Network Working Group
Internet-Draft
Expires: January 7, 2005

P. Eronen
Nokia

July 9, 2004

# Mobility Protocol Options for IKEv2 (MOPO-IKE)
draft-eronen-mobike-mopo-00.txt

## Status of this Memo



## Copyright Notice



## Abstract

This document describes a mobility and multihoming extension to the IKEv2 protocol. The main purpose of this extension is to update the (outer) addresses associated with IKE and IPsec Security Associations. The extension also includes features that assist in selecting which addresses to use, and verify return routability during and after updates. It is also able to work together with NAT Traversal in some scenarios.





# 1. Introduction

## 1.1 Features

This specification includes the following features. Note that some of them may be useful even when the endpoints are not mobile or multi-homed.

Continued return routability

Before establishing a CHILD_SA, IKEv2 verifies that the peer can receive packets at the address it uses as the source address (except in one corner case involving NAT translation, discussed in Section 4). However, this is done only when the IKE_SA is established, and does not guarantee that the peer stays at that address. In addition, if NAT Traversal is used, the address can be updated due to changes in NAT mappings.

This feature adds a payload that can be used in INFORMATIONAL exchanges to verify not only peer liveness ("dead peer detection"), but also the continued ability to receive packets at the given address ("return routability").

Additionally, the "Updating addresses in IKE and IPsec SAs" feature (described below) verifies the return routability of before modifying IPsec SAs.

NAT prevention

IKEv2/IPsec implementations that do not support NAT Traversal can, in fact, work across some types of one-to-one "basic" NATs and IPv4/IPv6 translation agents in tunnel mode. Some people feel that this is a problem that needs to be fixed, since in some sense any modification of the IP addresses could be considered to be an attack.

This feature adds a payload that can be used to verify that the addresses in the IP header have not been modified.

UDP encapsulation without NATs

There are cases when UDP encapsulation is needed even when no NATs are present. A typical example would be a stateful firewall that performs similar filtering as a NAT, but does not change the IP addresses (and therefore is not detected by NAT_DETECTION payloads).

This feature allows using UDP encapsulation without using the





other features of NAT Traversal, such as automatic update of peer address.

Path testing

Some MOBIKE protocol proposals have (implicitly) assumed that when something occurs, the parties know what is required to correct the situation. This assumption is not necessarily true when the only indication of a problem is a lack of responses to IKE requests.

The path testing features allows parties to find out what action is required when no responses are received; that is, to find a path (combination of addresses) that still works. It also removes the need to configure information about (lack of) routing relationships in the case where not all possible combinations of addresses work. Additionally, the PATH_TEST exchange plays a part in checking return routability before address updates.

Updating addresses in IKE and IPsec SAs

This feature allows each peer to notify the other peer of the addresses it has, update these in case of change due to e.g. mobility, and update the addresses used in IKE and IPsec SAs. Optionally this also includes updating NAT Traversal related state associated with IPsec SAs (that is, enabling and disabling NAT Traversal as needed).

## 1.2 Features not provided

- This extension considers only tunnel mode IPsec Security Associations. It does not modify the traffic selectors in the SPD or inbound IPsec SAs.

- This extension does not fully support all possible scenarios involving NATs. Many common cases do work, though.

- This extension does not provide any kind of load balancing between different addresses or Security Associations.

- This extension does not support the "zero address set" functionality, i.e. temporarily forwarding the traffic of some SA to /dev/null.

## 1.3 Security association viewpoint

The main purpose of this extension is to modify state associated with





IKE_SA and IPsec SAs that is normally initialized when the SA is created, and not changed afterwards.

