Flying Freedom: Location Privacy in Mobile Internetworking

Alberto Escudero, Martin Hedenfalk, Per Heselius
inet2001-freedom@flyinglinux.net
Royal Institute of Technology - KTH / IMIT
Electrum 204 - S 164 40 Kista
SWEDEN

April 26, 2001

Abstract

The Freedom System is a pseudonymous IP network that provides privacy protection by hiding the user's real IP addresses, email addresses, and other personal identifying information from communication partners and eavesdroppers. The following paper describes a set of protocol extensions to the Freedom System architecture to permit a *mobile node* to seamlessly roam among IP subnetworks and media types while remaining untraceable and pseudononymous.

These extensions make it possible to support transparency above the IP layer, including the maintenance of active TCP connections and UDP port bindings in the same way that $MobileIP_{v4}$ does but with the addition that the home and foreign network are unlinkable. We call this extension the *Flying Freedom System.*

1 Introduction

There are several important issues regarding security in wireless networks. As in all computer communications, these include message integrity, authentication, and confidentiality. Message integrity means that the message is transmitted without alteration, authentication means that the sending/receiving user is the one he claims to be, and confidentiality means that no one other than the intended party is able to read the transmitted message. In wireless networks, where users move between different networks and media types, another issue becomes equally important: **location privacy**. Location-aware services take advantage of the user's or terminal's location information, but what happens if the user doesn't want to be located? This means that it should be impossible to locate where a mobile user is currently working, if he/she so desires. [1]

In cellular mobile systems, such as GSM/GPRS or UMTS, it is also possible to locate users based on the cell they are in or in some cases even where whitin the cell the user is. In the future, a customer may choose whether this should be possible or not. Service providers could offer location privacy services as an add-on service for their customers.

1.1 Location privacy while seamlessly roaming

This paper presents a set of protocol extensions to the Freedom System [7] which provides similar functionality as in $MobileIP_{v4}$ [2] and also includes location privacy[4]. The Freedom System has been developed by the canadian company Zero Knowledge Systems Inc.

MobileIP allows users to move between different networks, while maintaining the same IP address. This is done by associating a care-of-address with the mobile node when it is away from home. All traffic to the mobile node is intercepted in the home network by a home agent that tunnels the data to the care-of-address.

When providing location privacy to the mobile node we need to ensure that:

- The home network should have no knowledge about which foreign network the mobile node is currently connected to.
- Similarly, the foreign or "roaming" network should have no knowledge about the mobile node's home network.
- An eavesdropper or man-in-the-middle should not be able to tell who the communicating parties are.
- In addition, all the usual communication security constraints must apply; i.e., message integrity, authentication and confidentiality.

2 A Pseudonymous IP Network: Freedom overview

This section is a quick overview of the Freedom System architecture and has been written with the intention of providing sufficient information to understand our protocol extensions to the Freedom System. For a detailed look at the entities, systems and protocols that make up the Freedom System we refer the reader to the Freedom Network architecture white papers [6,7,8].

The Freedom System is a **Pseudonymous IP,** PIP, network [9]. The PIP network provides privacy protection by hiding the user's real IP addresses, email addresses and other personal identifying information from counter-parts and eavesdroppers.

The Freedom System makes it possible for a user to access the Internet without revealing any location or personal information, through the use of so called *Nyms*. The user connects to the Internet via the Freedom System that encrypts the traffic and reroutes it through special servers. Which servers to be used in the routing is determined by the user before the connection is established. Each server only knows the next and the previous proxy on the route. This way a third person eavesdropping the channel can't find out the source and destination of the connection. Since all traffic is encrypted, the content is not visible to anyone else.

The Freedom System could be seen as an overlay network composed of globally distributed servers that runs on top of the Internet. Freedom routers or **Anonymous Internet Proxies** AIP_is are the core network privacy daemons and they are in charge of passing encapsulated packets between themselves until they reach an exit node AIP_{exit} or AIP wormhole. When a certain AIP_i runs as an AIP_{exit}, it works as a traditional network address translator, NAT.

Symmetric link encryption is applied between node pairs (AIP to AIP $\{AIP_i - AIP_{i+1}\}$ and freedom-client to entry-AIP $\{FC_j - AIP_1\}$) to hide the nature and characteristics of the traffic between them.

