IKEv2 Mobility and Multihoming
(mobike)
Internet-Draft
Expires: April 24, 2006

T. Kivinen
Safenet, Inc.
H. Tschofenig
Siemens

October 21, 2005

Design of the MOBIKE Protocol
draft-ietf-mobike-design-04.txt

Status of this Memo

Copyright Notice

Abstract

The MOBIKE (IKEv2 Mobility and Multihoming) working group is developing extensions for the Internet Key Exchange Protocol version 2 (IKEv2). These extensions should enable an efficient management of IKE and IPsec Security Associations when a host possesses multiple IP addresses and/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. Design decisions for the MOBIKE protocol, background information and discussions within the working group are recorded.

Table of Contents

1. Introduction

The purpose of IKEv2 is to mutually authenticate two hosts, establish one or more IPsec Security Associations (SAs) between them, and subsequently manage these SAs (for example, by rekeying or deleting). IKEv2 enables the hosts to share information that is relevant to both the usage of the cryptographic algorithms that should be employed (e.g., parameters required by cryptographic algorithms and session keys) and to the usage of local security policies, such as information about the traffic that should experience protection.

IKEv2 assumes that an IKE SA is created implicitly between the IP address pair that is used during the protocol execution when establishing the IKEv2 SA. This means that, in each host, only one IP address pair is stored for the IKEv2 SA as part of a single IKEv2 protocol session, and, for tunnel mode SAs, the hosts places this single pair in the outer IP headers. Existing documents make no provision to change this pair after an IKE SA is created.

There are scenarios where one or both of the IP addresses of this pair may change during an IPsec session. In principle, the IKE SA and all corresponding IPsec SAs could be re-established after the IP address has changed. However, this can be problematic, as the device might be too slow for this task. Moreover, manual user interaction (for example when using SecurID cards) might be required as part of the IKEv2 authentication procedure. Therefore, an automatic mechanism is need that updates the IP addresses associated with the IKE SA and the IPsec SAs. MOBIKE provides such a mechanism.

The work of the MOBIKE working group and therefore this document is based on the assumption that the mobility and multi-homing extensions are developed for IKEv2 [I-D.ietf-ipsec-ikev2]. As IKEv2 is built on the architecture described in RFC2401bis [I-D.ietf-ipsec-rfc2401bis], all protocols developed within the MOBIKE working group must be compatible with both IKEv2 and the architecture described in RFC2401bis. The document does not aim to neither provide support IKEv1 [RFC2409] nor the architecture described in RFC2401 [RFC2401].

This document is structured as follows. After introducing some important terms in Section 2 a number of relevant usage scenarios are discussed in Section 3. The next section Section 4 will describe the scope of the MOBIKE protocol. Finally, Section 5 discusses design considerations affecting the MOBIKE protocol. The document concludes in Section 7 with security considerations.

2. Terminology

This section introduces the terminology that is used in this document.

Peer:

A peer is an IKEv2 endpoint. In addition, a peer implements the MOBIKE extensions, as defined in this and related documents.

Available address:

An address is said to be available if the following conditions are met:

* The address has been assigned to an interface.

* If the address is an IPv6 address, we additionally require (a) that the address is valid as defined in RFC 2461 [RFC2461], and (b) that the address is not tentative as defined in RFC 2462 [RFC2462]. In other words, we require the address assignment to be complete.

  Note that this explicitly allows an address to be optimistic as defined in [I-D.ietf-ipv6-optimistic-dad].

* If the address is an IPv6 address, it is a global unicast or unique site-local address, as defined in [I-D.ietf-ipv6-unique-local-addr]. That is, it is not an IPv6 link-local. Where IPv4 is considered, it is not an RFC 1918 [RFC1918] address.

* The address and interface is acceptable for sending and receiving traffic according to a local policy.

This definition is taken from [I-D.arkko-multi6dt-failure-detection].

Locally Operational Address:

An address is said to be locally operational if it is available and its use is locally known to be possible and permitted. 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, if and only if bidirectional connectivity can be shown between the two addresses. Note that sometimes it is

necessary to consider connectivity on a per-flow level between two endpoints needs to be tested. This differentiation might be necessary to address certain Network Address Translation types or specific firewalls. This definition is taken from [I-D.arkko-multi6dt-failure-detection] and adapted for the MOBIKE context. Although it is possible to further differentiate unidirectional and bidirectional operational address pairs, only bidirectional connectivity is relevant to this document and unidirectional connectivity is out of scope.

Path:

The sequence of routers traversed by the MOBIKE and IPsec packets exchanged between the two peers. Note that this path may be affected not only by the involved source and destination IP addresses, but also by the transport protocol. Since MOBIKE and IPsec packets have a different appearance on the wire they might be routed along a different path, for example by load balancers. This definition is taken from [RFC2960] and adapted to the MOBIKE context.

Primary Path:

The sequence of routers traversed by an IP packet that carries the default source and destination addresses is said to be the Primary Path. This definition is taken from [RFC2960] and adapted to the MOBIKE context.