In particular, this extension considers the following state associated with IKE_SA and outbound IPsec SAs (conceptually speaking; an implementation could store this information in some other way as well):

- IKE_SA
  - local_address (source address for IKE requests)
  - local_port (source port for IKE requests, either 500 or 4500)
  - peer_address (destination address for IKE requests)
  - peer_port (destination port for IKE requests)
- outbound IPsec SAs
  - local_address (tunnel header source address)
  - peer_address (tunnel header destination address)
  - peer_port (destination port if UDP encapsulation is used)

* udp_encapsulation flag

* send_keepalives flag

* automatically_update_peer_address flag

Note that both IKE_SA and outbound IPsec SAs are considered to have a single pair of (source,destination) addresses at a time. These are the addresses used for IKE requests (including retransmissions of previous requests) and outbound ESP/AH packets.

In addition, the IKE_SA contains additional state specific to this extension. This state is used to to store information about addresses that are not currently active (see Section 7 for details).

This extension does not modify the SPD or inbound IPsec SAs.

## 1.4 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [1].





IPsec Security Association (SA)

An ESP or AH Security Association.

Path

A particular combination of source IP address and destination IP

address (and possibly ports?).

## 2. Signaling support for this specification

Implementations that support this specification MUST include a Vendor ID payload in the IKE_SA_INIT exchange (first two messages). The value for this payload is XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX (TBD).

This specification includes several optional features. In particular, implementations are not required to support the following aspects:

- Sending NAT_PREVENTION payloads.
- NAT_DETECTION_SOURCE_IP and NAT_DETECTION_DESTINATION_IP payloads.
- USE_UDP_ENCAPSULATION payload.

## 3. Continued return routability

In IKEv2, an empty INFORMATIONAL exchange does not guarantee return routability, since the peer can generate the response without actually seeing the request.

To improve this situation, a sender of an INFORMATIONAL request MUST include a COOKIE2 notification payload in the message. The data associated with this notification MUST be between 8 and 64 octets in length (inclusive), and MUST be chosen in a way that is unpredictable to the recipient.

The recipient of an INFORMATIONAL request MUST copy the payload as-is to the response. When processing the response, the original sender MUST verify that the values is the same as sent. If the values do not match, the IKA_SA MUST be closed (TBD details).

The Notify Message Type for this message is specified in Section 10. The Protocol ID field is set to one (1), and SPI Size is set to zero.

## 4. NAT prevention

IKEv2/IPsec implementations that do not support NAT Traversal can, in fact, work across some types of one-to-one "basic" NATs and IPv4/IPv6 translation agents in tunnel mode. Some people feel that this is a problem that needs to be fixed, since in some sense any modification of the IP addresses could be considered to be an attack.

This specification addresses the issue as follows. When an IPsec SA is created, the tunnel header IP addresses (and port if doing UDP encapsulation) are taken from the IKE_SA, not the message IP header. NAT_PREVENTION payloads are used to guarantee that NATs have not modified the address used in IKE_SA. However, all response messages are still sent to the address and port the corresponding request came from.

The initiator MAY include a NAT_PREVENTION payload in an IKE_SA_INIT request. The data associated with this notification is the SHA-1 hash [4] of the following data: the IP address and port from which the packet was sent, and the IP address and port to which the packet was sent. The Notify Message Type for this message is specified in Section 10. The Protocol ID field is set to one (1), and SPI Size is set to zero.

The responder MUST compare the NAT_PREVENTION payload with the values from the IP header. If they do not match, the responder replies with "HDR(A,0), N(NAT_PREVENTED)" and does not create any state.

If the values do match, the responder initializes (local_address, local_port, peer_address, peer_port) in the to-be-created IKE_SA with values from the IP header. The same applies if neither NAT_PREVENTION nor NAT_DETECTION_* payloads were included, or if the responder does not support NAT Traversal.

If the IKE_SA_INIT request included NAT_DETECTION_* payloads but no NAT_PREVENTION payload, the situation is different since the initiator may at this point change from port 500 to 4500. In this case, the responder initializes (local_address, local_port, peer_address, peer_port) from the first IKE_AUTH request, and schedules an INFORMATIONAL exchange to be sent soon after the IKE_AUTH exchanges have been completed.

