IETF Mobile IP Working Group
INTERNET DRAFT

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

Expires: April 2002

November 2001

Authentication of Mobile IPv6 Binding Updates and Acknowledgments <draft-roe-mobileip-updateauth-01.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 prevent old binding updates being replayed;

· To protect correspondents against denial of service attacks in which they are flooded with a large number of invalid binding updates;

· 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.

The three protocols differ in the amount of computation that they require and the assumptions that they make 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 Background to the Protocol Designs

1.1 IP Addresses derived from Cryptographic Keys

In the CAM-DH protocol (which will be described later), 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 [RFC2373]. 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 [RFC2373]. 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.

1.2 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 from 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 biding 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 checking the cryptography.

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 is 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 sucessfully. 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.

1.3 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 assymmetric 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 assymetric cryptography is being used to protect the mobility control portion of the message. This recommendation has affected the design of the procotols 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.

1.4 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.

2 Notation

This memo uses the following notation:

```
MN        A mobile node
CN        A correspondent node
A -> B    A sends a message to B
A -> B(HoA)  A sends a message to B at its home address
A -> B(CoA)  A sends a message to B at its care-of address
ADDRS     A node's home address and care-of address
MAC(m;K) A message authentication code computed on message m with key K
H(m)      A hash of message m
```

3 Abstract Protocols

3.1 Bake/2

Properties of the Protocol

The "Bake/2" protocol establishes a session key that is then used to authenticate binding updates sent from a mobile node to a correspondent node. The session key established by the protocol is not intended to be used for confidentiality, and is not intended to be used for authentication of messages in the reverse direction (e.g. binding acknowledgements). In particular, the correspondent node is not authenticated to the mobile node, which would be necessary if either of these two services were desired.

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 the 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 and the correspondent node, but not by attackers on the wireless network or elsewhere on the Internet.

Our motivation for designing this protocol 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 our earlier protocols for authenticating binding updates, such as CAM, 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-exisiting stored state about the correspondent and knows that it os 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.

Walkthrough

Each correspondent node has a secret key, KCN. 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, Nj, at regular intervals, for example every few minutes. A correspondent node uses the same KCN and Nj with all the mobiles it is in communication with, so that it does not need to generate and store a new Nj when a new mobile contacts it. Changing the value of Nj protects the correspondent against the replay of old messages. Each value of Nj is identified by the subscript j. j is communicated in the protocol, so that if Nj 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 Nj and the previous value N(j-1). Older values can be discarded, as messages using them will in any case be rejected as replays. KCN can be either a fixed value or regularly updated. An update of KCN can be done at the same time as an update of Nj, so that j identifies both the nonce and the key. A correspondent node can generate a fresh KCN each time that it boots to avoid the need for secure persistent storage for KCN.

1. MN -> CN : ADDRS

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 -> MN(HoA) : K0, j

K0 = MAC(HoA | Nj | 0; KCN)

The correspondent then generates two parts of a session key. One part of the session key is sent to the mobile node via its home agent; it is an assumption of the protocol that this route is secure.

3. CN -> MN(CoA) : K1, j

K1 = MAC(CoA | Nj | 1; KCN)

In parallel with step 2, the other half of the session key is sent directly to the mobile at its care of address. To obtain both halves of the key, the mobile needs to be able to receive messages that are sent via its home address, and able to receive messages sent to its care of address.

4. MN -> CN : ADDRS, 0, MAC(ADDRS, 0; K2), j

K2 = H(K0 | K1)

The mobile node hashes together the two halves of the key to form the session key, and then uses the session key to authenticate a binding update. Each binding update contains a sequence number, to prevent an attacker from replaying old binding updates. In the first binding update, the sequence number is zero. The first binding update contains j, so that the correspondent knows which value of Nj to use

to recompute the session key. Once it has verified the MAC, the correspondent can create state for the mobile. The correspondent

should remember the session key K2 and the mobile's addresses.