Preferred Address:

The IP address of a peer to which MOBIKE and IPsec traffic should be sent by default. A given peer has only one active preferred address at a given point in time, except for the small time period where it switches from an old to a new preferred address. This definition is taken from [I-D.ietf-hip-mm] and adapted to the MOBIKE context.

Peer Address Set:

We denote the two peers of a MOBIKE session by peer A and peer B. A peer address set is the subset of locally operational addresses of peer A that is sent to peer B. A policy available at peer A indicates which addresses are included in the peer address set. Such a policy might be created either manually or automatically through interaction with other mechanisms that indicate new available addresses.

Terminology regarding NAT types (e.g. Full Cone, Restricted Cone,

Port Restricted Cone and Symmetric), can be found in Section 5 of [RFC3489]. For mobility related terminology (e.g. Make-before-break or Break-before-make) see [RFC3753].

3. Scenarios

In this section we discuss three typical usage scenarios for the MOBIKE protocol.

3.1. Mobility Scenario

Figure 1 shows a break-before-make mobility scenario where a mobile node changes its point of network attachment. Prior to the change, the mobile node had established an IPsec connection with a security gateway which offered, for example, access to a corporate network. The IKEv2 exchange that facilitated the set up of the IPsec SA(s) took place over the path labeled as 'old path'. The involved packets carried the MN's "old" IP address and were forwarded by the "old" access router (OAR) to the security gateway (GW).

When the MN changes its point of network attachment, it obtains a new IP address using stateful address configuration techniques or via the stateless address autoconfiguration mechanism. The goal of MOBIKE, in this scenario, is to enable the MN and the GW to continue using the existing SAs and to avoid setting up a new IKE SA. A protocol exchange, denoted by 'MOBIKE Address Update', enables the peers to update their state as necessary.

Note that in a break-before-make scenario the MN obtains the new IP address after it can no longer be reached at the old IP address. In a make-before-break scenario, the MN is, for a given period of time, reachable at both the old and the new IP address. MOBIKE should work in both the above scenarios.

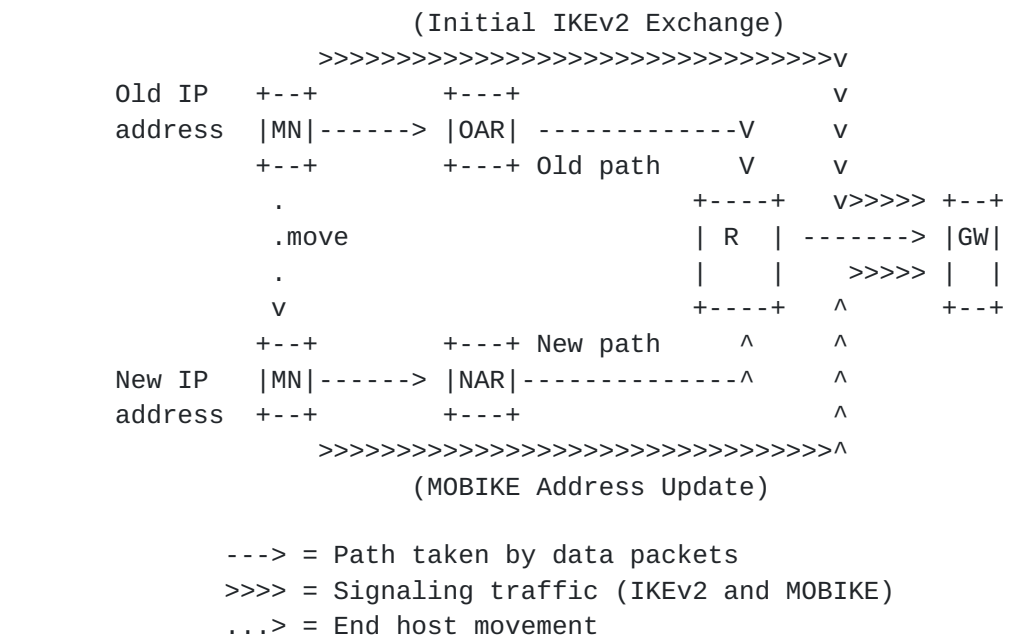

Figure 1: Mobility Scenario

3.2. Multihoming Scenario

Another MOBIKE usage scenario is depicted in Figure 2. In this scenario, the MOBIKE peers are equipped with multiple interfaces (and multiple IP addresses). Peer A has two interface cards with two IP addresses, IP_A1 and IP_A2, and peer B has two IP addresses, IP_B1 and IP_B2. Each peer selects one of its IP addresses as the preferred address which is used for subsequent communication. Various reasons, (e.g hardware or network link failures), may require a peer to switch from one interface to another.

