IETF Mobile IP Working Group
INTERNET DRAFT

M. Roe
T. Aura
G. O'Shea
Microsoft
J. Arkko
Ericsson
February 2002

Authentication of Mobile IPv6 Binding Updates and Acknowledgments <draft-roe-mobileip-updateauth-02.txt>

Status of this Memo

Abstract

This memo describes three protocols that may be used for authenticating binding updates in mobile IPv6. These protocols have the following goals:

- To prevent malicious nodes from forging binding updates for other nodes;
- To protect other nodes on the Internet from denial of service attacks in which a correspondent is tricked into sending them a large amount of data that they do not want;
- To make it difficult for an attacker to exhaust a node's resource by causing it to process large numbers of binding updates;
- To prevent binding updates being replayed for any of the above purposes.

The three protocols differ in the amount of computation that they require and the assumptions made about the environment in which they are used. The symmetric key method is efficient, but can only be used if the mobile and the correspondent have previously agreed a long-term secret. The BAKE/2 method is also efficient, but only works if some of the messages in the protocol take a route which is protected from attack by means outside the protocol. The CAM-DH protocol needs more processing power, because it involves asymmetric cryptography, but it can be used in situations where the other two protocols cannot.

1 Threats addressed by the protocols in this memo

We have identified the following threats to the mobile IPv6 protocol:

1. A malicious mobile node might lie about its home address.

   A malicious mobile node might send a correspondent node binding updates in which the home address is set to the address of another node ("the victim"). If the correspondent node accepted this forged binding update, then communications between the correspondent node and the victim would be disrupted, because packets that the correspondent node intended to send to the victim would be sent to the wrong care-of address.

   This is a threat to confidentiality as well as availability, because an attacker might redirect packets meant for another node to itself in order to learn the content of those packets.

2. A malicious mobile node might lie about its care-of address.

   A malicious mobile node might send a correspondent node binding updates in which the care-of address is set to the address of another node ("the victim node") or an address within another network ("the victim network"). If the correspondent node accepted this forged binding update, then the malicious mobile could trick the correspondent into sending data to the victim node or the victim network; the correspondent's replies to messages sent by the malicious mobile will be sent to the victim host or network. This could be used to cause a distributed denial of service attack; the malicious mobile could trick a large number of servers so that they all send a large amount of data to the same victim node or network.

   There are several variations of this threat:

   - A malicious mobile might start off by sending the correspndent node binding updates containing its true care of address, and then later (once its initial home and care of addresses had been authenticated) send binding updates containing the victim's care of address.
   - A malicious mobile might start of by sending the correspondent node binding updates contains its true care of address, and then continue to send binding updates containing that care-of address even after that care of address had been reallocated to a different node (the victim). This variation of the threat might be regarded as less serious than the previous two, because the attacker's choice of victim is restricted to nodes that are currently using a care of address that the attacker has used in the past.

3. A malicious node might send a large number of invalid binding updates to a victim correspondent node.

   If each invalid binding update took a significant amount of resources (such as CPU) to process before it could be recognized as invalid, then it might be possible to cause a denial of service attack by sending the correspondent so may invalid binding updates that it has no resources left for other tasks.

4. An attacker might reply an old binding update.

   An attacker might attempt to disrupt a mobile node's communications by replaying a binding update that the node had sent earlier. If the old binding update was accepted, packets destined for the mobile node would be sent to its old location and not its current location.

All of the above threats are concerned with denial of service. The first threat is the denial of service caused when the correspondent's state (its binding cache) contains incorrect information derived from forged messages. The second threat is the denial of service caused to a third party when the correspondent is tricked into consuming network resources. The third threat is the denial of service caused when the correspondent must consume a significant amount of resource such as CPU and memory to disttinguish genuine updates from forged ones.

2 Abstract Protocols

2.1 Notation

This memo uses the following notation:

MN	A mobile node
CN	A correspondent node
$A \rightarrow B$	Node A sends a message to B
$A \rightarrow B(HoA)$	Node A sends a message to B at its home address
$A \rightarrow B(CoA)$	Node A sends a message to B at its care-of address
HoA	Mobile node's home address
CoA	Mobile node's care-of address
$MAC_K(m)$	A message authentication code computed on message m with key K
$H(m)$	A hash of message m

2.2 The Shared Key Protocol

Properties of the Protocol

The shared key protocol is used to authenticate binding updates between a mobile node and a correspondent node that share a symmetric key (K_h). There are several different ways in which a correspondent and a mobile can agree on a shared key for use with this protocol; these be will be described later.

The protocol has the following properties:

- A node needs to know the shared secret (K_h) in order to create a binding update that will be accepted by the correspondent. This prevents a malicious mobile from forging binding updates containing another node's home address; the malicious mobile will not know the correct key.
- To create a binding update for a care-of address that is not equal to its home address, a mobile node needs to be able to receive messages sent to that care-of address.
- To create a binding update that deletes a binding cache entry, a mobile needs to know the secret K_h but does not need to be able to receive messages sent to a particular address.

Walkthrough

Each correspondent node has a secret key, K_{CN}. This key does not need to be shared with any other entity, so no key distribution mechanism is needed for it. Each correspondent node also generates a nonce, N_j, at regular intervals, for example every few minutes. A correspondent node uses the same K_{CN} and N_j with all the mobiles it is in communication with, so that it does not need to generate and store a new N_j when a new mobile contacts it. Each value of N_j is identified by the subscript j. j is communicated in the protocol, so that if N_j is replaced by N_{j+1} part way through a run of a protocol, the correspondent can distinguish messages that should be checked against the old nonce from messages that should be checked against the new nonce. Correspondent nodes keep both the current value of N_j and the previous value N_{j-1}. Older values can be discarded, as messages using them will in any case be rejected as replays. K_{CN} can be either a fixed value or regularly updated. An update of K_{CN} can be done at the same time as an update of N_j, so that j identifies both the nonce and the key. A correspondent node can generate a fresh K_{CN} each time that it boots to avoid the need for secure persistent storage for K_{CN}.

