IKEv2 Mobility and Multihoming   T. Kivinen
(mobike)   Safenet, Inc.
Internet-Draft   H. Tschofenig
Expires: July 1, 2005   Siemens
                                                     December 31, 2004

Design of the MOBIKE Protocol
draft-ietf-mobike-design-01.txt

Status of this Memo

Copyright Notice

Abstract

   The MOBIKE (IKEv2 Mobility and Multihoming) working group is
   developing protocol extensions to the Internet Key Exchange Protocol
   version 2 (IKEv2) to enable its use in the context where there are
   multiple IP addresses per host or where IP addresses of an IPsec host
   change over time (for example due to mobility).

This document discusses the involved network entities, and the
relationship between IKEv2 signaling and information provided by
other protocols and the rest of the network. Design decisions for
the base MOBIKE protocol, background information and discussions
within the working group are recorded.

Table of Contents

1. Introduction

IKEv2 is used for performing mutual authentication and establishing and maintaining IPsec security associations (SAs). IKEv2 establishes both cryptographic state (such as session keys and algorithms) as well as non-cryptographic information (such as IP addresses).

The current IKEv2 and IPsec documents explicitly say that the IPsec and IKE SAs are created implicitly between the IP address pair used during the protocol execution when establishing the IKEv2 SA. This means that there is only one IP address pair stored for the IKEv2 SA, and this single IP address pair is used as an outer endpoint address for tunnel mode IPsec SAs. After the IKE SA is created there is no way to change them.

There are scenarios where this IP address might change, even frequently. In some cases the problem could be solved by rekeying all the IPsec and IKE SAs after the IP address has changed. However, this can 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. MOBIKE provides solution to the problem of the updating the IP addresses stored with IKE SAs and IPsec SAs.

The charter of the MOBIKE working group requires IKEv2 [I-D.ietf-ipsec-ikev2], and as IKEv2 assumes that the RFC2401bis architecture [I-D.ietf-ipsec-rfc2401bis] is used, all protocols developed will use both IKEv2 and RFC2401bis. No effort is to be made to make protocols for IKEv1 [RFC2409] or old RFC2401 architecture [RFC2401].

This document is structured as follows. After introducing some important terms in Section 2 we list a few scenarios in Section 3, to illustrate possible deployment environments. MOBIKE depends on information from other sources (e.g., detect an address change) which are discussed in Section 4. Finally, Section 5 discusses design considerations effecting the MOBIKE protocol. The document concludes with security considerations in Section 6.

2. **Terminology**

   This section introduces some terms in useful for a MOBIKE protocol.

   Peer:

      Within this document we refer to IKEv2 endpoints as peers. A peer
      implements MOBIKE and therefore IKEv2.

   Available address:

      This definition is reused from
      [I-D.arkko-multi6dt-failure-detection] and refers to addresses
      which are available by an peer. A few conditions must hold before
      an address in such a state.

   Locally Operational Address:

      An available address is said to be locally operational when its
      use is known to be possible locally. This definition is taken
      from [I-D.arkko-multi6dt-failure-detection].

   Operational address pair:

      A pair of operational addresses are said to be an operational
      address pair, iff bidirectional connectivity can be shown between
      the two addresses. Note that sometimes it is necessary to
      consider a 5-tuple when connectivity between two endpoints need to
      be tested. This differentiation might be necessary to address
      load balancers, certain Network Address Translation types or
      specific firewalls. This definition is taken from
      [I-D.arkko-multi6dt-failure-detection] and enhanced to fit the
      MOBIKE context. Although it is possible to further differentiate
      unidirectional and bidirectional operational address pairs only
      bidirectional connectivity is relevant for this document and
      unidirectional connectivity is out of scope.

   Path:

      The route taken by the MOBIKE and/or IPsec packets sent via the IP
      address of one peer to a specific destination address of the other
      peer. Note that the path might be effected not only by the source
      and destination IP addresses but also by the selected transport
      protocol and transport protocol identifier. Since MOBIKE and
      IPsec packets have a different appearance on the wire they might
      be routed along a different path. This definition is taken from
      [RFC2960] and modified to fit the MOBIKE context.

Primary Path:

   The primary path is the destination and source address that will
   be put into a packet outbound to the peer by default. This
   definition is taken from [RFC2960] and modified to fit the MOBIKE
   context.

Preferred Address:

   An address on which a peer prefers to receive MOBIKE messages and
   IPsec protected data traffic. A given peer has only one active
   preferred address at a given point in time (ignoring the small
   time period where it needs to switch from the old preferred
   address to a new preferred address). This definition is taken
   from [I-D.ietf-hip-mm] and modified to fit the MOBIKE context.

Peer Address Set:

   A subset of locally operational addresses that will sent
   communicated to another peer. A policy available at the peer
   indicates which addresses to include in the peer address set.
   Such a policy might be impacted by manual configuration or by
   interaction with other protocols which indicate newly available
   addresses. Note that the addresses in the peer address set might
   change over time.

Preferred Address Pair:

   This address pair taken from the peer address set is used for
   communication. The preferred address pair is used (1) for MOBIKE
   communication where only two IP addresses are used and (2) as the
   outer IP addresses (source and destination IP address) of the
   IPsec packet in tunnel mode.

   Terminology for different NAT types, such as Full Cone, Restricted
   Cone, Port Restricted Cone and Symmetric, can be found in Section 5
   of [RFC3489]. For mobility related terminology, such as
   Make-before-break or Break-before-make see [RFC3753].

3. Scenarios

The MOBIKE protocol can be used in different scenarios. Three of them are discussed below.

