F I V E

Securing Mobile IPv6 Signaling

Mobility adds inherent security risks to those already in the Internet today. Some of these risks are introduced by the specific mobility protocol. Mobile IPv6 is a new protocol that attempts to do something that has not been done before on the Internet: redirect traffic between a mobile node and other correspondent nodes from one address to another. The signaling for such redirection is done between the mobile and correspondent nodes. To be able to design a protocol that avoids some or all of the security risks associated with it, we need to identify the types of threats specific to this protocol. Then we need to place requirements on the protocol to avoid some or all of these threats. In some cases, it is acceptable to have known threats associated with a protocol, provided that they are documented and understood. The output of the requirements study is used to test the protocol and see whether or not it conforms.

In this chapter, we focus on the security threats that result from the introduction of Mobile IPv6. We analyze different Mobile IPv6 messages and show how each one can be used by Bad Guy to produce undesired effects to the mobile node, correspondent node, and home agent. We then present the mechanisms used by Mobile IPv6 to secure its messages.

5.1 Why Do We Need to Secure Mobile IPv6?

Before we analyze the threats of Mobile IPv6's messages, we consider two different communication scenarios that are possible when Mobile IPv6 is used. Figure 5–1 shows the different cases.

A mobile node may tunnel its packets to the home agent, which in turn decapsulates and forwards them to the correspondent node. If route optimization were used (i.e., the mobile node sent a binding update to the correspondent

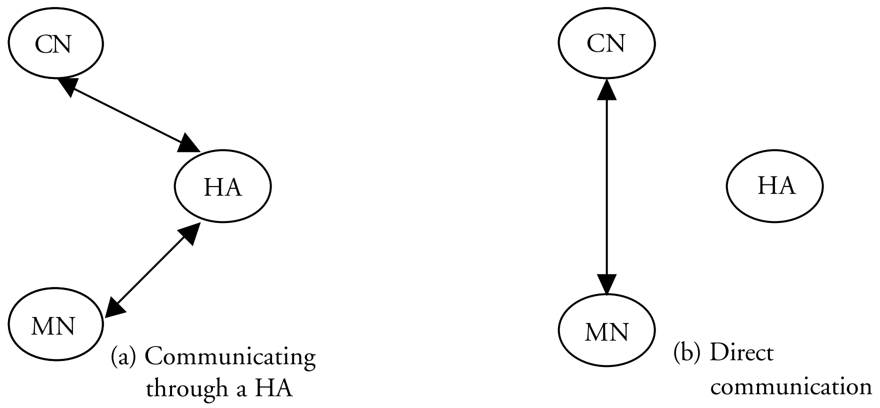


Figure 5–1 *Communication scenarios with and without route optimization.*

node), the mobile node would send packets directly to the correspondent node after adding a *home address* option, as described in Chapter 3. The correspondent node would also send packets directly to the mobile node using a *routing header type 2* that includes the mobile node's home address. We need to analyze the types of attacks that Bad Guy can launch when he is on-path or off-path. An on-path attacker is one that can see packets going through a certain link between two nodes. For instance, an attacker can be on-path between the mobile and correspondent nodes if he is located at the mobile node's link, the correspondent node's link, or any link between the two where packets between the two nodes are routed. On the other hand, an off-path attacker is unable to see packets sent between the two nodes he is trying to attack. Now let us consider the different attacks that Bad Guy can launch using this protocol.

5.1.1 Using Binding Updates to Launch Attacks

The binding update is used to redirect traffic from one address to another. If not used carefully, it can have some detrimental effects on the mobile–correspondent communication. It should be noted that while our examples are focused on the mobile–correspondent node communication, they can all be applied to the mobile node–home agent binding updates.

5.1.1.1 STEALING TRAFFIC

If Bad Guy knows the mobile node's home address (which is not difficult, since it is public knowledge and may be stored in the DNS), he can send a binding update to a correspondent node to redirect the mobile node's traffic to himself, as shown in Figure 5–2.

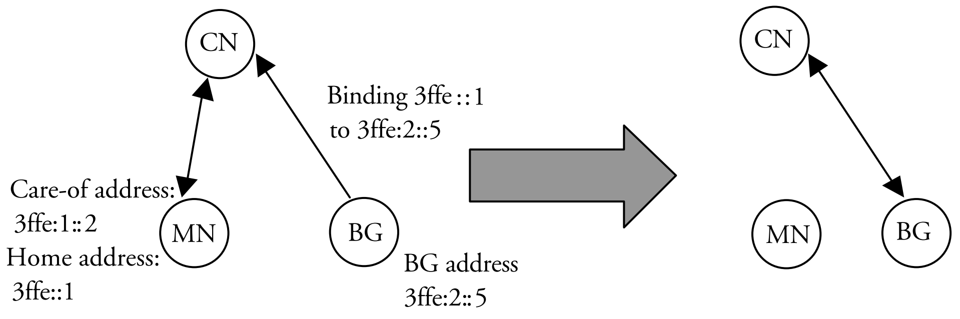


Figure 5–2 *Bad Guy (BG) stealing the mobile node's traffic.*

In this example, Bad Guy steals the traffic originally intended for the mobile node and could possibly hijack the mobile node's connection with a correspondent node and pretend to be the mobile node for the rest of the connection. Note that in this case, Bad Guy is not on the path between the mobile and correspondent node.

5.1.1.2 REFLECTION AND FLOODING ATTACKS

When Alice and Bob are communicating, it is perfectly acceptable that Alice send a packet to Bob, which causes Bob to send a reply back to Alice. However, if Alice sends a packet to Bob that causes Bob to reply to Angela, it is called a *reflection attack*. While it is perfectly acceptable for one node to communicate with another via a relaying agent, a packet sent from A to B should not cause B to **reply** to X. It is acceptable for B to **relay** the same message to X and for X to **reply** to A (note the distinction between relaying and replying), but it is unacceptable to have B replying to another node. Bad Guy (who need not be on-path) may use a binding update to launch a reflection attack. Figure 5–3 shows an example of a reflection attack.

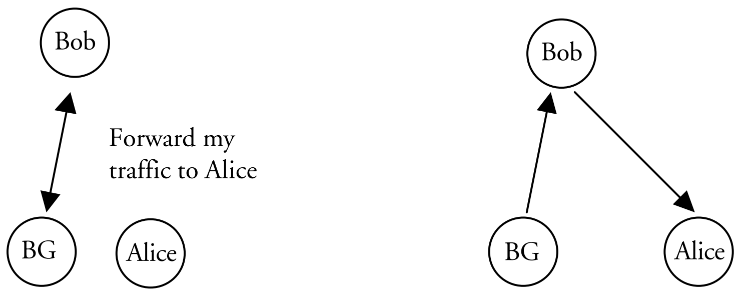


Figure 5–3 *Reflection attack.*

In this case, Bad Guy can start a connection with Bob. After the connection starts, Bad Guy can send a binding update to Bob asking him to direct his traffic to another address (pretending that he moved), which is assigned to Alice. This would cause Bob to send all packets to Alice. If Bob is a server streaming video, for example, or Bad Guy was downloading a large file from Bob, this would cause Bob to **flood** Alice's link with information that she is not interested in. Flooding attacks are sometimes called **bombing** attacks.

5.1.1.3 MAN IN THE MIDDLE (MITM) ATTACKS ON THE BINDING UPDATE

MITM attacks (Figure 5–4) involve Bad Guy on the communication path between two nodes. Bad Guy may interfere with the communication in different ways. The most obvious is to change the contents of a packet to cause some effects that were not intended by the original sender of the packet.

When Bad Guy is located on the path between a mobile and a correspondent node, he can modify the content of the binding update, possibly causing a reflection attack or hijacking of ongoing connection(s) between the mobile and correspondent nodes.

5.1.1.4 DENIAL OF SERVICE ATTACKS (DOS)

DoS attacks can be done in several ways, with or without Mobile IPv6. The aim of a DoS attack is to deny service to a legitimate node. For example, an attacker may send many bogus messages to a particular server (e.g., asking to initiate a connection) to keep it so busy that it cannot respond to legitimate nodes. These kinds of attacks are very difficult to stop. Avoiding them usually involves having a Firewall that detects that too many requests are coming from a certain address and can drop all or some of them. However, this is not a real solution, since the attacker can use more than one address (i.e., distributed DoS, or DDoS, attacks).

Using Mobile IPv6, however, Bad Guy may use the binding update to deny a mobile node service from a particular correspondent node. This can be done by sending a binding update to the correspondent node asking it to forward packets addressed to the mobile node to another address (not the mobile node's care-of address). Bad Guy may also continue to refresh the correspondent node's binding cache to stop it from directing traffic to the mobile node's real location. Clearly the mobile node would try to rectify the situation by sending a binding update that includes the real care-of address. However, if Bad Guy happens to be on the path between the mobile and correspondent nodes (MITM), he can modify the packet to suit his attack.

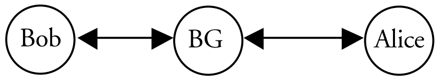


Figure 5–4 *Man in the middle attack.*

Other types of DoS attacks can be launched against a correspondent node by sending a large number of bogus binding updates (i.e., bindings that do not necessarily correspond to real home addresses) that can fill up the correspondent node's memory allocated for storing binding cache entries. This would result in the correspondent node being unable to process new messages sent by real mobile nodes. The same attack can be launched against a home agent.

A DoS attack may also involve Bad Guy somehow compromising and controlling a router on the path between the mobile node and correspondent node or the home agent. In this case, Bad Guy might decide to simply drop the mobile node's packet, hence denying it service. However, this type of attack is not specific to Mobile IPv6. In fact there is nothing that can be done for it but make it impossible (or extremely difficult) for Bad Guy to be able to compromise a router.

5.1.2 Attacks Using the Routing Header and Home Address Option

The routing header type 2 is included in packets sent from the correspondent node to the mobile node; it includes the mobile node's home address. Therefore, the routing header allows the mobile node, upon receiving the packet, to place its home address in the destination field and effectively forward the packet (internally) to itself. However, what would happen if the address in the routing header were not the mobile node's home address? The mobile node would follow the same steps, except the packet would be forwarded to the new destination address (which does not belong to any of the mobile node's interfaces). If Bad Guy were communicating with a mobile node, he might put another address in the routing header, causing the mobile node to forward his packets to another node. This is not the same type of attack as the reflection attack described earlier, since the mobile node would not be replying to another node, but relaying Bad Guy's packets instead. However, this is still an undesirable effect for two reasons: first, Bad Guy might be using the mobile node as a relay to another node that, under normal circumstances, he is not authorized to communicate with (see section 2.2.2); and second, this use of the routing header type 2 was not the intention of the Mobile IPv6 protocol.

If Bad Guy were a mobile node instead, he might attempt to create a reflection attack similar to the one in section 5.1.1.2, using the home address option. In this case, Bad Guy could send a packet containing a home address option to a correspondent node. Since correspondent nodes replace the source address of the packet with the one in the home address option (to maintain transparency to upper layers), the reply to this message would go to the address included in the home address option. If this is not Bad Guy's address, another node (victim) will receive this information, which could possibly flood its link.

