IKEv2 Mobility and Multihoming (mobike)
Internet-Draft
Expires: December 23, 2004

T. Kivinen
Safenet, Inc.

June 24, 2004

Design of the MOBIKE protocol
draft-ietf-mobike-design-00.txt

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on December 23, 2004.

Copyright Notice

Copyright (C) The Internet Society (2004). All Rights Reserved.

Abstract

This document discusses the potential design decisions in the base MOBIKE (IKEv2 Mobility and Multihoming) protocol. It also tries to provide some background information about different choices and tries to record the decisions made by the working group, so that we do not need to repeat discussion later.

Table of Contents

1. Introduction

The current IKEv2 and IPsec documents explicitly say that the IPsec and IKE SAs created implicitly between the IP-addresses used in the IKEv2 SA. This means that there is only one IP-address pair attached for the IKEv2 SA, and the only one IP-address pair used as a gateway endpoint address for tunnel mode IPsec SAs. Also after the SA is created there is no way to change those addresses.

There are scenarios which require that the IP address might change rapidly. In some cases the problem could be solved by rekeying all the IPsec and IKE SAs after the IP-address has changed. In some scenarios this might be problematic, as the device might be too slow to rekey the SAs that often, and other scenarios the rekeying and required IKEv2 authentication might require user interaction (SecurID cards etc). Due to these reasons, a mechanism to update the IP-addresses tied to the IPsec and IKEv2 SAs is needed.

The charter of the MOBIKE working group requires IKEv2, and as IKEv2 assumes that the RFC2401bis architecture is used, all protocols developed will use both IKEv2 and RFC2401bis (issue #9). No effort is to be made to make protocols for IKEv1 or old RFC2401 architecture.

MOBIKE protocol provides solution to the problem of the updating the IP-addresses. The MOBIKE protocol should take care following:

- Notifying the other end of IP-address(es) change
- Update the IKE SA endpoint addresses based on the notifications
- Switching to use new IP-address if old one does not work anymore
- Updating the tunnel mode IPsec SA tunnel endpoint addresses
- Ensuring that the given new addresses belong to the peer

The MOBIKE protocol can be used in different scenarios. Two such scenarios are discussed below.

1.1 Roaming Laptop Scenario

In the roaming laptop scenario the device that moves around is laptop, which might have several ways to connect to internet. It might for example have fixed ethernet, WLAN and GPRS access to net, and some of those can be used in different times. It tries to use the most efficient connection it has all the time, but that connection might change. For example user can disconnect himself from the fixed ethernet, and then use the office WLAN, and then later leave the office and start using GPRS during the trip to home. In home he might again use again WLAN (but with different IP-addresses) etc.

The device does not use Mobile IP or anything similar, it simply wants to keep the VPN connection to the corporate security gateway

(SGW) up and running all the time. Even if the interface or the IP-addresses change, the internal addresses used inside the IPsec tunnel remains same (allocated from the SGW), i.e. the applications might not detect the changes at all.

1.2 Multihoming SGW Scenario

Another possible scenario which might use MOBIKE is the SGW of the other end of the roaming laptop scenario. The SGW might have multiple interfaces to different ISPs, and wants to provide connection even when some of those connections are broken. One of the interface might also be the WLAN access point in the office. The SGW will know beforehand what set of IP-addresses it will use, but it might need to dynamically send update notifications the clients to tell them which addresses to use. It might also use this to do some sort of load balancing, i.e. giving different clients different preferred address, to utilize all the connections. This kind of load balancing is completely internal to the SGW (i.e. the clients will simply see that the preferred IP-address to be used for tunnel endpoint changes, but they do not know why or how the SGW decided to do that), and the actual algorithms how to do that is outside the scope of MOBIKE protocol (i.e. the whole issues is that MOBIKE does not disallow the SGW to give different sets of IP-addresses in different preference order to different clients).

Note, that the load-balancing inside the one IKE SA (i.e. one client) is not handled in the MOBIKE protocol. Each client uses only one of the IP-addresses given by the SGW at one time.

2. Issues

The base protocol needs to perform the following things:

- Ability to inform the peer about the current or changed address(es) of the sender
- Ability to inform the peer about the preferred address
- Ability to detect an outage situation and fall back to the use of another address
- Ability to prevent flooding attacks based on announcing someone else's address
- Ability to affect both the IKE and IPsec SAs

One of the key issues affecting the MOBIKE protocol is, whether MOBIKE protocol needs to recover from the case where packets simply dont get through. If the node can locally detect some problems with the interfaces (IP-address change, interface disappearing, link going down), it can act based on that and fix the situation. If the packets are simply disappearing somewhere in the net, the detection of the problem requires noticing that we cannot get packets through. If the protocol only need to fix problems appearing in the local interfaces, then the protocol is much simpler.