3.1 Mobility Scenario

Figure 1 shows a break-before-make mobility scenario where a mobile node attaches to, for example a wireless LAN, to obtain connectivity to some security gateway. This security gateway might connect the mobile node to a corporate network, to a UTMS network or to some other network. The initial IKEv2 exchange takes place along the path labeled as 'old path' from the MN using its old IP address via the old access router (OAR) towards the security gateway (GW). The IPsec tunnel mode terminates there but the decapsulated data packet will typically address another destination. Since only MOBIKE is relevant for this discussion the end-to-end communication between the MN and some destination server is not shown in Figure 1.

When the MN moves to a new network and obtains a new IP address from a new access router (NAR) it is the responsibility of MOBIKE to avoid restarting the IKEv2 exchange from scratch. As a result, some form of protocol exchange, denoted as 'MOBIKE Address Update', will perform the necessary state update. The protocol messages will travel along a new path whereby the old path and the new path will meet at the cross-over router.

Note that in a break-before-make mobility scenario the MN obtains a new IP address after reachability to the old IP address has been lost. MOBIKE is also assumed to work in scenarios where the mobile node is able to establish connectivity with the new IP address while still being reachable at the old IP address.
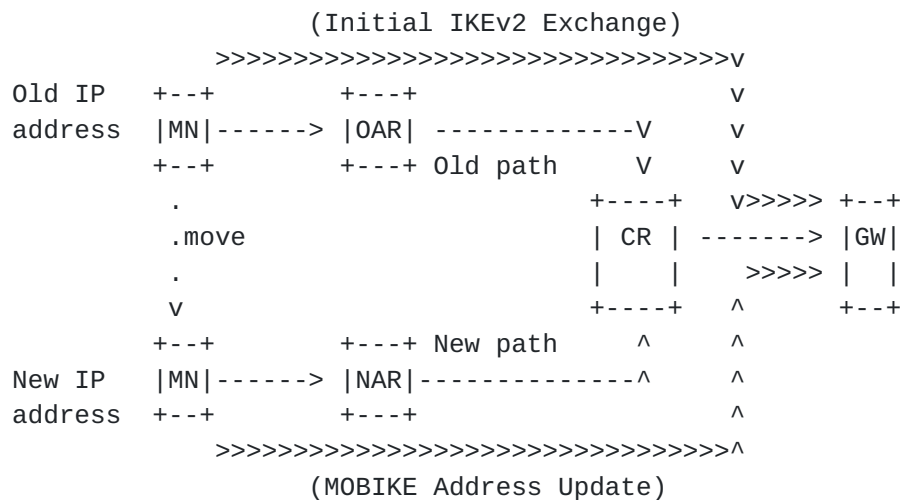
```
                      (Initial IKEv2 Exchange)
                 >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>v
      Old IP   +--+          +---+                    v
      address  |MN|------> |OAR| -------------V     v
               +--+          +---+ Old path     V     v
                .                              +----+   v>>>>> +--+
                .move                          | CR | ------> |GW|
                .                              |    |   >>>>> |  |
                v                              +----+   ^      +--+
               +--+          +---+ New path      ^      ^
      New IP   |MN|------> |NAR|--------------^      ^
      address  +--+          +---+                    ^
                 >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>^
                      (MOBIKE Address Update)


          ---> = Path taken by data packets
          >>>> = Signaling traffic (IKEv2 and MOBIKE)
          ...> = End host movement

              Figure 1: Mobility Scenario
```

3.2 Multihoming Scenario

Another scenario where MOBIKE might be used is between two peers
equipped with multiple interfaces (and multiple IP addresses).
Figure 2 shows two such multi-homed peers. Peer A has two interface
cards with two IP addresses IP_A1 and IP_A2. Additionally, Peer B
also has two IP addresses, IP_B1 and IP_B2. Each peer selects one of
its IP addresses as the preferred address which will subsequently be
used for communication. Various reasons, such as problems with the
interface card, might require a peer to switch from one IP address to
the other one.
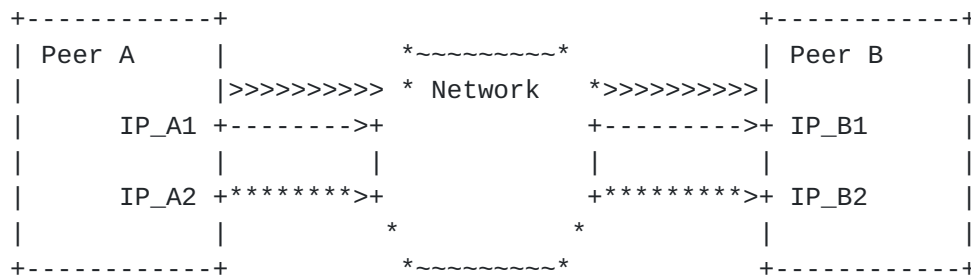
```
   +------------+                                    +------------+
   | Peer A     |            *~~~~~~~~~*             | Peer B     |
   |            |>>>>>>>>>> * Network   *>>>>>>>>>>|  |            |
   |     IP_A1 +-------->+              +--------->+ IP_B1      |
   |           |        |              |         |  |           |
   |     IP_A2 +********>+              +*********>+ IP_B2      |
   |           |        *              *         |  |           |
   +------------+            *~~~~~~~~~*             +------------+


              ---> = Path taken by data packets
              >>>> = Signaling traffic (IKEv2 and MOBIKE)
              ***> = Potential future path through the network
                     (if Peer A and Peer B chance their preferred
                      address)

                   Figure 2: Multihoming Scenario
```

   Note, that the load-balancing inside one IKE SA is not provided by
   the MOBIKE protocol. Each client uses only one of the available IP
   addresses at a given point in time.