1. $MN \rightarrow CN : HoA, CoA$

   In step 1, the mobile node informs the correspondent node that it is mobile, and gives both the mobile's home address and its care-of address.

2. $CN \rightarrow MN(CoA) : r_c, j$

   $r_c = MAC_{K_{CN}}(CoA|N_j|1)$

   In step 2, the correspondent node sends a binding request to the mobile node. The binding request contains a challenge (r_c), and a serial number (j) that indicates which version of N_j was used to generate the challenge. The challenge is generated from N_j so that the correspondent does not need to store state to remember which challenges it has sent to which mobiles — r_c can be recomputed from N_j as it is needed.

3. $MN \rightarrow CN : T_0, HoA, CoA, i, MAC_{K_{BU}}(T_0|HoA|CoA|i), j$

   $K_{BU} = H(K_h|r_c)$

   In step 3, the mobile node hashes together the shared secret and the challenge to form a session key (K_{BU}), and then uses this session key to authenticate a binding update. The binding update contains j, so that the correspondent knows which value of N_j to use to recompute the session key. Once it has verified the MAC, the correspondent can create a binding cache entry for the mobile. This message contains a tag (T_0) so that it can be distinguished from message 1 of the variant version of the protocol (described below). The binding update also contains a sequence number (i) so that if more than one binding update is sent within the lifetime of a single value of N_j, it is possible to determine their relative ordering.

When the correspondent's binding cache entry for the mobile node expires, the correspondent can refresh it by running the above protocol again, starting at step 2. The message sent in step 2 of this new run of the protocol will usually use a different value of the challenge from message that was sent in step 2 of the previous run with the same mobile, because the value of N_j has changed.

If the mobile changes its care-of address, but is still able to receive messages sent to the old care-of address, then it runs the above protocol again using its new care-of address.

If the mobile changes its care-of address, and is unable to receive messages sent to the old address, then it uses a variant of the protocol to give the correspondent an earlier notification that the old address is no longer valid:

1. $MN \rightarrow CN : T_1, HoA, CoA', i', MAC_{K_{BU}}(T_1|HoA|CoA', i'), j$

   In step 1, the mobile node sends a binding update authenticated using the key K_{BU} derived from the key that was sent to the mobile's old care-of address, CoA. (At this point in the protocol, the mobile has not yet received a challenge sent to its new care-of address, CoA'). This message contains a tag (T_1) so that it can be distinguished from the binding update sent in message 3 of the previous protocol.

   If the correspondent has a binding cache entry for the mobile, and it is able to verify the MAC correctly, then it should mark the binding cache entry as invalid. Note that the correspondent will only be able to verify the MAC if it has an existing binding cache entry for the mobile, because it will need to know the old care-old address to reconstruct the key K_{BU}. If the correspondent does not have an existing binding cache entry for the mobile node, then it does not need to verify the MAC because the binding cache entry has already been deleted.

2. $CN \rightarrow MN(CoA') : r'_c, j'$

   $r'_c = MAC_{K_{CN}}(CoA'|N_{j'}|1)$

   In step 2, the correspondent sends a new challenge to the new care-of address. It should send this challenge even if it was unable to verify the MAC on message 1. The reason for doing this is that it allows the protocol to resynchronise after messages have been lost or nodes have lost their state.

3. $MN \rightarrow CN : T_0, HoA, CoA', i'', MAC_{K'_{BU}}(T_0|HoA|CoA', i''), j'$

   $K'_{BU} = H(K_h|r'_c)$

   The third step if this protocol is the same as the third step of the previous protocol. Once the correspondent has verified the MAC, it can create a new binding cache entry for the mobile (or update the existing one).

Optimizations

1. It is not necessary to encode all the bits of j in the protocol messages; just the least significant bit is sufficient for the correspondent to tell whether to use N_j or N_{j-1}.
2. The values of N_j should be non-repeating, but do not need to be unpredictable. This means that N_j can be implemented as a counter. The secret K_{CN} should be changed if the counter wraps or is reset (e.g after a reboot)
3. It is not necessary to encode all the bits of the sequence number i. It is sufficient to encode enough of the lower bits of i so that it is possible to determine the relative ordering of binding updates sent within the lifetime of a single N_j.

Manually Configured Keys

This protocol can be used with a shared secret K_h that has been configured manually. This option might be appropriate for use between a mobile node and its home agent; the home agent can maintain a database of the keys that have been issued to the mobile nodes that it serves.

Use with a PKI

This protocol can also be used with a shared secret K_h that has been agreed using a certificate-based key agreement protocol. The certificates should associate a node's public key with its home address. That is, the public key infrastructure should be used to authenticate the node's homes address rather than its care-of address.

2.3 BAKE/2

Properties of the Protocol

The "Bake/2" protocol extends the shared key protocol of section 2.2 by providing a means to establish the shared secret dynamically.

This protocol is only suitable for use in an environment where communication from the correspondent through the home agent to the mobile, and between the home agent and the mobile node are protected from eavesdropping by means outside of this protocol. Examples of ways in which this protection could be provided include the use of IPSEC Encapsulating Security Payload, or a physically protected network.

An example of a situation where it would be appropriate to use this protocol is when the home agent and the correspondent node are both on a physically protected corporate intranet, the mobile node is connected via a public wireless network, and the mobile node has an encrypted tunnel between itself and the home agent.

This protocol may also provide a low level of protection when the correspondent node is (for example) a web server connected to the public Internet by a wired connection and the mobile node is connected via a wireless network. The protocol can be broken by an attacker on the route between the home agent and the correspondent node, but not by attackers on the wireless network or elsewhere on the Internet.

Walkthrough

1. $MN \rightarrow CN : HoA, CoA$

   In the first message, the mobile node contacts the correspondent node, giving both the mobile's home address and its care of address.

