Secure Mobile Networking
Sixth Quarterly Report – Winter 1997

James R. Binkley and John McHugh
Portland State University

15 April 1997

Project Status Overview

We have made progress on both security and redundancy issues this quarter despite the fact that system complexity and the number of moving targets (Mobile–IP, FreeBSD, IPSEC, ISAKMP, etc.) are ever–increasing.

We made two important releases to the Internet. We released a FreeBSD–IPSEC kernel in cooperation with MIT. We also made a second release of Mobile–IP at PSU that included our ad hoc and radio–based handoff code from summer and fall. We also have managed to acquire BBN's code and we sent a newly hired engineer to Boston in order to improve contacts and our knowledge of their system.

At the beginning of the quarter, we were extremely fortunate to hire Bill Trost as our second programmer (along with David Reeder). Both are focused on various security related issues. Bill has a fair degree of UNIX internals and networking experience. He is working on PCMCIA including Fortezza, kernel porting, and higher–level security issues including session–key mechanisms. David Reeder is continuing to work on tying Mobile–IP to IPSEC in terms of lower–level kernel and routing daemon mechanism. Our graduate students are working on redundancy issues.

Specific Accomplishments for Winter 1997

1. FreeBSD–IPSEC kernel release. This was a port of last summer's NRL code to FreeBSD modified to include our routing interface to IPSEC (ESP only). The code is split into two parts, one at PSU, the other at MIT. The first contains the PSU work, without the cryptographic code, the second contains only the cryptographic code.

   PSU – `http://zymurgy.cs.pdx.edu/freebsd--ipsec/freebsd.ipsec.tar.gz`

   MIT – `http://web.mit.edu/network/isakmp`

2. FreeBSD–PSU Mobile–IP release number 2. We released a second version (not IPSEC at this point) of our Mobile–IP for FreeBSD 2.1.0 that includes our authenticated ad hoc work, fast(er) radio–based handoff, and X–based mobile node daemon. It also includes improved and simplified kernel routing mechanisms.

3. IPSEC–integration with Mobile–IP. When the DARPA Project Manager visited us in January, we were able to demo static routing tunnels that used ESP/DES but the mobile node routing daemon were not yet able to use that kernel mechanism. At this point, our version of Mobile–IP can use 2–way ESP/DES tunnels between the Mobile Node and Home Agent, whether at home or away over a Foreign Agent. We can also, according to static configuration on a per IP address basis, install end to end 2–way DES routes between MNs that use the ad hoc routing protocol. We will discuss current status and emerging issues in more detail below. This work will form the basis of a future integrated Mobile–IP/IPSEC release.

4. PCMCIA kernel work bears fruit including Fortezza "contact." Bill Trost has been able to get Jon Inouye's (OGI[1]) supplied PCMCIA kernel infrastructure working and we have made "contact" with the Fortezza hardware. There will be a short section on this subject later in the report. There has been another interesting result here due to a better PCMCIA kernel infrastructure. We can now deploy a "mobile router" that has either a ethernet or serial card AND a wavelan card; i.e., we can make agents out of laptops. We have made use of this facility several times. We deployed a Foreign Agent at the Baltimore DARPA PI's meeting that was a laptop that made a serial connection and had a WaveLAN card. Jim Binkley gave a recent talk at OGI where we deployed a FA that used a lan card and a wavelan card.

5. MIP interoperation. We have successfully interoperated at FTP Inc.'s sponsored Mobile–IP interop. David Reeder attended the recent FTP Inc. interop and our implementation has successfully interoperated with other Mobile–IP implementations.

6. Bill Trost visited BBN in Boston. We have acquired their software. Bill's job will include integration of this code either in spring or more likely in summer.

7. Redundancy work. We have finished a specification of our multi–hop ad hoc routing protocol and also a first draft specification for the Home Agent Redundancy Protocol. We are also in the final stages of constructing a small lab for the Home Agent protocol. There will be a short discussion of both of these efforts below.

[1]Oregon Graduate Institute

Spring and Summer 1997 projects:

In this section, we are going to briefly discuss long term goals and planning. It is not clear that we can make a release in Spring as it has become readily apparent that we must include FreeBSD 2.2 in our next release. We also would like to include ISAKMP in the routing mechanism and as part of the release.

Our next release minimally will include the following attributes:

1. an integrated Mobile–IP/IPSEC based on routing where policy is specified very simply in configuration files.
2. Mobile–IP/IPSEC should include the ability to setup IPIP "tunnels" that use either AH/ESP as supplied in the current NRL kernel infrastructure and agents should be able to only accept packets over tunnels that have IPSEC attributes.
3. ISAKMP will be supported as part of the routing infrastructure. We will use previous Cisco releases for this (and in point of fact, have already gotten the daemon working on FreeBSD, but not the route to PF_KEY interaction).
4. the release will be targeted towards FreeBSD 2.2, as defined by the CDROM shipped by FreeBSD (that has not yet happened, although the release has been formally announced).

We do not choose to predict when we will make this release other than to say it will probably happen this summer. We must point out that we MUST use what we have and cannot afford to deal with all of the many moving targets at once. The list of possible targets includes ISAKMP, IPSEC proper (AH/ESP/combined) including PF_KEY, Mobile–IP, and FreeBSD itself. We must first limit the moving targets to FreeBSD. Of course, we still must integrate other existing sub–systems architecturally which includes more original routing/IPSEC work and ISAKMP, itself. For now, we plan on porting our entire software system to FreeBSD 2.2 as soon as possible. In point of fact, that work has already started and we have just gotten our mobile node daemon working under 2.2. When summer comes, we can hopefully address other moving targets that might include a new NRL release and/or PF_KEY changes.

Personnel and sub–projects:

Jim Binkley acts as project manager. He supervises David Reeder, Bill Trost, and the graduate students. Jim is usually involved in design, code release, review, specification review, and kernel work. Jim will also take on some of the 2.2 porting work as he is the most experienced kernel person.

John McHugh is working on the Fortezza part of the project and is in charge of high–level key issues. John will help manage Bill Trost.

David Reeder is a full–time programmer. David has worked this quarter on integration of Mobile–IP daemons and IPSEC/ESP in routing. He is currently working on finishing the kernel routing mechanism needed for binding IPSEC to Mobile–IP tunnels and other IPSEC related routing work in both the operating systems and Mobile–IP routing daemons.

Bill Trost is our new full–time programmer. Bill brings with him a fair degree of system programming experience. Bill is in charge of "higher–level" security issues such as ISAKMP, and the BBN session–key protocol. He visited BBN a few weeks ago and has been leading our effort to understand their work. For now, he has four sub–projects: 1. Fortezza AND kernel PCMCIA work, 2., taking part in the 2.2 port with Jim Binkley, 3. integration of ISAKMP into our IPSEC routing infrastructure, and 4. integration of the BBN session–key protocol into our version of Mobile–IP.