5.1.3 MITM Attacks on MPS/MPA

The Mobile Prefix Solicitation (MPS) and Mobile Prefix Advertisement (MPA) messages are used by the mobile node to learn about prefixes advertised on its home link. The MPS is sent by the mobile node and causes the home agent to send an MPA. The mobile node uses the information in the MPA to bind its home addresses to its current care-of addresses. Bad Guy can launch an MITM attack on these messages if he happens to be on the path between the mobile node and its home agent. Bad Guy can intercept packets and change their contents. Let's consider some of the undesirable effects:

- *Modifying lifetimes:* Suppose that the home network was being renumbered, or that for some reason, one of the prefixes on the home link was being removed. In this case, routers are likely to gradually reduce the lifetime on the prefix being removed to allow mobile nodes to pick the prefix with a higher lifetime. Eventually, a prefix will be deprecated and should not be used by the mobile node to form a home address. Suppose that Bad Guy saw one prefix (say prefix_x) with a lifetime of 10 minutes and another (say prefix_y) with a lifetime of 2 hours. He could swap the lifetimes of the prefixes. The mobile node would pick prefix_x, as it seems to have a longer lifetime. However, in 10 minutes prefix_x is going to be deprecated. When this happens, the mobile node will be unreachable.
- *Impersonating the home agent:* Bad Guy may intercept the mobile node's message (or simply read it) and generate his own replies. He could make up some prefixes or use prefixes that would cause the mobile node's traffic to be sent to his link. The mobile node would send a binding update to Bad Guy (thinking it is communicating with a home agent). If the binding acknowledgment were not authenticated, Bad Guy might simply reply to the mobile node, informing it of a successful binding. From this point on, all of the mobile node's traffic would be directed to Bad Guy. He might copy the packets and forward them, or he might drop the packets (DoS). Both actions will harm the mobile node and its correspondents, and should be avoided.

5.2 Requirements for Mobile IPv6 Security

We have now seen how dangerous Mobile IPv6 can be if not secured. The next step is to understand what type of security is needed and which threats are more relevant than others. Many of the requirements on Mobile IPv6 security are summarized by the IESG's recommendation to the Mobile IP WG: "Do no harm to the existing Internet." In other words, today's IPv4 Internet has known threats and is not perfect, but we ought not make it worse by introducing new threats. Hence, if a threat is known to exist in today's Internet, we do not have to solve it in this protocol (of course, solving it would be a bonus but is not required). From this high-level requirement, we can extrapolate that off-path attacks cannot be tolerated. MITM attacks can be done on today's Internet in some situations; however, off-path attacks like the ones described in section 5.1.1 should not be introduced by Mobile IPv6. That is, Mobile IPv6 signaling should be as secure as if the mobile node were located at the home link and no movement was involved.

Mobile devices vary in shapes and sizes; however, one of their common characteristics is that they are usually constrained by how much battery power they can have. Hence, the chosen cryptographic mechanism should not rely on computationally intensive algorithms (although this might be an optional feature for strong authentication, it should not be the only one).

In Chapter 4 we saw how the use of traditional public key cryptography needs Public Key Infrastructure (PKI) support—that is, hierarchies of certificate authorities—for proof of identity. Currently there is no global PKI on the Internet, and there is no expectation of global PKI deployment in the near future. This is mainly due to the difficulty involved in setting up trust between the different CAs. Mobile IPv6 should be deployed on a global scale; mobile nodes might be communicating with any random correspondent node on the Internet. Hence, a Mobile IPv6 security solution for mobile node to correspondent node signaling cannot assume the existence of global PKI.

On top of the general requirements discussed above, each function needs to have its own set of requirements.

5.2.1 Securing Communication Between Mobile and Correspondent Nodes

Messages between the mobile and correspondent nodes are needed for binding cache and binding update list management in the correspondent and mobile nodes respectively. The binding update is essentially acting as a redirection request, which if not secured can produce the undesirable result shown in section 5.1.1. Therefore, it is crucial that a correspondent node trust the mobile node to be able to accept this request. Let's consider what requirements this statement would generate.

5.2.1.1 HOW DOES A CORRESPONDENT NODE TRUST A MOBILE NODE?

Authorization is conditional. One might authorize a doctor to perform a certain surgical procedure because she can provide certain credentials that would make us believe that she is qualified for certain tasks. A heart surgeon is not generally trusted to perform plastic surgery, and an electrician is allowed to fix power problems but not problems related to a human's brain. All these people are trusted because they can prove that they meet a set of requirements that authorize them to perform their tasks (i.e., a trusted license). The same logic is needed when mobile nodes communicate with correspondent nodes. The important question is, What does a mobile node need to prove in order to be authorized to redirect packets from one address to another? Is a subject field of *Bob@example.com* in an X.509 certificate sufficient to authorize Bob to forward packets from `3ffe:1::1 to 3ffe::5`? The answer is no (in the general case, but of course if Bob is **known** to a correspondent, then this may be acceptable). If this were true, then we would be implying that Bob is a good person who would not do anything wrong, and if he does, we can sue him—which implies absolute trust and late detection of any breach of that trust, but does not **prevent** Bob from breaching the trust, which is our goal (prevention is better than late detection). The fact is, hosts on the Internet are allowed to forward traffic to themselves only. Hence, if a mobile node wishes to redirect traffic from one address to another, it needs to prove that it has such authority by proving that it is in fact assigned both addresses. This is the most important requirement for securing binding updates: proof of address ownership. The proof of home and care-of address ownership gives the mobile node the authority to request a redirection of traffic between those two addresses.

5.2.1.2 MESSAGE INTEGRITY: AVOIDING MITM ATTACKS

Even if Bob can prove address ownership, Bad Guy might be on the path between Bob (mobile node) and Alice (correspondent node). He can modify the contents of the binding update message and the IPv6 header to serve his purposes. Hence, Alice needs to check whether Bob's message was modified en route. This requires binding updates to include some form of message authentication. The same requirement can be placed on the binding acknowledgment; a mobile node needs to ensure that the binding acknowledgment came from the correspondent node. Otherwise, Bad Guy might impersonate the correspondent node and reply with a fake binding acknowledgment. Hence, mutual authentication between the mobile and correspondent nodes is needed.

5.2.1.3 AVOIDING DOS ATTACKS IN CORRESPONDENT NODES

The DoS attack mentioned in section 5.1.1.4 involves filling up the correspondent node's binding cache with unnecessary information. To avoid this type of attack, correspondent nodes must ensure that they do not maintain state per mobile node until the binding update is accepted. Hence, the protocol needs to be stateless from the correspondent node's viewpoint until an authenticated binding update message is received. This allows correspondent nodes to avoid the DoS attacks described earlier.

5.2.2 Securing Messages to the Home Agent

The same attacks on the binding update to correspondent nodes are relevant when a mobile node is communicating with its home agent. In addition, home agent impersonation (see sections 5.1.3 and 5.1.1.3) could be detrimental to the mobile and correspondent nodes. Hence, binding updates and acknowledgments and MPS/MPA exchanged between a mobile node and its home agent should be authenticated. Due to the importance of the home agent, it is important that mobile nodes' communication is secured using strong authentication. This is not a hard requirement, since the home agent(s) is known and mobile nodes are not expected to pick them at random, which is the case for correspondent nodes. There is no requirement for securing DHAAD messages; the reasons are shown in section 5.3.2.1.

5.2.3 Assumptions about Mobile IPv6 Security

We can certainly assume that any route optimization security solution that involves manual configuration is not useful as a generic solution. Hence, symmetric cryptography that requires manual configuration of keys in mobile and correspondent nodes is not a generic solution. A solution that requires no manual configuration for mobile–correspondent node bindings is needed. Furthermore, we can also assume that a solution that requires global PKI falls in the same—infeasible—category. We need a solution that requires no infrastructure support. Ideally, only nodes implementing Mobile IPv6 functions should be involved in securing the protocol between the mobile and correspondent nodes. On the other hand, a mobile node's home agent is a known node. Hence, manual configuration of security associations between the mobile node and its home agent is possible. This would allow for securing binding updates and MPS/MPA messages with strong authentication techniques. In addition, public keys can be used to secure binding updates between the mobile node and its home agent without requiring global PKI. The mobile node and its home agent can belong to the same trust domain; hence, only local PKI would be needed (e.g., a network operator may issue and sign certificates to its mobile nodes and home agents). Assuming a secure channel between the mobile node and its home agent also reduces the number of paths vulnerable to MITM attacks between the mobile and correspondent nodes. This reduces the number of locations that can be used by Bad Guy to launch attacks (see Figure 5–1).

5.3 Mobile IPv6 Security

Mobile IPv6 signaling utilizes several messages that require different levels of security. When dealing with the mobile node–home agent messages, it is possible to use manual configuration to set up an IPsec security association. However, this approach is not generally feasible between mobile and correspondent nodes. In the following sections we address each message and show how they can be secured.

5.3.1 Securing Binding Updates to the Home Agent

A home agent and a mobile node are expected to have prior knowledge of each other. Home agents could be provided by ISPs to serve their mobile subscribers. Subscribers typically have trusted credentials and a security relationship with their ISPs. The same type of relationship can be set up between mobile subscribers and their home agents. This would allow home agents to know the right credentials needed by mobile nodes to use certain home addresses. Furthermore, it would allow a home agent to impose certain actions on misbehaving mobile nodes, for instance, by terminating the home agent service provided to these nodes. This special relationship between a mobile node and its home agent allows us to relax the requirements on some of the threats mentioned earlier; specifically, reflection attacks shown in section 5.1.1.2 can be traced to the misbehaving mobile node. A responsible network manager would not want to have mobile nodes that use the network to flood other links in his network or in other networks on the Internet. For this reason, a mobile node and its home agent can be treated as a single entity when it comes to these attacks. A consequence to this assumption is that we do not require mobile nodes to prove ownership of their care-of addresses to their home agents.

It should be noted that home agents and mobile nodes are not required to have a subscriber–ISP relationship. A user might wish to have a home agent at home (in a house) that serves her mobile devices (mobile phone, PDA, laptop, etc.). The same assumptions are still valid for this case. If someone owns a home agent, there is no difference between using that home agent to flood other nodes' links on the Internet and using her mobile node to directly flood these links. The fact that packets are reflected makes no difference, since the attacker administers both entities (mobile node and home agent). Therefore, when sending a binding update to a home agent, the only things a mobile node needs to prove are that

- it has been allocated the home address (i.e., it is authorized to redirect traffic), and
- it is authorized to use this particular home agent to forward its packets (i.e., it was authorized to use the forwarding service).

To satisfy these requirements, an IPsec security association is needed between the mobile node and its home agent. IPsec uses IP addresses to identify security associations in its Security Association Database (SAD). Hence, by knowing the mobile node's home address and using that to look up a corresponding security association, a home agent can verify both points above in a simple way if the security association was manually configured. That is, since the network administrator configures a security association to protect the mobility header, then he is implicitly allowing a mobile node (which knows the secret key) to use this home agent to forward its traffic; and since each address has a single key associated with it, knowing that key is a proof that the mobile node has been allocated a particular home address.

If the security association were established dynamically, using public keys and certificates, the mobile node would need to provide credentials showing that it has been allocated this particular home address. This can be done in different ways. One way is to include the mobile node's home address in a certificate signed by a trusted CA. This might be the ISP's own CA or another one trusted by the home agent. Clearly, if the CA does not belong to the ISP, it needs to verify with the ISP (through an out-of-band method) that the ISP has allocated a particular home address to the mobile node. Alternatively, the association between a particular certificate and a corresponding home address can be manually configured in both the home agent and the mobile node. This adds more manual configuration in the home agent; however, it is not a significant amount. In both cases, the mobile node needs to know its home address. However, the latter method does not require any cooperation between the CA and the ISP (except, of course, the home agent must trust the CA). This is a significant advantage and involves less work when the home network is being renumbered.