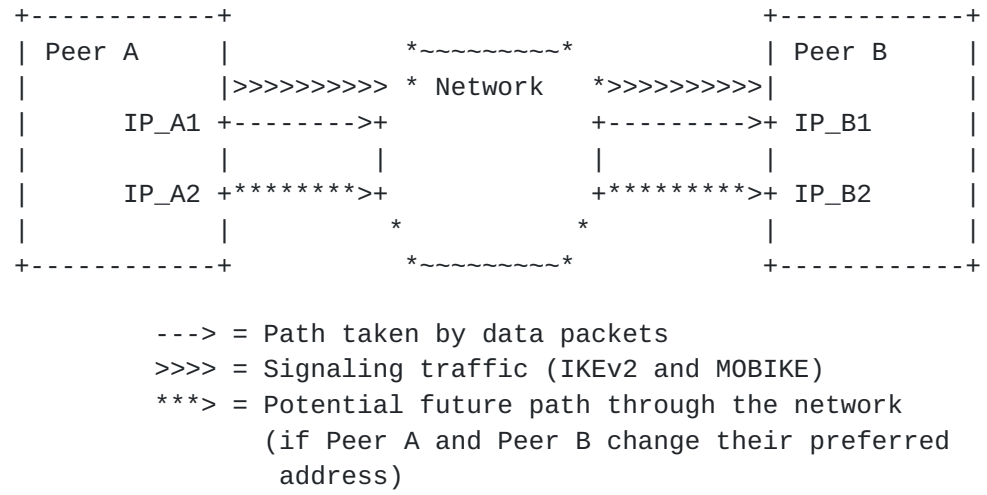

Figure 2: Multihoming Scenario

Note that MOBIKE does not aim to support load balancing between multiple IP addresses. That is, each peer uses only one of the available IP addresses at a given point in time.

3.3. Multihomed Laptop Scenario

The third scenario we consider is about a laptop, which has multiple interface cards and therefore several ways to connect to the network. It may for example have a fixed Ethernet card, a WLAN interface, a GPRS adaptor, a Bluetooth interface or USB hardware. Not all interfaces are connected to the network at all times for a number of reasons (e.g., cost, availability of certain link layer technologies, user convenience). The mechanism that determines which interfaces are connected to the network at any given point in time is outside the scope of the MOBIKE protocol and, as such, this document. However, as the laptop changes its point of attachment to the network, the set of IP addresses under which the laptop is reachable, changes too.

Even if IP addresses change due to interface switching or mobility, the IP address obtained via the configuration payloads within IKEv2 remain unaffected. The IP address obtained via the IKEv2 configuration payloads allow the configuration of the inner IP address of the IPsec tunnel. As such, applications might not detect any change at all.

4. Scope of MOBIKE

Getting mobility and multihoming actually working requires lots of different components working together, including coordinating things and making consistent decisions in several link layers, the IP layers, different mobility mechanisms in those layers, and IPsec/IKE. Most of those aspects are beyond the scope of MOBIKE: The MOBIKE focuses on what two peers need to agree in IKEv2 level (like new message formats and some aspects of their processing) for interoperability.

The MOBIKE is not trying to be full mobility protocol; there is no support for simultaneous movement or rendezvous mechanism, and there is no support for route optimization etc. This current design document focuses mainly on the tunnel mode, everything going inside the tunnel is unaffected by the changes in the tunnel header IP address, and this is the mobility feature provided by the MOBIKE. I.e. the applications running through the MOBIKE IPsec tunnel cannot even detect the movement, their IP address etc stay constant.

A MOBIKE protocol should be able to perform the following operations:

- inform the other peer about the peer address set
- inform the other peer about the preferred address
- test connectivity along a path and thereby to detect an outage situation
- change the preferred address
- change the peer address set
- Ability to deal with Network Address Translation devices

Figure 3 shows an example protocol interaction between a pair of MOBIKE peers. MOBIKE interacts with the IPsec engine using the PF_KEY API [RFC2367]. Using this API, the MOBIKE daemon can create entries in the Security Association (SAD) and Security Policy Databases (SPD). The IPsec engine may also interact with IKEv2 and MOBIKE daemon using this API. The content of the Security Policy and Security Association Databases determines what traffic is protected with IPsec in which fashion. MOBIKE, on the other hand, receives information from a number of sources that may run both in kernel-mode and in user-mode. Information relevant for MOBIKE might be stored in a database. The contents of such a database, along with the occurrence of events of which the MOBIKE process is notified, form the basis on which MOBIKE decides regarding the set of available

addresses, the peer address set, and the preferred address. Policies may also affect the selection process.

The a peer address set and the preferred address needs to be available to the other peer. In order to address certain failure cases, MOBIKE should perform connectivity tests between the peers (potentially over a number of different paths). Although a number of address pairs may be available for such tests, the most important is the pair (source address, destination address) of the primary path. This is because this pair is selected for sending and receiving MOBIKE signaling and IPsec traffic. If a problem along this primary path is detected (e.g., due to a router failure) it is necessary to switch to a new primary path. In order to be able to do so quickly, it may be helpful to perform connectivity tests of other paths periodically. Such a technique would also help in identifying previously disconnected paths that become operational.