5. MN -> CN : ADDRS, i, MAC(ADDRS, i; K2)

In subsequent binding updates, the mobile node increments the sequence number by one each time. It is no longer necessary to include j, because the correspondent will have remembered the session key. Including j would not help because Nj is no longer fresh, and the correspondent node might have forgotten it.

Optimisations

The binding update only needs to include the bottom few bits if the sequence number, as long as the MAC is computed over the full sequence number. The network might discard or re-order messages even when it is not under attack, but messages will not encounter very long delays. On the other hand, a malicious attacker might replay a very old message. The bottom few bits are sufficient to detect non-malicious re-ordering hence to determine the full value of the sequence number to use in computing the MAC. The MAC must be computed over the full sequence number to prevent malicious replays. We explain this optimisation because in draft 14 of the mobile IP specification the sequence number field is only 8 bits wide.

If this optimisation is used, the last message is replaced with the following:

MN -> CN : ADDRS, i mod 256, MAC(ADDRS, i; K2)

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 Nj or N(j-1).

3.2 CAM-DH

The "CAM-DH" combines the Bake/2 protocol with a digitally signed Diffie-Hellman key exchange. In CAM-DH, each mobile node's home address is algorithmically related to its public signature key. The use of key-related addresses avoids the need for X.509 certificates or similar mechanisms that associate keys with addresses [OShea2001]. 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 key-related 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 -> CN : ADDRS

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

2. CN -> MN(HoA) : K0, j, g^y

K0 = MAC(HoA, Nj, 0 ; KCN)

In the second and third messages, the correspondent node sends the mobile node a key in two parts, one half being sent to the care-of

address and the other being sent via the home agent. 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 KCN, y can be constant (eliminating the need for the correspondent node to perform a modular exponentiation) or periodically updated. If y is changed, the subscript j indicates which version of y to use (as well as which KCN and Nj).

3. CN -> MN(CoA) : K1, j

K1 = MAC(CoA, Nj, 1; KCN)

4. MN -> CN : 0, ADDRS, MAC(0, ADDRS ; K3), g^x, Signature(TypeTag, g^x, HoA PK, MAC(... ; K2), j

K2 = h(K0, K1)

K3 = h(K0, g^{xy})

When it has received the two halves of the key, the mobile hashes them together to form a key that will authenticate the fourth message. 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 the session key, 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 the key that was sent in two pieces.

When the correspondent receives the fourth message, it should check the outer MAC with K2 first. It should only attempt to compute K3 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 session key K3.

5. MN -> CN : i, ADDRS, MAC(i, ADDRS ; K3)

Subsequent binding updates only need to be protected with K3. They include a sequence number to protect against replay attacks.

Optimizations

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 -> HA : K0, j, g^y

HA -> MN : K0, K3, j

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 should contain an additional flag that indicates to the mobile node that the correspondent is using this optimisation. When this flag is present in the second message, the mobile should send the third 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.

3.3 Shared Key

Properties of the Protocol

The shared key protocol is used to authenticate binding updates between a mobile node and a correspondent node that already share a symmetric key. For example, a mobile node might use this protocol to send binding updates to its home agent. This protocol protects binding updates against replays by using the long-term shared key and a random challenge to derive a fresh session key. This protocol also checks that messages sent to the care of address reach an entity that is willing to participate in the protocol; this helps prevent denial of service attacks in which a mobile sets their care of address to the victim's address and then causes them to be flooded with data sent by a third party.