The next issue is how to protect Mobile IPv6 messages between the mobile node and home agent using the established security association. IPsec has two different mechanisms, AH and ESP (discussed in Chapter 2). Both can be used to get authentication and replay protection. However, strong replay protection can only be achieved with dynamic keying. When preconfigured keys are used (secret key cryptography), only the sequence number in the binding update can be used. However, this would not provide strong replay protection (the sequence number rolls over and the same numbers are used again with the same key). The major difference between AH and ESP is that ESP provides confidentiality (encryption) while AH does not. Another difference is that AH protects the entire packet, including the IPv6 header; that is, the care-of address (in the source address) is protected. On the other hand, ESP only protects the headers below it; hence, the source address is not protected. Consequently, when using ESP, the *alternate care-of address* option must be included in the mobility header, which is covered by ESP (and should contain the same address as the source address field).

When considering the binding update and acknowledgment messages, we can see that the most important requirement is mutual authentication and message integrity. The mobile node and the home agent need to authenticate each other and ensure that the message was not modified by MITM. Both AH and ESP (for ESP, authentication must be enabled) can achieve this. Since binding updates are sent to set up a tunnel between the mobile node and the home agent (i.e., before the tunnel is set up), either AH or ESP can protect the binding update only when used in *transport* mode.

The IPsec security association should be based on the mobile node's home address. Therefore, when sending a binding update to the home agent, the mobile node needs to include its home address in the packet (remember that the source address is the care-of address). This is done by including the home address inside the *home address* option, which allows the home agent to look up the security association based on the mobile node's home address. Similarly, when sending a binding acknowledgment to the mobile node, the home agent needs to include a routing header type 2, including the mobile node's home address, to allow mobile nodes to find the right security association. An overview of the mobile node and home agent processing of binding updates is shown in Figure 5–5. Figure 5–6 shows an overview of the binding acknowledgment processing by the mobile node and home agent.

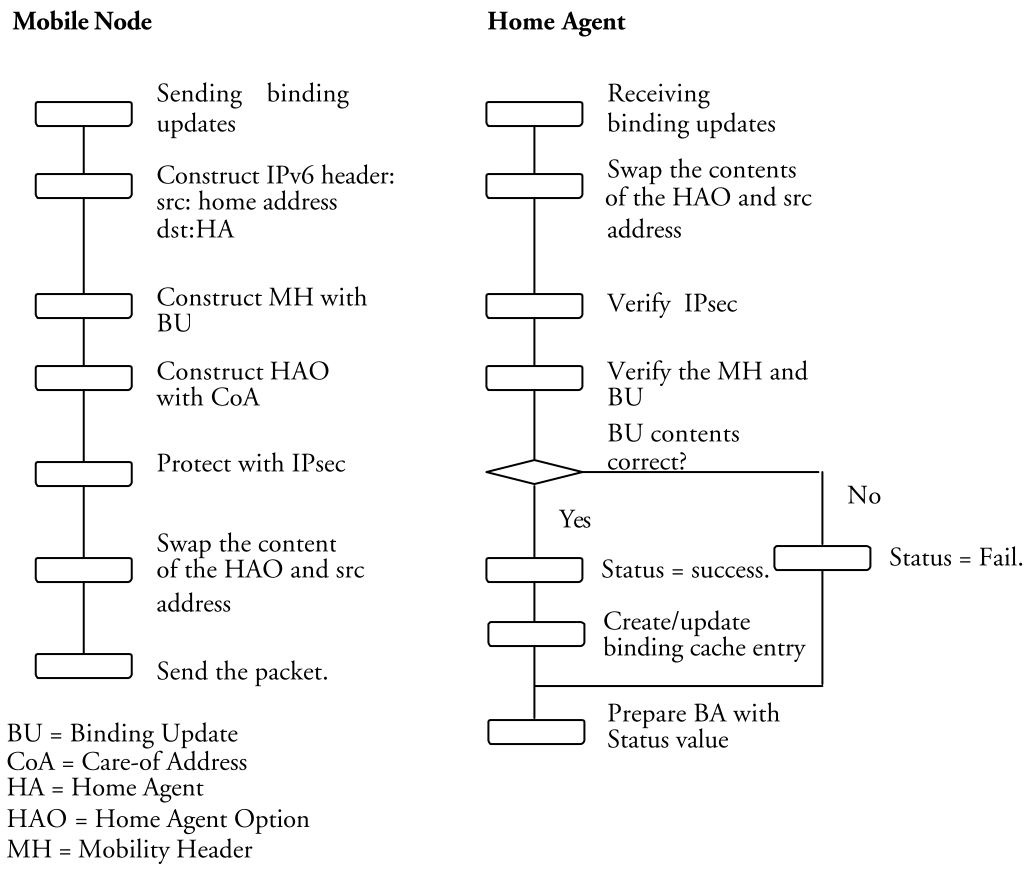


Figure 5–5 *A simplified flowchart for sending and receiving binding updates by the mobile node and home agent respectively.*

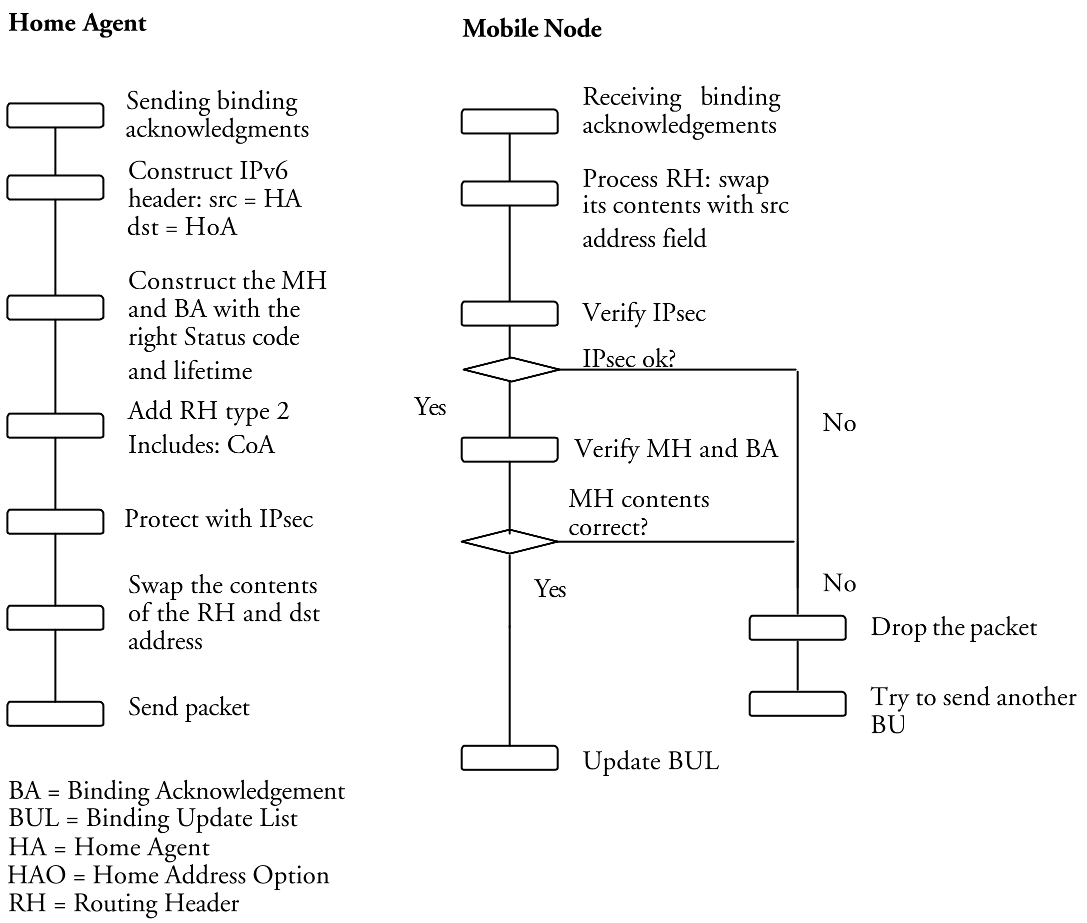

Home Agent
Sending binding acknowledgments
Construct IPv6 header: src = HA dst = HoA
Construct the MH and BA with the right Status code and lifetime
Add RH type 2 Includes: CoA
Protect with IPsec
Swap the contents of the RH and dst address
Send packet
Mobile Node
Receiving binding acknowledgements
Process RH: swap its contents with src address field
Verify IPsec
IPsec ok?
Yes
No
Verify MH and BA
MH contents correct?
Yes
No
Drop the packet
Try to send another BU
Update BUL
BA = Binding Acknowledgement
BUL = Binding Update List
HA = Home Agent
HAO = Home Address Option
RH = Routing Header

Figure 5–6 *A simplified flowchart for sending and receiving binding acknowledgments by the home agent and mobile node respectively.*

Note that while the contents of the binding update list and binding cache are stored in volatile memory, the sequence number requires a different treatment when dynamic keying is not used (i.e., no IKE). The home agent needs to make sure that the last valid sequence number used is stored in nonvolatile memory to be able to survive reboots. Otherwise, the home agent will accept a replayed binding update message after a reboot, causing it to tunnel traffic to a potentially wrong address. This problem does not exist when dynamic keying is used, as a reboot results in losing the contents of the SAD (stored in volatile memory). Hence, a security association must be negotiated before a binding update can be accepted.

Note that when dynamic keying is used, it is possible for the mobile node to establish a security association with its home agent that requires renegotiation every time the mobile node moves. In this case, the *K* flag in the binding update (refer to Chapter 3) would be cleared. However, this implies that the mobile node needs to renegotiate a new security association whenever its care-of address changes (i.e., every time it moves). This case is not covered in the flowcharts shown in this section.

5.3.2 Securing Mobile Prefix Solicitations and Advertisements

The requirements and security mechanism for MPS and MPA are basically the same as those for the binding update/acknowledgment. The main difference is the configuration of the SAD and Security Policy Database (SPD); they clearly need to indicate ICMP for the protocol type, as opposed to the *mobility header* used for binding updates and acknowledgments.

5.3.2.1 WHAT ABOUT DHAAD?

Dynamic Home Agent Address Discovery allows a mobile node to discover home agent addresses while located on a foreign link. The actual messages do not need to be hidden; the home agent's address is not a secret. It is exposed in all packets tunneled to and from the home agent and cannot be hidden. Bad Guy might intercept packets, modify them, or reply on behalf of the home agent. However, none of this would cause traffic disruption, since the mobile node sends an authenticated binding update and expects an authenticated binding acknowledgment, as shown in the previous sections. The attacker cannot send an authenticated binding acknowledgment to the mobile node, and no harm can be done. The best that Bad Guy can do is launch a DoS attack, which is already possible today. Therefore, no Mobile IPv6-specific attacks can be launched using DHAAD.

5.3.3 Manual Versus Dynamic SAs Between the Mobile Node and Its Home Agent Configuration

Two types of keying mechanisms can be used when setting up a security association with the home agent: static and dynamic keying. In the static case, a network administrator configures a security association on the home agent and the mobile node; in the dynamic keying case (e.g., IKE), public key cryptography can be used with appropriate certificates, as discussed earlier. We now see how static security associations can be established.

IPsec security associations are unidirectional. That is, when establishing a security association between two nodes for bidirectional communication, one needs to configure a security association for each direction: IN (for traffic routed toward a node) and OUT (for traffic sent from the node). In our example, we set up a security association between the mobile node and its home agent to protect the *mobility header*. Each node contains the necessary information for both the SAD and the SPD. In this example, we use the same encryption and authentication keys for both directions of communication.

Example

Let's consider an example where the mobile node's home address is `3ffe:200:8:1:1234:5678::1`, its care-of address is `3ffe:ffff:2:1:1234:5678::1`, and its home agent's address is `3ffe:200:8:1::5`. To set up a security association for binding update protection, we need to configure the mobile node and the home agent with the necessary keys, Security Parameter Index (SPI), and the protocol to be protected.