2. $CN \rightarrow MN(HoA) : K_h, j$

   $K_h = MAC_{K_{CN}}(HoA|N_j|0)$

In the second step, the correspondent generates a value (K_h) that will be used as a shared secret between the mobile and the correspondent. This shared secret is sent to the mobile node via its home agent; it is an assumption of the protocol that this route is secure. K_h also acts as a challenge to test that the mobile can receive messages sent to its home address.

3. $CN \rightarrow MN(CoA) : r_c, j$

   $r_c = MAC_{K_{CN}}(CoA|N_j|1)$

   The correspondent also sends a challenge to the mobile's care-of address. This step is the same as step 2 of the shared key protocol described in section 2.2.

4. $MN \rightarrow CN : T_0, HoA, CoA, i, MAC_{K_{BU}}(T_0|HoA|CoA|i), j$

   $K_{BU} = H(K_h|r_c)$

   In the third step, the mobile sends an authenticated binding update.

2.4 CAM-DH

Properties of the Protocol

The "CAM-DH" protocol combines the BAKE/2 protocol with a digitally signed Diffie-Hellman key exchange. In CAM-DH, each mobile node's home address is generated from its public signature key. The use of cryptographically-generated addresses (CGA) avoids the need for X.509 certificates or similar mechanisms that associate keys with addresses [5]. The mobile node uses its private signature key to sign a Diffie-Hellman exponent which is then used to negotiate a session key. The underlying BAKE/2 protocol provides the correspondent node with protection against denial of service attacks - the correspondent will not perform any asymmetric cryptographic operations until it knows it is talking to a mobile which has been authenticated with BAKE/2 - while the signature mechanism provides a higher level of security than would be available with BAKE/2 used on its own.

This protocol could have been simplified by deriving mobile's home address from the Diffie-Hellman exponent, rather than deriving it from the public key that verifies the signature on the Diffie-Hellman exponent. However, the extra level of indirection allows the signature key to be used to sign messages that are used with other protocols. We anticipate that there will be other protocols that would like to use cryptographically generated addresses. Our approach allows a node to use several such protocols simultaneously. Each signed key is accompanied by a tag that indicates the protocol it is used for, to prevent attacks based on interactions between protocols.

Walkthrough

1. $MN \rightarrow CN : HoA, CoA$

   In the first message, the mobile node contacts the correspondent node, giving the mobile's home and care-of addresses.

2. $CN \rightarrow MN(HoA) : r_h, j, g^y$

   $r_h = MAC_{K_{CN}}(HoA|N_j|0)$

   In the second and third messages, the correspondent node sends the mobile node two challenges, one to the care-of address and one to the home address. The correspondent also sends the mobile a Diffie-Hellman exponent. The correspondent can use the same exponent with all mobiles it is communicating with, so there is no need to generate a new exponent for each protocol run. Like K_{CN}, y can be constant (this reduces by one the number of modular exponentiations that the correspondent needs) or periodically updated. If y is changed, the subscript j indicates which version of y to use (as well as which K_{CN} and N_j).

3. $CN \rightarrow MN(CoA) : r_c, j$

   $r_c = MAC_{K_{CN}}(CoA|N_j|1)$

4. $MN \rightarrow CN : T_0, HoA, CoA, i, MAC_{K_{BU}}(T_0|HoA|CoA|i), g^x, S_{PK}(TypeTag|g^x|HoA), PK, MAC_{K_3}(\ldots), j$

   $K_3 = h(r_h|r_c)$

   $K_h = h(g^{xy}|r_h)$

   $K_{BU} = h(K_h|r_c)$

   When it has received the two challenges, the mobile hashes them together to form a key (K_3), and then uses this key to compute a message authentication code on its public key and signed Diffie-Hellman parameter. The purpose of this MAC is to convince the correspondent that the risk of the message being a forgery is low enough that it is worthwhile expending computational resources on checking the signature and calculating the Diffie-Hellman exponent g^{xy}. The mobile also uses Diffie-Hellman key agreement to calculate a session key that can be used to authenticate binding updates. The fourth message consists of a binding update, a message authentication code on the binding update computed using K_{BU}, the mobile's public signature key, the mobile's Diffie-Hellman exponent signed with its private signature key, and a message authentication code on all of the aforementioned data, computed using a key derived from the two challenges.

   When the correspondent receives the fourth message, it should check the outer MAC with K_3 first. It should only attempt to compute K_{BU} and verify the inner MAC with it if the outer MAC verifies correctly. This protects the correspondent against denial of service attacks in which it is flooded with many bogus fourth messages. If both MACs verify correctly, the correspondent should store state related to the mobile, including the key K_h.

5. $CN \rightarrow MN : r'_c, j'$

   When the correspondent node's binding cache entry is about to expire, the correspondent sends the mobile node a binding request containing a fresh challenge. (Typically, N_j will have changed since the last time a challenge was sent to the mobile).

6. $MN \rightarrow CN : T_0, HoA, CoA, i, MAC_{K'_{BU}}(T_0|HoA|CoA|i), j'$

   $K_h = h(g^{xy}|r_h)$

   $K'_{BU} = h(K_h|r'_c)$

   The mobile node hashes the old value of K_h together with the new challenge to compute a new key K'_{BU}, and sends a binding update authenticated using this key.

Optimizations

1. All of the asymmetric cryptographic operations that the mobile carries out can be performed instead by the home agent, provided that the home agent is given access to the appropriate keys. An example of a situation where the optimisation might be useful is a low-power wireless mobile device that does not have enough computational power for asymmetric cryptography. If this optimisation is used, the home agent intercepts the second message (which is routed via the home agent) and performs certain processing on in before forwarding it on to the mobile node.

   That is, the second message is replaced with the following:

   $CN \rightarrow HA : r_h, j, g^y$

$HA \rightarrow MN$: CN's address, K_h, j

$K_h = h(g^{xy}|r_h)$

To use this optimization, communications between the home agent and the mobile node must be protected against eavesdropping (e.g. by using IPSEC ESP).