Walkthrough

1. MN -> CN : ADDRS

2. CN -> MN(CoA) : K1, j

K1 = MAC(CoA | Nj | 1 ; KCN)

3. MN -> CN : ADDRS, 0, MAC(ADDRS, 0 ; K2), j

K2 = H(K0, K1)

4. MN -> CN : ADDRS, i, MAC(ADDRS, i ; K2)

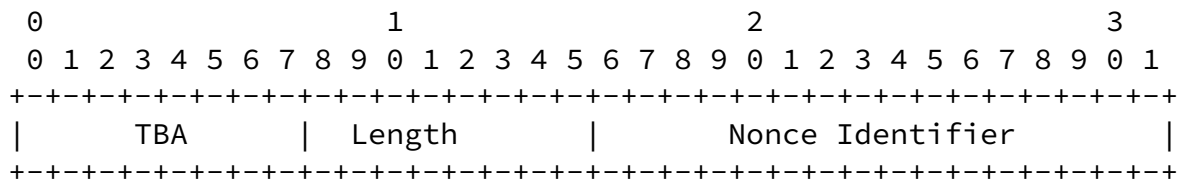
The variables KCN and j are the same as in the BAKE/2 protocol. K0 is the secret shared between the mobile node and the correspondent node.

4 New IPv6 Sub-option Types

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

4.1 Challenge Serial Number

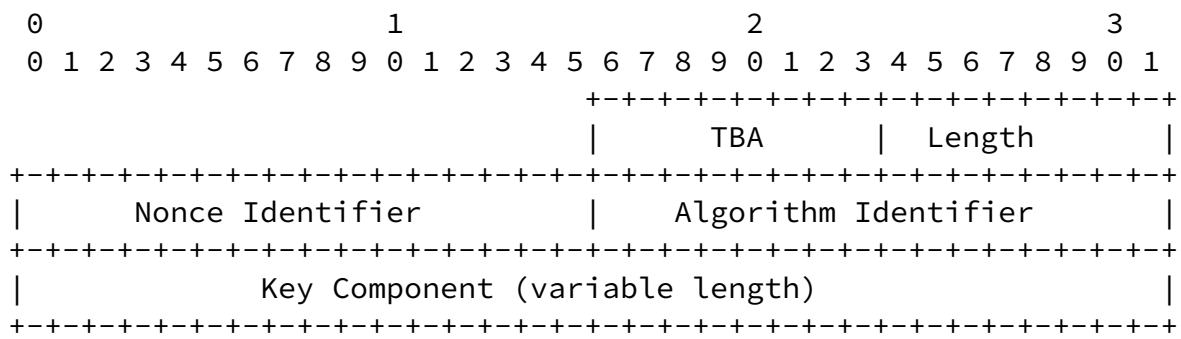
Alignment requirement: none



The challenge serial number sub-option is valid in a Binding Update destination option. The Nonce Identifier field contains the variable j in the BAKE/2 and CAM-DH protocols. That is, it tells the correspondent node that receives the suboption which of the challenge values (Nj) are to be used to authenticate the binding update.

4.2 Home Address Key Component

Alignment requirement: none



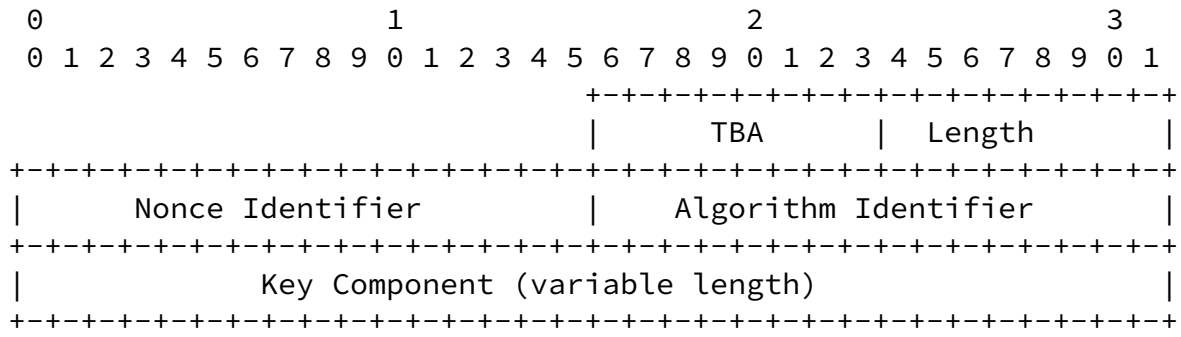
The Home Address Key Component sub-option is valid in a Binding Request destination sub-option. The The Nonce Identifier field contains the variable j in the BAKE/2 and CAM-DH protocols. The Algorithm Identifier field indicates which cryptographic algorithm

should be used to compute the authentication information field in the Binding Update that is sent in response to this option. The Key Component field contains the key component K0 in the BAKE/2 and CAM-DH protocols; it is the first of two components which are to be

concatenated and hashed to form a key which is then used to authenticate binding updates.