The SPD configuration has the following format:

```
spdadd <source> <destination> <protocol> <dir> <protection>
<mode>
```

where `<source>` is the source address, `<destination>` is the destination address, `<protocol>` is the protocol to be protected, `<dir>` is the direction of communication, `<protection>` is the type of IPsec protection (ESP or AH), and `<mode>` specifies *transport* or *tunnel* modes. Note that the source and destination may contain a range of addresses, for instance, by specifying a prefix. Using this definition, let's see how this can be used to configure the SPD in the mobile node and home agent for binding update/acknowledgment protection using ESP transport mode. The following is the SPD configuration in the mobile node.[1]

```
#setkey –c

spdadd 3ffe:200:8:1:1234:5678::1 - 3ffe:200:8:1::5 mh –P out
esp/transport/require;

spdadd 3ffe:200:8:1::5 - 3ffe:200:8:1:1234:5678::1 mh –P in
esp/transport/require;
```

In English, this essentially implies two Boolean conditions:[2]

- If any outgoing packet has a source address of `3ffe:200:8:1:1234:5678::1`, a destination address of `3ffe:200:8:1::5`, and carries an extension header of type *mobility header*, it must (the "require" part) be protected by ESP in transport mode.
- If any incoming packet has a source address of `3ffe:200:8:1::5`, a destination address of `3ffe:200:8:1:1234:5678::1`, and carries an extension header of type *mobility header*, it must be protected by ESP in transport mode. If not, the packet must be dropped.

The same configuration for the SPD must be done for the home agent:

```
#setkey –c

spdadd 3ffe:200:8:1::5 - 3ffe:200:8:1:1234:5678::1 mh –P out
esp/transport/require;

spdadd 3ffe:200:8:1:1234:5678::1 - 3ffe:200:8:1::5 mh –P in
esp/transport/require;
```

Now let's consider how the SAD can be configured:

```
Add <source> <destination> <protection> <mode> <spi> <algconf>
```

The first four fields are the same as those mentioned earlier for the SPD. The `<spi>` is the SPI used by the IPsec header, and `<algconf>` is a set of parameters describing the keys and algorithms to be used for encryption, authentication, or both. The following is an example of an SAD configuration command in the mobile node (this is a continuation of the previous command, i.e., `setkey –c`):

```
add 3ffe:200:8:1:1234:5678::1 - 3ffe:200:8:1::5 esp 0x10001 -m
transport –E 3DES "maryhasalittlelamb121234"

-A hmac-sha1 "this is an mn example";

add 3ffe:200:8:1::5 - 3ffe:200:8:1:1234:5678::1 esp 0x10002 -m
transport –E 3DES "maryhasalittlelamb121234"

-A hmac-sha1 "this is an mn example";

EOF
```

In this example we configured the mobile node to use ESP in transport mode to protect any packets containing the mobility header. The packets will be encrypted with 3DES (key is `maryhasalittlelamb121234`) and authenticated with the message authentication code HMAC-SHA1 (key is `this is an mn example`). The switch `–E` indicates encryption algorithm; `-A` indicates authentication algorithm, and `-m` indicates mode.

Continuing with the home agent configuration, the SAD configuration is done as follows:

```
add 3ffe:200:8:1:1234:5678::1 - 3ffe:200:8:1::5 esp 0x10001 -m
transport –E 3DES "maryhasalittlelamb121234"

-A hmac-sha1 "this is an mn example";

add 3ffe:200:8:1::5 - 3ffe:200:8:1:1234:5678::1 esp 0x10002 -m
transport –E 3DES "maryhasalittlelamb121234"

-A hmac-sha1 "this is an mn example";

EOF
```

[1] We use an example configuration for a FreeBSD operating system (www.freebsd.org).

[2] There was no protocol type "mh" defined in FreeBSD at the time of writing because the Mobile IPv6 implementation was not complete yet. This (mh) was added for the purpose of illustration.

At this stage, the mobile node and home agent can send and receive encrypted and authenticated packets that contain binding updates and acknowledgments. To configure a security association for MPS and MPA messages, the same commands need to be done, with the exception that the protocol is ICMPv6 (as opposed to the mobility header). Note that this results in protecting **all** ICMPv6 messages between the mobile node and the home agent due to the inability to define higher granularity (to look for certain *types* of ICMPv6 messages) in the SPD (some, but not all, implementations may support a finer granularity). However, since there are no other likely ICMPv6 packets (except for ICMPv6 errors, which should be rare), the additional overhead of encrypting all ICMPv6 packets is minimal.

This configuration example clearly shows some of the reasons for processing the home address option and the routing header shown in Figures 5–5 and 5–6. The security association depends on the home address, which is also seen by applications as the source address of all packets. Therefore, IPsec needs to see this address in the source address field of the packet. If the security association were based on the care-of address, the mobile node would need to set up a new security association every time it moves, which is impossible with manual configuration and would cause unnecessary signaling when IKE is used. It would also cause additional delays before the mobile node can send an authenticated binding update.

5.3.4 Securing Binding Updates to Correspondent Nodes

Unlike the home agent, the correspondent node is not likely to have established any security relationship with a mobile node. A correspondent node can be a Web server, a news channel, or any random destination that the mobile node wishes to communicate with. This random node will not have the resources, or the desire, to keep track of mobile nodes' actions and report them if they misbehave. For instance, if a mobile node launches a flooding attack on another node through *www.bbc.co.uk* (e.g., downloading a large video file), we cannot expect *www.bbc.co.uk* to keep track of that or punish the mobile node for doing so. It is simply unrealistic to assume that correspondent nodes will police the mobile nodes' actions; it is much more realistic to assume that the protocol will prevent these attacks from taking place. Preventing the mobile node from stealing another node's traffic or directing traffic to a node other than itself can be done if the mobile node proves to correspondent nodes that it is authorized to redirect traffic from address A to address B; that is, the mobile node must prove that it **owns**[3] both addresses.

[3] We use the term "owns" loosely. Nodes do not own addresses; they are assigned addresses that can be changed at any time. Therefore, a host that "owns" an address at a particular moment in time is one that has rightfully configured that address on its interface, provided that the address is still valid.

5.3.4.1 OVERVIEW OF THE RETURN ROUTABILITY PROCEDURE

Suppose that the correspondent node sent a message to the mobile node's home address and another one to its care-of address; and suppose that each one of those messages required a particular reply that depends on the content of the message. If the mobile node answers both messages correctly, then we can assume that it **received** both messages. Since the messages were sent to the mobile node's claimed home address and care-of address, then answering them correctly implies that the mobile node actually received both messages and therefore owns both addresses. This is basically how a return routability test works in a Bad Guy–free Internet. However, since this ideal Internet does not exist, we need to somehow authenticate these messages. We now show how a return routability test works and how it can be used to secure binding updates to correspondent nodes.

A correspondent node generates a key, K_{cn}, which can be used with any mobile node. In addition, the correspondent node generates nonces at regular intervals. For instance, a correspondent node may generate K_{cn} when it boots, then generate a nonce every two minutes. The size of the nonce can vary depending on the correspondent node's implementation; however, the Mobile IPv6 specification recommends that nonce size be 64 bits. Nonces and K_{cn} will be used to generate a security association between the mobile and correspondent nodes. Nonces are stored for some time in an indexed list. Each nonce will have a 16-bit index. The return routability procedure is shown in Figure 5–7.

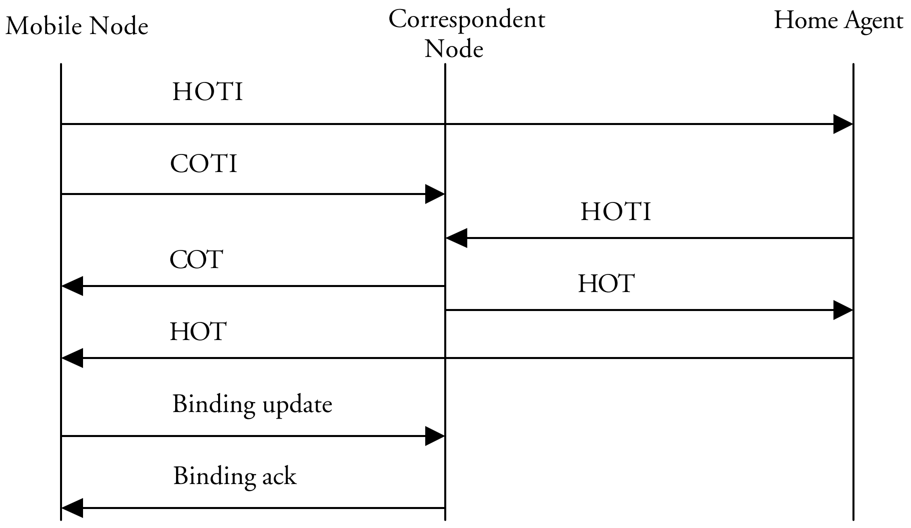


Figure 5–7 *Operation of the return routability procedure.*

The essence of the return routability procedure is that the mobile node requests that the correspondent node test its ownership of the home and care-of addresses. This is done by sending two independent messages: the HOme address Test Init (HOTI) and Care-Of address Test Init (COTI). The correspondent node creates two tokens that only the correspondent node can create and sends one token to each address (home and care-of addresses) in two separate messages: HOme Test (HOT) and Care-Of Test (COT). The mobile node uses both of these tokens to create a key (K_{bm}) that can be used to authenticate a binding update message to the correspondent node. Since the correspondent node knows all the information needed to produce the key, it can reproduce it when the binding update is received, and so authenticate the message. The same key is used to authenticate the binding acknowledgment.

The HOTI message is sent by the mobile node to request a test of the home address. The source address used in the IPv6 header is the mobile node's home address, and the destination is the correspondent node's address. Hence, this message has to be tunneled to the home agent (since the home address is not topologically correct in the visited network), which decapsulates the message and forwards it to the correspondent node. The HOTI message is transported inside a mobility header type 1. This message contains a cookie (called *home init cookie*) generated by the mobile node and later returned by the correspondent node. The cookie is a random number that has no significance; it is included to ensure that the entity responding to the HOTI message has actually received it. This message is protected on the mobile node–home agent path by ESP in *tunnel* mode. When sent by the mobile node, the HOTI message has the following format:

```
IPv6 header
   src = care-of address
   dst = home agent
ESP header
IPv6 header
   src = home address
   dst = correspondent node
Mobility Header type 1
   Home init cookie
```

The home agent verifies the ESP header and forwards the internal message to the correspondent node. Note that in this case the home agent is not provided with a home address option in the outer header (unlike the binding update message) to use in order to locate the right security association in the SAD. In this scenario, the home agent's SPD is configured to treat the mobile node's care-of address as a security gateway address. The implication of this configuration is that the home agent can associate a security association entry in the SAD with a specific tunnel interface, identified by the mobile node's care-of address. Hence, the home agent will be able to identify the security association based on the interface it was received from. Why is this message (and the HOT message) treated differently by not including the home address option? The reason is that the binding update is sent before establishing the tunnel; therefore, no tunnel interface can be used to identify the security association.