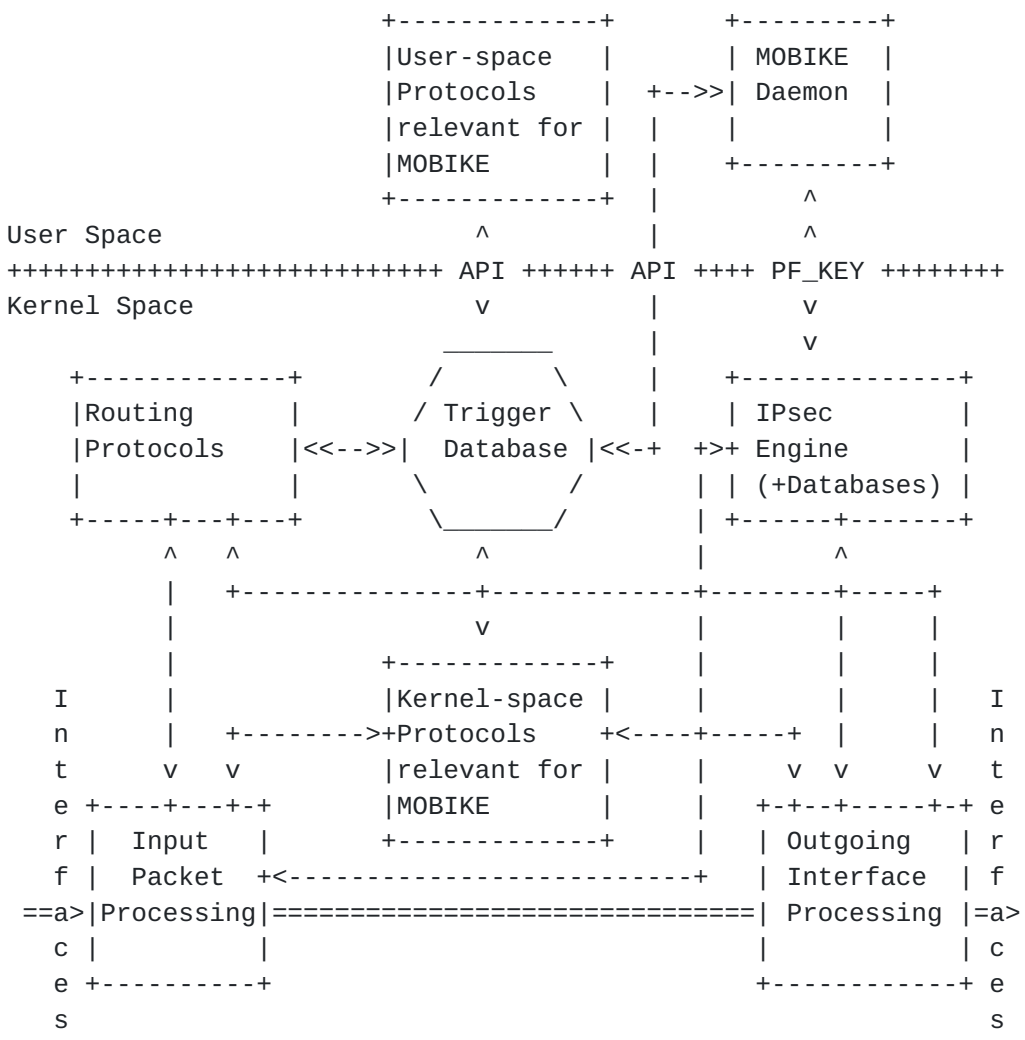

Figure 3: Framework

Please note that Figure 3 illustrates an example of how a MOBIKE implementation could work. Hence, it serves illustrative purposes only.

Extensions of the PF_KEY interface required by MOBIKE are also within the scope of the working group. Finally, certain optimizations for wireless environments are also covered.

5. Design Considerations

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

5.1. Choosing addresses

One of the core aspects of the MOBIKE protocol is the selection of the address for the IPsec packets we send. Choosing addresses for the IKEv2 request is somewhat separate problem: in many cases, they will be the same (and in some design choice they will always be the same).

5.1.1. Inputs and triggers

How the address changes are triggered are largerly beyond the scope of MOBIKE. The triggers can include e.g. changes in the set of addresses, various link-layer indications, failing dead peer detection, and changes in preferences and policies. Furthermore, there may be less reliable sources of information (such as lack of IPsec packets and ICMP packets) that do not trigger any changes directly, but rather cause DPD to be performed sooner than it otherwise would have been (and if that DPD fails, that may trigger changing of addresses).

These triggers are largerly the same as for, e.g. Mobile IP, and are beyond the scope of MOBIKE.