3. Adopting a new address / multihoming support

From the MOBIKE's point of view the multihoming support is the set of rules how and when to change to use new IP-address for the other end.

3.1 IP-address list or one IP-address

One option is that the other end can provide a list of addresses which can be used as destination addresses, and the local end needs to decide which of them to use. The MOBIKE does not include load-balancing, i.e. the local end only uses one IP-address at time, and it only changes to use new IP-address after some kind of indication.

Another option is to only communicate one address for each end, and both ends only use that address when communicating. When the something changes, the end whose situation changes, sends update notification to the other end, changing that one address.

If the other end provides the full list of possible IP-addresses, then the other end can recover from the movements on its own, meaning that when it detects it cannot get packets through it can try another IP-address. If the other end only provides one IP-address to be used, then the other end has to wait for the new IP-address before the situation is fixed. The good thing about only one IP-address for the remote host is that it makes retransmission easy, and it also makes it clear which end should do the recovery (i.e. the end, whose IP-address changed, MUST start recovery process and send the new IP-address to the other end).

The one IP-address approach will not work if both ends happen to loose their IP-address at the same time (routing problems, which causes the one link between the hosts to go down, thus either end cannot get recovery packets through as the link is down). It also may cause the requirement for the IKEv2 window size larger than 1, especially if only direct indications are used. This is because the host needs to be able to send the IP-address change notifications before it can switch to another address, and depending on the return routability checks, retransmissions policies etc, it might be hard to make the protocol such that it works with window size of 1 too (issue #11). Also one IP-address approach does not really benefit much from the indirect indications as the end getting those indirect indications cannot often fix the situation by itself (i.e. even if the host gets ICMP host unreachable for the old IP-address, it cannot try other IP-addresses, as it does not know them).

The problems with IP-address list are mostly in its complexity. Notification and recovery processes are more complex, as both end can

recover from the IP-address changes. There is also possibilities that both ends tries to recover at the same time and this must be taken care in the protocol.

3.2 Indirect or direct indication (issue #1)

The indication that the situation regarding the IP-address has

changed might be either direct or indirect. The direct indication means that the other end will send specific indication that now something changed. The indirect indication is something which can be observed from infrastructure or lack of packets, not directly from the other end.

The direct indication can be for example the other end IKEv2 sending authenticated address update notification, which have different IP-address(es) than used earlier.

The indirect indication can be many things. One example might be that the local end notices that suddenly the other ends start using different source address for the packets than what it used before, or ICMP message or routing information change.

Another type of indirect information might that there has been no traffic from the other end for some time (i.e. the current connection might be broken).

This kind of indirect information should not directly cause any changes to the IP-addresses, but they should be used as indication that there might be need to do dead-peer-detection for the currently used address. I.e. when the local end detects that the other end started to use different source IP-address than which was used before, it should initiate dead-peer-detection for the address currently in use. If that dead-peer-detection tells that the connection is alive, then there is no need to do anything. If local end does not receive any reply to the dead-peer-detection, then it should do dead-peer-detection for the other addresses in the list (if available, in the preferred order). If it can find an address which works, it will switch to that.

3.3 Dead peer detection and IKEv2 (issue #11)

The IKEv2 dead-peer-detection is done by sending empty informational exchange packet to the other end, in which case the other end will acknowledge that. If no acknowledge is received after certain timeout (and after couple of retransmissions), the local end should try other IP-addresses (if available). The packets to other IP-addresses should use the same message-id as the original dead-peer-detection (i.e. they are simply retransmissions of the dead-peer-detection packet

using different destination IP-address). If different message-id is used that violates the IKEv2 constraints on the mandatory ACK for each message-id, causing the IKEv2 SA to be teared down.

If the local end does not receive acknowledge message back from any of the IP-addresses, it should mark the IKE SA dead, and delete it (as mandated by the IKEv2 specification).

Note, that as IKEv2 implementations might have window size of 1, it means that while we are doing some other exchange, we cannot initiate dead-peer-detection. This means that all other exchanges should also receive identical retransmission policy than what is used for the dead-peer-detection (issue #11).