3.3 Multihomed Laptop Scenario

   In the multihomed laptop scenario we consider a laptop, which has
   multiple interface cards and therefore several ways to connect to a
   network. It might for example have a fixed Ethernet, WLAN, GPRS,
   Bluetooth or USB hardware in order to sent IP packets. Some of these
   interfaces might connected to a network over the time depending on a
   number of reasons (e.g., cost, availability of certain link layer
   technologies, user convenience). For example, the 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. At home he might again use again WLAN. At a certain
   point in time multiple interfaces might be connected. As such, the
   laptop is a multihomed device. In any case, the IP address of the
   individual interfaces are subject to change.

   The user desires to keep the established IKE-SA and IPsec SAs alive
   all time without the need to re-run the initial IKEv2 exchange which
   could require user interaction as part of the authentication process.
   Furthermore, even if IP addresses change due to interface switching
   or mobility, the IP source and destination address obtained via the
   configuration payloads within IKEv2 and used inside the IPsec tunnel
   remains unaffected, i.e., applications might not detect any change at
   all.

[4](#). **Framework**

   Initially, when a MOBIKE peer starts and executes the initial
   protocol exchange with its MOBIKE peer it needs to setup a peer
   address set based on the available addresses. It might want to make
   this peer address set available to the other peer. The Initiator
   does not need to explicitly indicate its preferred address since
   already using its preferred address. The outgoing IKEv2 and MOBIKE
   messages use this preferred address as the source IP address and
   expects incoming signaling messages to be addressed to this address.

   Interaction with other protocols at the MOBIKE host is required to
   build the peer address set and the preferred address. In some cases
   the peer address set is available before the initial protocol run and
   does not change during the lifetime of the IKE-SA. The preferred
   address might change due to policy reasons. In many other cases, as
   motivated in [Section 3](#) the peer address set is modified (by adding or
   deleting addresses) and the preferred address needs to be changed as
   well.

   Modifying the peer address set or changing the preferred address is
   effected by the host's local policy and by the interaction with other
   protocols (such as DHCP or IPv6 Neighbor Discovery).

   Figure 3 shows an example protocol interaction at a MOBIKE peer.
   MOBIKE interacts with the IPsec engine using the PF_KEY interface to
   create entries in the Security Association and Security Policy
   Databases. The IPsec engine might also interact with IKEv2 and
   MOBIKE. Established state at the IPsec databases has an impact on
   the incoming and outgoing data traffic. MOBIKE receives information
   from other protocols running in both kernel-mode and user-mode, as
   previously mentioned. Information relevant for MOBIKE is stored in a
   database, referred as Trigger database which guides MOBIKE in its
   decisions regarding the available addresses, peer address set and the
   preferred address. Policies might affect the selection process.

   Building and maintaining a peer address set and selecting or changing
   a preferred address based on locally available information is,
   however, insufficient. This information needs to be available to the
   other peer and in order to address various failure cases it is
   necessary to test connectivity along a path. A number of address
   pairs might be available for connectivity tests but most important is
   the source and the destination IP address of the preferred address
   pair since these addresses are selected for sending and receiving
   MOBIKE signaling messages and for sending and receiving IPsec
   protected data traffic. If a problem along this primary path is
   detected (e.g., due to a router failure) it is necessary to switch to
   the new preferred address pair. Testing other paths might also be

