising Route Optimization. Our scheme requires two new options in IP, and processing of these options involves symmetric-key cryptography by the routers. A major concern is whether such cryptography would be too computation expensive to be of practical use. To address this concern, we have modified the NetBSD router kernel, to support these options and have demonstrated on a test network that the increase in end-to-end delay and decrease in Throughput are both marginal. The increase in delay is between 6% and 9%, while the decrease in Throughput is between 1% and 10%. We have used the RC4 algorithm, which while not as secure as DES is computationally much cheaper, and is sufficiently secure for our purposes.

While the results are encouraging, we cannot be too hasty in generalizing our results. Specifically, the test network that we have used requires only two routers to perform encryption/decryption. A realistic scenario may require many more routers to perform such functions. Secondly, our experiments are on software-based routers. In practice, commercial routers are hardware based, and are extensively optimized for forwarding packets without options, while packets with options follow a slower software path.

Work also needs to be done in rigorously specifying the modified Route Optimization Protocol. We have left many questions unanswered. What are the overheads associated with sending random binding cache update messages? When must these messages be sent? What happens if a particular route that a correspondent host uses breaks down (in our current design, it has to patiently wait until the next update message)? A comprehensive implementation would be in order to expose any issues that may have been missed.

Our current design requires that a mobile host use its usual IP address as the source address when it is communicating with another host. However, this might be complicated by the growing concern for Ingress Filtering.

Finally, our approach is also suitable for providing Identity Privacy for all nodes in the Internet. It would be interesting to examine if any modifications have to be made to our scheme, and if any new issues crop up.

References

[1] David Chaum, "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms", CACM, 24(2):84-88, February, 1981.

[2] David B. Johnson, "Scalable support for transparent mobile host internetworking", Mobile Computing, Chapter 3, pp 103-128.

[3] Michael G. Reed, Paul F. Syverson, and David M. Goldschlag, "Anonymous Connections and Onion Routing", to appear in the IEEE Journal on Selected Areas in Communication Special Issue on Copyright and Privacy Protection, 1998.

[4] M. Reiter and A. Rubin., "Crowds: Anonymity for Web Transactions (preliminary announcement)", DIMACS Technical Report 97-15, April 1997.

[5] Bruce Schneier, "Applied Cryptography".