The dead-peer-detection for the other IP-addresses can also be done simultaneously, meaning that after the initial timeout of the preferred address expires, we send packets simultaneously to all other IP-addresses. The problem here is that we need to distinguish from the acknowledge packets which IP-address actually works now (i.e. we will check the acknowledge packets source IP-address, as it should match the destination IP we sent out).

Also the other end is most likely going to reply only to the first packet it receives, and that first packet might not be the most preferred IP-address. The reason the other end is only responding to the first packet it receives, is that implementations should not send retransmissions if they have just sent out identical retransmissions. This is to protect the packet multiplication problem, which can happen if some node in the network queues up packets and then send them to the destination. If destination will reply to all of them then the other end will again see multiple packets, and will reply to all of them etc.

The protocol should also be nice to the network, meaning, that when some core router link goes down, and all those MOBIKE clients notice that, they should not start sending lots of messages while trying to recover from the problem. This might be especially bad if this happens because packets are dropped because of the congested network. If MOBIKE clients will try simultaneously test all IP-addresses sending lots of packets to the net, because they lost one packet because of the congestion, it simply make problem worse.

Also note, that IKE dead-peer-detection is not sufficient for the return routability check. See Section 7 for more information.

4. Simultaneous Movements (issue #2)

We do not need to solve the simultaneous movement recovery problem, as we are not creating full mobility solution (charter forbids that), but are instead concentrating on the VPN style scenarios. In the scenarios we assume that the one end (SGW) will have fixed set of addresses (from which some subset might be in use), thus it cannot move to the address not known by the other end. This means that the solutions how to recover from cases where both ends move and the movement notifications do not reach other ends, is outside the scope of the MOBIKE WG.

Note, that if we use only one address per each end, instead of address list, we might end up in the case where it seems that both ends changed their addresses at the same time. This is something that the protocol must take care of.

There is three different cases here:

Two mobile nodes getting a new address at the same time, and then being unable to tell each other where they are. This problem is called the rendevouz problem, and is traditionally solved using home agents (Mobile IPv6) or forwarding agents (Host Identity Protocol). Essentially, solving this problem requires the existence of a stable infrastructure node somewhere. Example: roaming laptop to another roaming laptop, no SGW involved.

Simultaneous changes to addresses such that at least one of the new addresses was known by both peers before the change occurred. The primary problem in first case was not knowing the new addresses beforehand. Here we know the address so there is no problem. Example 1: two SGWs failover to another path. Example 2: roaming laptop gets a new address at the same time as its SGW's primary interface goes down.

No simultaneous changes at all.

5. Interaction with NAT-T (issue #3)

In some way the MOBIKE and NAT-T are not compatible. The NAT-T tries to work regardless of the IP-addresses, i.e. regardless whether someone modifies its IP-address or not. One of the goals in the MOBIKE is to AUTHENTICATE the change of the IP-address, i.e. when the IP-address changes we want to verify that this change is actually legitimate change done by the other end, not something done by the attacker along the path.

There is no way to distinguish the cases where there is NAT along the path which modifies the packets or if it is an attacker doing that. If NAT is detected in the IKE SA creation, that should automatically disable the MOBIKE extensions and use NAT-T.

If MOBIKE is enabled for the IKE SA (i.e. no NAT along the path when the IKE SA was created), then if NAT is later added then MOBIKE can detect that, but it cannot securely do anything for the issue. We can disable MOBIKE extensions completely at that time and move to use NAT-T, but as we loose all the security offered by the MOBIKE, it might be better to rekey the IKE SA (if policy allows that) so that we do not use MOBIKE at all and start using normal NAT-T.

If we start using NAT-T, then there is no defined way to detect that we moved away from the inside of the NAT. Thus we need to modify NAT-T and add that kind of detection capability there, if we want to start using MOBIKE at that point.

As a summary, if the policy accepts the risks caused by enabling NAT-T, then it can switch to NAT-T when it detects we are behind NAT. Easiest way to do it is to create new IKE SA, as NAT-T can only be enabled by the initial IKE SA creation, and it cannot be enabled by rekeys. Moving back from NAT-T to MOBIKE is harder as it requires changes to NAT-T. On the other hand keeping NAT-T enabled simply adds few bytes of extra overhead.

If we define some additions and extensions to NAT-T we can probably make it work better with MOBIKE, but there are quite a lot open issues. One way of seeing this is, that we have few other parameters we might want to turn on during the address update, i.e. the NAT-T parameters. Those include turning on or off keepalives, UDP encapsulation, or automatic address updating.