IKEv2 requires that if an IPsec endpoint discovers a NAT between it and its correspondent, it MUST send all subsequent traffic to and from port 4500. To simplify things, implementations that support both this specification and NAT Traversal MUST change to port 4500 if the correspondent also supports both, even if no NAT was detected between them.





The initiator initializes its IKE_SA with the values used for sending the first IKE_AUTH request.

The use of NAT_PREVENTION payloads with later updates is described in Section 7.

## 5. UDP encapsulation without NATs

There are cases when UDP encapsulation is needed even when no NATs are present. A typical example would be a stateful firewall that performs similar filtering as a NAT, but does not change the IP addresses (and therefore is not detected by NAT_DETECTION payloads).

This feature allows using UDP encapsulation without using the other features of NAT Traversal, such as automatic update of peer address.

To enable this feature, a peer MAY include a USE_UDP_ENCAPSULATION

notification payload in a request message that also includes an SA payload requesting a CHILD_SA, or includes a CHANGE_PATH payload. If the recipient supports this feature and its use is allowed by local policy, it includes a USE_UDP_ENCAPSULATION notification payload in the response.

The Notify Message Type for this message is specified in Section 10. The Protocol ID field is set to one (1), and SPI Size is set to zero. There is no data associated with this Notify type.

## 6. Path testing

Some MOBIKE protocol proposals have (implicitly) assumed that when something occurs, the parties know what is required to correct the situation. This assumption is not necessarily true when the only indication of a problem is a lack of responses to IKE requests.

The path testing feature allows parties to find out what action is required when no responses are received; that is, to find a path (combination of addresses) that still works. It also removes the need configure information about (lack of) routing relationships in the case where not all possible combinations of addresses work. Additionally, the PATH_TEST exchange plays a part in checking return routability before address updates.

If both parties have several addresses, path testing may require testing all N*M combinations, even when only failures at the "first" hop (local link) are considered. To see why this is the case, consider a case where endpoint A has N links to a global "Internet cloud" and endpoint B has M links. If all but one of A's and B's links are down, finding the one that works requires either local

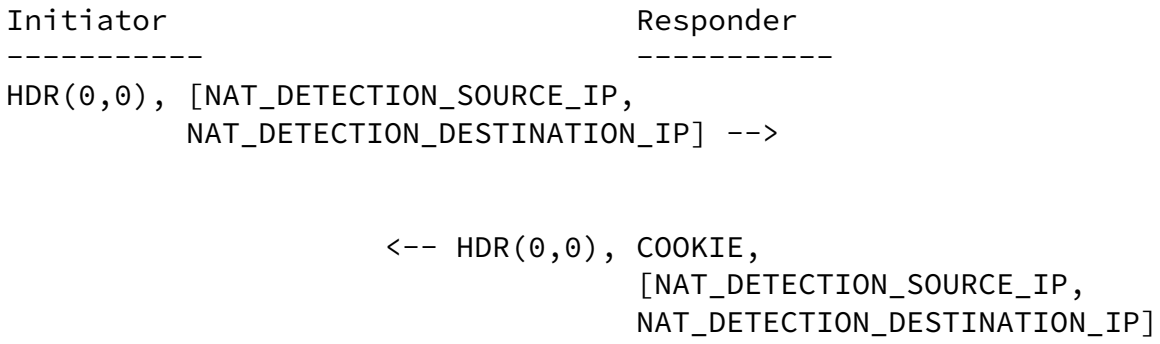




information (something better than lack of responses to IKE requests), or trying N*M combinations.

In general, it may also be the case that not addresses have routing between them. For instance, A and B might have IP connections, one from ISP1 (with addresses A1 and B1), and another one from ISP2 (with addresses A2 and B2). In this case, combinations (A1,B2) or (A2,B1) do not necessarily work. Thus, when one of the links goes down, it is necessary that both ends change their addresses simultaneously (changing them one-by-one does not necessarily work).