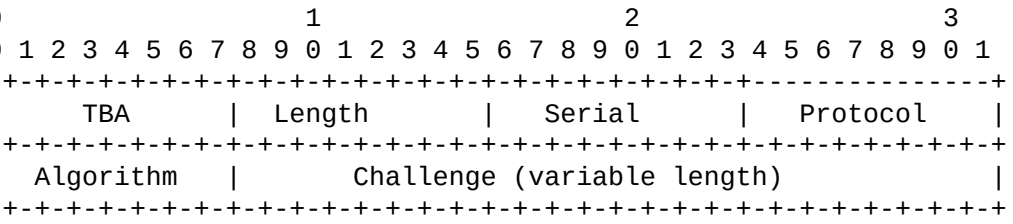
2. In the case when the correspondent node is also a mobile node, all of the asymmetric cryptographic operations that the correspondent performs can instead be performed by the correspondent's home agent. To enable this optimisation, the second message of the protocol contains a flag that indicates to the mobile node whether the correspondent is using this optimisation. When this flag is set in the second message, the mobile should send the fourth message to the correspondent's home address, rather than its care-of address. That is, the mobile should disable route optimisation when sending the third message.

3 New IPv6 Sub-option Types

This memo defines the following IPv6 destination option sub-option types:

3.1 Care-of Address Challenge

Alignment requirement: none



The Care-of Address Key Challenge sub-option is valid in a Binding Request destination sub-option. The Serial field contains the variable j in the BAKE/2 and CAM-DH protocols. The Algorithm field indicates which cryptographic algorithm should be used to compute the authentication information field in the Binding Update that is sent in response to this option. The Challenge field contains the challenge r_c in the shared key, BAKE/2 and CAM-DH protocols; it is the second of two components which are to be concatenated and hashed to form a key which is then used to authenticate binding updates.

The Protocol field indicates which authentication protocol the correspondent requires. The following values are defined:

1	The shared key protocol
2	BAKE/2
3	CAM-DH

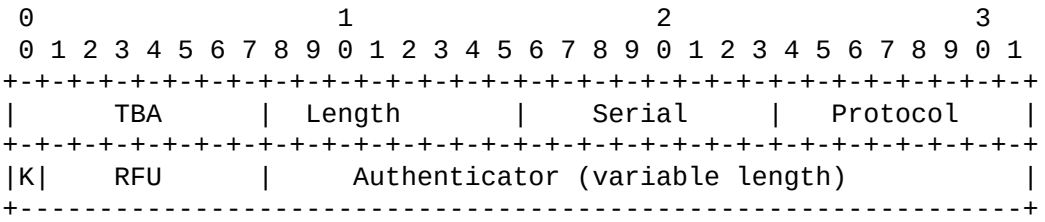
The Algorithm field indicates which cryptographic algorithms are to be used in the authentication protocol. The following values are defined:

1	HMAC-SHA1

The Challenge field is of variable length. It is recommended that this field be 4 bytes long.

3.2 Response to Challenge

Alignment requirement: none



The Response to Challenge sub-option is valid in a Binding Update destination option.

The Serial field contains the variable j in the shared key, BAKE/2 and CAM-DH protocols. That is, it tells the correspondent node that receives the suboption which of the challenge values (N_j) are to be used to authenticate the binding update.

The Protocol field indicates which protocol (shared key, BAKE/2 or CAM-DH) was used to construct the authenticator. The value of this field is the same as the value of the Protocol field in the Challenge sub-option.

The K bit corresponds to the tags T_0 and T_1 in the shared key protocol. It is set to zero if the MAC on the binding update is to be verified using the challenge that was sent to the mobile node's current care-of address, and is set to 1 if the binding update is authenticated using the challenge that was sent to the mobile node's previous care-of address.

The RFU bits are reserved for future use and shall be set to zero.

The Authenticator field is computed by applying HMAC-SHA1-80 to the following data:

AHSD, Reserved	3 bytes
Sequence Number	1 byte
Lifetime	4 bytes
Home Address	16 bytes
Care-of Address	16 bytes
K, RFU	1 byte

The A, H, S, D, Reserved, Sequence Number and Lifetime fields show above have the same value as the corresponding fields in the Binding Update. The Home Address field contains the Home Address from the Home Address option earlier in the packet. The Care-Of Address field contains the IP source address of the packet. The K and RFU fields shown above have the same value as the corresponding fields in the Response To Challenge sub-option.

4 Other Message Formats

4.1 DHChallenge

```
HomeAddressChallenge ::=
    [0] SEQUENCE
    {
        serial INTEGER,
        CHOICE
            {
                bake2 [0]  SEQUENCE
                    {
                        key BIT STRING
                    }
                camDH [1] SEQUENCE
                    {
                         challenge BIT STRING,
                         exponential INTEGER,
                         disableRouteOptimization BOOLEAN
                    }
```

```
        }
    }
```

The serial field contains the value of j. The key field contains K_h. The challenge field contains r_h. The exponential field contains g^y. If the disableRouteOptimization field is set to TRUE, then the response to this message should be sent to the correspondent's home address, not its care-of address.

4.2 DHResponse

```
DHResponse ::=
    [1] SEQUENCE
    {
        serial INTEGER,
        signedExponential SIGNED SEQUENCE
        {
            tag OBJECT IDENTIFIER,
            homeAddress BIT STRING,
            exponential INTEGER
        }
        publicKey PublicKey,
        innerMAC BIT STRING,
        outerMAC BIT STRING
    }
```

The serial field contains the value of j. The exponential field contains the value of g^x. The innerMAC field contains a MAC computed using K_{BU} with HMAC-SHA1-80. The outerMAC field contains a MAC computed using K_3 with HMAC-SHA1-80.

The value of the tag field is to be assigned.