Almost simultaneously, the mobile node can send a COTI message. The COTI message is sent from the mobile node's care-of address directly to the correspondent node. It is transported in a mobility header type 2. The message contains another random cookie (called *care-of init cookie*). The COTI cookie is a random number used to ensure that the responder to a COTI message has actually received the original (COTI) message. The content of the message is shown below:

```
IPv6 header
   src = care-of address
   dst = correspondent node
Mobility Header type 2
   Care-of init cookie
```

When the correspondent node receives the HOTI message, it generates a 64-bit *home keygen* token (the token generated based on the home address). The home keygen token is generated by taking the first 64 bits of the output of a message authentication code function using K_{cn} and computed on the concatenation of the home address and a nonce generated by the correspondent node as follows:

```
Home keygen token = First(64, HMAC_SHA1(Kcn, home
address|nonce|0))
```

where `First(n, j)` represents the first `n` bits in `j`. `HMAC_SHA1(K, info)` means a hashed message authentication code (or a keyed hash) based on the SHA1 hash algorithm and uses `K` to key the function, which operates on `info`. The 0 is used to distinguish the home keygen token from the care-of keygen token, shown later. Hash functions and MACs were explained earlier in Chapter 4.

The correspondent node then constructs a HOT message and sends it to the mobile node. This message contains the home init cookie originally sent by the mobile node and the home keygen token. Since the correspondent node generates nonces frequently, it needs to be aware of the nonce used to generate a particular cookie. Nonces are stored in an indexed list. Therefore, a correspondent node only needs to know the index corresponding to a particular nonce to be able to generate the *home keygen token* again. The *nonce index* is included in the HOT message. This will be needed later by the correspondent node to authenticate the binding update. The content of the HOT message is

```
IPv6 header
   src = correspondent node
   dst = home address
```

```
Mobility Header type 3
   Home nonce index
   Home init cookie
   Home keygen token
```

The message will be intercepted by the home agent and tunneled to the mobile node's care-of address. A secure tunnel (ESP) is used to forward this message to the mobile node.

A similar operation is done when the correspondent node receives the COTI message. It generates a care-of keygen token, where

Care-of keygen token = First(64, MAC (K_{cn}, care-of address | nonce|1))

The nonce used in this operation might not be the same nonce used to create a *home keygen token*, depending on when the COTI message was received (the correspondent node might have generated a new nonce). Therefore, the nonce index should be sent to the mobile node in the COT message. The COT message is formed as follows:

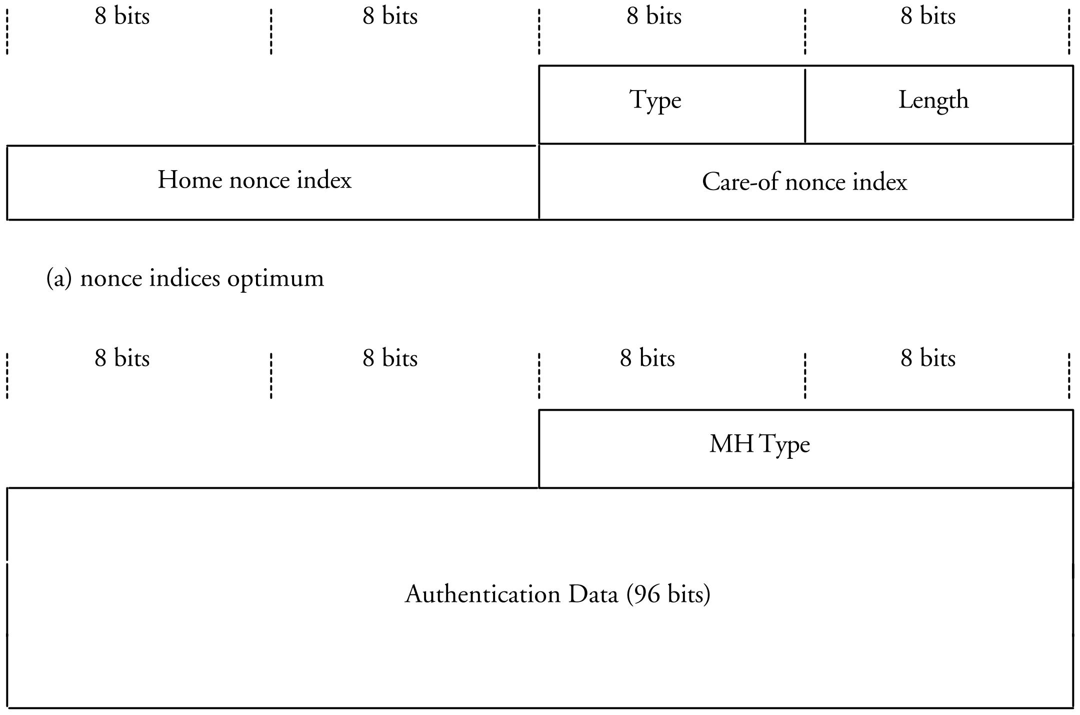
```
IPv6 header
   src = correspondent node
   dst = care-of address
Mobility Header type 4
   Care-of nonce index
   Care-of init cookie
      Care-of keygen token
```

This message concludes the return routability procedure. At this point, the correspondent node has not yet stored any more information than it had at the beginning of this procedure: K_{cn} and an indexed list of nonces. The correspondent node stores neither the home keygen token nor the care-of keygen token. When needed, these tokens can be regenerated, given the nonce indices originally used to generate them.

After receiving the HOT (tunneled from the home agent) and COT messages, the mobile node is in a position to generate a *binding management key*, K_{bm}. This is done as follows:

K_{bm} = SHA1 (home keygen token|care-of keygen token)

The mobile node can now construct the mobility header used for the binding update message. The mobility header includes the binding update, a *nonce indices* option, and a *binding authorization data* option (shown in Figure 5–8). The nonce indices option contains the two indices received in the HOT and COT messages.



(a) nonce indices optimum

(b) Authorisation data option

Figure 5–8 *Nonce indices and binding authorization data options.*

The authentication data are calculated as follows:

```
Auth_data = First (96, MAC(K_bm, Mobility_data)
```

where

```
Mobility_data = care-of address|final dst|Mobility header data
```

The mobility header data includes the content of the mobility header with the exception of the authorization data option itself. The `final dst` is the packet's final destination, that is, the correspondent node's address. If the correspondent node were also a mobile node, a routing header type 2 (containing its home address) would be included in the packet. Since the routing header is processed before the mobility header, the `final dst` field should contain that correspondent node's home address.

Since the correspondent node does not keep state for any mobile nodes during the return routability procedure, the mobile node needs to include its home and care-of addresses in the binding update. The home address is included in a home address option (in a destination options extension header), which precedes the mobility header. If the care-of address were different from the packet's source address, it should be included in the alternate-care-of address option; otherwise, the packet's source address is assumed to be the care-of address. In any case, the care-of address should always be the one used in the source address field of the COTI message; otherwise, the wrong care-of keygen token will be used to generate K_{bm} when the binding update is received at the correspondent node.

After the binding update message is constructed, the mobile node sends it to the correspondent node with the following format:

```
IPv6 header
     src = care-of address
     dst = correspondent node
DST-options header
     Home address option
Mobility header type 5
     Binding update
     Nonce indices option
     [optional alternate-care-of address option]
     Authorization data option
```

Alternatively, if the correspondent node were also a mobile node, and it had a binding cache entry in the mobile node, the packet format would differ slightly:

```
IPv6 header
     src = care-of address
     dst = correspondent node
Routing header type 2
     Home address_cn
DST-options header
     Home address option
Mobility header type 5
     Binding update
     Nonce indices option
     [optional alternate-care-of address option]
     Authorization data option
```

where `Home address_cn` is the correspondent node's home address.

When the correspondent node receives the binding update, it looks into the nonce indices option and finds the corresponding nonces. The correspondent node will be able to regenerate K_{bm} as follows:

1. Generate home keygen token: `First(64, MAC(`K_{cn}`, home address|nonce|0))`. The home address is taken from the home address option.
2. Generate care-of keygen token: `First(64, MAC(`K_{cn}`, care-of address| nonce|1))`. The care-of address is taken from the alternate-care-of address option when present; otherwise, the source address is used.
3. Generate K_{bm}: `Hash(`*`home keygen token`*`|`*`care-of keygen token`*`)`.
4. Calculate `Auth_data`: `First(96, MAC(`K_{bm}`, Mobility_data)`.
5. If `Auth_data` is equal to the content of the binding authorization data option, accept the binding update.

If an acknowledgment is requested, the correspondent node must send a binding acknowledgment. The binding acknowledgment should also contain the binding authorization data option in Figure 5–8. The contents of the binding acknowledgment message are shown below:[4]

```
IPv6 header
   src: correspondent node
   dst: care-of address
Routing header type 2
   mobile node's home address
DST-options header
      Home address option (if the correspondent node were
      also a mobile node)
      Mobility header type 6
Binding acknowledgment
[optional binding refresh advice option]
Authorization data option
```

where

`Auth_data = First(96, MAC(`K_{bm}`, Mobility_data),`

and

`Mobility_data = care-of address|`*`CN_addr`*`|Mobility header data`

`CN_addr` is the correspondent node's address. The *binding refresh advice* option informs the mobile node about the time when a new binding update is needed.

[4] There are exceptions to this format, which are described in the following section.

5.3.4.2 REMOVING BINDING CACHE ENTRIES IN THE CORRESPONDENT NODE

The mobile node attempts to delete its binding cache entry in the correspondent node's binding cache when it moves to its home link to allow packets to be sent directly to its home address. In this case, the mobile node would not test a care-of address because it does not have one. Hence, only a home keygen token is needed; that is, only a HOTI/HOT exchange is needed.

When sending a binding update to remove a binding cache entry, the source address in the IPv6 header will contain the mobile node's home address, and the destination address will contain the correspondent node's address. K_{bm} is calculated as follows:

K_{bm} = SHA1 (Home keygen token)

The binding update message sent to the correspondent node will contain the following information:

```
IPv6 header
     src = home address
     dst = correspondent node
Routing header type 2 [optional if the correspondent
node were also mobile]
     Home address_cn
Mobility header type 5
     Binding update (lifetime set to zero)
     Nonce indices option
     Authorization data option
```

Note that no home address option or alternate-care-of address can be included in this message. In this scenario, the care-of nonce index is irrelevant, since K_{bm} was formed based on the home keygen token. Therefore, this nonce index is ignored by the correspondent node.

When the correspondent node accepts the binding update, it sends a binding acknowledgment (if requested) with the following format:

```
IPv6 header
   src: correspondent node
   dst: home address
DST-options header
   Home address option (if the correspondent node were also a
mobile node)
Mobility header type 6
   Binding acknowledgment
   Authorization data option
```

In this scenario, the routing header type 2 is not included in the binding acknowledgment. The authorization data is calculated based on K_{bm}, as shown in the previous section.

5.3.4.3 MAINTAINING NONCES AND K_{CN} CORRESPONDENT NODES

The correspondent node is the only node that can generate the home and care-of keygen tokens based on two secrets: K_{cn} and a nonce. K_{cn} is 160 bits, and the nonce may have an arbitrary size; however, 64 bits is the recommended size in the Mobile IPv6 specification. A 64-bit nonce is large enough to minimize the probability that it will be reproduced in the future (for the same mobile node). If K_{cn} and the nonce stay constant for a long period of time, both tokens will remain the same, allowing Bad Guy to steal a legitimate mobile node's traffic. For instance, suppose that Bad Guy snooped a HOT message sent to the mobile node's home address (if he happened to be on the path between the correspondent node and the mobile node's home agent). He can later move to another location, send a COTI message to the correspondent node, then use the snooped token with that received in the COT message to direct the mobile node's traffic to his new location (see section 5.1.1.1). However, if tokens change regularly (because the nonce or K_{cn} changes), this attack will eventually fail. Hence, it is wise to require the correspondent node to regularly generate a new nonce that can be used for token generation. It is also necessary that tokens have short lifetimes. When a mobile node moves, it can reuse the same home keygen token while it is valid (the care-of keygen token is clearly invalid because the address changed). After the token expires, the mobile node must run the return routability procedure again in order to send an authenticated binding update.