When a freedom client with IP address IP_{FC_j} communicates with a correspondent node CN_m via a previously built virtual circuit VC_x in the Freedom System, the correspondent node sees that the traffic as coming from the wormhole IP address $IP_{AIP_{exit}}$ instead of the client's real IP address.

The client creates a virtual circuit inside the freedom network by sending a route creation packet which contains secrets S_N[1] to be shared, with each AIP_i in a chosen chain. The route create packet uses *Nested ElGamal encryption* to securely transmit the shared secrets and to ensure that each AIP_i can only read the part of the *route create packet* destined for itself. Hence, AIP_i only knows the previous AIP_{i-1} and the next AIP_{i+1} in the chain as given in the *route create packet*. The Nested ElGamal encryption is performed using the AIPs' public keys $K_{publicAIP_i}$.

The set of encryption layers (multilayer nested encryption) is called **"telescope encryption"** and it is used to provide *"freedom client-to-wormhole"* confidentiality to both route creation and data packets.

Once the route VC_x is created from the freedom client to the wormhole AIP_{exit}, the data packets travel towards the wormhole over the virtual circuit, being link decrypted, telescope unwrapped and finally link encrypted at each point.

The data is routed to the next hop by use of an **Anonymous Circuit Identifier** (ACI) mapping table. The ACIs indicate, along with a packet's implicit source address and port, the next hop in a particular route.

Data coming in over a given $[IP_{AIP_i}, Port_{AIP_i}, ACI_k]$ is first link decrypted and then telescope encrypted with the key generated from S_N. Finally the data is link encrypted and sent on its way to $IP_{AIP_{i+1}}$ with a rewritten ACI value, ACI_{k+1}.

When the ACI_k from the incoming data packet indicates that the packets from that entity need to be sent to the wormhole, the AIP_{exit} acts as a NAT_x for that connection. In this case, the wormhole will map the ACI value ($ACI_k = ACI_{exit}$) with a TCP (or UDP) local port.

2.1 Freedom virtual circuit example.

Let us consider a freedom client FC_j that wants to communicate with a correspondent node CN_m. The FC_j chooses a set of three AIP_i from the globally distributed Freedom AIPs $\{AIP_1 - AIP_2 - AIP_{exit}\}$. The chosen chain establishes one virtual circuit VC_x between the freedom client and the wormhole AIP_{exit}.

The freedom client negotiates a link encryption key with $AIP_1 = AIP_{entry}$ for the $\{FC_j - AIP_1\}$ link.

During the route creation process each AIP_i receives from the FC_j a unique shared secret $S_N = preKeySeed_N$ for that session. The shared secret is mapped to the ACI_k field of the incoming route create packet.

It is also during the route creation when each AIP_i is responsible for choosing a random locally unique ACI_{k+1} that will be used to send packets to the next AIP_{i+1}. The first ACI_1 in the virtual circuit chain is selected by the FC_j.

[1] The S_N, named *preKeySeed,* is a *key seed* that is used to generate keys for the three symmetric algorithms (routeCrypt, bckSymAlg and fwdSymAlg) [7].

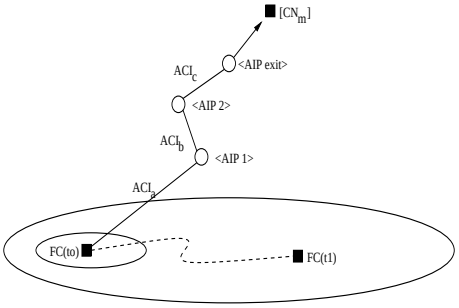

Figure 1: Freedom Overview

In figure 1 we can see that:

- FC_j chooses ACI_a and shared secret S_A to communicate with the freedom entry AIP, AIP_1.
- After applying link decryption using the previously negotiated key with FC_j; AIP_1 knows that packets coming from IP_{FC_j} address with ACI_a can be decrypted with the key generated from shared secret S_A[2] and have to be link encrypted and forwarded to AIP_2 with rewritten ACI value, ACI_b.
- In the same way AIP_2 (after link decryption) uses S_B to decrypt packets coming from AIP_1 with ACI_b and link encrypts and forwards them to AIP_{exit} with ACI_c.
- After link decryption in AIP_{exit} the last layer of the *telescope encryption* is removed using the key derived from the shared secret S_C. AIP_{exit} also maps the packets coming from AIP_2 with ACI_c and certain port number to a local non routable IP address that will act as the source of a NAT_x session.