To overcome these limitations, a new IKEv2 exchange type, PATH_TEST, is introduced. This exchange is not part of any IKE_SA, so it cannot be cryptographically protected. It also does not result in the responder keeping any state.




Performing path testing over several different paths is not required if the node has other information that enables it to select which path should be used. In this case, a single PATH_TEST exchange to retrieve a COOKIE is sufficient.

Implementations MAY do path testing even if the currently used path is working to e.g. detect when a better but previously unavailable path becomes available, or to speed up recovery in fault situations.

Implementations that perform path testing MUST take steps to avoid causing unnecessary congestion. TBD: add some more details here.

## 7. Updating addresses in IKE and IPsec SAs

Finally, we get to the part of this document that actually explains how the IKE and IPsec Security Associations are updated.

This extension is based on the idea that same as in ordinary IKEv2, the initiator decides what addresses are used in the IPsec SAs. That

is, the responder never updates any IPsec SAs without receiving an explicit CHANGE_PATH request from the initiator. As described below, the responder can however update the IKE_SA in some circumstances.





An implementation of this specification maintains some additional information associated with the IKE_SA. This includes the latest_update_received and latest_update_sent counters, a pending_update flag, additional_addresses list, and results of path testing.

## 7.1 In the beginning

Both the initiator and responder MAY include one or more ADDITIONAL_ADDRESS notification payloads in the IKE_AUTH exchange (in case of multiple IKE_AUTH exchanges, in the message containing the SA payload).

The recipient stores this information, together with peer_address/ peer_port from the IKE_SA, to the "additional_addresses" list in the IKE_SA.

The Notify Message Type for this message is specified in Section 10. The Protocol ID field is set to one (1), and SPI Size is set to zero. The data associated with this Notify type is either an IPv4 address or an IPv6 address (the type is determined by payload length).

## 7.2 Updates by responder

When the responder's set of addresses changes, it proceeds as follows.

o If the current path in IKE_SA is no longer valid (e.g. the

current local_address is no longer in the set), it uses path testing to select new (local_address, peer_address, peer_port) from (local addresses) X (additional_addresses)

- Updates (local_address,peer_address,peer_port) in IKE_SA.

- Sets the pending_update to flag.

- When window size allows, sends an INFORMATIONAL request containing the following payloads:

  HDR, SK {N(ADDITIONAL_ADDRESS), [N(ADDITIONAL_ADDRESS), ..., ],
           N(COOKIE2), [NAT_PREVENTION]} -->

  and clears the "pending_update" flag. The message includes one ADDITIONAL_ADDRESS for each address the responder has (and is willing to use with this peer), including the one used in IP header.

When the initiator receives this, it





- If the NAT_PREVENTION payload is present, TBD.

- Compares the Message ID with the latest_update_received counter in the IKE_SA. If latest_update_received is greater than this one, a reply is sent but the addresses are not updated.

- Updates the latest_update_received counter in the IKE_SA.

- Replaces the additional_addresses list in IKE_SA with this list, and if NAT_PREVENTION was not present, also the address from the

IP header (TBD).

- Replies with "HDR,SK {N(COOKIE2)}".

- If current peer_address is NOT contained in additional_addresses, triggers an update to be done (described at the next section).

When the responder receives the reply, it

- Verifies the COOKIE2 payload as described in Section 3.

## 7.3 Updates by initiator

When the initiator wishes to change the path, it does the following:

- Uses the PATH_TEST exchange to obtain a COOKIE for the new local_address (if it does not already have one).

- Updates IKE_SA with the new (local_address, peer_address, peer_port) information.

- Sets pending_update flag.