5.1.2. Connectivity

There can be two kind of "failures" in the connectivity; local or middle. Local failure is a property of an address (interface), while the failures in the middle is property of address pair. MOBIKE does not assume full connectivity, but it does not try to use unidirectional address pairs (multi6 has discussed about how to deal with unidirectional paths). Unidirectional address pairs is complicated issue, and supporting it would require abandoning the principle that you always send the IKEv2 reply to the address the request came from. Because of that MOBIKE decided to deal only with bidirectional address pairs. It does consider unidirectional address pairs as broken, and not use them, but the connection will not break even if unidirectional address pairs are present, provided there is at least one bidirectional address pair it can use.

Note, that MOBIKE is not really concerned about the actual path used, it cannot even detect if some path is unidirectional, if the same address pair has some other unidirectional path back. Ingress filters might still cause such path pairs to be unusable, and in that

case MOBIKE will detect that there is no connection between address pair.

In a sense having both IPv4 and IPv6 address is basically a case of partial connectivity (putting both IPv4 and IPv6 address in the same IP header does not work). The main difference is that it is known beforehand, and there is no need to discover that IPv4/IPv6 combination does not work.

5.1.3. Discovering connectivity

To detect connectivity, i.e failures in the middle, MOBIKE needs to have some kind of probe which it can send to the other end and get a reply back to that. If it will see the reply it knows the connection works, if it does not see the reply after multiple retransmissions it may assume that the address pair tested is broken.

The connectivity tests do require to take in to account the congestion problems, because the connection failure might be because of congestion, and the MOBIKE should not make it worse by sending lots of probe packets.

5.1.4. Decision making

One of the core decisions to the MOBIKE protocol is to who makes the decisions to fix situations, i.e. symmetry in decision making vs. asymmetry in decisions. Symmetric decision making may cause the different peers to make different decisions, thus causing asymmetric upstream/downstream traffic. In mobility case it is desirable that the mobile peer can move both upstream and downstream traffic to some particular interface, and this requires asymmetric decision making.

Working with stateful packet filters and NATs is easier if same address pair is used in both upstream and downstream directions. Also in common cases only the peer behind NAT can actually do actions to recover from the connectivity problems, as it might be that the other peer is not able to initiate any connections to the peer behind NAT.

5.1.5. Suggested approach

Because of those issues listed above, the MOBIKE protocol decided to select method where the initiator will decide which addresses are used. This has the benefits that it makes one peer to decide, thus we cannot end up in the asymmetric decisions, and it also works best with NATs, as the initiator is the most common peer to be behind NAT, and thus is the only peer which can recover in those cases.

5.2. NAT Traversal

5.2.1. Background and constraints

Another core aspect of the MOBIKE was the co-operation and working with NATs. In IKEv2 the tunnel header IP addresses are not sent inside the IKEv2 payloads, and thus there is no need to do unilateral self-address fixing (UNSAF). The tunnel header IP addresses are taken from the outer IP header of the IKE packets, thus they are already processed by the NAT.

The NAT detection payloads are used to detect if the addresses in the IP header were modified by a NAT between the peers, and that enables UDP encapsulation of ESP packets if needed. MOBIKE is not to change how IKEv2 NAT-T works, in particular, any kind of UNSAF or explicit interaction with NATs (e.g. midcom or nsis natfw) are beyond the scope. The MOBIKE will need to define how MOBIKE and NAT-T are used together.

The NAT-T support should also be optional, i.e. if the IKEv2 implementation does not implement NAT-T, since it is not required in some particular environment, implementing MOBIKE should not require adding support for NAT-T as well.

The property of being behind NAT is actually property of the address pair, thus one peer can have multiple IP-addresses and some of those might be behind NAT and some might not be behind NAT.

5.2.2. Fundamental restrictions

There are some cases which cannot be made work with the restrictions provided by the MOBIKE charter. One of those cases is the case where the party "outside" a symmetric NAT changes its address to something not known by the the other peer (and old address has stopped working). It cannot send a packet containing the new addresses to the peer, because the NAT does not contain the necessary state. Furthermore, since the party behind the NAT does not know the new IP address, it cannot cause the NAT state to be created.

This case could be solved using some rendezvous mechanism outside IKEv2, but that is beyond the scope of MOBIKE.

5.2.3. Moving to behind NAT and back

MOBIKE provides mechanism where peer not initially behind the NAT, can move to behind NAT, when new address pair is selected. MOBIKE does not need to detect when someone attach NAT to the currently working address pair, i.e. the NAT detection is only done when MOBIKE

changes the address pair used.

Similarly the MOBIKE provides mechanism to move from the address pair having NAT to the address pair not having NAT.

As we only use one address pair at time, effectively MOBIKE peer is either behind NAT or not behind NAT, but each address change can change the situation. Because of this and because initiator always chooses the addresses it is enough to send keepalive packets only to that one address pair.

5.2.4. Responder behind NAT

MOBIKE can work in cases where the responder is behind static NAT, but in that case the initiator needs to know all possible addresses where the responder can move to, i.e. responder cannot move to the address which is not known by the initiator.