3 Protocol extensions to the Freedom System

Our protocol extensions to the Freedom System can be divided into two types. The first type concerns location privacy when the mobile node is run only as a client, i.e. the mobile node is only making outbound connections *(mobile client location privacy)*. The second set of extensions concerns location privacy when the mobile node also wants to act as a server accepting inbound connections from corresponding hosts *(mobile server location privacy)*.

We have identified three different subcases for mobile client location privacy:

[2]The S_A is used as key seed material to generate the key for the algorithm *(fwdSymAlg)* that is used to decrypt the corresponding encryption layer when data travels towards the AIP_{exit}.

STATE	from	to
$Before$	$FC_j[IP, Port](t_0)$ - $ACI_1(t_0)$	AIP_2 - ACI_2
$After$	$FC_j[IP, Port](t_1)$ - $ACI_1(t_1)$	AIP_2 - ACI_2

Table 1: Mappings in AIP_{entry} before and after a handover.

- **Case 0** (full route create): The mobile node sends a new **ROUTE CREATE** message after changing its point of attachment rebuilding the whole virtual circuit but keeping the same AIP_{exit} and ACI_{exit}, i.e., to preserve TCP connections and UDP port bindings.
- **Case 1** (partial route creating preserving AIP_{entry}): The mobile node sends a **ROUTE CREATE**$^{v.3}$ message[3] to the AIP_{entry} that updates the partial route. The information in the *route create packet* is used to renew the $[IP_{FC_j}(t), Port_{FC_j}(t), ACI_1(t)]$ parameters while preserving the mapping with $[IP_{AIP_2}, Port_{AIP_2}, ACI_2]$. If we represent the stages before and after a handover with t_o and t_1, then the ACI mappings in an entry AIP (AIP_{entry}) are represented as: [table 1].
- **Case 2** (partial route creating non-preserving AIP_{entry}): The mobile node sends a **ROUTE CREATE**$^{v.3}$ message upwards in the hierarchy of AIPs until the message reaches the "switching" AIP. All the routes under the switching AIP are updated preserving the higher part of the hierarchy [5].

When talking about mobile server location privacy:

- **Case** 3: The mobile server is reachable at IP_{FS_j} and $Port_{FS_j}$ allocated in a choosen AIP_{exit}, but the care-of-address of the server is not known by any AIP_i. In this case the mobile server registers an IP_{FS_j} and $Port_{FS_j}$ served by some AIP_{exit} in the freedom system. When packets arrive to that IP and port, the data travels back over the network, the information is passed along the route indicated by the ACI_k until it reaches the AIP_{entry} and link encrypted to the care-of-address of the freedom mobile server $IP(coa)_{FS_j}$. The data is transported from the AIP_{exit} to the client in the same way as data packets are transported in the normal mode of operation, i.e. the data packets are link decrypted, telescope encrypted and finally link encrypted in each AIP_i.

[3]The ROUTE CREATE and ROUTE CREATE ACK messages described in case 1-3 are not supported in current Freedom 2.x and they are part of our protocol extensions proposal.

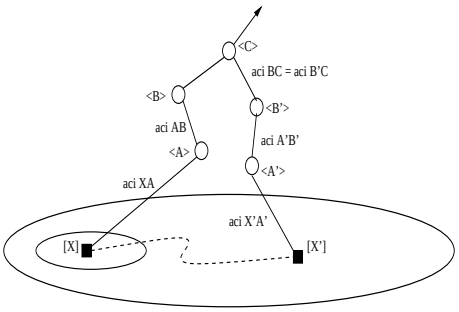

Figure 2: Case 0: ROUTE CREATE $AIP_{exit} = AIP_{switch}$

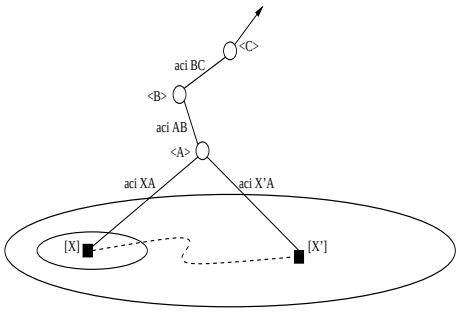

Figure 3: Case 1: ROUTE CREATE $AIP_{entry} = AIP_{switch}$

3.1 Mobile client location privacy

One of the possible scenarios looks as follows: A freedom client FC_j running an IEEE 802.11 wireless interface $IP_{FC_j}(t_0)_{WLAN}$ (while communicating with correspondent node CN_m) is moving from an indoor environment to an outdoor GPRS wireless network. A "vertical handover" is performed from one media type to another and the *mobile node* obtains a new IP address $IP_{FC_j}(t_1)_{GPRS}$ from the GPRS network.