Xu Hao is a M.S. graduate student who graduated at the end of winter quarter. He completed a first cut multi–hop ad hoc routing during winter quarter.

Jennie Ye is a M.S. student and joined the project last quarter. She has taken over Xu Hao's work and has begun to implement the ad hoc protocol. Jennie spent last quarter coming up to speed on the project.

Bjorn Chambless is a M.S. graduate student who has been with us since last summer. Bjorn made a first draft of the Home Agent Redundancy Protocol (HARP) during winter quarter and will start implementation in spring. He has also setup a small lab as HARP minimally needs 3 routers for testing. (Two routers will act as peered Home Agents, and the third will act as an upstream control router).

IPSEC and Mobile–IP

In this section we will discuss the keys that are currently available in our system, current IPSEC/MIP project status and emerging issues, and provide a short discussion of firewall issues and work we have done in that area.

We have written a rough draft "appendix" on Mobile–IP and Security Policy issues. The appendix is at the end of this document.

Keys In Our Current Software System

There are currently three classes of keys in our Mobile–IP system. The classes include: 1. Mobile–IP authentication keys, 2. IPSEC keys (both AH and ESP), and 3. ad hoc authentication keys. The first and last might be viewed as "routing" keys.

The Mobile–IP keys are indexed by IP–address. There are three kinds:

1. MN–HA,
2. MN–FA,
3. FA–HA.

We support all three but use of the latter two would make the mobile system closed; i.e., external FAs (and MNs) in terms of key administration could not participate.

We typically only use the MN–HA keys. Each MN has its own unique HA key (although it is actually indexed by the HA IP address, thus the mechanism could support multiple HAs). The Home Agent configuration file of course is indexed by MN IP address. A sample key at the MN looks like this:

`mnha_key 204.203.1.2 md5 256 0x12345678901234567890123456789012`

The key type is named by a tag, and is followed by the HA IP address, transform, spi, and key. MN–FA and FA–HA key syntax is very similar. We borrowed the syntax from NRL's /etc/keys format.

The IPSEC keys apply to IP source and destination addresses and not only may be unique in terms of each Mobile Node but can be unique for outgoing and incoming packets. We do not distinguish between socket keys and route keys. The key simply is mapped into a kernel Security Association and is used by either mechanism in the stack. A typical ESP key key pairing as found in a Mobile Node might look like this:

esp 5000 204.203.67.203 204.203.67.210 des-cbc 0123456789abcdef 11223344

esp 5001 204.203.67.210 204.203.67.203 des-cbc 0123456789abcdef 11223344

The NRL format names the type, spi, IP source, IP destination, transform, key and iv (in the case of ESP).

A Home Agent with manually distributed keys would have the same per MN key pairing. Note that we also use the same mechanism with the MN to MN relationship.

The ad hoc keys are currently network wide but could be agent specific or even end system specific. There are two keys since one is intended to be used at home (the network authorization key) and one is intended to be manufactured on the spot and given away (the ad hoc authorization key). The formats are very similar to the Mobile IP key format but currently lack an IP address as an index. Note that any beacon can have BOTH keys or only one.

IPSEC/MIP status and issues

We have spent a good deal of the last quarter debugging our current Mobile–IP and IPSEC routing system. At this point we are able to dynamically install (or remove) 2–way ESP/DES tunnels between the MN and HA, whether the MN is away or at home. We use a very simple policy extension to Mobile–IP (that could *charitably* be viewed as a form of in–line ISAKMP) whereby the MN tells the HA whether or not it wants ESP. Of course the keys are a priori loaded and known to both systems. FAs are not involved and are simply used as routers, hence this 2–way tunneling could also work with the MN that has acquired a COA and has no FA. Further note that the proverbial chicken and egg problem of routing before IPSEC is solved by a simple hint in the MIP packet itself. Presumably we can modify this mechanism to handle either static key management or tell the other side to setup routes that can kick ISAKMP into play.

Getting the system working has not been easy. We have experienced some rather large bugs including the bug (recently solved by us) whereby systems under stress corrupted TCP data in transit that had been encrypted with DES. We finally determined that the problem was an NRL bug and removed the offending code which offered a security feature that we do not need. This gave us a cheap workaround. NRL has been informed and is in the process of fixing the bug. Ironically the Finnish ssh utility, which might be viewed as IPSEC competition, blew up when it detected file corruption - hence it showed us that the bug existed).

During the next quarter we intend to finish our IPSEC/route binding and deploy the IPSEC system internally. At this point we can only bind one SA (e.g., ESP) to any route. We may however want more than one binding, particularly for IPIP tunnel routes, so that we can establish authentication between a HA and FA, and at the same time have an IPSEC association between the MN and HA. We also of course want to be able to do AH as well as ESP. Much of the work to do the latter two mechanisms has been done, but route(4) API work, final hookup, and testing remains.

In addition, we can also address ad hoc topologies where keys have been manually distributed. This is doable because of our beacon–based authenticated ad hoc protocol. A mobile node daemon when it receives a authenticated beacon can lookup the beacon sender by IP address in a policy list and determine whether or not there is a policy installed for that MN (e.g., use ESP or do not use ESP). Given a pre–loaded SA again, when a beacon arrives for MN X, each system simply installs a link layer route with ESP for the other system. We assume any such routes will be symmetric (2–way). Note that unlike the MN to HA case, this use of route/IPSEC is End System to End System and that as always we assume our systems are single user systems.

We should point out that we have somewhat refined various problems that are built–in to the notion of binding IPSEC to routing, especially between IS to ES. We implemented two new switches in the TCP/IP stack that act as IPSEC–oriented packet filters. One switch applies to only IP forwarding (routing and not ES functionality). We call this the IPSEC–forwarding switch. If a system forwards packets, it may or may not apply IPSEC attributes to them. As a result, our MNs only apply IPSEC attributes to packets that originate on them. A spoofing system can not make a proposed plaintext attack on a MN where the IPSEC–forwarding switch is in use. By definition, this switch cannot be used at a HA that forwards packets into a combined IPIP and IPSEC tunnel. The other new switch applies to the input side of the IPIP virtual tunnel driver in the operating system. It can insist that packets that arrive to be detunneled, must either have already had IPSEC successfully performed on them lower down in the stack OR will have IPSEC performed on them later on in this computer. As a result, we have a simple filter that can guarantee that only IPIP packets with an IPSEC attribute can be forwarded off of a particular machine.