- When the window size allows, sends an INFORMATIONAL request

```
HDR, SK {N(CHANGE_PATH), N(COOKIE), N(COOKIE2), N(ADDITIONAL_ADDRESS),..
          [N(NAT_DETECTION_*),]
          [N(NAT_PREVENTION)]} -->
```

  and clears the pending_update flag and sets the latest_update_sent to the Message ID of this message. The message includes one ADDITIONAL_ADDRESS for each address the responder has (and is willing to use with this peer), including the one used in IP header.

When the responder receives this message, it

- Compares the Message ID with the latest_update_received counter in the IKE_SA. If latest_update_received is greater than this one, replies with "HDR,SK {COOKIE2}", but no other action is taken.

- Updates the latest_update_received counter in the IKE_SA.

- If the NAT_PREVENTION payload is present, compares it with the information in the IP header. If they do not match, replies with "HDR, SK {COOKIE2,N(NAT_PREVENTED)}".

- Compares the COOKIE payload with the source IP address and port in the IP header. If the cookie is not valid, replies with "HDR, SK {COOKIE2, N(NEW_COOKIE_REQUIRED)}".

- Checks that using the destination IP address in the IP header is allowed. If this is not the case, replise with "HDR, SK {COOKIE2, N(UNACCEPTABLE_PATH)}". (This case could occur even legally, if the set of addresses has changed but the initiator has not yet received this message. TBD if tere are there other valid causes for this?).

- Updates (local_address,peer_address, peer_port) in the IKE_SA and any outbound IPsec SAs with the values from the IP header.

- Stores athe additional addresses, together with the peer_address/ peer_port from the IKE SA, to the "additional_addresses" list.

- If NAT Traversal is supported and NAT detection payloads were included, updates the NAT-related flags in outbound IPsec SAs.

- Replies with "HDR,SK {COOKIE2, [NAT_DETECTION_*]}".

When the initiator receives the reply, it

- Verifies the COOKIE2 payload as described in Section 3.

- Compares the Message ID with the latest_update_sent counter in the IKE_SA. If latest_update_sent is greater, stops processing the response.

- If the response contains a NAT_PREVENTED payload, TBD (probably we should retry this a couple of times, to make sure that a single packet can't kill us. But if the NAT stays there, and we don't allow it, there's nothing much we can do.)

- If the response contains a NEW_COOKIE_REQUIRED payload, removes the cookies for this source address, and starts from the beginning (obtains new cookie with path testing, sets pending_update, and so





on).

- If the response contains a UNACCEPTABLE_PATH payload, TBD.

- Otherwise, updates the outbound IPsec SAs with (local_address,peer_address,peer_port) from the IKE_SA.

- If NAT Traversal is supported and NAT detection payloads were included, updates the NAT-related flags in outbound IPsec SAs.

The Notify Message Types for CHANGE_PATH, NEW_COOKIE_REQUIRED, and UNACCEPTABLE_PATH are specified in Section 10. The Protocol ID field is set to one (1), and SPI Size is set to zero. There is no data

associated with these Notify types.

# 8. Discussion

## 8.1 NAT support

This section discusses what cases involving NATs are and are not supported by this specification. The details also depend on exactly what kind of NAT is present; see [9] for discussion about NAT variations.

The following cases work:

- The responder is single-homed, its address does not change, and it is not behind a NAT. The initiator can be multi-homed, its addresses can change, and it can be behind a NAT (or stateful firewall).

- The responder is multi-homed, its addresses do not change, and it is not behind a NAT. The initiator can be multi-homed, its addresses can change, and it can be behind a NAT (or stateful firewall).

- The responder is multi-homed, its addresses can change, and it is not behind a NAT. The initiator can be multi-homed, its addresses can change, and it can be behind a "full cone" NAT.

The following cases DO NOT work.

- The responder's addresses can change, but the initiator is behind a "restricted cone", "port restricted cone", or "symmetric" NAT, or a stateful firewall. (If the responder sends packets from a new address, they will be blocked by the NAT or firewall.)

TBD: This section needs more details; in particular, there are

probably some tricky details in the second and third cases.