The correspondent node is not aware of the mobility of the *mobile node* and futhermore the AIP_{exit} is also not aware of which foreign networks the mobile node is roaming in. The AIP_{exit} acts as a $MobileIP_{v4}$ home agent for the freedom client and the AIP_{entry} acts as a $MobileIP_{v4}$ foreign agent. The AIP_{entry} and the AIP_{exit} are unlinkable [3].

3.1.1 Case 0: Full route create

The option for a full recreation of a route maintaining the same TCP/UDP bindings is a present feature in the Freedom 2.x architecture [Fig.2] . The *specAci* field in the **ROUTE CREATE**$^{v.2}$ packet allows the freedom client to specify the ACI_{exit} that the wormhole AIP_{exit} should use so that a route can be extended or changed using the same exit hop. This feature allows a freedom client to dynamically add a new AIP in the chain, preserving the previously allocated ACI_{exit} - NAT_x mapping.

A successful route creation is completed when the AIP_{exit} checks and validates the **Nym signature**. All packets received by AIP_{exit} from AIP_{exit-1} with a certain source port number and an ACI value (ACI_{exit}) are mapped to a local non routable IP address that will act as the source of a NAT_x session.

This reserved ACI_{exit} in the AIP_{exit} is used to identify the socket used to communicate through TCP/UDP with the corresponding host CN_m. The allocated ACI_{exit} is sent back in the **ROUTE CREATE ACK**$^{v.2}$ answer in response to the client's initial **ROUTE CREATE**$^{v.2}$ message. A **ROUTE CREATE ACK**$^{v.2}$ packet is just a data packet with the datapacket type field set to a special value. The payload carries the 2 byte ACI_{exit} number allocated in the AIP_{exit} for that session.

When the client wants to change its point of attachment, it sends a new **ROUTE CREATE**$^{v.2}$ message using the *specAci* field that is set to the $ACI_{exit}(t_0)$. The $ACI_{exit}(t_0)$ is acquired from the **ROUTE CREATE ACK**$^{v.2}$ message from the previous route creation. This way the TCP connection and UDP port bindings between the AIP_{exit} and the CN_m are preserved, and thus the connections between the mobile client and the corresponding host.

The whole route is rebuilt between the AIP_{entry} and AIP_{exit}, even where those routes are unchanged. The **Nym signature** is also rechecked at the AIP_{exit}.

From the point of view of all applications running on the freedom client the connection looks unchanged though the client IP address has changed ($IP_{FC_j}(t_0) \neq IP_{FC_j}(t_1)$).

3.1.2 Case 1: Route creating a preserving AIP_{entry}

This is the first of the proposed extensions of the Freedom System, to be able to change the point of attachment, while preserving TCP/UDP connections, but without rebuilding the whole route [Fig 3]. The freedom client gets a new IP address $IP_{FC_j}(t_1)$ (perhaps due to a move to another network) but uses the same $AIP_{entry}(t_0) = AIP_{entry}(t_1)$.

As shown in *case 0* the current solution requires that the whole route is rebuilt (except for the socket binding in the AIP_{exit}) and that the **Nym signature** is rechecked.

Case 1 presents an alternative when the mobile node wants to keep using the same entry AIP ($AIP_{entry}(t_0) = AIP_{entry}(t_1)$). In this case, the whole route does not need to be rebuilt.

Case	AIP_{switch}
Case 0	AIP_{exit}
Case 1	AIP_{entry}
Case 2	AIP_i

Table 2: AIP_{switch} depending on the case.

In order to update the route binding for the mobile node, the AIP_{entry} needs to be notified that:

- the freedom client has a new IP address $IP_{FC_j}(t_1)$.
- the freedom client already has had a route binding for the old IP address $[IP_{FC_j}(t_0), Port_{FC_j}(t_0), ACI_1(t_0)]$.

The mobile node first has to exchange a new shared secret with the entry AIP ($AIP_{entry}(t_1)$) to be able to establish new link encryption between the freedom client and the entry AIP. $\{FC_j(t_1) - AIP_{entry}(t_1)\}$.