The Mobile IPv6 specification requires that a nonce have a maximum lifetime of 240 seconds. On the other hand, a token is valid for 210 seconds. Hence, a fast-moving mobile node may reuse its home keygen token to generate K_{bm} for 210 seconds, and only a new COTI/COT exchange would be needed every time the mobile node moves within the 210 second period following the first binding update.

Since the correspondent node needs to know the nonce used to create the home keygen token, it must "remember" nonces for the lifetime of the token. Note that the difference between the nonce and token lifetimes is 30 seconds. Hence, a correspondent node can generate a new nonce every 30 seconds and accept bindings for the last eight nonces. Using this mechanism, only eight nonces would need to be remembered by the correspondent node at any given time (assuming that it uses the latest nonce to generate tokens for any new HOT/COT messages). If the correspondent node updates K_{cn} regularly, it needs to know which K_{cn} was used with a particular nonce to generate a token. Each one of the eight nonces stored should point to a particular K_{cn}. To simplify matters, the correspondent node could update K_{cn} at the same time it creates a new nonce; this allows it to use the nonce index to identify a nonce and the corresponding K_{cn}.

If a correspondent node receives a binding update with nonce indices pointing to nonces that have expired, it should respond with a binding acknowledgment, including an appropriate error code in the status field. The following error codes can be used for this case:

136	Expired home nonce index
137	Expired care-of nonce index
138	Expired nonces (if both nonces expired)

Upon reception of one of these errors, the mobile node should restart the return routability procedure. Note that if one of these errors occurs, the correspondent node cannot authenticate the binding acknowledgment, as it would not share a valid K_{bm} with the mobile node. Therefore, when these error codes are used, the correspondent node should not include an authorization data option in the binding acknowledgment.

5.3.4.4 SECURING HOTI AND HOT BETWEEN THE MOBILE NODE AND THE HOME AGENT

The strength of return routability relies to a large extent on a secure tunnel between the mobile node and its home agent (this is analyzed in detail in sections 5.3.4.6 and 5.3.4.7). For HOTI/HOT messages, confidentiality is required, hence the need for protection using ESP. The HOTI/HOT messages are protected by ESP in tunnel mode. As we saw earlier, those messages do not require the addition of a home address option (when sent by the mobile node) or a routing header type 2 (when sent by the home agent). The SAD entry required for protecting both of these messages would point to a tunnel mode ESP protection, which includes the tunnel exit point. Hence, the IPsec implementation in the sending node would add the outer header used to send the packet. Similarly, when one of these messages is received, the receiver would look at the interface (tunnel interface) that it was received on, and based on this information, identify the right security association in the SAD.

How does a mobile node configure its SPD to ensure that outgoing HOTI packets are protected? The mobile node's SPD entry checks the source address in the packet (home address), then checks whether a mobility header is included in the packet. If both of these conditions are satisfied, then the packet is a HOTI (pardon the pun!). Fortunately, these conditions are sufficient regardless of the content of the destination address field (the correspondent node's address), which could not be preconfigured, as no one knows which correspondent nodes the mobile node might communicate with in the future.

When a home agent tunnels a HOT message to the mobile node, its SPD entry contains checks on both the destination address (mobile node's home address) and whether a mobility header is included. Again, these conditions are sufficient independently of the source address field (the correspondent node's address).

5.3.4.5 SECURING BINDING REFRESH REQUESTS

Binding refresh requests are used to request a binding update from the mobile node. However, the mobile node is not required to respond to this message with a binding update.

The only potential vulnerability that can be introduced with binding refresh requests is a resource-exhaustion DoS attack. Bad Guy could send many binding refresh requests to trick the mobile node into believing that a return routability test and a consequent binding update is required. If this is done too many times, it can use processing power and memory in the mobile node. However, this kind of attack is not considered to be serious for two main reasons:

- The mobile node is not obliged to start a return routability test or send a binding update when this message is received. As shown earlier, the binding update and acknowledgment messages synchronize the binding cache and binding update list in the correspondent and mobile node respectively. Therefore, the mobile node should know when a binding update needs to be sent to a correspondent node or a home agent.
- The same effect can be produced without sending a binding refresh request. Bad Guy can simply request a connection with the mobile node and send many packets initially to make the mobile node think that it should optimize its path to the correspondent. Therefore, this message does not add new vulnerabilities to the existing ones on the Internet.

5.3.4.6 WHY DOES RETURN ROUTABILITY WORK?

The return routability procedure implicitly assumes that the routing infrastructure is secure and trusted. Thus, it is appropriate to design a protocol to secure the binding update as long as it is no less secure than the underlying routing infrastructure. In other words, if a packet is sent to a particular destination, the routing system delivers it to that destination. If Bad Guy(s) compromise the routing infrastructure and manage to control one or more routers, several serious attacks can be launched independently of return routability procedures.

Another assumption made by return routability is that it is difficult for Bad Guy to be located on two different paths at the same time and receive both tokens needed to generate K_{bm}. This could happen if Bad Guy is sharing a link with the correspondent node; he would be able to see all of the return routability packets, construct a binding update message, send it to the correspondent node, and receive all of the correspondent node's traffic addressed to the mobile node. However, Bad Guy does not need to go through all this trouble to hijack the correspondent node's connections with the mobile node if he shares a link with the correspondent node; he can simply pretend to be a router by stealing the default router's link-layer address and sending a fake router advertisement to the correspondent node. Alternatively, he can send a Neighbor Discovery redirect message to the correspondent node requesting that all its traffic be sent to his link-layer address. Thus, an attacker sharing a link with the correspondent node can cause serious harm without Mobile IPv6; that is, Neighbor Discovery messages are the weakest link when an attacker is sharing a link with the correspondent node.

Since our main goal was to ensure that securing route optimization does not make things worse than they are in today's Internet, the above case can be ignored. However, it is worth noting that this type of attack will become significant as soon as a mechanism is devised to secure Neighbor Discovery messages. When this happens, the return routability procedure will become the weakest link.

Bad Guy can be located on the mobile node–correspondent node path. In this location, he would only be able to see the care-of keygen token, which would not allow him to construct K_{bm} correctly to steal the mobile node's traffic.

Bad Guy might send a large number of HOTI and COTI messages to try to consume the correspondent node's resources in a way that makes it unable to process legitimate requests from real mobile nodes. The return routability procedure is designed to allow correspondent nodes to be protected from memory-exhaustion attacks; a correspondent node would only keep state when it receives an authenticated binding update from a mobile node. Clearly, this procedure cannot protect against an attacker aiming at using up the correspondent node's link bandwidth by sending a very large number of HOTI/COTI messages. However, this attack can be launched without return routability by simply sending a large number of bogus messages. It is worth noting, though, that the correspondent node can simply decide to not receive any HOTI/COTI messages if it detects that it is being attacked. That is, the correspondent node can "turn off" route optimization; communication with mobile nodes will still take place through the home agent.

One of the most important advantages of the return routability procedure is that it does not require any manual configuration or infrastructure support. This feature assists with the quick deployment of Mobile IPv6 and encourages vendors to support route optimization, which would have been much harder if route optimization came with the burden of infrastructure support or the unrealistic assumption of manual configuration. However, it is important to note that this comes at the cost of having weak authentication compared to the more traditional applications of public key cryptography.

5.3.4.7 HOW SECURE IS RETURN ROUTABILITY?

When assessing the security level of the return routability procedure, we need to recall the most important requirement that we had to deal with: Do no harm to the current Internet. The current Internet is predominantly based on IPv4 but moving toward IPv6 in the near future. Hence, we need to understand the types of attacks that can be launched today in IPv4, attacks that can be launched in IPv6, and finally compare that situation with the one created by introducing Mobile IPv6.

In the current IPv4 Internet, Bad Guy can launch several attacks while located on the path between two communicating nodes (on-path includes the case where Bad Guy is sharing a link with one or both of the nodes). Bad Guy can modify data between the two nodes (MITM attack), impersonate one of the nodes (masquerading), or perform DoS, including connection termination (e.g., sending a TCP RST message; more on TCP in the next chapter). The same attacks can be launched in an IPv6 Internet, but we currently have mechanisms that can protect against these attacks. Authentication and message integrity can protect against masquerading. Finally, encryption can provide protection against eavesdropping. Off-path attacks can be classified into three categories:

- *DoS attacks:* Bad Guys can launch several types of DoS attacks. These include memory/processing capability exhaustion attacks. Bad Guy can also terminate ongoing connections, as discussed above.
- *Reflection attacks:* Reflection attacks can be performed if the attacker spoofs a victim's IP address; the receiver of a message with a spoofed IP address will respond to that address, causing the message to be sent to the victim. Ingress filtering can minimize the range of addresses that can be spoofed (i.e., to the addresses that can be derived from the prefixes on one link) and expose the location (link) of the attacker. However, it cannot stop this attack.
- *MITM attacks:* These attacks require the attacker to be able to remotely compromise one of the routers on-path to allow him to be **present** on-path. This is extremely difficult and practically impossible today. Alternatively, the attacker may corrupt the routing system to be able to divert traffic to his location. For instance, an attacker may inject routes to a particular destination, hence fooling other routers to believe that traffic to that location should be delivered to him. However, routing protocols are typically protected against such attacks by setting up security associations between routers to allow for authentication and message integrity.

Thus far, it seems that both IPv4 and IPv6 suffer the same types of vulnerabilities when considering on-path, on-link, and off-path attackers. The important question is, What does return routability add to these vulnerabilities? To answer this question, let's consider the attacker's possible locations (Figure 5–9).

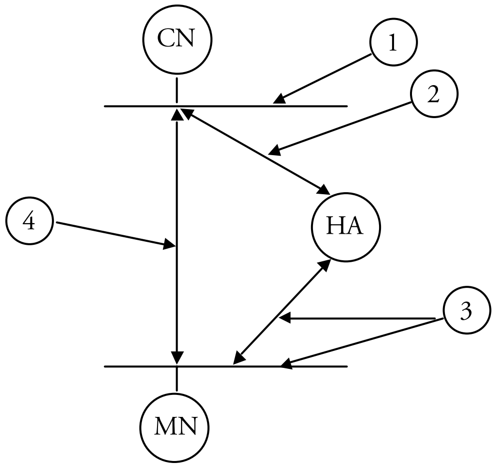


Figure 5–9 *Possible locations for Bad Guy.*

- *Location 1:* Here, Bad Guy is sharing a link either with the correspondent node or somewhere else where path 2 intersects path 4. We've already shown that when Bad Guy is sharing a link with the correspondent node, several attacks can be launched without using Mobile IPv6 or return routability. However, things can be made worse with return routability. Bad Guy can pick one or more home addresses for one or more mobile nodes. He can send both HOTI and COTI messages to the correspondent node. Since he can see both HOT and COT messages (assuming a shared link, e.g., Wireless LAN), he can construct K_{bm} and send a binding update for those home addresses. This would stop the legitimate mobile nodes that own those addresses from communicating with the correspondent node for the duration of the binding. The same attack can be launched against a mobile node that is already communicating with the correspondent node. The difference from using Neighbor Discovery is that Bad Guy can move to another location (off-path), and continue to deny service between the mobile and correspondent node by refreshing the existing binding. If Bad Guy were somewhere else where paths 2 and 4 intersect, he can essentially launch off-path attacks, provided that ingress filtering allows him to generate a packet containing the mobile node's home address (for HOTI). In addition, Bad Guy would need to be able to snoop the HOT message containing the home keygen token.