5 Assigned Numbers

5.1 Ports

`UDP_PORT_CAM` to be assigned

5.2 Object Identfiers

SignedExponent to be assigned

5.3 Binding Acknowldgement Status Values

`AUTHENTICATION_REQUIRED` to be assigned

6 Realisation of the Abstract Protocols

6.1 The Shared Key Protocol

1. The mobile sends the correspondent a packet containing a Binding Update destination option.

2. The correspondent sends the mobile a packet containing a Binding Acknowledgment destination option, with the status field set to `AUTHENTICATION_REQUIRED`. The Binding Acknowledgment contains a Care-Of Address Challenge sub-option.

3. The mobile sends the correspondent a packet containing a Binding Update destination option, which in turn contains a Response to Challenge sub-option. The Flags field of this sub-option will be set to 0. The Serial field of this sub-option will be the same as the Serial field of the Care-Of Address Challenge sub-option in the previous step. The other fields are computed as described in section 3.2.

4. The correspondent sends the mobile a Binding Acknowledgement, with the status field set to indicate success.

5. When the correspondent's Binding Cache Entry is about to expire, the correspondent sends the mobile a Binding Request containing a Care-Of Address Challenge sub-option.

6. The mobile replies to the request by sending a Binding Update containing a Response to Challenge sub-option.

7. When the mobile's Binding Entry is about to expire, it sends the correspondent a Binding Update containing a Response to Challenge sub-option.

8. The correspondent replies with a Binding Acknowledgment.

   - If the value of the Serial field in the Binding Update is the one which the correspondent is currently using, the status field of the Binding Acknowledgement is set to indicate success.
   - If the value of the Serial field in the Binding Update is not the most recent one, but is recent enough to be acceptable to the correspondent, then the Binding Acknowlegment's status field is set to indicate success and the Binding Acknowledgment contains a Care-Of Address Challenge sub-option with the most recent value in the Serial field.
   - If the value of the Serial field in the Binding Update is too old to be acceptable, the status field of the Binding Acknowledgment is set to indicate failure and the Binding Acknowledgment contains a Care-Of Address Challenge sub-option with the most recent value in the Serial field.

     In this case, the mobile will reply with another Binding Update containing a Response to Challenge sub-option.

6.2 BAKE/2

1. The mobile sends the correspondent a packet containing a Binding Update destination option.

2. The correspondent sends a UDP packet of format HomeAddressChallenge to the mobile at the port `UDP_PORT_CAM`. The optional exponential field is not present in this packet.

3. The correspondent sends to the mobile (at its care-of address) a packet containing a Binding Acknowledgement destination option with the status field set to `AUTHENTICATION REQUIRED`. The Binding Acknowledgment contains a Care-Of Address Challenge sub-option.

4. The mobile sends the correspondent a packet containing a Binding Update destination option, which in turn contains a Response to Challenge sub-option. The Authenticator field of this sub-option is computed as described in section 3.2.

5. The correspondent sends the mobile a packet containing a Binding Acknowledgment, with the status field set to indicate success.

The procedures taken when the correspondent's Binding Cache Entry is about to expire, and when the mobile's Binding Entry is about to expire, are the same as for the shared key protocol.

6.3 CAM-DH

1. The mobile sends the correspondent a packet containing a Binding Update destination option.
2. The correspondent sends a UDP packet of format HomeAddressChallenge to the mobile's home address at port `PORT_UDP_CAM`.
3. The correspondent sends the mobile (at its care-of address) a packet containing a Binding Acknowledgment destination option with its status field set to `AUTHENTICATION_REQUIRED`. The Binding Acknowledgment contains a Care-Of Address Challenge sub-option.
4. The mobile sends a UDP packet of format DHResponse to the correspondent at port `PORT_UDP_CAM`.
5. The correspondent sends the mobile (at its care-of address) a packet containing a Binding Request destination option, which in turn contains a Care-Of Address Challenge sub-option.
6. The mobile sends the correspondent a packet containing a Binding Update destination option, which in turn contains a Challenge Serial Number sub-option.

7 Finite State Machines

7.1 The Shared Key Protocol

Mobile Node

- Event: Mobile receives a Binding Request

  Action: Add the correspondent to the Binding Entry List, if it isn't already on it. Send a Binding Update. If the Binding Request contained a Care-of Address Challenge sub-option, include a Response to Challenge sub-option in the Binding Update, and store the value of the challenge in the Binding Entry List.

- Event: Mobile's Care-Of Address changes

  Action: Send a Binding Update to all correspondents in the Binding Entry List. If it is no longer possible to receive packets sent to the old care-of address, set the T_1 tag in the binding update, and compute the authentication field based on the challenge that was sent to the old care-of address. If it is still possible to receive packets sent to the old care-of address, send a binding update without authentication.

- Event: Mobile's Binding Entry about to expire

  Action: Send a Binding Update containing a Response to Challenge sub-option to the correspondent.

- Event: Mobile receives a packet from the correspondent routed via its home agent

  Action: Check the Binding Entry List to see if a binding update has been sent to this correspondent recently. If a binding update has not been sent recently, send one. If the Binding Entry List contains a recent challenge from the correspondent, use that to construct a Response to Challenge sub-option that is included in the Binding Update; otherwise, do not include a Response to Challenge sub-option.

- Event: Mobile receives a Binding Acknowledgment

  If the Binding Acknowledgment contains a Care-Of Address Challenge sub-option, then the mobile stores the value of the challenge in its binding entry list. If the Binding Acknowledgment contains a Care-Of Address Challenge sub-option and the status field is set to indicate failure, then the mobile sends a Binding Update containing a Response to Challenge sub-option.

Correspondent Node

- Event: Binding Cache Entry about to expire

  Action: Send the mobile a Binding Request containing a Care-of Address Challenge sub-option

- Event: Receive a Binding Update

  Action: If the Binding Update contains a Response to Challenge sub-option, and the Serial field is sufficiently recent, and the Authenticator field contains the right value, then update the Binding Cache and send a Binding Acknowledgment with the status field set to indicate success. If the value in the Serial field was not the most recent one, include a Challenge sub-option in the Binding Acknowledgment.