If the responder is behind NAPT then it might need to communicate with the NAT to create mapping so initiator can connect to it. Those external hole punching mechanisms are beyond the scope of MOBIKE.

In case the responder is behind NAPT then also finding the port numbers used by the responder, is outside the scope of MOBIKE.

5.2.5. NAT Prevention

One new feature created by the MOBIKE, is the NAT prevention, i.e. if we detect NAT between the peers, we do not allow that address pair to be used. This can be used to protect IP-addresses in cases where it is known by the configuration that there is no NAT between the nodes (for example IPv6, or fixed site-to-site VPN). This gives extra protection against 3rd party bombing attacks (attacker cannot divert the traffic to some 3rd party). This feature means that we authenticate the IP-address and detect if they were changed. As this is done on purpose to break the connectivity if NAT is detect, and decided by the configuration, there is no need to do UNSAF processing.

5.2.6. Suggested approach

The working group decided that MOBIKE uses NAT-T mechanisms from the IKEv2 protocol as much as possible, but decided to change the dynamic address update for IKEv2 packets to MUST NOT (it would break path testing using IKEv2 packets). Working group also decided to only send keepalives to the current address pair.

5.3. Scope of SA changes

Most sections of this document discuss design considerations for updating and maintaining addresses in the database entries that relate to an IKE-SA. However, changing the preferred address also affects the entries of the IPsec SA database. The outer tunnel header addresses (source and destination IP addresses) need to be modified according to the primary path 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). If the MOBIKE messages and the IPsec protected data traffic travel along a different path then NAT handling is severely complicated.

The basic question is then how the IPsec SAs are changed to use the new address pair (the same address pair as the MOBIKE signaling traffic). 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 a separate exchange to move the IPsec SAs separately.

If IPsec SAs should be updated separately then a more efficient format than the notification payload is needed to preserve bandwidth. A notification payload can only store one SPI per payload. A separate payload could have list of IPsec SA SPIs and new preferred address. If there is a 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 from IKE SA, i.e. no need to store IP addresses per IPsec SA. 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 SA address set 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 prevent the move, for example, of 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.

MOBIKE protocol decided to move all IPsec SAs implicitly when the IKE

SA address pair changes. If more granular handling of the IPsec SA is required, then multiple IKE SAs can be created one for each set of IPsec SAs needed.

5.4. Zero address set functionality

One of the features which is potentially useful is for the peer to announce 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 it 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 and therefore stop sending traffic. 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 indicate how long the peer should allow remote peers to remain disconnected.

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

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

- MOBIKE signaling messages are also ignored. The IKE-SA must not be deleted and the suspend functionality (realized with the zero address set) may require the IKE-SA to be tagged with a lifetime value since the IKE-SA should not be kept in alive for an undefined period of time. Note that IKEv2 does not require that the IKE-SA has a lifetime associated with it. In order to prevent the IKE-SA from being deleted the dead-peer detection mechanism needs to be suspended as well.

Due to the fact that this extension could be complicated and there was no clear need for it, a first version of the MOBIKE protocol will not provide this feature.

5.5. Return routability test

Changing the preferred address and subsequently using it 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 determine the answer to this question:

- The addresses a peer is using are 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 appear in the certificate.

- A return routability check is performed by the remote peer before the address is updated in that peer's Security Association Database. This is done in order provide a certain degree of confidence to the remote peer that local peer is reachable at the indicated address.

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

A MOBIKE peer should not use an IP addressed provided by another MOBIKE peer as a primary address without computing the authorization decision. If the addresses are part of the certificate then it is not necessary to execute the weaker return-routability test. The return-routability test is a form of authorization check, although it provides weaker guarantees then the inclusion of the IP address as part of a certificate. If multiple addresses are communicated to the remote peer then some of these addresses may 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 perform a return routability test only when an address update needs to be sent from some address other than the indicated preferred address.

Finally it would be possible not to execute return routability checks at all. In case of indirect change notifications we only move to the new preferred address after successful dead-peer detection (i.e., a response to a DPD test) on the new address, which is already a return routability check. With a direct notification the authenticated peer may have provided an authenticated IP address. Thus it is would be possible to simply trust the MOBIKE peer to provide a proper IP address. There is no way an adversary can successfully launch an attack by injecting faked addresses since it does not know the IKE SA and the corresponding keying material. A protection against an internal attacker, i.e. the authenticated peer forwarding its traffic to the new address, is not provided. This might be an issue when extensions are added to IKEv2 that do not require authentication of end points (e.g., opportunistic security using anonymous Diffie-Hellman). On the other hand we know the identity of the peer in that

case.

There is also a policy issue when to schedule a return routability test. Before moving traffic? After moving traffic?

The basic format of the return routability check could be similar to dead-peer detection. There are potential attacks if a return routability check does not include some kind of nonce. The valid end point could send an address update notification for a third party, trying to get all the traffic to be sent there, causing a denial of service attack. If the return routability checks does not contain any cookies or other random information not known to the other end, then that valid node could reply to the return routability checks even when it cannot see the request. This might cause a peer to move the traffic to a location where the original recipient cannot be reached.