useful to notice when a disconnected path is operational again.

```
                        +-------------+        +---------+
                        |User-space   |        | MOBIKE  |
                        |Protocols    | +-->>| Daemon  |
                        |relevant for |  |    |         |
                        |MOBIKE       |  |    +---------+
                        +-------------+  |        ^
   User Space                  ^         |        ^
   ++++++++++++++++++++++++++++ API ++++++ API ++++ PF_KEY ++++++++
   Kernel Space                v         |        v
                             _____      |        v
      +-------------+       /       \     |    +--------------+
      |Routing      |      / Trigger \    |    | IPsec        |
      |Protocols    |<<-->>|  Database |<<-+   +>+ Engine       |
      |             |      \         /     | | (+Databases) |
      +-----+---+---+       _____/      | +------+-------+
           ^   ^                ^          |        ^
           |  +--------------+------------+--------+-----+
           |   |             v           |        |    |
           |   |      +-------------+     |        |    |
   I       |   |      |Kernel-space |     |        |    |  I
   n       |  +-------->+Protocols    +<----+-----+  |    |  n
   t       v  v      |relevant for |     |    v  v      v  t
   e +----+---+-+    |MOBIKE       |     |    +-+--+-----+-+ e
   r |  Input  |     +-------------+     |    | Outgoing  | r
   f |  Packet  +<--------------------------+    | Interface  | f
  ==a>|Processing|============================| Processing |=a>
   c |         |                            |           | c
   e +---------+                            +-----------+ e
   s                                                     s
```

```
            ===> = IP packets arriving/leaving a MOBIKE node
            <-> = control and configuration operations
```
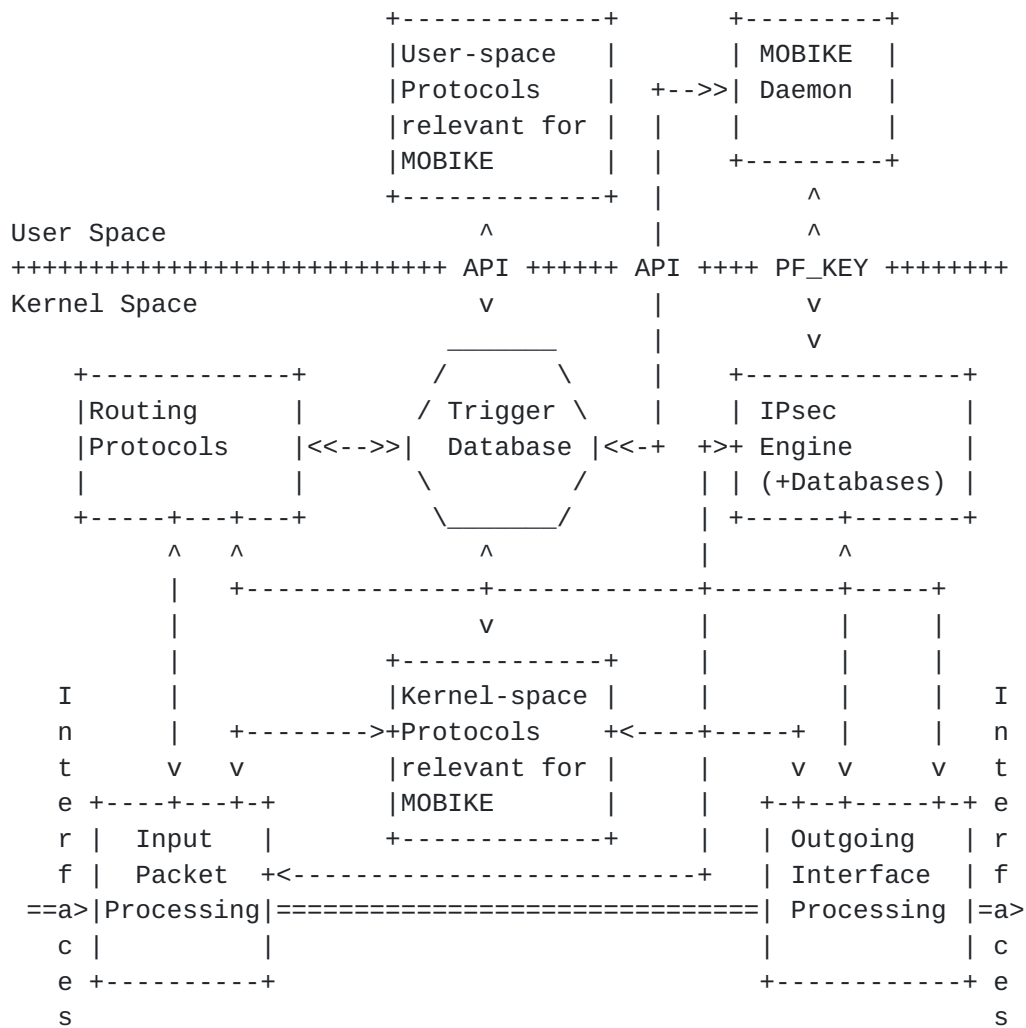
Figure 3: Framework

Although the interaction with other protocols is important for MOBIKE
to operate correctly the working group is chartered to leave this
aspect outside the scope. The working group will develop a MOBIKE
protocol which needs to perform the following functionality:
o Ability to inform a peer about the peer address set
o Ability to inform a peer about the preferred address
o Ability to test connectivity along a path and thereby to detect an
   outage situation in order to fall back to another preferred
   address, if necessary, or to change the peer address set

o Ability to deal with Network Address Translation devices

In addition to the above-listed functionality security is important
to the working group. For example, the ability to prevent flooding
attacks based on announcing someone else's address needs to be dealt
with.

Extensions of the PF_KEY interface required by MOBIKE are also within
the scope of the working group. Finally, optimizations in wireless
environment will also be covered.

Note that MOBIKE is somewhat different compared to, for example, SCTP
mobility since both the IKE-SA and the IPsec SA is affected by the
change of addresses.

5. Design Considerations

   This section discusses aspects affecting the design of the MOBIKE
   protocol.

5.1 Indicating Support for MOBIKE

   A node needs to be able to guarantee that its address change messages
   are understood by its corresponding peer. Otherwise an address
   change may render the connection useless, and it is important that
   both sides realize this as early as possible.

   Ensuring that the messages are understood can in be arranged either
   by marking some IKEv2 payloads critical so that they are either
   processed or an error message is returned, by using Vendor ID
   payloads or via a Notify.

   The first solution approach is to use Vendor ID payloads during the
   initial IKEv2 exchange using a specific string denoting MOBIKE to
   signal the support of the MOBIKE protocol. This ensures that in all
   cases a MOBIKE capable node knows whether its peer supports MOBIKE or
   not.

   The second solution approach uses the Notify payload which is also
   used for NAT detection (via NAT_DETECTION_SOURCE_IP and
   NAT_DETECTION_DESTINATION_IP).

   Both, a Vendor ID and a Notify payload, might be used to indicate the
   support of certain extensions.

   Note that the node could also attempt MOBIKE optimistically with the
   critical bit set to one when a movement has occurred. The drawback
   of this approach is, however, that the an unnecessary MOBIKE message
   round is introduced on the first movement.

   Although Vendor ID payloads and Notifications are technically
   equivalent Notifications are already used in IKEv2 as a capability
   negotiation mechanism and is therefore the preferred mechanism.

5.2 Changing a Preferred Address and Multihoming Support

   From MOBIKE's point of view multihoming support is integrated by
   supporting a peer address set rather than a single address and
   protocol mechanisms to change to use a new preferred IP address.
   From a protocol point of view each peer needs to learn the preferred
   address and the peer address set somehow, implicitly or explicitly.