During the course of the quarter two thorny issues have emerged. The first is how do we combine ARP and IPSEC/routing? The second is how do we describe policy in terms of configuration? We will not discuss the second issue at this time and only wish to point out that when one combines certain topological possibilities that include using IPSEC, acquiring COAs (there are a number of possible ways), always routing to home or to individual nets or hosts, and other issues, the policy combinatorics become non–trivial.

The ARP issue emerged because we simply lacked kernel mechanism to control ARP access to computers. For example, assume a Home Agent with a wireless link and further assume that you want to control access to the interior so that only Mobile Nodes using IPSEC can qualify. One might (wrongly) think that the MIP registration packet could be used as a policy lookup token so that a agent daemon could determine if ESP was wanted or not. Unfortunately this does not apply to systems that do not play the Mobile–IP game. Any system that uses ARP can get access and send packets through a router as long as its subnet IP address makes sense.

There is a further worrisome detail and that is that the BSD link layer clone route would automatically clone a link layer "host" route (gateway == MAC address) without notification of any daemon upstairs. As a result, our agent daemon that implements the HA function really does not manage ARP routing state for Mobile Nodes that are at Home since the routes are automatically created by the operating system (although it does delete them when a MN moves away). We must be able to determine at the application layer if a system has sent us an ARP message and act on it. We also must make sure that IPSEC attributes apply to all packets sent through a link layer route (created by ARP) if that is the desired policy.

After much puzzlement, we decided on a plan and have implemented the kernel mechanism part of it. Next quarter we will modify our routing daemons to take advantage of the kernel mechanism and thus finally implement our MN–HA (at home) and MN–MN secure topologies using ARP. The basic idea is that we will install the clone route (a necessity for ARP) so that it has two new attributes:

1. the XRESOLVE attribute. This causes an UPCALL to be made when ARP cloning occurs; i.e., a daemon listening on a route(4) message socket will be notified that a system with IP destination X has shown up. This mechanism applies both to packets send and ARP packets received. As a result, we can implement a IP address based policy lookup mechanism.

2. a new BLACKHOLE attribute. The clone route will be loaded on a per interface basis with this flag as well. As a result, cloned routes will not let packets escape until the routing daemon shows up and changes the route (e.g., binds ESP to it and/or removes the BLACKHOLE attribute or logs a breakin attempt). This mechanism oddly existed in terms of route(4) flags but did not actually do anything.

Both of the above attributes should give us enough control to implement IPSEC based routing using ARP.

IPSEC and MIP - Firewalls and Tunnel Considerations

We have made more trips to the Oregon Graduate Institute because the Foreign Agent we installed there (which is now a HA for OGI at the same time) is as previously reported, outside our NWNET border router, hence we cannot talk to home without taking "measures."

It would appear that we have done something clever (or foolish?). We were able to demonstrate that two MNs could simultaneously "borrow" the FA's COA and setup IPIP tunnels back to home systems where the individual tunnels either had or did not have ESP bound to them. We wrote a script that would determine the COA from existing MIP utilities and then automatically install the tunnels. There are at least two points worthy of mention:

1. With ESP, all packets sent from the HA were sent using IPSEC, but we could either bind the default route to send all packets home using IPSEC or leave it alone, and bind IPSEC to the individual tunnel routes that sent packets home. This sort of policy choice is yet another mechanism that we would like to somehow build into our configuration.

2. There is no problem in using the FA's COA since the FA itself (as is probably the case with routers where firewall–oriented checking is not on) do not examine incoming packets to see if their own IP source is used. Since the FA's COA is really only applied to get the packets from the FA to the HA where the outer wrapper is stripped and thrown out, there is no reason more than one MN can't avail itself of that address.

We do not understand the current Mobile–IP effort to allow MNs to "reverse tunnel" home courtesy of the FA. We do not think that MNs should ask FAs to break into their homes and we do much like the idea that routers that are not performing firewall functions, should examine the IP source addresses of incoming packets. Of course, this whole notion is lunatic in terms of "ingress filters" since it serves to institutionalize bypassing them. As a result, a future CERT advisory may very well suggest that unicast IPIP should be blocked. Might one timorously suggest that IPSEC is the answer here?

It is true that stealing the FA's COA may be tacky. Possibly the right answer is that the FA might make COAs available to MNs. A COA could be a useful disguise since an IP outer header with such an address can cross a firewall and if followed by ESP could even hide the MN's own IP address. Of course, in that case, the MN must dynamically setup a Security Association between itself and the HA that binds the SA to the MN's COA.

Fortezza under FreeBSD

We have ported the Fortezza interface library to FreeBSD 2.1.0 using the "PAO" PC–Card support for built–in laptop PC–Card slots distributed by Tatsumi Hosakawa.

In so doing, we discovered an apparent bug in the interface library's code to check for operation completion, which we had to correct before the library would function properly. At this point, it appears that we can talk to the Fortezza cards and other PCMCIA cards at the same time under our hybrid FreeBSD 2.1.0 system.

We have timed the SHA implementation present on the Fortezza card, thereby demonstrating our ability to access the card's cryptographic functions. The Fortezza's implementation of SHA is significantly slower than that provided with NRL's IPSEC distribution, which is software in the operating system. The NRL version is approximately 200 times faster on a 100 MHz Pentium with large (800K) inputs, and faster still with small inputs. The Fortezza documentation points out that the card's SHA implementation is slow, and they recommend using a software version of SHA they provide with the documentation that is approximately half as fast as the NRL supplied SHA. (The NRL version was supplied by NIST.)

Redundancy

In this section, we will present a short description of our redundancy work over the last quarter. This work is being done by graduate students and has focused on the specification of protocols for:

1. a multi–hop authenticated ad hoc routing protocol
2. and a protocol/system for Mobile–IP Home Agent redundancy.

We have also developed and implemented a very simple "Mobile Router" protocol that allows radio–based mobile agents to extend the wired infrastructure. This was an unplanned effort that occurred due to project synergy.

At this point we have developed three different ad hoc protocols, although the multi–hop version can be viewed as a logical extension of the first cut ARP replacement system. These protocols are:

1. a single–hop authenticated ad hoc replacement for ARP. Key ideas here include using a shared key as opposed to a shared subnet, and making address spoofing more difficult by authenticating the MAC address, IP address pair. All systems beacon. This work has been commented on in previous reports.

2. a simple mobile router system where Foreign Agents can also act as mobile nodes and hence become "mobile routers." Such systems only use one wireless interface and have no wired interface. Thus they act as a mobile node and a FA out the same interface. We have invented a very simple protocol that we will discuss below that allows wireless–based mobile routers to not choose each other (and avoid a route loop) if they can hear an agent that is closer to the wired infrastructure. This work is complete and will be included in our next release.