6. Changing addresses or changing the paths (issue #10, #14)

The question here is, if it is enough for the MOBIKE to detect the dead-address, or do it need to detect also dead-paths. Dead-address detection means that we only detect that we cannot get packets through to that remote address by using the local IP-address given by the local IP-stack (i.e. local address selected normally by the routing information). Dead-path detection means that we need to try all possible local interfaces/IP-addresses for each remote addresses, i.e. find all possible paths between the hosts and try them all to see which of them work (or at least find one working path).

Doing the dead-address detection is simpler, and there is less probe packets to be sent, thus it does not cause that much stress to the network. It also is enough for the scenarios where the connection problems are local (i.e. interfaces going down, WLAN access disappearing etc). It does not help if some router somewhere along the path breaks down, in which case rerouting the packets along another path might get around that broken router. The question is, whether rerouting around that problem inside the core network is a problem that MOBIKE needs to solve.

7. How and When to do Return Routability Checks (issue #6, #12, #15)

One of the decisions that needs to be done is when to do return routability checks. The simple approach is to do it always. Another option is to do it every time new IP-address is taken in to use. The basic format of the return routability check could be similar than dead-peer-detection, but the problem is that if that fails then the IKEv2 specification requires the IKE SA to be deleted. Because of this we might need to do some kind of other exchange.

If the other end is SGW with limited set of fixed IP-addresses, then the SGW may have a certificate having all the IP-addresses in the certificate. If the certificate includes all the IP-addresses, it is no point to do weaker return routability check, the data in the certificate is already properly authenticated after the IKE SA is created, so the peer might simply use that and ignore return routability checks for the addresses of the SGW.

Another option is to use draft-dupont-mipv6-3bombing [I.D.dupont-mipv6-3bombing] approach: do it only if you had to send the update from some other address than indicated preferred address.

Final option would simply not to do return routability checks at all. If we use indirect change notifications then we only move to the new IP address after successful dead-peer-detection on the new address, which is already return routability check. In the direct change notifications the authenticated peer have given out authenticated IP-address, thus we could simply trust the other end. There is no way external attacker can cause any attacks, but we are not protected by the internal attacker, i.e. the authenticated peer forwarding its traffic to the new address. On the other hand we do know the identity of the peer in that case.

There are some attacks which can be launched unless the return routability checks include some kind of nonce (issue #15). In this attack the valid end points sends address update notification for the third party, trying to get all the traffic to be sent there, causing denial of service attack. If the return routability checks does not contain any cookies or other random information not known by the other end, then that valid node could reply to the return routability checks even when it cannot see the request. This might cause the other end to turn the traffic to there, even when the real original recipient isn't at that address.

The IKEv2 NAT-T does not do any return routability checks. It simply uses the last address used by the other end, as and address where to send return packets back. The attacker can change those IP-addresses, and can cause the return packets to be sent to wrong IP-address. The

situation is fixed immediately when the attacker no longer changes the packets, and the first packet with real IP-address reaches the other end. In IKEv2 NAT-T the valid client can cause third party bombing by redirecting all the traffic pointed to him to third party. As the MOBIKE tries to provide better security than IKEv2 NAT-T the MOBIKE protocol should protect against those attacks.

There might be also return routability information available from the other parts of the system too (IP-stack, Mobile IP or so), but the checks done might be different (issue #12). There are multiple different levels for the return routability checks:

- None, no tests
- A party willing to answer is on the path to the claimed address. This is the basic form of return routability test.
- There is an answer from the tested address, and that answer was

authenticated (including the address) to be from our peer.

- There was an authenticated answer from the peer, but it is not guaranteed to be from the tested address or path to it (because the peer can construct a response without seeing the request).

The basic return routability checks do not protect against 3rd party bombing, if the attacker is along the path, as the attacker can forward the return routability checks to the real peer (even if those packets are cryptographically authenticated)

If the address to be tested is inside the packet too, then attacker cannot forward packets, thus it prevents 3rd party bombings.

If the reply packet can be constructed without seeing the request packet (i.e. there is no nonce or similar), then the real peer can cause 3rd party bombing, by replying to those requests without seeing them at all.

Other levels might only return information saying that yes, there is someone there in the IP-address which replied to my request. Or say that I sent request to IP-address and got reply back, but I am not sure whether that reply was freshly generated or repeated, or sent from different address. The MOBIKE requirements for the return routability checks could be to verify that there is same (cryptographically) authenticated node in the other end and it can now receive packets from the IP-address it claims to have.