5.2.1 **Storing a single or multiple addresses**

   One design decision is whether an IKE-SA should store a single IP
   address or multiple IP addresses as part of the peer address set.
   One option is that the other end can provide a list of addresses
   which can be used as destination addresses. MOBIKE does not include
   load balancing, i.e., only one IP address is set to a preferred
   address at a time and changing the preferred address typically
   requires some MOBIKE signaling.

   Another option is to only communicate one address towards the other
   peer and both peers only use that address when communicating. If
   this preferred address cannot be used anymore then an address update
   is sent to the other peer changing the primary address.

   If peer A provides a peer address set with multiple IP addresses then
   peer B can recover from a failure of the preferred address on its
   own, meaning that when it detects that the primary path does not work
   anymore it can either switch to a new preferred address locally
   (i.e., causing the source IP address of outgoing MOBIKE messages to
   have a non-preferred address) or to try an IP address from A's peer
   address set. If peer B only received a single IP address as the A's
   peer address set then peer B can only change its own preferred
   address. The other end has to wait for an address update from peer A
   with a new IP address in order to fix the problem. The main
   advantage about using a single IP address for the remote host is that
   it makes retransmission easy, and it also simplifies the recover
   procedure. The peer, whose IP address changed, must start recovery
   process and send the new IP address to the other peer. Failures
   along the path are not well covered with advertising a single IP
   address.

   The single IP address approach will not work if both peers happen to
   loose their IP address at the same time (due to, say, the failure of
   one of the links that both nodes are connected to). It may also
   require the IKEv2 window size to be 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. Furthermore, the
   single IP address approach does not really benefit much from indirect
   indications as the peer receiving these indications might not be able
   to fix the situation by itself (e.g., even if a peer receives an ICMP
   host unreachable message for the old IP address, it cannot try other
   IP addresses, since they are unknown).

   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.

Please note that the discussed aspect is partially different from the question how many addresses to sent in a single MOBIKE message. A MOBIKE message might be able to carry more than one IP address (with the IP address list approach) or a single address only. The latter approach has the advantage of dealing with NAT traversal in a proper fashion. A NAT cannot change addresses carried inside the MOBIKE message but it can change IP address (and transport layer addresses) in the IP header and Transport Protocol header (if NAT traversal is enabled). Furthermore, a MOBIKE message carrying the peer address set could be idempotent (i.e., always resending the full address list) or does the protocol allow add/delete operations to be specified. [I-D.dupont-ikev2-addrmgmt], for example, offers an approach which defines add/delete operations. The same is true for the dynamic address reconfiguration extension for SCTP [I-D.ietf-tsvwg-addip-sctp].

5.2.2 Indirect or Direct Indication

An indication to change the preferred IP address might be either direct or indirect.

Direct indication:

   A direct indication means that the other peer will send an message with the address change. Such a message can, for example, accomplished by having MOBIKE sending an authenticated address update notification with a different preferred address.

Indirect indication:

   An indirect indication to change the preferred address can be obtained by observing the environment. An indirect indication might, for example, be be the receipt of an ICMP message or information of a link failure. This information should be seen as a hint and might not directly cause any changes to the preferred address. Depending on the local policy MOBIKE might decide to perform a dead-peer detection exchange for the preferred address pair (or another address pair from the peer address set). When a peer detects that the other end started to use different source IP address than before, it might want to authorize the new preferred address (if not already authorized). A peer might also start a connectivity test of this particular address.If a bidirectionally operational address pair is selected then MOBIKE needs to

communicate this new preferred address to its remote peer.

MOBIKE will utilize both mechanisms, direct and indirect indications, to make a more intelligent decision which address pair to select as the preferred address. The more information will be available to MOBIKE the faster a new operational preferred address pair can be selected among the available candiates.

5.2.3 Connectivity Tests using IKEv2 Dead-Peer Detection

This section discusses the suitability of the IKEv2 dead-peer detection (DPD) mechanism for connectivity tests between address pairs. The basic IKEv2 DPD mechanism is not modified by MOBIKE but it needs to be investigated whether it can be used for MOBIKE purposes as well.

The IKEv2 DPD mechanism is done by sending an empty informational exchange packet to the other peer, in which case the other end will respond with an acknowledgement. If no acknowledgement is received after certain timeout (and after couple of retransmissions), the other peer is considered to be not reachable. Note that the receipt of IPsec protected data traffic is also a guarantee that the other peer is alive.

When reusing dead-peer detection in MOBIKE for connectivity tests it seems to be reasonable to try other IP addresses (if they are available) in case of unsuccessful connectivity test for a given address pair. Dead-peer detection messages using a different address pair should use the same message-id as the original dead-peer detection message (i.e. they are simply retransmissions of the dead-peer detection packet using different destination IP address). If different message-id is used then it violates constraints placed by the IKEv2 specification on the DPD message with regard to the mandatory ACK for each message-id, causing the IKEv2 SA to be deleted.

If MOBIKE strictly follows the guidelines of the dead-peer detection mechanism in IKEv2 then an IKE-SA should be marked as dead and deleted if the connectivity test for all address pairs fails. Note that this is not in-line with the approach used, for example, in SCTP where a failed connectivity test for an address does not lead to (a) the IP address or IP address pair to be marked as dead and (b) delete state. Connectivity tests will be continued for the address pairs in hope that the problem will recover soon.

Note, that as IKEv2 implementations might have window size of 1, which prevents to initiate a dead-peer detection exchange while doing another exchange. As a result all other exchanges experience the

identical retransmission policy as used for the dead-peer detection.

The dead-peer detection for the other IP address pairs can also be
executed simultaneously (with a window size larger than 1), meaning
that after the initial timeout of the preferred address expires, we
send packets simultaneously to all other address pairs. It is
necessary to differentiate individual acknowledgement messages in
order to determine which address pair is operational. Therefore the
source IP address of the acknowledgement should match the destination
IP towards the message was previously sent.