- *Location 2:* Bad Guy is somewhere on the path between the correspondent node and the home agent. He can see the HOT message, but cannot see the COT message, and therefore cannot claim to own the mobile node's care-of address. However, he can claim to own the mobile node's home address. He can initiate a return routability procedure, intercept the HOT message (or simply snoop it), and use his current address as a care-of address; that is, the COT message will be sent to his current address. Similar to attacks from location 1, the attacker can then move to another location, keep refreshing the binding, and consequently deny service between the mobile and correspondent nodes. The fact that the attacker can move to another location, off-path, and continue to deny communication between the mobile and correspondent nodes makes this attack different from a typical MITM attack that can take place today.
- *Location 3:* We have combined the two locations (on the mobile node's link and on-path between the mobile node and home agent) for simplicity, since they have the same implications, with one exception described below. Basically, this location is different from 1 and 2 because we have a secure channel between the mobile node and home agent, which is used to encrypt both the HOTI and HOT messages. Hence, if Bad Guy is in this location, there is not much he can do that cannot be done without Mobile IPv6. However, Bad Guy can in fact be a malicious mobile node, which has a real home address. In this case, he can use his valid home address and a care-of address that belongs to another node on the visited link. He would be able to receive the HOT message securely and snoop the COT message (on a shared link). Hence, K_{bm} can be generated, and Bad Guy can divert his traffic to another node on the link. That is, Bad Guy can bomb a neighbor.
- *Location 4:* In this location, Bad Guy can launch the same attacks as can be launched without Mobile IPv6 with MITM cases. He cannot see the HOTI/HOT messages and therefore cannot construct K_{bm}. Hence, no new vulnerabilities are added by return routability.

An interesting observation is that attacks from locations 1 and 2 can be launched even on encrypted/authenticated traffic (e.g., using ESP or AH). Recall that the identifiers used to authenticate a binding update are completely different from those used by a client to authenticate traffic for her Internet banking or to run an IPsec tunnel to her corporate network. Return routability and the resulting binding update redirect traffic from one location to another. Whether that traffic is encrypted or not is irrelevant to the binding update. This is a result of the requirements placed on identifiers used to authorize the binding update (home and care-of addresses ownership), which have nothing to do with the real identity of a user or a machine that might have been used to set up a security association between the mobile and correspondent nodes.

From the above analysis we can see that the vulnerabilities of return routability are associated with an attacker being located on the correspondent node's link, somewhere where he can see both HOT and COT, or on the path between the correspondent node and the home agent. In addition, there are two main differences between these attacks and today's attacks on the Internet:

- The attacker can disallow future connections (i.e., ones that do not exist yet) between a correspondent node and any other mobile node by picking a home address and performing a return routability procedure.
- The attacker can move to another location and refresh the binding in the correspondent node to continue to deny service to the mobile node, even while located off-path.

So what can we do to stop these attacks or minimize their impact? Let's consider these vulnerabilities and how to minimize their impact. Vulnerabilities related to the first scenario are addressed in sections 5.3.4.8 and 5.4.

Vulnerabilities related to the second scenario can be addressed using two different approaches:

1. Conservative approach: Every time a mobile node moves, it must run the entire return routability procedure. This will ensure that Bad Guy cannot move somewhere off-path and continue to refresh the correspondent node's binding cache based on the previous home keygen token.
2. Moderate approach: Set a lifetime for the home and care-of keygen tokens. A mobile node will only be able to refresh a binding while these tokens are still valid. After that, the mobile node must run the entire return routability test. This approach will limit the time during which the attacker is refreshing bindings while located off-path. That is, if Bad Guy was in location 1 (or 2), then moved off-path, he would be able to refresh the binding (and hence deny future communication between the mobile and correspondent nodes) for the lifetime of the home keygen token (he would only need to send a COTI message from his new location).

The conservative approach has the advantage of eliminating the vulnerability in question, but requires sending HOTI/COTI and receiving HOT and COT every time the mobile node moves. During this period, the mobile node cannot communicate with the correspondent node directly; outbound packets (sent from the mobile node) would need to be routed via the home agent, but packets sent from the correspondent node are routed to the mobile node's previous location and effectively lost. Therefore, we can assume that communication is impossible for the duration of the return routability procedure. The moderate approach has the advantage of sending a single message (COTI) while the home keygen token is still valid. While the overall disruption time may not vary significantly (since HOTI and COTI are sent at the same time), the moderate approach causes fewer packets to be sent by the mobile node. The Mobile IPv6 specification chose the moderate approach over the conservative one. This allows Bad Guy to continue to refresh the binding for the lifetime of the home keygen token (240 seconds). Moreover, Mobile IPv6 restricts the overall lifetime of a binding cache entry created using the return routability procedure to 7 minutes. After 7 minutes, the mobile node must perform the entire procedure to be able to send a new binding update. This must be done regardless of the mobile node's location—even if it never moved since the last binding update.

A new mobility protocol is bound to have new threats associated with it; hence, it is pertinent to understand these threats and decide which ones must be avoided and which ones can be "lived with." It is certainly likely that new threats will also emerge when the protocol is deployed. Therefore, stronger security mechanisms might be required in the future to eliminate some of these threats. Some of the efforts to develop more secure mechanisms are discussed in the next section and in section 5.4.

5.3.4.8 SECURING BINDING UPDATES USING PUBLIC KEYS AND CERTIFICATES

In Chapter 4 we saw how Alice and Bob used public keys and certificates to authenticate each other and encrypt traffic. These mechanisms can also be used to authenticate binding updates between mobile and correspondent nodes. In fact, earlier revisions of Mobile IPv6 assumed that public key cryptography would be used between mobile and correspondent nodes to authenticate binding updates. However, the earlier approach was not clear on two aspects:

1. How does the mobile node prove ownership of the home address?
2. How does the mobile node prove ownership of the care-of address?

Public key cryptography typically requires a key exchange mechanism (e.g., IKE) and a method of binding the public key of the correspondent to the identity that it claims. The identity needed to answer the first question above is the home address. The home address can be included in a certificate. Two possible options exist:

1. Include the home address in the *subject* field, or
2. Include the home address in the *subjectUniqueIdentifier* field.

Including the home address in the latter field is probably a better option, as it allows the subject field to be used for other cases where a different identifier may be needed (i.e., where current applications normally expect the identifier). Hence, a correspondent node receiving a mobile node's certificate would be able to prove (given the CA's public key) that the mobile node owns the home address.

Answering the second question would involve a reachability test for the care-of address. That is, if the correspondent node can send a message to the care-of address and receive a correct reply (authenticated by the same node that claims to own the home address), then the correspondent can authorize the mobile node to direct traffic from its home address to its current care-of address. Once a security association is established, the mobile node could reuse that security association to send future binding updates. However, the mobile node still needs to prove its ownership of the care-of address every time it moves. This is needed to prevent the mobile node from using a fake care-of address or another node's care-of address (flooding attack). An example of a protocol that can be used to solve this problem is shown in Figure 5–10.

When using this protocol, the mobile node uses its care-of address as a source address in IP packets containing the IKE messages. In phase 2, the mobile node negotiates the security association that will be used to protect the binding update. IKE allows the mobile node to specify an address other than the source address to be used to identify the security association (recall Chapter 4, Figure 4–10). The mobile node can use its home address for this purpose. However, the correspondent node would need to ensure that the mobile node is authorized to use such address. This can be done by reading the subjectUniqueIdentifier field in the mobile node's certificate and making sure

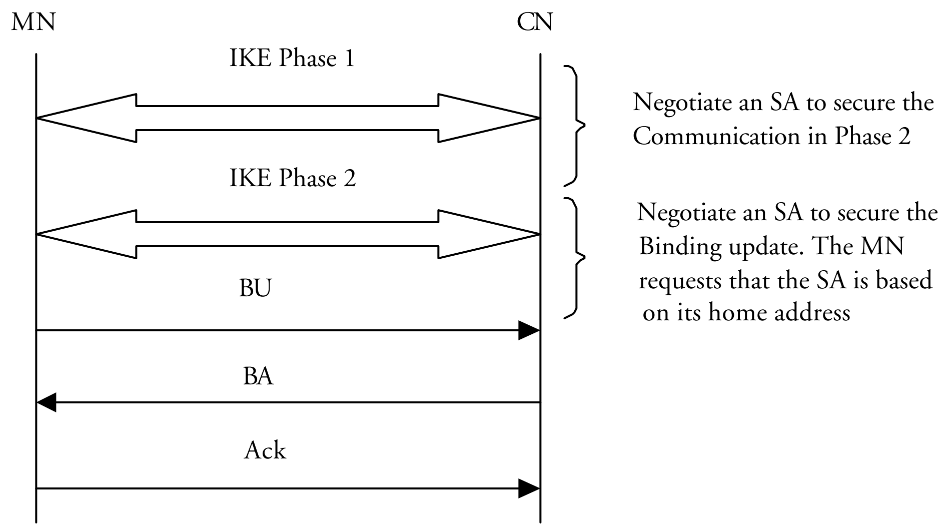

MN
CN
IKE Phase 1
Negotiate an SA to secure the Communication in Phase 2
IKE Phase 2
Negotiate an SA to secure the Binding update. The MN requests that the SA is based on its home address
BU
BA
Ack

Figure 5–10 *Authenticating binding updates using IKE and IPsec headers (ESP/AH).*

that it includes the mobile node home address. If it does, the security association is established. At this stage, the mobile node can send a binding update to the correspondent node. The correspondent node acknowledges the binding update by sending an encrypted (with ESP) binding acknowledgment to the mobile node's care-of address. The binding acknowledgment should contain a sequence number that matches the one included in the binding update. In addition, the binding acknowledgment should contain a nonce (e.g., 64 bits). So far, the mobile node has proven that it owns the home address, but not necessarily the care-of address. For example, the mobile node could have used its real care-of address during the IKE exchange but sent the binding update with a fake one. If the mobile node replies with a message containing the same nonce sent by the correspondent node (the new ACK message), then the binding can be accepted.

We must note that this scheme has the same vulnerability associated with location 3 in the return routability case (or, more accurately, a subset of location 3; that is, if Bad Guy is sharing a link with the victim). In this case, Bad Guy can actually be the mobile node itself, which is trying to bomb a victim, sharing a link with it. The mobile node would be able to successfully set up a security association with the correspondent node. It can also use a care-of address that belongs to another node on the same link (on a shared link). The mobile node would be able to see the binding acknowledgment, decrypt it, and read the nonce (since it knows the key). The mobile node can then send the correct reply in the ACK message.

5.3.5 Preventing Attacks Using Home Address Options and Routing Headers

In section 5.1.2, we saw how Bad Guy can launch reflection attacks using the home address option and the routing header. In Chapter 2, we saw how the routing header can be abused by Bad Guy to gain access to nodes that he is not authorized to gain access to. In Mobile IPv6, restrictions were placed on the home address option and the routing header to prevent these types of attacks. Both the home address option and the routing header (type 2) can be used only **after** the correspondent node has accepted a binding update from the mobile node. When receiving a packet containing a home address option, the correspondent node checks its binding cache to see if a binding exists between the home address (included in the option) and the care-of address included in the source address of the packet. If such entry does not exist, the mobile node sends a *binding error* message to the source address in the packet. This stops Bad Guys from using the home address option without proving their ownership of both the home address and the care-of address.

To prevent attacks against the routing header, Mobile IPv6 requires that two rules be satisfied in the mobile node in order to accept a packet containing a routing header:

1. The routing header must contain one address only.
2. The address in the routing header must be one of the mobile node's home addresses.