MOBIKE might also want to export the information it has done the return routability checks to the other modules, like Mobile IP, so Mobile IP does not need to do return routability checks again, if it is satisfied with the level of checks done by the MOBIKE.

8. Scope of SA changes (issue #8)

The basic question is that how the IPsec SAs are changed to use new address. One option is that when the IKE SA address is changed then automatically all IPsec SAs associated with it are moved along with it to new address. Another option is to have separate exchange to move the IPsec SAs separately.

If we want to update each IPsec SA separately, we probably need more efficient format than notification payload, as it can only store one SPI per payload. I.e. we want separate payload which have list of IPsec SA SPIs and new address set for them. If we have lots of IPsec SAs, those payloads can be quite large unless we support ranges in SPIs or at least have some kind of notation of move those SAs not moved separately (i.e. rest of the SAs indication). The implementations need also keep state of IP-addresses per IPsec SA, not per IKE SA. If we automatically move all IPsec SAs when the IKE SA moves, then we only need to keep track which IKE SA was used to create the IPsec SA, and fetch the IP-addresses from that (Note, that IKEv2 [I-D.ietf-ipsec-ikev2] already requires implementations to keep track which IPsec SAs are created using which IKE SA).

If we do allow each IPsec SAs address sets to be updated separately, then we can support scenarios, where the machine have fast and/or cheap connection and slow and/or expensive connection, and it wants to allow moving some of the SAs to the slower and/or more expensive connection, and forbid some SAs to move. I.e. never move the news video stream from the WLAN to the GPRS link.

On the other hand, even if we tie the IKE SA update to the IPsec SA update, then we can create separate IKE SAs for this scenario, i.e. we create one IKE SA which have both links as endpoints, and it is used for important traffic, and then we create another IKE SA which have only the fast and/or cheap connection, which is then used for that kind of bulk traffic.

9. Zero Address Set (issue #5)

One of the features which might be useful would be for the peer to announce the other end that it will now disconnect for some time, i.e. it will not be reachable at all. For instance, a laptop might go to suspend mode. In this case the peer could send address notification with zero new addresses, which means that it will not have any valid addresses anymore. The responder of that kind of notification would then acknowledge that, and could then temporarily disable all SAs. If any of the SAs gets any packets they are simply dropped. This could also include some kind of ACK spoofing to keep the TCP/IP sessions alive (or simply set the TCP/IP keepalives and timeouts large enough not to cause problems), or it could simply be left to the applications, i.e. allow TCP/IP sessions to notice the link is broken.

The local policy could then decide how long the peer would allow other peers to be disconnected, i.e. whether this is only allowed for few minutes, or do they allow users to disconnect Friday evening and reconnect Monday morning (consuming resources during weekend too, but on the other hand not more than is normally used during week days, but we do not need lots of extra resources on the Monday morning to support all those people connecting back to network).

10. What modes we support (issue #7)

The charter mostly talks about VPN style usage, and all scenarios are using tunnel mode, so that is where this document mostly concentrates. For transport mode to be used we first need to define the scenarios where it is needed. XXX someone needs to write more text here.

11. Message representation

One of the basic design choices that is needed for the MOBIKE is the format of the messages. The IKEv2 offers some formatting alternatives for the protocol. The basic IP-address change notifications can be sent either via informational exchange already specified in the IKEv2, or we could also have our own MOBIKE specific exchange. Using informational exchange has the main advantage that it is already specified in the IKEv2 and the implementations should already have code for those.

One advantage of creation of the new exchange would be that we could incorporate the return routability checks to the exchange in this state (i.e. create 3-4 packet exchange). The problem here is that we might need to do the return routability checks for each IP-address separately, thus we might not be able to do it in this phase.

Another question is the basic format of the address update notifications. The address update notifications can include multiple addresses, which some can be IPv4 and some IPv6 addresses. The number of addresses is most likely going to be quite small (less than 10). The format might need to give out senders preference of the use of the addresses, i.e. the sender will tell this is the preferred address to be used. The format could either contain the preference number, giving out the relative order of the addresses, or it could simply be ordered list of IP-addresses in the order of the most preferred first. Benefits of the ordered list is, that then we do not need to define what happens if the preference numbers are identical, and we do not need to reserve space for the numbers. Normally we do not need any priority values, we simply need an ordered list.