3. the current multi–hop source–routing based ad hoc protocol that we have been specifying this quarter.

Redundancy - Mobile Routers (Mobile FAs)

We discovered that mnd (the Mobile Node daemon) and mipd (our HA/FA agent daemon) can operate on the same computer without noticeably interfering with each other's operation. Such a system consists of one laptop with one PCMCIA wavelan card (radio only). The mobile node side communicates with a wired Foreign Agent and hence can roam. The Foreign Agent side allows any Mobile Nodes that are too far from the wired FA to operate. This allows us to deploy a wireless–based Foreign Agent as long as the Mobile Node (mnd) side can hear a wired infrastructure agent. We have actually used this system in a real life situation to extend the reach of our wireless infrastructure.

However, running multiple mobile nodes as FAs causes problems if two such mobile routers can hear each other. If the two Mobile Nodes are adjacent to each other and distant from any wired FAs, our current Mobile Node radio signal–strength heuristics will cause the two mobile nodes to attempt to use each other as Foreign Agents, causing a "routing loop" that prevents the Mobile Nodes from establishing long–term Internet connectivity.

To address this problem, we added a new optional tag–length–value extension to the ICMP Router Advertisement messages broadcast by Foreign Agents. This TLV contains an integer value that indicates the Foreign Agent's distance from the wire in hops. Zero or the absence of the TLV indicates the FA is wired. Thus, our modifications should interoperate with existing Mobile IP implementations that may or may not use our extension. (In theory, MIP implementations that do not use our extension should ignore it because the tag is in the range of Mobile–IP tags that may be unknown but should be ignored.) Mobile routers must have a specific configuration switch in their configuration file to cause them to even include the extension.

The protocol extension is extremely simple. Mobile routers (as FAs) send out the "hops" extension and calculate it by sending out a value that is one more than the smallest value they can hear. If they cannot hear any Foreign Agent, the value gallops to infinity (255) via a timer based mechanism. Thus a FA that is by itself "goes bad" relatively quickly (in seconds). This is a very simple vector distance protocol. The Mobile Node side (be it normal non–router MN or mobile router) uses the information as a first cut on the signal–strength based heuristic. We only keep the set of agents sorted according to signal strength where that set has the same "hops–to–wire" value. This system is not perfect but it does allow Mobile routers to be sloppily deployed so that agent cells can overlap. A con is that a normal MN that hears both a "mobile router" and a wired infrastructure FA may very well choose the wired FA over the wireless FA, even though the wireless system might be a better choice. On the pro side, we can deploy a daisy–chain of Mobile Routers so that we do not have to worry about whether or not they will point at each other as Mobile

Nodes. The picture below shows what is possible:

```
------------ wired FA  (hops = 0)-------------------
               |
               |
               |
             wireless FA #1  (hops = 1)
               |
               |
               |
             wireless FA #2  (hops = 2)
               |
               |
               |
             wireless MN hooked up to to FA #2
```

Although we have not measured the latency of such a system (from the MN point of view), we should do so.

Redundancy – Multi–Hop Ad Hoc Routing

In this section we will provide a short discussion of our proposed multi–hop ad hoc routing protocol. We will first provide an overview, a few details, and some possible experimental goals.

Protocol Overview

The protocol establishes routing from one MN to another MN or to an agent when the MN is not available locally. It will be an on–demand source–route based protocol. Roughly when MN X wishes to find end system Z, it may establish a short–term path from X - Y - Z, where all systems are routers. The routing system is end host oriented because we want it to be subnet free. X will send a multicast–based request out and intermediate systems will read that request, subject it to a (anti-) flooding filter, and if the request survives, forward the request until it wears itself out (due to too many hops) or until it arrives at the desired destination. Source requests are made only on demand; i.e., when a system actually wants to talk to another system. The request mechanism may be viewed as a expanding multicast ring search. Agents always answer the request as a possible destination. Thus an agent is essentially advertising itself as a possible default route. Each request (or multicast reply) builds up a variable length hop tuple that consists of (IP address, metrics) as the source request goes from start to finish. Forwarding nodes must setup routes pointing backward as the request goes through them. The reply packet must also cause routes to be setup to the destination as it goes back to the sender.

The protocol should be viewed as consisting of requests and replies which if multicast are very similar to the request.

Replies may be multicast or unicast. We will make this a configuration option. It might make sense for agents to only unicast their existence (although multicast of agent availability may also be useful - usage situations may differ.) Multicast replies (like requests) cause the forwarding node to append its own IP hop tuple to the payload of the packet. Unicast replies are simply returned hop by hop in a unicast fashion from the destination (or agent). The source path is frozen in a unicast reply and represents sender to destination but rebuilt from scratch if multicast and may find more than one path back. Multicast replies hence consume more resources but at the same time are more adaptive. In fact, multicast replies have one very interesting property and that is that a multicast reply essentially informs the mesh of the "answer" (destination). Thus nodes might learn about a shared DNS server or agent (or command node). It may be useful to think more how this quality might be exploited.

The protocol is source route based as intermediate systems that choose to forward a packet will append a IP "hop tuple" that consists of (IP address, metric) to the packet. Thus a source route is built up. Each system that gets such a packet must set a route back to the sender (or multicast destination for replies) and may also set a route back to any intermediate system in the path. Routes already contained in the routing table should be refreshed. Routes timeout according to sender supplied values and/or forwarding system parameters, whichever are smaller. This might help to accommodate motion as for example, a truly mobile node would have to keep state based on a smaller timeout set by any sender or forwarding system (if it is moving).

How flooding is handled is very important. The true essence of such a protocol is that it is flooding–oriented and one would like to predict that it would improve the odds of packets getting through in a mobile mesh (as opposed to traditional routing protocols that try to achieve convergence first and simply use flooding as a tool to achieve that end). Flooding attributes include:

1. Requests that arrive at agents are not forwarded onto the wired infrastructure. We expect only mobile mesh systems sharing the same kind of link to use this system. However a agent may forward the packet to the mobile mesh itself (on the same kind of link).

2. Any multicast request has a id value set by the sender. Forwarding systems will cache the packet and not forward it if they have seen the id previously and the id itself has not timed out. On the other hand, forwarding systems MIGHT use the built–in route metrics to determine the best path and ignore the id. The latter notion is more costly again in terms of mesh bandwidth but might improve redundancy.

3. Forwarding systems will subject the packet to a loop test. If they see their own IP address in the packet, they will drop it.

4. The sending system will set a high–level (application, not IP) TTL. If the TTL is exceeded, the packet will be dropped. This is not the IP TTL which will be set by any forwarding or originating sender to 1 to prevent any possible multicast routing of the packet.