## 8.2 Triggers

TBD: describe what kind of situations might lead to a node using the mechanisms specified here. E.g. explicit "use local address X from now on" triggers, and indirect triggers that might lead to e.g. path testing.

# 9. Security considerations

The main goal of this specification has been not to reduce any security offered by normal IKEv2.

(TO BE WRITTEN: more text is needed here.)

If NAT Traversal is not supported, no IPsec (ESP/AH) traffic is sent to an address before it is verified that the peer of the corresponding IKE_SA can actually receive packets at the address.

This return routability check is not inherently incompatible with NATs; as explained in Section 4 IKEv2/IPsec can in fact work across some kind of NATs even without NAT Traversal support. In this specification, "NAT prevention", or integrity protection for the addresses in the IP header, is a separate feature.

When NAT Traversal is supported, the peer's address may be updated automatically to allow changes in NAT mappings. The "continued return routability" feature, implemented by the COOKIE2 payload, allows verification of the new address after the change. This limits the duration of any "third party bombing" attack by off-path (relative to the victim) attackers.

# 10. IANA considerations

This document does not create any new namespaces to be maintained by

IANA, but it requires new values in namespaces that have been defined in the IKEv2 base specification [3].

This document defines one new IKEv2 exchange whose value is to be allocated from the "IKEv2 Exchange Types" namespace.

| Exchange type | Value |
|---|---|
| PATH_TEST | TBD-BY-IANA (38...239) |

This document defines eight new IKEv2 notification payloads whose values are to be allocated from the "IKEv2 Notification Payload





Types" namespace.

| Notify message | Value |
|---|---|
| ADDITIONAL_ADDRESS | TBD-BY-IANA (16396..40959) |
| CHANGE_PATH | TBD-BY-IANA (16396..40959) |
| COOKIE2 | TBD-BY-IANA (16396..40959) |
| NAT_PREVENTED | TBD-BY-IANA (40..8191) |
| NAT_PREVENTION | TBD-BY-IANA (16396..40959) |
| NEW_COOKIE_REQUIRED | TBD-BY-IANA (40..8191) |
| UNACCEPTABLE_PATH | TBD-BY-IANA (40..8191) |
| USE_UDP_ENCAPSULATION | TBD-BY-IANA (16396..40959) |

## 11. Acknowledgements

Everyone in MOBIKE WG, especially Jari Arkko, Francis Dupont, Paul Hoffman, Tero Kivinen, and Hannes Tschofenig. This document also borrows many ideas and even some text from [5], [6] and [7].

## 12. References

### 12.1 Normative references


[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.

[2] Huttunen, A., Swander, B., Volpe, V., DiBurro, L. and M. Stenberg, "UDP Encapsulation of IPsec Packets", draft-ietf-ipsec-udp-encaps-09 (work in progress), May 2004.

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

[4] National Institute of Standards and Technology, "Specifications for the Secure Hash Standard", Federal Information Processing Standard (FIPS) Publication 180-2, August 2002.


### 12.2 Informative references


[5] Dupont, F., "Address Management for IKE version 2", draft-dupont-ikev2-addrmgmt-05 (work in progress), June 2004.

[6] Eronen, P. and H. Tschofenig, "Simple Mobility and Multihoming Extensions for IKEv2 (SMOBIKE)", draft-eronen-mobike-simple-00 (work in progress), March 2004.

[7] Kivinen, T., "MOBIKE protocol", draft-kivinen-mobike-protocol-00







(work in progress), February 2004.

[8] Kivinen, T., "Design of the MOBIKE protocol", draft-ietf-mobike-design-00 (work in progress), June 2004.

[9] 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.

## Author's Address

Pasi Eronen
Nokia Research Center
P.O. Box 407
FIN-00045 Nokia Group
Finland

EMail: pasi.eronen@nokia.com

## Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

## Disclaimer of Validity

## Copyright Statement



## Acknowledgment

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