  Otherwise, send a Binding Acknowledgment with the status field set to indicate failure and containing a Care-of Address Challenge sub-option.

8 Background to the Protocol Designs

8.1 IP Addresses derived from Cryptographic Keys

In the CAM-DH protocol, a node uses a home address that is derived from the node's public key. The idea behind this is that if the address is the same as the public key, nodes can work out which key corresponds to an address without needing to use a secure key distribution mechanism such as X.509 certificates. Such key distribution mechanisms typically need to be configured manually, and this conflicts with the design goal of IPv6 that it should be possible to configure hosts automatically. However, it is not possible to set the IP address equal to the public key, because they will normally be of different length, and the network part of the address must be set to the right value for the packet to be routed correctly. Instead, we use a more complex relationship between the address and the key, in which the last 64 bits of the address (the "Interface ID") are defined as follows:

```
InterfaceId = First64(SHA1(Route Prefix | M | RFU | Public Key))
                              & 0xfcffffffffffffff
```

The field "RFU" is reserved for future use, and shall be set to zero. The field "M" is a modifier, which is used in the following way. A node generates a private/public key pair, and then attempts duplicate address detection for an address generated using the above equation with M set to zero. It is very unlikely that a collision will occur except as a result of an attack on the protocol. However, if a collision is detected the host MAY attempt duplicate address detection again with a different address, generated using the same public key and with M equal to one. If necessary, this process may be be repeated with M equal to 2 and M equal to 3. Nodes MUST NOT use values of M greater than three. Four collisions in a row are very, very unlikely to occur by chance, and are almost certainly the result of either an attack on the protocol or an error in the implementation.

Bit 6 of the host part of the address is the universal/local bit[3]. It is set to zero to indicate that the address generated is not guaranteed to be globally unique. This ensures that it will not collide with an IP address derived from an ethernet address. It is important to avoid such collisions, because hosts that use their MAC address to derive their IP address will not expect such collisions, and they might not have a means to recover from them when they occur. Bit 7 of the host part of the address is the individual/group bit[3]. It is set to zero to indicate that it is the address of an individual node, not a group of nodes.

The route prefix is included in the input to the hash function to prevent an attack in which the attacker expends a very large initial set-up cost, but is then able to attack many different nodes at a relatively low cost per node. If the route prefix was not included, an attacker could, at great expense, compute a lookup table that contains a suitable key pair for each of the 2^{62} possible values of the InterfaceId. Such a lookup table could then be used to masquerade as any mobile node. Including the route prefix makes this attack not economically viable (from the point of view of the attacker), because it means that such a look-up table can only be used to masquerade as nodes which have the same route prefix. Typically, there will not be enough nodes with the same route prefix to justify the expense of constructing the lookup table.

8.2 Resource Exhaustion and other Denial of Service Attacks

When designing these protocols, we found it useful to distinguish between two different types of denial of service attack. Resource exhaustion attacks are attacks in which the victim has only a limited amount of some resource (such as network bandwidth or CPU cycles), and the attack consumes some of this resource, leaving the victim with not enough of it left to carry out the other work it needs to do. There are denial of service attacks that are not resource exhaustion attacks. For example, forged binding updates can lead to denial of service, because packets will be sent to the wrong care-of address. This is not an example of resource exhaustion; a host with an unlimited supply of network bandwidth and CPU would still be vulnerable to denial of service attacks based on forged binding updates. This attack works by corrupting a host's state (its binding cache), not by running it out of resources. That is, a failure of integrity and authentication then leads to denial of service.

The binding updates that are used in mobile IPv6 are only an optimisation. A mobile node can communicate with a correspondent node even if the correspondent refuses to accept any of its binding updates. However, performance will suffer because packets from the correspondent to the mobile will be routed via the mobile's home agent rather than a more direct route. A correspondent can protect itself against some of the resource exhaustion attacks by stopping processing binding updates when it is flooded with a large number of binding updates that fail the cryptographic integrity checks. If a correspondent finds that it is spending more resources on checking bogus binding updates than it is likely to save by accepting genuine binding updates, then it can decide to reject all binding updates without performing any cryptographic operations.

Nodes that are willing to expend significant resources responding to anyone, no matter who they are, will often be vulnerable to resource exhaustion attacks. The DoS protection mechanisms described in this memo will only be useful if each node has some means of deciding whether it should expend resources on behalf of a particular peer. This information needed to make this decision will usually originate in layers above IP. For example, TCP knows if the node has a queue of data that it is trying to send to a peer. It is possible to produce a conforming implementation of the protocols in this memo without making use of information from higher protocol layers, but implementations may be able to manage resources more effectively by making use of such information.

In general, a node will be willing to devote resources to a run of an authentication protocol for one of two reasons. In the first case, the node itself is trying to carry out some work, and knows that completing the authentication protocol run is necessary (or helpful) in getting that work done. In the second case, the node's peer is trying to carry out some work for which the authentication protocol run is necessary or helpful. In this case, the node does not know directly that the protocol run is worthwhile, but may be prepared to expend resources on behalf of certain peers when they ask it to. There is a problem with this case that is specific to authentication protocols, and does not occur with other types of protocol. The node will only know that is is worthwhile expending resources on a protocol run when it knows that the run has been initiated by a peer that is willing to devote resources to. However, it will only know this when the peer has been successfully authenticated, that is when protocol run has been completed and the resources have already been spent. One way in which this situation may be improved is to divide the authentication protocol in to two phases. The first phase consumes very little resources, but does not provide a very high level of security. The second phase provides a higher level of security, but requires more resources to provide this level of security. The second phase is only started if the first phase completes successfully. In this way, only attackers who can break the security of the first phase can cause a resource exhaustion attack using the second phase. We have used this approach in the protocols described in this memo.