5. Packets that fail authentication are dropped.

The protocol will be authenticated, and will use shared secret keys in a manner similar to the first ad hoc routing protocol. It will carry over the major attributes from that protocol in that it will be subnet–free and authenticate the MAC, IP address pair of the sender. In our first cut, we will actually build the protocol on top of the current ad hoc protocol; i.e., the current ad hoc protocol can be viewed as establishing the link layer and the next generation establishes multi–hop connectivity. However the protocols can be combined since the multi–hop protocol can carry a MAC address over the first hop. Thus we can get to a complete demand driver system for link–layer establishment. In addition, we will carry over our current ad hoc authentication for now, but later on we will also try and make use of AH authentication tied in to the multicast addresses (it is not clear that our current NRL code supports that, but it should not be hard to make it work).

The metric part of the packet tuple will consist of a composite metric: (hop count, signal strength (or snr), and power). To begin with, we will only apply the metric at the endpoints of the source route path; i.e., forwarding agents will simply toss multiply received requests (same sender ID). By endpoint, we mean the destination (or sender on the multicast return route). The composite metric formula is:

$m = (PV * PS) + (SV * SS) + (HV * HS)$

where P stands for POWER, S for SNR, and H for HOP COUNT. The PV, SV, and HV values are variables that will arbitrarily range from 0..100 where 0 is worst and 100 is best. The PS, SS, and HS are weights (scalars) and are chosen at mesh configuration time. PS, SS, HS must add up to some integral value (say 1) and m itself will be scaled to 0..100. Thus one can choose to emphasize hop count, signal strength, or power according to various hardware attributes of the mesh itself. Forwarding systems will store the signal strength as recorded from the last link and their own current power value in the tuple as they forward it. Hop count can be determined by counting the number of tuples. It should be obvious that this composite metric is oriented towards mobile systems.

Implementation Details

The protocol will use a multicast IP address in the range 224.0.0.1 to 224.0.0.255; i.e., it is not routable in MBONE terms. It will use UDP. Mobile routing daemons will send and receive multicast packets (and optionally unicast packets sent to the IP address indicated in the hop path tuple). It should be pointed out that routing table entries are setup by this protocol and normal traffic will hence be carried by normal forwarding mechanisms until timers remove routes.

One important question that we have finally settled in the last quarter is how this integrates with our current routing daemons and kernel routing mechanism. We need a mechanism that can drive the source routing algorithm in the sender. We want said mechanism to not be dependent on the state of the routing table but to be driven by routing misses; i.e., the lack of specific host routes in the routing table. We also want the mechanism to be kernel initiated but inform an application daemon so that policy and mechanism can be cleanly separated. We do not want the mechanism to be affected by whether or not a default route is present as this would require a potentially hazardous and painful rewrite of the current code in our mobile routing daemons (which is complex enough). Currently a default route is present in Mobile Nodes if they can hear an acceptable agent directly (agent beacons). It is not present if they cannot hear an agent. We need to tolerate both sets of circumstances. There are also redundancy arguments for keeping a default route if available (since it gives you a default path where a node might be found).

In order to satisfy these requirements we have determined that the pre–existing BSD routing attribute called the XRESOLVE switch can be used. As a result, we can use the route(4) socket together with the pre–existing raw ICMP socket in our daemons to look for certain messages either from the network or the kernel which can drive the source route lookup. We had to make a small modification to the IP_output function so that XRESOLVE would also apply to raw socket packets (for example, ping). We do not understand why raw sockets would not cause XRESOLVE to work, as it was originally intended as a routing upcall so that a routing daemon could somehow externally resolve a non–ARP link address (X.25).

There are three possible events that can drive the source routing:

1. if the default route is present, we tag it with the XRESOLVE flag. When IP packets are sent to a new host IP destination, that has not previously been present in the routing table, a route(4) upcall is made to the routing daemons informing them that a cloned default host route to X was created. Normally such a route might be used for TCP metrics or for PATH_MTU state. The upcall will be used to drive the source route request. If source routing fails, packets will still be routed to the default route (which might be correct although it may or may not have an optimal path). Thus there is an important redundancy aspect here since our goal is to try and get packets through end to end even if the path is not optimal.

2. if no default route is present, the kernel will generate a routing MISS message on the route socket. We can of course use this to initiate a search.

3. In addition, we may receive an ICMP_UNREACHABLE message. We can use this to try the source routing again, although we might build in a back off timer on such retries. Note that the unreachable message is by definition not generated by multicast since one does not return ICMP errors for multicast

messages. However it may at any time be generated by a path failure and should be regarded as a potentially useful input.

Experimental Goals

1. Obviously the protocol should work and not immobilize the local mesh with a broadcast or multicast storm. We need to observe local operation and simply make sure that the protocol does indeed operate as intended. It should be noted that we do not have enough computers to make wide scale testing in terms of even 10's of units. We will have to do the best we can with a handful of machines.

2. We need to make some attempt to quantify at least the following:

   (a) how much traffic overhead is induced?

   (b) can we determine any measure of "packet success" in terms of survival as opposed to loss?

   (c) how far can we actually get in terms of nodes away from the wired infrastructure and still have decent performance?

3. multicast versus unicast replies. We need to ask ourselves whether or not the multicast mechanism's cost is worth the overhead as opposed to the less–costly but less redundant unicast reply. As part of this, we must ask if the multicast return of a given destination to nodes that didn't even ask the question may be useful. Certainly it would seem a priori that a trait whereby many nodes learn and share a DNS or local web server could be very useful. Nodes could cache information that they might use later.

4. effectiveness of metrics. We have access to signal strength information currently in our WaveLAN drivers (although we still need to deploy the signal strength code in the desktop platform (agents)). We do not have means to actually gauge power strength but we can configure values in by hand and test it. We intend to make a small trial of a number of signal strength and power metrics.

5. motion and loss of intermediate systems. We certainly intend to setup paths and then eliminate forwarding systems "in between" the sender and destination. Hopefully this will give us some feel about how the algorithm will react in a truly mobile environment. In the final analysis, this is certainly an area that could stand simulation. We do not presently have the means for such a study.

Home Agent Redundancy

The Home Agent Redundancy system and protocol is still not completely specified as we are conducting some simple prototyping experiments. However we can tentatively outline how the protocol itself will work and outline a few requirements. When we have this system working, we will write an Internet draft on the protocol.

The protocol itself is called HARP, which stands for "Home Agent Redundancy Protocol." HARP will be layered on top of ordinary unicast routing and as a protocol will be very simple. In general two Home Agents will be designated as HARP peers (yes, they will be called "harpies") and will send HARP packets over both UDP and TCP. For a change, we can certainly use IPSEC/socket mechanisms (authentication in particular) as these are simply ordinary unicast sockets.