Also the other peer is most likely going to reply only to the first
packet it receives, and that first packet might not be the best (by
whatever criteria) address pair. The reason the other peer is only
responding to the first packet it receives, is that implementations
should not send retransmissions if they have just sent out identical
response message. 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 a large number of MOBIKE clients
notice it, they should not start sending a large number 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 simultaneously try to test all
possible address pairs by executing connectivity tests then the
congestion problem only gets worse.

Also note, that the IKEv2 dead-peer detection is not sufficient for
the return routability check. See Section 5.6 for more information.

5.3 Simultaneous Movements

MOBIKE does not aim to provide a full mobility solution which allows
simultaneous movements. Instead, the MOBIKE working group focuses on
selected scenarios as described in Section 3. Some of the scenarios
assume that one peer has a fixed set of addresses (from which some
subset might be in use). Thus it cannot move to the address unknown
to the other peer. Situations where both peers move and the movement
notifications cannot reach the other peer, is outside the scope of
the MOBIKE WG. MOBIKE has not being chartered to deal with the
rendezvous problem, or with the introduction of any new entities in
the network.

Note, that if only a single address is stored in the peer address set

(instead of an address list) we might end up in the case where it
seems that both peers changed their addresses at the same time. This
is something that the protocol must deal with.

Three cases can be differentiated:

o Two mobile nodes obtain a new address at the same time, and then
   being unable to tell each other where they are (in a
   break-before-make scenario). This problem is called the
   rendezvous problem, and is traditionally solved by introducing
   another third entity, for example, the home agents (in Mobile
   IPv4/IPv6) or forwarding agents (in Host Identity Protocol).
   Essentially, solving this problem requires the existence of a
   stable infrastructure node somewhere.

o Simultaneous changes to addresses such that at least one of the
   new addresses is known to the other peer before the change
   occurred.

o No simultaneous changes at all.

5.4 NAT Traversal

IKEv2 has support of legacy NAT traversal built-in. This feature is
known as NAT-T which allows IKEv2 to work even if a NAT along the
path between the Initiator and the Responder modified the source and
possibly the destination IP address. With NAPT even the transport
protocol identifiers are modified (which then requires UDP
encapsulation for subsequent IPsec protected data traffic).
Therefore, the required IP address information is taken from the IP
header (if a NAT was detected who rewrote IP header information)
rather than from the protected payload. This problem is not new and
was discovered during the design of mobility protocol where the only
valuable information is IP address information.

One of the goals in the MOBIKE protocol is to securely exchange one
or more addresses of the peer address set and to securely set the
primary address. If not other protocol is used to securely retrieve
the IP address and port information allocated by the NAT then it is
not possible to tackle all attacks against MOBIKE. Various solution
approaches have been presented:

o Securely retrieving IP address and port information allocated by
   the NAT using a protocol different from MOBIKE. This approach is
   outside the scope of the MOBIKE working group since other working
   groups, such as MIDCOM and NSIS, already deal with this problem.
   The MOBIKE protocol can benefit from the interaction with these
   protocols.

o   Design a protocol in such a way that NAT boxes are able to inspect
    (or even participate) in the protocol exchange. This approach was
    taken with HIP but is not an option for IKEv2 and MOBIKE.

o   Disable NAT-T support by indicating the desire never to use
    information from the (unauthenticated) header. While this
    approach prevents some problems it effectively disallows the
    protocol to work in certain environments.

There is no way to distinguish the cases where there is NAT along the
path which modifies the header information in packets or whether an
adversary mounts an attack. If NAT is detected in the IKE SA
creation, that should automatically disable the MOBIKE extensions and
use NAT-T.

A design question is whether NAT detection capabilities should be
enabled only during the initial exchange or even at subsequent
message exchanges. If MOBIKE is executed with no NAT along the path
when the IKE SA was created, then a NAT which appears after moving to
a new network might cannot be dealt with if NAT detection is enabled
only during the initial exchange. Hence, it might be desirable to
also support scenario where a MOBIKE peer moves from a network which
does not deploy a NAT to a network which uses a private address
space.

A NAT prevention mechanism can be used to make sure that no adversary
can interact with the protocol if no NAT is expected between the
Initiator and the Responder.

The support for NAT traversal is certainly one of the most important
design decisions with an impact on other protocol aspects.

5.5 Changing addresses or changing the paths

A design decision is whether it is enough for the MOBIKE protocol to
detect dead address, or does 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).

While performing just an address change is simpler, the existence of
locally operational addresses are not, however, a guarantee that
communications can be established with the peer. A failure in the
routing infrastructure can prevent the sent packets from reaching
their destination. Or a failure of an interface on one side can be

related to the failure of an interface on the other side, making it
necessary to change more than one address at a time.

5.6 Return Routability Tests

Setting a new preferred address which is subsequently used for
communication is associated with an authorization decision: Is a peer
allowed to use this address? Does this peer own this address? Two
mechanisms have been proposed in the past to allow a peer to compute
the answer to this question:

o The addresses a peer is using is part of a certificate. [RFC3554]
   introduced this approach. If the other peer is, for example, a
   security gateway with a limited set of fixed IP addresses, then
   the security gateway may have a certificate with all the IP
   addresses in the certificate.

o A return routability check is performed before the address is used
   to ensure that the peer is reachable at the indicated address.

Without performing an authorization decision a malicious peer can
redirect traffic towards a third party or a blackhole.