8.3 Piggybacking and Jitter

The mobile IPv6 specification allows for "piggybacking". That is, control messages such as binding updates may be combined with other messages. Piggybacking will delay these other messages in two ways. Firstly, it will make them larger, and larger messages usually take longer to transmit. Secondly, it will increase the amount of processing that is needed to send and receive the messages because the mobility information in the message will need to be processed as well. When the control messages are authenticated with asymmetric cryptography, they will add a large amount of jitter, because digital signatures requires many bytes to represent and take many CPU cycles to compute or verify. Some applications, for example real-time voice, are very sensitive to jitter.

Some networks have "quality of service" facilities whereby an application can reserve a particular amount of bandwidth. Piggybacking can interfere with these quality of service facilities, because when packets are made bigger by adding mobility headers they may exceed the size that has been reserved, and this may cause them to be discarded or severely delayed by the network.

Accordingly, we recommend that piggybacking should not be used when quality of service facilities are in use (e.g. the IPv6 flow id is nonzero) and should not be used when asymmetric cryptography is being used to protect the mobility control portion of the message. This recommendation has affected the design of the protocols described in this memo; digital signatures are carried in UDP messages, not IPv6 destination options. UDP messages cannot be piggybacked, but this is not a serious problem as we recommend that these messages should not be piggybacked.

8.4 Length of Suboptions

The IPv6 option length limits the amount of data that may be passed in a destination option or as a suboption within a destination option. The maximum length of a suboption is 255 bytes, or 2040 bits excluding any other data in the protocol. Since both a public key and a Diffie-Hellman value needs to be passed in the CAM-DH protocol, passing these in a suboption would limit the key size to 1020 bits. These values are just about enough for current security needs, but seem low in view of future developments. They also preclude the use of the same long key for both MIPv6 and other purposes. Therefore, we have chosen to run the authentication protocol as an independent protocol on top of UDP.

8.5 Rationale for BAKE/2

Our motivation for designing BAKE/2 was that we wanted to add support for mobile IP without creating major new security problems. We wanted a protocol that would protect against the new vulnerabilities that were introduced by IP mobility. It was not our goal to protect against attacks that were already possible before the introduction of IP mobility. This protocol does not defend against an attacker who can monitor the home agent to correspondent node route. Our justification for this is that if such an attacker exists, they are able to attack the system before IP mobility is enabled, because they can mount an active attack against the mobile node when it is at its home location. Prevention of such attacks is outside the scope of this protocol. The possibility of such attacks is not an impediment to the deployment of mobile IP, because these attacks are possible irrespective of whether mobile IP is in use or not.

Some of our earlier protocols for authenticating binding updates, such as CAM [5], ran the complete protocol for each binding update. The protocol described here establishes a session key which can then be used for many binding updates between the same nodes without running the whole protocol again. This can result in an efficiency saving, because binding updates are resent at regular intervals. This efficiency saving will usually be realised when a mobile node communicates with the same correspondent node for an extended period of time. If the mobile node communicates with a correspondent briefly and then never talks to it again, then the establishment of a session key does not result in efficiency savings.

This protocol protects the correspondent node against denial of service attacks in which the correspondent is flooded with many bogus messages. The correspondent does not have to store state or consume a large amount of processing time handling messages from a source which has not yet been authenticated. The protocol does not protect the mobile against these attacks. This means that this protocol is suitable for use when a client on a mobile node accesses a server on a non-mobile node, but may not be suitable for use when accessing a server on a mobile node. It is an assumption of the protocol that at the start of a run the mobile node already has stored state about the correspondent (perhaps at a level above IP, such as TCP or the application), and knows that it is worthwhile expending resources on the run. There is a clear need for a protocol for the opposite case, where the correspondent has pre-existing stored state about the mobile and knows that it is worthwhile expending resources on the protocol run. This is a matter for further study.

This protocol also protects against denial of service attacks in which the attacker pretends to be a mobile, but uses the victim's address as the care of address, and so causes the correspondent to send the victim traffic that it does not want. For example, suppose that the correspondent is a news site that will send a high-bandwidth stream of video to anyone who asks for it. Note that the use of flow-control protocols such as TCP does not necessarily defend against this type of attack, because the attacker can fake the acknowledgements. Even keeping TCP initial sequence numbers secret

doesn't help, because the attacker can receive the first few segments (including the ISN) at its own address, and then redirect the stream to the victim's address. This protocol defends against these attacks by only completing if packets sent by the correspondent to the care of address are received and processed by an entity that is willing to participate in the protocol. Normally, this will be the mobile node.

9 Intellectual Property Rights Notice

The CAM-DH variant of our protocols uses public keys and hashes to prove address ownership [4, 5]. In case there would be any Ericsson IPR on such methods, the Ericsson policy on IPR issues can be checked from the Ericsson General IPR statement for IETF, http://www.ietf.org/ietf/IPR/ERICSSON-General. Microsoft's IPR statement concerning this memo is available at http://www.ietf.org/ietf/IPR/MICROSOFT-MOBILEIP-UPDATEAUTH.txt.

10 Security Considerations

10.1 Risks of unauthenticated binding updates

If a node accepts binding updates without authentication, then it is vulnerable to several attacks in which an attacker sends forged binding updates for other nodes. These include a denial of service attack in which the attacker sends the victim a forged binding update for a service that the victim relies on (e.g. the domain name service), and sets this service's care of address to a non-existent address. The victim will be unable to contact the service at the falsified care of address, and henceforth will be unable to make use of the service. A variation on this attack with consequences beyond denial of service is when the attacker sets the service's care of address to the attackers own address, and the attacker then provides a maliciously modified version of the service.

For this reason, it is recommended that nodes on the Internet SHOULD use some form of authentication for binding updates. Nodes on private intranets that use other means to exclude potential attackers MAY accept binding updates without authentication.