System requirements of interest include:

1. IPSEC (which is easy to do) for all protocol messages.

2. the harpy peers will present a consistent virtual mobile IP subnet to the world; i.e., they advertise one subnet to the interior routing domain. This will be discussed below in more detail.

3. any RFC 2002 compliant Mobile Node will not have to know about HARP. Thus agents take part, but mobile nodes do not need software changes. One consequence of this is that a Mobile Nodes' Home Agent can be changed, but the MN itself will not need to know that the change occurred. The MN need be configured with only one HA IP address.

4. both harpy systems must maintain routing tunnels (say IPIP from HA to MN) in parallel. When MNs are at home, one harpy will tunnel to the other HA where the MN is actually resident.

HARP consists of three kinds of messages that will be exchanged over a UDP socket. We expect that peered HARP routers will not be on the same network since one would not want them both to disappear due to the loss of a shared network card in a single router, but they are not likely to be far apart either. Exterior routing protocols cannot support the idea of a subnet being in two directions. It is likely that the internal path between them will be better than the external "to the MN" (or from the CH) path. As a result (unlike say with BGP), we feel that we can get by with mostly stateless servers and let the MNs send registration messages to one harpy, which in turn will forward the message to the other system. The MNs must retry with a certain periodicity since a HA might reboot at any time. As a result, the MNs will essentially keep the HA state "fresh" and the protocol (or server implementations) itself need not overly concern itself too much with redundancy. Redundancy boils down to having two HAs in two different places.

On the other hand, as an optimization, we will allow one HA that has rebooted to use TCP to read out the entire MN routing table from the other HA. Thus the protocol in some respects will be somewhat like DNS (use TCP to exchange tables, and UDP to exchange messages). We agonized over whether to use TCP in all cases (more BGP like), but decided that we simply could not trust the semantics of TCP when routing connections fail or backup due to buffer latency. We do not want any given MIP agent to hang on an internal write because TCP cannot find room in the peer receiving window. The surviving agent should still be able to process all messages with a rough but constant fairness. If an interior network path fails, we expect dynamic routing to send all packets to the remaining peer. We don't want it to fail because its peer failed. We could mitigate this TCP worry by always opening a connection, writing a message, and closing the connection, but this could lead to too much overhead. We suspect that taking advantage of the MIP stateless property here will lead to making HARP simple and more fault tolerant as well. It may also make it a bit more extensible, as possibly a unicast UDP protocol could be easily replaced with a reliable multicast UDP protocol.

UDP–based:

1. hello messages. When one harpy has not heard from the other for awhile they will exchange simple UDP unicast "hello" packets. If the other harpy is deemed not available due to the failure of N out of Y tries, a message will be logged, but it will not stop forwarding MN requests. Eventually more aggressive TBD "out of band" mechanisms might be implemented in order to notify local network managers that a serious internal routing failure has occurred.

2. MIP registration forwarding: Any MN registration message received by one peer will be encapsulated in a very small wrapper and forwarded to the other harpy, where it will be processed "normally" (routes may be setup to either a COA or peer HA), but no MIP ACK will be returned by the HARP receiver peer. The system that initially receives MIP messages is responsible for ACKs and NAKs and will not forward NAK'ed registrations.

3. upon boot, a HARP peer will know its peer due to static configuration. It will initiate a TCP connection to its peer and attempt to exchange in bulk the other's MN routing table. Thus one peer can learn the entire routing table of the other. If two systems reboot at the same time, there will be nothing to learn. HAs minimally should have UPNs. One might think about stable storage at some point.

The basic HARP topology

The two most important topological aspects of HARP are that we intend to interoperate with Mobile–IP MNs "as is" and in addition we will take advantage of normal IP unicast interior gateway protocol routing (e.g., OSPF or RIP). The trick is that we expect dynamic routing to find a path to the "nearest" HA since both peered HAs advertise a path to the "same" subnet. If one peer is lost, we expect dynamic routing to send all packets to the other peer. (Note that the mobile subnet may be partitioned and hence "virtual"). Thus a Mobile–IP registration request and a normal packet sent from a Correspondent Host will likely follow the same path to one or the other HA. Either packet may at any time go to either HA. Each paired HA must tell the other where the MN has gone, and they both must track the MN in terms of routing state. A MN will really have "two" homes with two HAs (and there is nothing to limit the protocol to two, but two will suffice for our purposes.) For the HAs to hide their differences from the MN, both must share the same "virtual subnet" IP address. They cannot communicate with each other as a result with that address, but we expect such systems to be multi–homed by definition. One "real" IP address must be used at each HA, paired with one virtual IP subnet address that is shared and attached (possibly as an alias) to an interface. This is a small price to pay for MN interoperability.

Our lab topology will be something like the following: note that both HAs are multi–homed systems and each has both a WaveLAN and ethernet card.

```
Cisco router ---------- test router (FreeBSD)
                         |
                         |
                    | hub  |  interior ethernet subnet Z
                    |      |
                HA #1      HA #2
                !          !
                wavelan subnet  X
```

The Cisco router routes mobile subnet X to the test router. The test route statically routes that subnet to either HA #1 or HA #2 in terms of their respective IP addresses for interior ethernet subnet Z. This gives us a way to simulate dynamic unicast routing as we can simply change the route to the mobile subnet by hand. The Wavelan interfaces will be configured with both the same (and different) IP addresses. From a practical standpoint, the HA WaveLAN cells should not overlap since the use of the same IP address (Home is the same) will cause confusion to the MNs when they are actually resident in both cells. We can either only really have one home (by not sending HA beacons from one system) or configure the WaveLAN Network IDs differently so that HA beacons from both systems cannot simultaneously be overheard.

Outreach

Bill Trost has visited BBN for a few days to discuss issues of porting their code into our Mobile–IP/IPSEC system.

David Reeder has attended recent IETF and the recent FTP Inc. Mobile–IP InterOP.

Jim Binkley and John M^{c}Hugh have given recent talks of various lengths at the Oregon Graduate Institute, at the Baltimore DARPA PI meeting, and the GLOMO PI meeting on the UCLA campus.

A Appendix - IPSEC, Mobile–IP and Policy

John M^{c}Hugh has recently mailed out a short treatise entitled "Keys, Chickens, and Eggs" in which in the final section he discussed capital P 'Policy' vs little p 'policy; i.e., the desired attributes of a security system versus how the attributes were actually implemented. The point needs to be reiterated that it would be useful to actually think aloud and possibly talk to industry, firewall vendors, and to IPSEC people, about what kind of Policy might make sense in terms of Mobile–IP. Once we know, we could try and implement it. Up until now, we have addressed a very limited range of topologies, but as we begin to explore "dynamic key" issues, it would be nice to try and determine what actually might be useful and/or feasible. We shall attempt a short discussion here in order to begin to explore some of the issues. This discussion will not be complete, but it is a start. Hopefully it might fuel some high–level discussion about what needs to be done and how such Policies might be accomplished.