4.3 Care-of Address Key Component

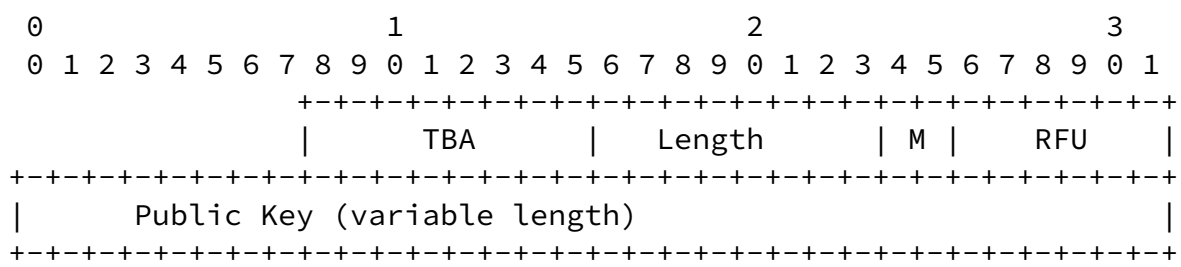
Alignment requirement: none



The Care-of Address Key Component sub-option is valid in a Binding Request destination sub-option. The The Nonce Identifier field contains the variable j in the BAKE/2 and CAM-DH protocols. The Algorithm Identifier 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 Key Component field contains the key component K1 in the 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.

4.4 Address-related RSA Public Key Sub-option

Alignment requirement: none



The Address-related RSA Public Key sub-option is valid only in Home Address destination options. The Public Key field contains an RSA public key, encoded as the ASN.1 Basic Encoding Rules [ISO8825] representation of the type PublicKey.

```
PublicKey ::= SEQUENCE
              {
                  modulus INTEGER,
                  exponent INTEGER
              }
```

The RFU (reserved for future use) field SHALL contain zero bits. Packets in which these bits are non-zero MUST be rejected as invalid. (See the security considerations section of this memo for the rationale for this)

The following relationship SHALL hold between the public key field and the network part of the home address, where SHA1 is the SHA-1 message digest algorithm [SHA1] and First64 extracts the first 64 bits of the 160 bit hash value.

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

Packets where this relationship does not hold MUST be rejected as invalid.

5 Other Message Formats

NB: This protocol specification is currently incomplete, as it does not fully described how the abstarct protocols of section 4 are encoded, This information will be provided in the next version of the document.

6 Intellectual Property Rights Notice

The CAM-DH variant of our protocols uses public keys and hashes to prove address ownership [Nikander2001,OShea2001]. 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.

7 Security Considerations

7.1 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.

7.2 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 updates from a home agent, but to accept unauthenticated binding updates from other correspondents.

7.3 Risks of not verifying the care-of address

The BAKE/2 and CAM-DH protocols described in this memo verify that 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.

7.4 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.

7.5 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 consideration 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

[ISO8825] 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.

[SHA1] Secure Hash Standard. FIPS PUB 180-1, NIST, April 1995

[RFC2373] R. Hinden and S. Deering, IP Version 6 Addressing Architecture. RFC 2372, July 1998.

[Nikander2001] P. Nikander, A Scaleable Architecture for IPv6 Address

Ownership. Internet Draft, March 2001.

[OShea2001] Greg O'Shea and Michael Roe, Child-proof authentication for MIPv6 (CAM). Computer Communications Review, April 2001.

8 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