The mobile node then sends a **ROUTE CREATE**$^{v.3}$ message, as described in [10], that contains the old IP address $IP_{FC_j}(t_0)$, old port $Port_{FC_j}(t_0)$, old $PreKeySeed_{AIP_{entry}}(t_0)$ and old ACI $ACI_1(t_0)$. The AIP_{entry} then checks the authenticity of the message by checking that the $PreKeySeed_{AIP_{entry}}(t_0)$ sent with the update is the same as the one that was previously exchanged between the client and entry AIP. ($IP_{FC_j}(t_0)$ with $ACI_1(t_0)$). If the message is verified to be correct, it then updates its route binding (uniquely identified with the $[IP_{FC_j}(t_0), Port_{FC_j}(t_0), ACI_1(t_0)]$) with the new $[IP_{FC_j}(t_1), Port_{FC_j}(t_1), ACI_1(t_1)]$ which is extracted from the **ROUTE CREATE**$^{v.3}$ header, see [table 1]

3.1.3 Case 2: Route creating a non-preserving AIP_{entry}

To generalize *case 1*, we introduce the concept of a "switching AIP", AIP_{switch} [Fig. 4].

When the mobile node changes its point of attachment (IP address), it may not want to use the same AIP_{entry}. For example, it may be impossible to use the same AIP_{entry} becouse it resides in a private network.

However, some $AIPs$ in the route can be the same, so the minimum route that needs to be rebuilt, is the partial route upwards to the first common AIP (AIP_{switch}).

If the mobile node selects $AIP_{switch} = AIP_{entry}$, this would behave as in *case 1.* If $AIP_{switch} = AIP_{exit}$, this case would behave as in *case 0* [table 2].

In any case the mobile node first has to perform a new key exchange with the new $AIP_{entry}(t_1)$ to be able to establish link encryption between the client with the new IP address and the entry AIP $\{FC_j(t_1) - AIP_{entry}(t_1)\}$.

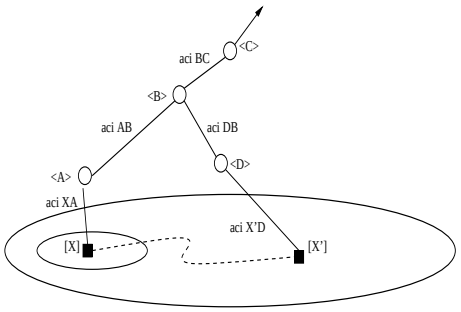

Figure 4: Case 2: ROUTE CREATE $AIP_i = AIP_{switch}$

The client sends a **ROUTE CREATE**$^{v.3}$ message along the new specified path, up to the switching AIP, AIP_{switch}. The AIP_{switch} discovers that this is actually an update of an existing route, updates its bindings and disables the old route by sending a teardown message down the old path. If this teardown message is lost, the old route will eventually time out, since no new data will go that way.

In the same way as in *case 1* the **ROUTE CREATE**$^{v.3}$ message verifies to the AIP_{switch} that this message is authorized to update the binding represented by $IP_{AIP_{switch-1}}(t), Port_{AIP_{switch-1}}(t), ACI_{switch}(t)]$ from the values at t_0 to the new ones at t_1.

To succeed with this, the **ROUTE CREATE** $^{v.3}$ contains the old IP address and port of the next lower entity ($IP_{AIP_{switch-1}}(t_0)$)- ($Port_{AIP_{switch-1}}(t_0)$), the old $PreKeySeed_{AIP_{switch}}(t_0)$ and the old ACI $ACI_{switch}(t_0)$.

This means that the client must know all ACI_i used along the route. This can be accomplished by modifying the **ROUTE CREATE ACK**$^{v.3}$ message, sent back from the initial **ROUTE CREATE**$^{v.3}$ message, so that each AIP_i in the chain adds its own ACI_k number to the message before it is passed on along the route. The client already knows the IP addresses of all AIPs (AIP_i), since it is the client's responsibility to choose the chain of AIPs (AIP_i) in the first place.

If the $PreKeySeed_{AIP_{switch}}(t_0)$ is verified to be correct, the AIP_{switch} sends a teardown message down the old route, and updates its bindings to reflect the change.

The **ROUTE CREATE**$^{v.3}$ message is similar to the standard **ROUTE CREATE**$^{v.2}$ message, in the way that new shared secrets $S_N(t_1)$ are exchanged between the client and the new set of AIPs ($AIP_i(t_1), i < switch$), within each layer of the *telescope encryption.*

The multilayer encryption ensures that each secret is only known by the respective AIP_i. The client reuses the secret established with the AIP_{switch} in the initial **ROUTE CREATE**$^{v.3}$ message $S_{switch}(t_0) = S_{switch}(t_1)$.