As a word of caution, the issues for security and Mobile–IP are complex and they are changing . Mobile–IP itself is a moving target as recently the subject of a "reverse tunnel" protocol setup by MNs where the local FA tunnels back to the HA has come up in the IETF Mobile–IP working group. As is often the case, it seems little attention is paid to security issues, even though an entirely new topological dimension has been revealed.

For the sake of argument, let us establish first some topological notions. We will try to bind our Policies according to the topologies. In conclusion, we will try to summarize any lessons learned.

Mobile–IP topologies of interest might include:

1. MNs/HAs.
   - .at home
     - on a wireless system (WaveLAN)
     - on a wired system (ethernet)
   - .away
     - via a FA
     - via a COA (but no FA)
2. MNs/FAs It is possible that a FA might have a pool of COAs. This might be useful although Mobile–IP currently does not address it as an possibility.

3. MNs/COA Here an MN acquires a COA as a local–link IP address. The IP address may be acquired via DHCP or some other mechanism including PPP and manual administration.

4. FAs/HAs

5. ad hoc MNs

6. route optimization (MNs talking directly to CHs).

There are two kinds of protocol considerations that are fundamental. One can fundamentally distinguish between "control" packets (routing including link–layer mechanisms like ARP), and "data" packets that can be transmitted after routing is established. Routing must be setup first. In certain topologies (barring ad hoc), this is what Mobile–IP tries to do. This is of course, the chicken and egg problem that BBN has referred to elsewhere. Without routing, data cannot be sent.

In the discussion below, we will mostly neglect any Policy that says "you may not do this," and always neglect Policies of the form "we don't care if you do this." We will also neglect the likely useful area of IPSEC/Firewall policy combinations.

A.1 MNs/HAs

A.1.1 At home

Policies:

1. A wireless link may be felt to be less secure than a wired link. As a result, one might require wireless security along the usual lines with the usual possibilities: encryption, authentication, session–keys.

2. Ideally such security should encompass both routing and normal data.

3. For reasons of scalability, one may very well require dynamic key management (as opposed to static). If all employees at General Motors have laptops, that will be a lot of keys. (There is not much difference between /etc/keys and /etc/hosts from the scalability POV.)

4. One may view a Mobile system as a threat simply because it can get up and walk away, possibly with valuable information. Hence it may be subject to any number of special restrictions even when at home. (can only use subnet X, and must use Mobile–IP authentication to even get routability).

5. One may have policies for the HA as a router that limits ingress to interior non–mobile networks.

Observations: Each relationship (like any other network conversation) is symmetrical. Here we have MN to HA and HA to MN. As John pointed out, in some cases, one of the sub–relationships may be less important. In this case, the HA is probably more important simply because it represents an entrance to the secure enclave. (This is of course, even more important when we are talking about routing to and from the HA when an MN is abroad).

One might view the MN at home as being within the secure enclave, unless one is worried about uncontrolled wireless exposure or uncontrolled taps into local links. The MN at home is not very different from conventional systems at home. ARP is a security exposure anywhere, although it may be deemed adequate for links like ethernet.

Certainly small–scale static key distribution for both control/data is possible. But if dynamic key distribution will occur anywhere, surely it can occur here.

Away

Policies:

1. Mobile–IP may be banned. Of course, such a Policy may be viewed as not interesting but it is of fundamental importance. Until recently firewalls existed to keep spoofers out and one form of spoofing is borrowing IP addresses that should be "inside." Therefore Mobile–IP can be viewed as institutionally (IETF) approved spoofing. Certainly many organizations will never permit it either for their employees to get back in or for "visitors" from abroad. We will not refer to such a negative policy again, but it is a fundamental challenge for IPSEC to see what inroads if any can (or should) be made about such world views.

2. One may require the MN to authenticate its control packets to the HA (and vice versa).

3. One may require the MN to secure all of its packets including data and either:

    (a) send packets to home systems securely but not care about packets sent to CH systems not at home

    (b) send all packets home first so that the MN appears to originate to CHs from home. The enclave itself may choose to not allow CHs to forward packets to MNs to suppress proposed plaintext attacks through the HA.

    (c) only talk to systems at home.

    (d) never talk to systems at home.

    Note that again "secure" can be extended to include authentication, encryption, and session keys.

4. One may require the MN to NOT form any sort of security relationship other than with Home. For example, MNs may use FAs but may not form SAs with them, and must not have the FA forward packets into an enclave (reverse tunnel). Why, because we choose to not trust non–local FAs.

5. One may require the MN to hide any IP addresses that are native to "home" since exposure of such address topology may lead to external attacks.

Observations:

We are talking about the MN and the HA when the MN is not present on the HA's subnet. The MN/HA relationship is potentially useful in many ways, simply because the MN can pre–establish a security relationship with the HA before it wanders abroad. We can also view a MN that is abroad as creating a possible extension of a local secure enclave. Another variation on this theme is that the MN and HA by definition belong to one adminstrative/security domain.

As we have pointed out previously, ARP spoofing could be viewed alone as a denial of service attack, but if coupled with a previous acquisition of any home IP address or telnet/FTP password sent in the clear, it may become a very potent means of attack on the home enclave. As a side issue, note that not giving out IP addresses for home systems (which our current 2–was DES tunnel scheme implements) is an important attribute as well.

IPSEC as a mechanism here is very important. We also need to be able to dynamically setup "webs of trust" that include HA–FA–MN pairings.

Specification of routing policy based on IP addresses that leads to flexible routing could be useful. One might want to talk to CHs directly and avoid home OR send everything home, etc.

A.2 MNs/FAs

policies:

1. One may rightfully worry about wireless links again, in that they may be deemed less secure. If a network security officer cannot control wireless links at home, he/she can certainly not control foreign links. We may hence require link security when abroad and might also ban ARP. Foreign links can be viewed as much more dangerous.

2. A reasonable rationale for the existence of FAs is to charge MNs as customers. As a result, a likely policy here is that any MN will have to use a very good cryptographic means to prove its identity. Dynamic negotiation of trust (for identity as well as usage) would be a requirement.

3. Local security may require that FAs form a closed trust system (no outside MNs may use them).

4. Local security may require that FAs be outside of any secure enclave (and hence there may be no need for security relationships).

5. Certainly the home security organization might desire that MNs reveal as little as possible about themselves to a FA. This might include a requirement that the MN is NOT to form a SA with a FA.