10.2 Risks of unauthenticated binding acknowledgements

The consequences of forged binding acknowledgements are, in general, less serious that those of forged binding updates. The usual consequence of forging a binding acknowledgement is that the victim's correspondent will fail to obtain an up-to-date binding for the victim, the correspondent's previous binding for the victim will expire, and the correspond will revert to sending packets via the victim's home agent. Communications between the victim and its correspondent will still work, but may suffer degraded performance. In some circumstances this degradation of performance will be sufficiently severe to constitute a denial of service attack.

Forged binding acknowledgements that appear to come from the victim's home agent have more serious consequences than forged acknowledgements that appear to come from other correspondent nodes. If a mobile node is away from home, and its home agent does not have a valid binding for it, then the mobile node will become uncontactable. As a result, it is possible to carry out a denial of service attack on a mobile node by blocking the binding updates it sends to its home agent and forging the acknowledgements. Even if the attacker cannot prevent packets getting through, they may still be able to use forged acknowledgements to cause denial of service some of the time; if a binding update is lost for normal reasons (not as a result of the attack), then the forged acknowledgements will prevent it from being retransmitted.

This attack might also make it possible to intercept packets destined for a mobile node. Suppose that a particular network does not allow two nodes to have the same address at the same time, but will allow one node to take over another's address when the original user of the address has left the network. (This assumption does not hold with many network technologies). Then the attacker waits for a mobile node to leave the network, takes over its old care-of address, and uses forged binding acknowledgements and/or blocks the binding updates so that the mobile's home agent

never learns that mobile's care-of address has changed. Packets sent to the mobile's home address will continue to be forwarded to the old care-of address, which is now under the control of the attacker.

One possible security policy that takes into account these considerations is to require authenticated binding acknowledgments from a home agent, but to accept unauthenticated binding acknowledgments from other correspondents.

10.3 Risks of not verifying the care-of address

The BAKE/2 and CAM-DH protocols described in this memo verify that packets sent to a mobile node's claimed care-of address reach an entity that is willing to participate in the protocol. If this check was not performed, a malicious mobile node could perform a denial of service attack by asking a correspondent node to send it a high volume stream of data, and then sending the correspondent a binding update that redirects the stream of data to the victim of the denial of service attack. The acknowledgements and initial sequence number of TCP do not protect against this attack. A malicious mobile node can send the acknowledgements for the stream of data even if it is not actually receiving it. Unpredictable initial sequence numbers do not prevent a malicious mobile forging acknowledgements because the mobile sees the beginning of the stream of data (including the initial sequence number) before it redirects it to the victim.

The BAKE/2 and CAM-DH do not authenticate the care-of address. An attacker who can intercept packets sent to the care-of address can complete the protocol and cause the care-of address to be flooded with data, even if the host that actually owns the care-of address is not willing to participate in the protocol.

An alternative method of authenticating the care-of address would have been to derive the care-of address (as well as the home address) from the node's public key. We did not adopt this approach, because some subnetworks may impose constraints on the care-of addresses that can be used.

10.4 Risks of Not Authenticating Home Agents

If a mobile node is willing to allow anyone to act as its home agent (for example. suppose that it uses multicast to locate a suitable home agent) then it is vulnerable to a number of attacks in which the attacker pretends to be a home agent. For example, by acting as a node's home agent the attack can intercept packets sent to the node (a threat to confidentiality), and can cause denial of service. We observe that if an attacker is in a position to carry out these attacks, then it is also in a position to carry out other attacks of equal or greater seriousness, for example pretending to be a router.

In environments where this is a concern, the mobile should authenticate its home agent (and the next hop router, and many other services it relies on). In this case, it is not sufficient to check that the home agent's address is statistically unique; it is also necessary to check that the address corresponds to a "good" home agent, that is one that will behave in a particular way. This means that the technique of deriving addresses from public keys is not sufficient for authenticating the home agent to the mobile, because it only guarantees that the address is almost certainly not being used by anyone else. An IPSEC security association established using certificate-based key management may not be sufficient either; it is not enough to know that some authority has associated a particular key with a particular IP address, as this on its own does not provide assurance that the node at that address is a good home agent.

10.5 Denial of Service Attacks against Home Agents

Home agents are vulnerable to denial of service attacks carried out by mobile nodes for which they are the home agent. For example, a malicious mobile node that has two different home addresses from two different home agents can create a routing loop by sending the first home agent a binding update with the mobile's second home address as a care-of address, and sending the second home agent a binding update with the mobile's first home address as a care-of address. Packets caught in this routing loop will eventually time out, but there is a considerable degree of traffic amplification: for each packet that the attacker sends into the routing loop, the home agents will have to send and receive many packets.

Home agents can defend against these attacks in several ways. A home agent that will only act as home agent for mobile nodes that it knows to be trustworthy will not be vulnerable to these attacks.

References

[1] Information processing systems — Open Systems Interconnection — Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1). ISO 8825, International Organization for Standardization, 1987.

[2] Secure hash standard. FIPS PUB 180-1, NIST, April 1995.

[3] R. Hinden and S. Deering. IP Version 6 Addressing Architecture. RFC 2373, July 1998.

[4] P. Nikander. A Scaleable Architecture for IPv6 Address Ownership. Internet draft, March 2001.

[5] Greg O'Shea and Michael Roe. Child-proof authentication for MIPv6 (CAM). *Computer Communications Review*, April 2001.

11 Author's Addresses

Michael Roe
Microsoft Research Limited
7 J J Thomson Avenue
Cambridge CB3 0FB
UK
Email: mroe@microsoft.com

Tuomas Aura
Microsoft Research Limited
7 J J Thomson Avenue
Cambridge CB3 0FB
UK
Email: tuomaura@microsoft.com

Greg O'Shea
Microsoft Research Limited
7 J J Thomson Avenue
Cambridge CB3 0FB
UK
Email: gregos@microsoft.com

Jari Arkko
Oy LM Ericsson Ab
02420 Jorvas
Finland

Phone: +358 40 5079256
EMail: jari.arkko@ericsson.com