The AIP_{switch} has to send an acknowledgement that the route actually has been updated. This message is identical to the **ROUTE CREATE ACK**$^{v.3}$ except it only contains the newly chosen ACIs (ACI_i, $i < switch$) in the partial route.

3.1.4 Switching policies

How does the client decide what AIP should be the switching one? Three possible policies are:

- Preserve as much of the old route as possible. This yields a shorter path for the **ROUTE CREATE** message, which in turn yields faster handover.
- Optimize the route length. This yields fewer hops in the route from the client to the destination.
- Change more of the route than actually needed, to increase the privacy level.

3.1.5 Entry AIP discovery

In our scenario we used a mobile client with both a wireless LAN like 802.11b and a GPRS interface. The mobile client wants to roam between different IP networks hiding the mobility from both the corresponent node and the wormhole.

The mobile node needs to know which entry AIPs (AIP_{entry}) are available in the different IP networks it is roaming in.

The mobile client determines which AIP_{entry} to use based on the following discovery procedure:

All AIPs (AIP_i) sends out an "AIP advertisement" periodically. The "AIP advertisement" message is a standard ICMP router advertisement message with a *Freedom AIP advertisement extension,* see [10]. The TTL field should be set to 1, and the destination should be 255.255.255.255 (limited broadcast).

The client can also force an advertisment by sending out "AIP solicitation" messages. The "AIP solicitation" message is a standard ICMP router solicitation message with TTL set to 1.

Which AIPs to use in the rest of the route created is determined by the user, based on the information retrieved from the freedom core servers.

One interesting feature of Freedom System that can be used to speed up handovers is that the client can create secure links between itself and *more than one* AIP_i. The $\{FC_j(t_1) - AIP_{entry}(t_1)\}$ encryption link can be established prior a new route creation is requested.

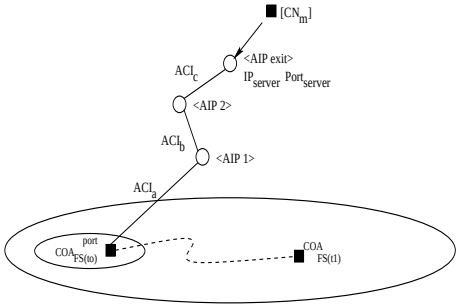

Figure 5: Mobile Server

3.1.6 Handover

The mobile node performs handover when a change in point of attachment has been detected. The change can be detected either becouse the old connection is lost or by the client receiving agent advertisement messages from a new network. If a connection is lost then the client sends an agent solicitation message to trigger an agent advertisment message.

3.2 Mobile server location privacy

With this second type of protocol extension we want to allow an external node to start a connection to a *mobile server*, using the Freedom System, via an IP address IP_{FS_j} and port $Port_{FS_j}$ previously registered in the AIP_{exit}.

The AIP_{exit} acts as a home agent for the mobile server, accepting incoming connections and making the data travel back over the network to the care-of- address that the mobile server is using while moving.

The information is passed along the route indicated by the $ACIs$ until it reaches the AIP_{entry} and then it is link encrypted to the mobile server IP address. The data is transported from the AIP_{exit} to the client in the same way as data packet is transported in the normal mode of operation, i.e. the data packet is link decrypted, telescope encrypted and finally link encrypted in each AIP. The AIP_{entry} acts a foreign agent for the mobile server. The IP address of the server is in fact its care-of-address $IP(coa)_{FS_j}$.

The mobile server that wants to be reachable via the Freedom System opens a "control connection" to the AIP_{exit} and registers an IP address and port where the AIP_{exit} should listen to incoming connections. This registration is mapped with an ACI_{exit}. This ACI_{exit} binding is created by sending a **ROUTE CREATE**$^{v.3}$ message that includes the number of IP addresses and ports to be registered with the exit AIP and how those IP addresses and ports should be mapped to the remote local ports that the service is listening to on the mobile server.

In the previous cases the AIP_{exit} maps the mobile node port number and ACI_{exit} with a random port selected by the NAT for the outbound connection.[4] In our new case the **ROUTE CREATE**$^{v.3}$ message would explicitly inform the AIP_{exit} that inbound connections to the mobile server's "home address" with *port X* should be mapped to ACI_{exit} and *port X'*.