The IKEv2 NAT-T mechanism does not perform return routability checks. It simply uses the last seen source IP address used by the other peer as the destination address 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 an unmodified IP address reaches the other peer. Mobility environments make this attack more difficult for an adversary since it requires the adversary to be located somewhere on 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 user 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 mechanism it should protect against these attacks.

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

- None, no tests
- A party willing to answer the return routability check is located along 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, integrity and replay protected.

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

The 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 carried inside the MOBIKE payload, then the adversary cannot forward packets. Thus 3rd party bombings are prevented.

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 a guarantee that there is a node at the IP address which replied to the request. There is no indication as to whether or not the reply is fresh, and whether or not the request may have been transmitted from a different source address.

5.5.1. Employing MOBIKE results in other protocols

If MOBIKE has learned about new locations or verified the validity of a remote address through a return routability check, 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 are unaware of 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 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 potentially be useful for SCTP to learn about the new addresses at the same time as MOBIKE. Similarly, various IP layer mechanisms may make use of the fact that a return routability test of a specific type has been performed. However, care should be exercised in all these situations.

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

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

5.5.2. Suggested approach

MOBIKE protocol selected to use IKEv2 INFORMATIONAL exchanges as a return routability tests, but added random cookie there to prevent redirections done by authenticated attacker. Return routability tests are done by default before moving the traffic. However these tests are optional. Nodes MAY also perform these tests upon their own initiative at other times.

It is worth noting that the return routability test in MOBIKE is not he same as return routability test in MIP6: The MIP6 WG decided that it is not necessary to do return routability tests between the mobile node and the home agent at all.

5.6. IPsec Tunnel or Transport Mode

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

6. Protocol detail issues

6.1. Indicating support for mobike

In order for MOBIKE to function, both peers must implement the MOBIKE extension of IKEv2. If one or none of the peers supports MOBIKE, then, whenever an IP address changes, IKEv2 will have to be re-run in order to create a new IKE SA and the respective IPsec SAs. In MOBIKE, a peer needs to be confident that its address change messages are understood by the other peer. If these messages are not understood, it is possible that connectivity between the peers is lost.

One way to ensure that a peer receives feedback on whether or not its messages are understood by the other peer, is by using IKEv2 messaging for MOBIKE and to mark some messages as "critical". According to the IKEv2 specification, such messages either have to be understood by the receiver, or an error message has to be returned to the sender.

A second way to ensure receipt of the above-mentioned feedback is by using Vendor ID payloads that are exchanged during the initial IKEv2 exchange. These payloads would then indicate whether or not a given peer supports the MOBIKE protocol.

A third approach would use the Notify payload which is also used for NAT detection (via NAT_DETECTION_SOURCE_IP and NAT_DETECTION_DESTINATION_IP payloads).

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

Note that a MOBIKE peer could also attempt to execute MOBIKE opportunistically with the critical bit set when an address change has occurred. The drawback of this approach is, however, that an unnecessary message exchange is introduced.

Although Vendor ID payloads and Notifications are technically equivalent, Notifications are already used in IKEv2 as a capability negotiation mechanism. Hence, notification payloads are used in the MOBIKE to indicate support of it.

Also as the information of the support of the MOBIKE is not needed during the IKE_SA_INIT exchange, the indication of the support is done inside the IKE_AUTH exchange. The reason for this is to need to keep the IKE_SA_INIT messages as small as possible, so they do not get fragmented. The idea is that responder can do stateless processing of the first IKE_SA_INIT packet, and request cookie from

the other end if it is under attack. To mandate responder to be able to reassemble initial IKE_SA_INIT packets would not allow fully stateless processing of the initial IKE_SA_INIT packets.

6.2. Path Testing and Window size

As the IKEv2 has the window of outgoing messages, and the sender is not allowed to violate that window (meaning, that if the window is full, then he cannot send packets), it do cause some complications to the path testing. The another complication created by IKEv2 is that once the message is first time sent to the other end, it cannot be modified in its future retransmissions. This makes it impossible to know what packet actually reached first to the other end. We cannot use IP headers to find out which packet reached the other end first, as if responder gets retransmissions of the packet it has already replied (and those replies might have been lost due unidirectional address pair), it will retransmit the previous reply using the new address pair of the request. Because of this it might be possible that the responder has already used the IP-address information from the header of the packet, and the reply packet ending up to the initiator has different address pair.

Another complication comes from the NAT-T. The current IKEv2 document says that if NAT-T is enabled the node not behind NAT SHOULD detect if the IP-address changes in the incoming authenticated packets, and update the remote peers addresses accordingly. This works fine with the NAT-T, but it causes some complications in the MOBIKE, as it needs an ability to probe the another address pairs, without breaking the old one.