An IP address should not be used as a primary address without
computing the authorization decision. If the addresses are part of
the certificate then it is not necessary execute the weaker return
routability test. If multiple addresses are communicated to another
peer as part of the peer address set then some of these addresses
might be already verified even if the primary address is still
operational.

Another option is to use the [I-D.dupont-mipv6-3bombing] approach
which suggests to do a return routability test only if you have to
send an address update from some other address than the indicated
preferred address.

Finally it would be possible not to execute return routability checks
at all. In case of indirect change notifications then we only move
to the new preferred address after successful dead-peer detection on
the new address, which is already a return routability check. With a
direct notifications the authenticated peer may have provided an
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.

As such, it sees that there it is also a policy issue when to

schedule a return routability test.

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.

There are potential attacks if a return routability checks include
some kind of nonce. 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 true original recipient cannot be reached at
this address.

The IKEv2 NAT-T mechanism does not perform any return routability
checks. It simply uses the last seen source IP address used by the
other peer where to send response packets. An adversary can change
those IP addresses, and can cause the response packets to be sent to
wrong IP address. The situation is self-fixing when the adversary is
no longer able to modify packets and the first packet with true IP
address reaches the other peer. In a certain sense mobility handling
makes this attack difficult for an adversary since it needs to be
located somewhere along the path where the individual paths ({CoA1,
..., CoAn} towards the destination IP address) have a shared path or
if the adversary is located near the MOBIKE client then it needs to
follow the users mobility patterns. With IKEv2 NAT-T the genuine
client can cause third party bombing by redirecting all the traffic
pointed to him to third party. As the MOBIKE protocol tries to
provide equal or better security than IKEv2 NAT-T the MOBIKE protocol
should protect against these attacks.

There might be also return routability information available from the
other parts of the system too (as shown in Figure 3), but the checks
done might have a different quality. There are multiple levels for
return routability checks:

o None, no tests

o A party willing to answer is on the path to the claimed address.
   This is the basic form of return routability test.

o There is an answer from the tested address, and that answer was
   authenticated (including the address) to be from our peer.

o 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 MOBIKE packet too, then the
adversary cannot forward packets, thus it prevents 3rd party
bombings.

If the reply packet can be constructed without seeing the request
packet (for example, if there is no nonce, challenge or similar
mechanism to show liveness), then the genuine peer can cause 3rd
party bombing, by replying to those requests without seeing them at
all.

Other levels might only provide information that there is someone at
the IP address which replied to the request. There might not be an
indication that the reply was freshly generated or repeated, or the
request might have been transmitted from a different source address.

Requirements for a MOBIKE protocol for the return routability test
might be able to verify that there is same (cryptographically)
authenticated node at the other peer and it can now receive packets
from the IP address it claims to have.

5.7 Employing MOBIKE results in other protocols

If MOBIKE has learned about new locations or verified the validity of
an address through a return routability test, can this information be
useful for other protocols?

When considering the basic MOBIKE VPN scenario, the answer is no.
Transport and application layer protocols running inside the VPN
tunnel have no consideration about the outer addresses or their
status.

Similarly, IP layer tunnel termination at a gateway rather than a
host endpoint limits the benefits for "other protocols" that could be
informed -- all application protocols at the other side are running
in a node that is unaware of IPsec, IKE, or MOBIKE.

However, it is conceivable that future uses or extensions of the
MOBIKE protocol make such information distribution useful. For
instance, if transport mode MOBIKE and SCTP were made to work

together, it would likely be useful for SCTP to learn about the new
addresses at the same time as MOBIKE learns them. Similarly, various
IP layer mechanisms might make use of the fact that a return
routability test of a specific type has already been performed.
However, in all of these cases careful consideration should be
applied. This consideration should answer to questions such as
whether other alternative sources exist for the information, whether
dependencies are created between parts that prior to this had no
dependencies, and what the impacts in terms of number of messages or
latency are.

[I-D.crocker-celp] discussed the use of common locator information
pools in IPv6 multihoming context; it assumed that both transport and
IP layer solutions would be used for providing multihoming, and that
it would be beneficial for different protocols to coordinate their
results in some manner, for instance by sharing experienced
throughput information for address pairs. This may apply to MOBIKE
as well, assuming it co-exists with non-IPsec protocols that are
faced with the same multiaddressing choices.

Nevertheless, all of this is outside the scope of current MOBIKE base
protocol design and may be addressed in future work.

5.8 Scope of SA changes

Most sections of this document discuss design considers for updating
and maintaining addresses for the IKE-SA. However, changing the
preferred address also has an impact for IPsec SAs. The outer tunnel
header addresses (source IP and destination IP addresses) need to be
modified according to the preferred address pair to allow the IPsec
protected data traffic to travel along the same path as the MOBIKE
packets (if we only consider the IP header information).

The basic question is that how the IPsec SAs are changed to use the
new address pair (the same address pair as the MOBIKE signaling
traffic -- the preferred address pair). 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 pair. Another option
is to have separate exchange to move the IPsec SAs separately.

If IPsec SAs should be updated separately then a more efficient
format than notification payload is needed. A notification payload
can only store one SPI per payload. A separate payload which would
have list of IPsec SA SPIs and new preferred address. If there are
large number of IPsec SAs, those payloads can be quite large unless
ranges of SPI values are supported. 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. 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 has fast and/or
cheap connections and slow and/or expensive connections, and it wants
to allow moving some of the SAs to the slower and/or more expensive
connection, and prevents to 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, e.g.,
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.

5.9 Zero Address Set

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, e.g., 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, e.g., 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).