Each AIP_i keeps a *control timer* for each virtual circuit that is reset when data is transmitted. When the timer expires a *teardown message* is generated to destroy the virtual circuit. If a corresponding host tries to connect when the route has been destroyed, this would of course lead to the inbound connection being lost.

To avoid the destruction of the route we could regularly send "keep alive" traffic. The best way would be to send a **ROUTE KEEPALIVE** message regularly at some time interval suitable compared to the route teardown timer. This message is basically a normal *data packet* with a special type [10]. Data packets of this type are passed along the route to the ACI_{exit} where they are discarded. Each AIP resets their expiration timers before passing this message to the next AIP.

4 Conclusions

Our protocol extensions of the Freedom System permits a mobile client to seamlessly roam among IP subnetworks and media types while remaining untraceable. These extensions makes it possible to support transparency above the IP layer, including the maintenance of active TCP connections in the same way that $MobileIP_{v4}$ does but with the addition that the **home network and foreign network are unlinkable** [4].

The new proposed routing messages provide the mobile node the **flexibility of rebuilding partial routes** hiding the mobility associated with certain pseudo-*Nym*.

We have also introduced the possibility of having an **unlocated mobile server** roaming behind the Freedom System. The mobile server is able to accept incoming connections via a home address/port previously registered in one of the Freedom System's wormholes.

About the authors

Alberto Escudero <alberto@it.kth.se>, is graduate student at the Institution of Microelectronics and Information Technology. (formerly Teleinformatics). KTH. Sweden. As a researcher in IMIT, Alberto is currently focusing on *location privacy* in mobile internetworking and privacy protection of personal information and over the last year was involved in the IT University - Kista Wireless Network, FlyingLinux.NET and The Big Brother "storebror" Project. http://www.it.kth.se/~aep

Martin Hedenfalk <mhe@home.se>, is an undergraduate student in Electrical Engineering at the Royal Institute of Technology, Stockholm, Sweden. Martin has many years of experience with GNU/Linux and has developed networking software for the Open Source community and currently combines his studies by working part time as a Linux consultant.

Per Heselius d97-phe@nada.kth.se, is an undergraduate student in Computer Science at the Royal Institute of Technology in Stockholm, Sweden. Per is interested in developing systems for Computer Security, Cryptography and Computer Communications and is currently finishing the last year of his M.Sc. in Computer Science and Engineering.

[4] In order to guarantee that IP and port are unique for the NAT, the AIP_{exit} allocates one fake non routable IP address per ACI_{exit}. The mobile node local port number and ACI_{exit} are mapped to IP_{fake} and a random port number. IP_{fake} and $Port_{IP_{fake}}$ are used as source of *requests connections* to the NAT.

References

[1] Forrester Report. *"Surviving the privacy revolution"*. February 2001

[2] Perkins, C. *"IP Mobility support, RFC 2002"*. 1996

[3] Pfitzmann, A., Köhntopp, M. *"Anonymity, Unobservability, and Pseudonymity - A Proposal for Terminology"*. 2000
http://www.koehntopp.de/marit/publikationen/

[4] Fasbender, A., Kesdogan, D., Kubitz, O. *"Variable and Scalable Security: Protection of Location Information in Mobile IP"*.

[5] Forsberg, D., Malinen, J.T., Maline, J.K., Weckström, T. *"Dynamics - HUT Mobile IP a Technical Document"*. 1999
http://www.cs.hut.fi/Research/Dynamics/documents.html

[6] Boucher, P., Shostack, A., Goldberg, I. *"Freedom System 2.0 Architecture"*. 2000
http://www.freedom.net/info/whitepapers

[7] Goldberg, I., Shostack, A. *"Freedom Network 1.0 Architecture and Protocols"*. 1999
http://www.freedom.net/info/whitepapers

[8] Back, A., Goldberg, I., Shostack, A. *"Freedom 2.0 Security Issues and Analysis"*. 2000
http://www.freedom.net/info/whitepapers

[9] Goldberg, I. *"A pseudonymous communications infrastructure for the internet"*. Fall 2000
http://www.isaac.cs.berkeley.edu/~iang/

[10] Hedenfalk, M., Heselius P., Escudero A. *"Location privacy protocol extensions to the Freedom System"*. April 2001
http://www.it.kth.se/~aep/publications