Even when the load-balancing inside the one connection is outside the scope of the MOBIKE, there might be future work to include that. The format selected needs to be flexible enough to allow addition of some kind of extra information for the load-balancing features in the future. This might be something like one reserved field, which can later be used to store that information. As there is other potential information which might have to be tied to the address in the future, a reserved field seems like a prudent design in any case.

There are two basic formats for putting IP-address list in to the exchange, we can include each IP-address as separate payload (where the payload order indicates the preference value, or the payload itself might include the preference value), or we can put the IP-address list as one payload to the exchange, and that one payload will then have internal format which includes the list of IP-addresses.

Having multiple payloads each having one IP-address makes the protocol probably easier to parse, as we can already use the normal IKEv2 payload parsing procedures to get the list out. It also offers easy way for the extensions, as the payload probably contains only the type of the IP-address (or the type is encoded to the payload type), and the IP-address itself, and as each payload already has length associated to it, we can detect if there is any extra data after the IP-address. Some implementations might have problems parsing too long list of IKEv2 payloads, but if the sender sends them in the most preferred first, the other end can simply only take n first addresses and use them. It might loose connection in some cases if all the n first address are not in use anymore, and the other end hasn't sent new list, but in most cases everything will still work.

Having all IP-addresses in one big payload having MOBIKE specified internal format, provides more compact encoding, and keeps the MOBIKE implementation more concentrated to one module.

The next choice is which type of payloads to use. IKEv2 already specifies a notify payload, which could be used for that. It includes some extra fields (SPI size, SPI, protocol etc), which gives 4 bytes of the extra overhead, and there is the notify data field, which could include the MOBIKE specific data.

Another option would be to have our own payload type, which then include the information needed for the MOBIKE protocol.

The basic protocol is most likely going to be something where we send list of all IP-addresses every time the list changes (either addresses are added, removed, or the preferred order changes). Another option is that we send some kind of incremental updates to the IP-address list. Sending incremental updates provides more

compact packets (meaning we can support more IP-addresses), but on the other hand have more problems in the synchronization and packet reordering cases i.e. the incremental updates must be processed in order, but for full updates we can simply use the most recent one, and ignore old ones, even if they arrive after the most recent one (IKEv2 packets have message id which is incremented for each packet, thus we know the sending order easily).

The address update notification protocol is not restricted to only one way, i.e. both ends might have multiple IP-addresses and both ends might send address updates. Example of that is when the roaming laptop connects to the multihoming SGW.

12. Security Considerations

As all the messages are already authenticated by the IKEv2 there is no problem that any attackers would modify the actual contents of the packets. The IP addresses in IP header of the packets are not authenticated, thus the protocol defined must take care that they are only used as an indication that something might be different, they should not cause any direct actions.

Attacker can also spoof ICMP error messages in an effort to confuse the peers about which addresses are not working. At worst this causes denial-of-service and/or the use of non-preferred addresses, so it is not that serious.

One type of attacks which needs to be taken care of the MOBIKE protocol is also various flooding attacks. See [I-D.nikander-mobileip-v6-ro-sec] and [Aur02] for more information about flooding attacks.

13. IANA Considerations

No IANA assignments are needed.

14. References

14.1 Normative references

[I-D.ietf-ipsec-ikev2]
Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", draft-ietf-ipsec-ikev2-14 (work in progress), June 2004.

[Kiv04] Kivinen, T., "MOBIKE protocol", draft-kivinen-mobike-protocol-00 (work in progress), February 2004.

14.2 Non-normative references

[I-D.nikander-mobileip-v6-ro-sec]
Nikander, P., "Mobile IP version 6 Route Optimization Security Design Background", draft-nikander-mobileip-v6-ro-sec-02 (work in progress), December 2003.

[I-D.dupont-mipv6-3bombing]
Dupont, F., "A note about 3rd party bombing in Mobile IPv6", draft-dupont-mipv6-3bombing-00 (work in progress), February 2004.

[Aur02] Aura, T., Roe, M. and J. Arkko, "Security of Internet Location Management", In Proc. 18th Annual Computer Security Applications Conference, pages 78-87, Las Vegas, NV USA, December 2002.

Author's Address

Tero Kivinen
Safenet, Inc.
Fredrikinkatu 47
HELSINKI FIN-00100
FI

EMail: kivinen@safenet-inc.com

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to

pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

Full Copyright Statement

Copyright (C) The Internet Society (2004). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assignees.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION

HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.