From a technical point of view this features addresses two aspects:

o There is no need to transmit IPsec data traffic. IPsec protected
   data needs to be dropped which saves bandwidth. This does not
   provide a functional benefit i.e, nothing breaks if this feature
   is not provided.

o MOBIKE signaling messages are also ignored and need to be
   suspended. The IKE-SA must not be deleted and the suspend
   functionality (realized with the zero address set) might require
   the IKE-SA to be tagged with a lifetime value since the IKE-SA
   will not be kept in memory an arbitrary amount of time. Note that
   the IKE-SA has no lifetime associated with it. In order to
   prevent the IKE-SA to be deleted the dead-peer detection mechanism
   needs to be suspended as well.

Due to the enhanced complexity of this extension a first version of
the MOBIKE protocol will not provide this feature.

5.10 IPsec Tunnel or Transport Mode

Current MOBIKE design is focused only on the VPN type usage and
tunnel mode. Transport mode behaviour would also be useful, but will
be discussed in future documents.

5.11 Message Representation

The basic IP address change notifications can be sent either via an
informational exchange already specified in the IKEv2, or via a
MOBIKE specific message exchange. Using informational exchange has
the main advantage that it is already specified in the IKEv2 and
implementations incorporated the functionality already.

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 indicate a preference value for each
address Furthermore, one of the addresses in the peer address set
must be labeled as the preferred address. 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. While two addresses can have the
same preference value an ordered list avoids this problem.

Even if load balancing is currently outside the scope of MOBIKE,
there might be future work to include this feature. The format
selected needs to be flexible enough to include additional
information (e.g., to enable load balancing). This might be
something like one reserved field, which can later be used to store
additional 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.

Having all IP addresses in one big 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. 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 a custom payload type, which then
includes the information needed for the MOBIKE protocol.

MOBIKE might send the full peer address list once one of the IP
addresses changes (either addresses are added, removed, the order
changes or the preferred address is updated) or an incremental
update. 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).

Note that each peer needs to communicate its peer address set to the
other peer.

6. **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.ietf-mip6-ro-sec] and [Aur02] for more information about
   flooding attacks.

7. IANA Considerations

   This document does not introduce any IANA considerations.

8. Acknowledgments

This document is the result of discussions in the MOBIKE working
group. The authors would like to thank Jari Arkko, Pasi Eronen,
Francis Dupont, Mohan Parthasarathy, Paul Hoffman, Bill Sommerfeld,
James Kempf, Vijay Devarapalli, Atul Sharma, Bora Akyol and Joe Touch
for their discussion input.

9. References

9.1 Normative references

   [I-D.ietf-ipsec-ikev2]
              Kaufman, C., "Internet Key Exchange (IKEv2) Protocol",
              draft-ietf-ipsec-ikev2-17 (work in progress), October
              2004.

   [I-D.ietf-ipsec-rfc2401bis]
              Kent, S. and K. Seo, "Security Architecture for the
              Internet Protocol", draft-ietf-ipsec-rfc2401bis-05 (work
              in progress), December 2004.

9.2 Informative References

   [I-D.arkko-multi6dt-failure-detection]
              Arkko, J., "Failure Detection and Locator Selection in
              Multi6", draft-arkko-multi6dt-failure-detection-00 (work
              in progress), October 2004.

   [RFC2409] Harkins, D. and D. Carrel, "The Internet Key Exchange
              (IKE)", RFC 2409, November 1998.

   [RFC2401] Kent, S. and R. Atkinson, "Security Architecture for the
              Internet Protocol", RFC 2401, November 1998.

   [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.

   [I-D.ietf-mip6-ro-sec]
              Nikander, P., "Mobile IP version 6 Route Optimization
              Security Design Background", draft-ietf-mip6-ro-sec-02
              (work in progress), October 2004.

   [I-D.ietf-hip-mm]
              Nikander, P., "End-Host Mobility and Multi-Homing with
              Host Identity Protocol", draft-ietf-hip-mm-00 (work in
              progress), October 2004.

   [I-D.crocker-celp]
              Crocker, D., "Framework for Common Endpoint Locator
              Pools", draft-crocker-celp-00 (work in progress), February
              2004.

   [RFC3489] Rosenberg, J., Weinberger, J., Huitema, C. and R. Mahy,

                "STUN - Simple Traversal of User Datagram Protocol (UDP)
                Through Network Address Translators (NATs)", RFC 3489,
                March 2003.

   [RFC2960]   Stewart, R., Xie, Q., Morneault, K., Sharp, C.,
                Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M.,
                Zhang, L. and V. Paxson, "Stream Control Transmission
                Protocol", RFC 2960, October 2000.

   [RFC3753]   Manner, J. and M. Kojo, "Mobility Related Terminology",
                RFC 3753, June 2004.

   [I-D.ietf-tsvwg-addip-sctp]
                Stewart, R., "Stream Control Transmission Protocol (SCTP)
                Dynamic Address Reconfiguration",
                draft-ietf-tsvwg-addip-sctp-09 (work in progress), June
                2004.

   [I-D.dupont-ikev2-addrmgmt]
                Dupont, F., "Address Management for IKE version 2",
                draft-dupont-ikev2-addrmgmt-06 (work in progress), October
                2004.

   [RFC3554]   Bellovin, S., Ioannidis, J., Keromytis, A. and R. Stewart,
                "On the Use of Stream Control Transmission Protocol (SCTP)
                with IPsec", RFC 3554, July 2003.

   [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.

Authors' Addresses

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

   EMail: kivinen@safenet-inc.com

Hannes Tschofenig
Siemens
Otto-Hahn-Ring 6
Munich, Bavaria 81739
Germany

EMail: Hannes.Tschofenig@siemens.com
URI: http://www.tschofenig.com