6. FAs detunnel IPIP packets. One may require a SA between the tunnel endpoints.

Observations: It is not clear in how many ways an MN and FA might combine (or not) in security terms. It is clear that it runs the gamut from complete and dynamic security both for routing control and data (wireless link) to nothing at all. In our current Mobile–IP/IPSEC system we can run all packets to home with IPSEC and simply use a FA as a router. We could currently setup our FAs to form a "closed" trust system since we have implemented MN/FA static keys. By definition, this would exclude MNs from other domains. However we currently have no means to dynamically negotiate anything between MNs and FAs.

Note that the relationship could be assymetrical. The MN might not care so much about whether or not the FA can prove itself, since the MN is very much at the FA's mercy. If the FA will ship packets out and in, that may do from the MN's point of view. The FA however might care greatly since it may represent a portal to a "secure enclave" or might be interested in economic transactions.

On the other hand, the truly paranoid MN may be concerned about the FA since the FA is certainly a "man in the middle" and might do funny things to the MN's packets. The MN may be able to take advantage of its permanent trust relationship with the HA and use the HA to determine the FA's trustworthiness. Is "Joe Fa" really a FA? Possibly such a determination could be used to setup 2–way trust between both the MN–FA and FA–HA.

A.3 MNs/COAs

policies:

1. an MN might be required to ONLY use a COA ip address on outer IP packets sent home so that it can tunnel packets through a border router. Thus even the MN's IP address could be hidden from view.

2. The MN may be required to setup a SA based on the COA with the HA. This would prevent external non–secured IP addresses from entering a secure enclave. Such a Policy could extend to both control and data.

Observations: An MN may acquire a COA via some TBD mechanism (DHCP). It might even do so at a FA. Some of the previous Policies apply here (so we left them out). There is however a new wrinkle, and that is that the MN has acquired an IP address with local significance. This is a two–edged sword. It may be useful in that an MN with a local COA can use it to bypass an ingress–filter. On the other hand, one could view such a COA mechanism as a security liability.

A.4 HAs/FAs

policies:

1. One may require either a static or dynamic trust negotiation for at least routing packets (MIP). Dynamic keying would be needed for FAs not under local control.

2. IPIP packets may not be permitted. One might require IP IPSEC IP or IP IP IPSEC where there is some way to bind at least one of the local headers to the box "at home" that performs the IPSEC operation.

observations: We feel the most important point to make here is that IPIP can be viewed as a liability especially across security domains as it is basically a mechanism for poking an IP datagram anywhere (and shedding the IP outer header or skin that got the datagram to the tunnel endpoint). One could use conventional firewalls to protect internal "detunneling" devices that lack protection of their own. On the other hand, IPSEC is an obvious mechanism for making IPIP safer. Further one needs dynamic key distribution to enable IP IPSEC IP as a mechanism for boxes not in the same security domains.

A.5 ad hoc MNs

policies:

1. MNs should authenticate routing packets.

2. MNs should secure control information.

3. MNs should only talk to their own kind and maybe only when they see their own kind.

observations: The first two policies above are nothing new and other policies previously mentioned can certainly apply (e.g., concern about wireless links abroad).

It is important to note that MNs in an ad hoc situation may be worse off than an MN at a FA, simply because they can be cut off from the wired infrastructure. Both DNS and any existing Certificate Authority may not be available. (The lack of DNS certainly stops some not very bright applications from even running).

A.6 Route Optimization

policies:

1. We may simply disallow it as we do not want our MNs to talk to CHs except through home. Of course we may not want them to talk to non–home CHs either.

2. control packets should be authenticated.

3. previous concerns about tunnels apply as well.

observations: One can point out that routing optimization is an interesting name for a noble attempt to make MNs behave like ordinary hosts; i.e., avoid the routing triangle problem. However from a security point of view, one might not only like CH packets to go through the HA (you could log who the MNs are talking to), but worse you may want all of your traffic from the MNs to come back home first as well. Hence one would have a double triangle (MN - HA - CH and CH - HA - MN) and be happy about it.

Routing optimization is a form of source routing (as is any use of IPIP tunnels) and one might think that minimally authentication for tunneled packets is a requirement.

A.7 Summary

1. MNs in ad hoc situations and at FAs are at a disadvantage because they either lack connectivity or must do it through a man in the middle. How do you get a certificate from the DNS if there is no DNS? You may have to bring something with you to either minimize exchanges due to lack of bandwidth or fast motion, OR you may simply not have access to the wired infrastructure in an ad hoc situation OR you may have to establish a "face–to–face" web of trust.

2. MNs can take advantage of the a priori security relationship with the HA. We agree with BBN that a MN need not ship a certificate to the HA to prove itself since it can do something more lightweight and/or prepare a bit before it leaves. The MN can also ask the HA about a FA since the MIP protocol itself is extensible. Some sort of "in–line keying" could be done there in the MIP protocol. A FA might ask a HA about an MN (as well as about the HA itself) as we can presume that the FA/HA bandwidth might be greater than the MN/FA bandwidth.

3. Tunnels need to be secured by IPSEC and be setup via some dynamic exchange of key material since security domains may be crossed. Note that Home Agents may detunnel packets as well as Foreign Agents.

4. If Policy dictates that FAs need keys, FAs need more than manual key mechanisms, although such a mechanism may have limited utility in a small network

infrastructure. FAs need to dynamically negotiate with HAs in order to setup tunnels and to make sure that all packets coming through a tunnel are secure. This is an absolute MUST. FAs MAY need to negotiate with MNs where a security policy dictates that MNs must authenticate themselves and MUST negotiate with MNs wherever FAs might charge for their services.

5. One must utter the "scalability" word in terms of manual keying. This may be deemed obvious, but one will have more FAs than HAs, and if you use manual keying with FAs, you must add each new FA key to all MNs and teach that FA about all MNs. This is less scalable than the MN–HA relationship. Manual key distribution may also be hard to automate. It can be done where FAs are concerned since we assume they are reachable. But given that laptops may be off or away, the laptop side is very hard to automate in practical terms.

6. In an environment where an MN acquires and uses a COA, the neighborhood local router might choose to impose security policies on the MN. The kinds of policies between such a router and a FA may not differ very much.

7. This last set of router policy considerations may be extended to filtering border routers and might require some TBD protocol that can discover the number of policy–oriented routers between here and there (and back). Such a system begins to sound similar to other problems facing the Internet (e.g., RSVP setup management, or nested tunnels and "soft state"). We need to consider the dangers of creating a system that requires "hard state" end to end, so that if dynamic routing must occur, a "connection" will be lost. Such a system would not be very much like the current Internet and might be very hard if not impossible to debug.