One approach to fix those would be to add completely new protocol that is outside the IKE SA message id limitations (window code), outside identical retransmission requirements, and outside the dynamic address updating of the NAT-T.

Another approach is to make the protocol so that it does not violate window restrictions and does not require changing the packet is sent, and change the dynamic address updating of NAT-T to MUST NOT in case MOBIKE is used. To not to violate window restrictions, it means that the addresses of the currently ongoing exchange needs to be changed, to test different paths. To not to require changing the packet after it is first sent, requires that the protocol needs to restart from the beginning in case packet was retransmitted to different addresses (so sender does not know which packet was the one that responder got first, i.e. which IP-addresses it used).

MOBIKE protocol decided to use normal IKEv2 exchanges for the path testing, and decided to change the dynamic address updating of NAT-T

to MUST NOT.

6.3. Message presentation

The 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 incorporate the functionality already.

Another question is the format of the address update notifications. The address update notifications can include multiple addresses, of which some may be IPv4 and some IPv6 addresses. The number of addresses is most likely going to be limited in typical environments (with less than 10 addresses). The format may need to indicate a preference value for each address. The format could either contain a preference number that determines the relative order of the addresses, or it could simply be ordered, according to preference, list of IP addresses. While two addresses can have the same preference value an ordered list avoids this situation.

Even if load balancing is currently outside the scope of MOBIKE, future work might include support for it. The selected format needs to be flexible enough to include additional information (e.g. to enable load balancing). This may be realized with an reserved field, which can later be used to store additional information. As there may arise other information which may have to be tied to an address in the future, a reserved field seems like a prudent design in any case.

There are two formats that place IP address lists into a message. One includes 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 with each one having carrying one IP address makes the protocol probably easier to parse, as we can already use the normal IKEv2 payload parsing procedures. It also offers an 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 IKEv2 payloads that are longer than a certain threshold, but if the sender sends them in the most preferred first, the receiver can only use the first addresses.

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. It also avoids problems of packets arriving in an order different from what they were sent.

Another 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 decided to use IKEv2 NOTIFY payloads, and put only one data item per notify, i.e. there will be one NOTIFY payload for each item to be sent.

6.4. Updating address list

Because of the initiator decides, the initiator needs to know all the addresses used by the responder. The responder needs also that list in case it happens move to the address unknown by the initiator, and needs to send address update notify to the initiator, and it might need to try different addresses for the initiator.

MOBIKE could send the full peer address list every time any 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).

MOBIKE decided to use protocol format, where both ends can send full list of their addresses to the other end, and that list overwrites the previous list. To support NAT-T the IP-addresses of the received packet is added to the list (and they are not present in the list).

7. Security Considerations

As all the messages are already authenticated by the IKEv2 there is no problem that any attackers would modify the contents of the packets. The IP addresses in the 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, and that do not cause any direct actions.

An 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.

One type of attack that needs to be taken care of in the MOBIKE protocol is the 'flooding attack' type. See [I-D.ietf-mip6-ro-sec] and [Aur02] for more information about flooding attacks.

8. IANA Considerations

This document does not introduce any IANA considerations.

9. 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, Joe Touch, Udo Schilcher, Tom Henderson, Andreas Pashalidis and Maureen Stillman for their input.

We would like to particularly thank Pasi Eronen for tracking open issues on the MOBIKE mailing list. He helped us to make good progress on the document.

10. References

10.1. 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-06 (work in progress), April 2005.

10.2. 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-02 (work in progress), June 2005.

[I-D.ietf-mip6-ro-sec]
Nikander, P., "Mobile IP version 6 Route Optimization Security Design Background", draft-ietf-mip6-ro-sec-03 (work in progress), May 2005.

[I-D.ietf-hip-mm]
Nikander, P., "End-Host Mobility and Multihoming with the Host Identity Protocol", draft-ietf-hip-mm-02 (work in progress), July 2005.

[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-12 (work in progress), June 2005.

[I-D.dupont-ikev2-addrmgmt] Dupont, F., "Address Management for IKE version 2", draft-dupont-ikev2-addrmgmt-07 (work in progress), May 2005.

[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.

[I-D.ietf-ipv6-optimistic-dad] Moore, N., "Optimistic Duplicate Address Detection for IPv6", draft-ietf-ipv6-optimistic-dad-06 (work in progress), September 2005.

[I-D.ietf-ipv6-unique-local-addr] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", draft-ietf-ipv6-unique-local-addr-09 (work in progress), January 2005.

[RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.

[RFC2367] McDonald, D., Metz, C., and B. Phan, "PF_KEY Key Management API, Version 2", RFC 2367, July 1998.

[RFC2462] Thomson, S. and T. Narten, "IPv6 Stateless Address Autoconfiguration", RFC 2462, December 1998.

[RFC2461] Narten, T., Nordmark, E., and W. Simpson, "Neighbor

Discovery for IP Version 6 (IPv6)", RFC 2461, December 1998.

[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

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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