These rules can certainly be satisfied when using the existing routing header (with type 0). However, other concerns were also brought up during the design of Mobile IPv6. Network administrators might get concerned about attacks like the one shown in Chapter 2. After all, a firewall has no way of knowing if the address contained in the routing header belongs to the same node receiving the packet or to another node. Hence, network administrators might configure their firewalls to drop any packet containing a routing header. This would effectively prohibit the use of Mobile IPv6 (recall that binding acknowledgments from the home agent contain a routing header). To avoid the routing header (type 0) overload, which might force network administrators to throw away the baby with the bath water, Mobile IPv6 uses a new routing header (type 2). This allows firewalls to distinguish between a routing header type 0 and Mobile IPv6's routing headers, which are restricted to particular use and policed by mobile nodes. That is, a node would process routing header type 2 only if it were a mobile node. As a consequence, it must know the rules of processing such header. Therefore, it will not relay the packet to any other node.

5.4 Future Mechanisms for Authenticating Binding Updates

In Chapter 4, we saw how a node can generate a cryptographically generated address by generating a public/private key pair, then using the public key to produce *h*(PK, *something*). The hash function will produce 62 bits that can be used (in addition to the u- and g-bits) to generate the interface identifier. If *something* is a number that includes the prefix on a particular link, then the CGA produced can prove that a node owns an address on a particular link. For instance, the mobile node might have a cryptographically generated home address as follows:

```
64-bit Interface_identifier = First(64, (H(home_prefix|PK|
context) & 0xfcffffffffffffff
```

where `context` is a number associated with the context in which CGAs are used (e.g., set to 1 for Mobile IPv6). The `home_prefix` is the mobile node's home prefix. The `& 0xfcffffffffffffff` indicates a bitwise AND operation used to set the u- and g-bits to their correct values (zero, because the interface identifier cannot be guaranteed to be globally unique and is not part of a group).

The mobile node can prove that it owns such an address and authenticate messages sent from this address by signing the message with its private key and including its public key inside the message. The signature is calculated as follows:

```
SIG = SIGALG (H(m), SK)
```

where `SIGALG` is a signature algorithm, `H(m)` is a hash of the message being authenticated, and `SK` is the mobile node's private key (we use **S** for private key, since **P** was already used for the public key). Since the message includes the mobile node's public key, the receiver of such message is able to verify two things:

1. The sender knows the public/private key pair.
2. The sender owns the address used in the source address.

The first can be verified by verifying the signature sent in the message. The second can be verified because a change in address would show that the packet has been modified (i.e., assuming that the receiver knows **context** and can recalculate the interface identifier).

Before we consider how this mechanism can be used to secure binding updates, we need to understand an important limitation: CGAs prove that a node owns a particular address, but do not prove that this node is currently located at that address. A node may have formed this address based on a fictitious prefix or while being on that link, but it is no longer there. Hence, there is no guarantee that the node owning a CGA is actually located on that link at a particular moment in time. To prove this, we need to test that address to see if it is "live." In other words, we still need a return routability test.

5.4.1 Alternative 1: Using a Cryptographically Generated Home Address

Consider the case where the mobile node has a cryptographically generated home address. Such home address can be used when performing the return routability procedure to provide stronger authentication. Note that we still need to perform return routability to make sure that the mobile node is actually located where it claims to be. Figure 5–11 shows the return routability procedure.

In this protocol, the mobile node sends the HOTI and COTI messages as described earlier. After receiving the HOT and COT messages, the mobile node

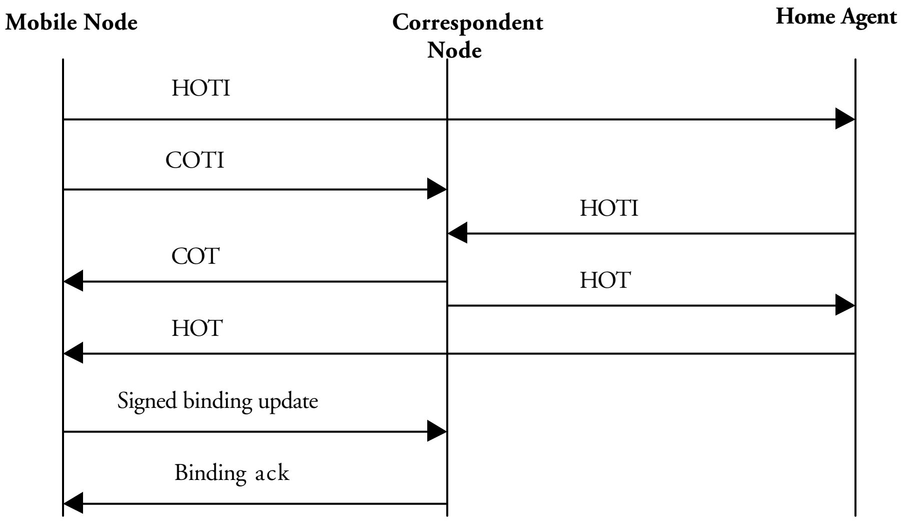


Figure 5–11 *Return routability with cryptographically generated home addresses.*

can form K_{bm} and authenticate the binding update. In addition, the mobile node can sign the binding update with its private key, as shown earlier. This assures the correspondent node that the mobile node owns that home address.

The advantage of this addition to return routability is that it protects from Bad Guys on the path between the correspondent node and the home agent (locations 1 and 2 in Figure 5–9). Bad Guy can no longer claim to own the mobile node's address without generating the corresponding key pair, which is extremely difficult. Thus, Bad Guy cannot steal the mobile node's traffic or prevent future communication between the correspondent and mobile nodes. Bad Guy cannot hijack a connection either, since he cannot sign the binding update message with the right private key. Furthermore, due to the strong cryptographic nature of this scheme, the mobile node could be allowed to generate a security association that exempts it from running the home address test. This would allow for a much longer lifetime for the home keygen token.

A disadvantage of this mechanism is that it is computationally intensive (to generate the public/private key pair). However, this might not be a major obstacle to deploying CGAs, since mobile nodes will not need to generate these keys very often. The strong cryptographic property of CGAs allows the mobile node to use the same key pair for a long period of time. A more significant disadvantage of this approach is that verifying the mobile node's signature could be computationally intensive for the correspondent node (depending on the signature algorithm used). If the correspondent node is a battery-powered wireless device, this could be a concern. Finally, public keys are very large; on a bandwidth-challenged link, sending these packets can block the link for some time and cause some jitter for ongoing sessions. In Chapter 7, we show how binding updates can be reduced for fast moving mobile nodes.

5.4.2 Alternative 2: Using Cryptographically Generated Home and Care-of Addresses

CGAs are useful for any application in which a node is required to prove that it owns a particular address. One of these applications (discussed below) is securing Neighbor Discovery messages. A mobile node might wish to ensure that no other node on a visited link will steal its address and might consequently produce a cryptographically generated care-of address. This is useful in providing more security to the binding update. In this case, the messages used are identical to the procedure shown in Figure 5–11, with the addition that the mobile node would use a cryptographically generated care-of address. Hence, the mobile node's signature will prove that the mobile node owns both the home and care-of addresses (both generated from the same public key but with different prefixes).

This scenario is useful because it stops a malicious node from bombing another neighbor on-link, since only the mobile node will have the right key pair to generate both home and care-of addresses. When using this approach, all known vulnerabilities of the return routability procedure can be addressed.

5.4.3 Other Improvements Gained from CGAs

As shown earlier, CGAs are useful for securing Neighbor Discovery messages like the following:

- *Duplicate Address Detection* (DAD): In this case, a node can prove that it owns a particular address. The DAD message can be signed by the node generating the address. If another node claims the same address (very unlikely), it must sign its Neighbor Advertisements (NAs) to prove that it generated the same key pair. This technique prevents DoS attacks that can be launched by Bad Guy to prevent a node from configuring an address by claiming that it owns that address (i.e., replying to all DAD messages with corresponding NAs).
- *Neighbor Solicitation* (NS): The Neighbor Discovery specification allows nodes to include their own link-layer addresses in an NS. This optimization saves the responding node from sending another NS to the soliciting node. If a node signs the NS with its private key and uses a CGA, the receiver would be sure that such node owns the address that it claims to own.
- *Neighbor Advertisements* (NAs): NAs can also be protected if the sending node signs the message with its private key and uses a

CGA as a source address. Since these messages bind a node's IP address to its link-layer address (not covered by IP layer security), somehow the receiver of this message will need to check (possibly from its own link layer) that the message was in fact received from the right link-layer address.

- *Router Advertisements* (RAs): This message is a bit different from the others. A router can certainly use a CGA for its source address to allow nodes to verify that the same router is sending the same advertisement. However, this would not allow other nodes to know that this address (the router's) is in fact authorized to send a router advertisement. For instance, Bad Guy could generate a CGA and a prefix (may or may not be valid, depending on the intention of the attack) and start sending RAs on the link. This could fool other nodes into believing that Bad Guy is actually a router and start sending their traffic to him. This scenario illustrates, once more, the value of authorization. Bad Guy can be authenticated using CGAs, but such authentication does not authorize Bad Guy to be a router.
- *Redirect Messages:* A redirect message instructs the sender of a particular message to send future messages to a different node; such node may be the ultimate destination of those messages (a neighbor on a link) or a router, which has a better route to the ultimate destination. If not authenticated, these messages can be used to bomb other neighbors. Bad Guy could instruct node A to use node C as a default route to send packets addressed to node B. However, node C is not a router; it is another host on the link. In this case, node A needs to ensure that a redirect message actually came from a router authorized to send this message. If the RA authorization problem is solved (e.g., using certificates that are signed by the authority delegating the prefix to the router), a node would only need to authenticate a redirect message provided that it knows that the source address (a CGA) of this message is authorized to be a router.

The SEcure Neighbor Discovery (SEND) Working Group in IETF was chartered in October 2002 to solve the problem of securing Neighbor Discovery messages. In [2] and [5], the different threats associated with Neighbor Discovery messages and ways of addressing them using CGAs are discussed.

5.5 Summary

This chapter started by justifying the need for Mobile IPv6 security. Mobile IPv6 messages were analyzed, and a brief threat analysis was presented to show the impact of an insecure mobility management protocol like Mobile IPv6. Next, we discussed the main requirements used to design the security solution for Mobile IPv6.

We presented Mobile IPv6 security in the context of two different security relationships:

1. mobile node–home agent relation
2. mobile node–correspondent node relation

Mobile IPv6 signals to the home agent are protected with IPsec (ESP), and an example showing manual configuration of security associations was shown. Route optimization signaling to the correspondent node is done using the return routability procedure. This procedure was discussed in detail, emphasizing its strengths and vulnerabilities.

Finally, we showed different alternatives for securing Mobile IPv6 with public keys and certificates, or with CGAs. The use of CGAs for other applications was also discussed.

This chapter concludes our look at Mobile IPv6's operation. The next part of this book looks into the performance of Mobile IPv6 handovers and different optimizations that can be used to enhance it. This be shown in the context of wireless IP networks.

Further Reading

[1] Arkko, J., P. Nikander, G. Montenegro, E. Nordmark, and T. Aura, "Residual threats," note from the design team to the Mobile IP mailing list, 2002.

[2] Arkko, J., P. Nikander, and V. Mantyla, "Securing Neighbor Discovery Using Cryptographically Generated Addresses (CGAs)," draft-arkko-send-cga-00, work in progress, June 2002.

[3] Arkko, J., V. Devarapalli, and F. Dupont, "Using IPsec to Protect Mobile IPv6 Signaling between Mobile Nodes and Home Agents," draft-ietf-mobileip-mipv6-ha-ipsec-03, work in progress, February 2003.

[4] Aura, T., and J. Arkko, "MIPv6 BU Attacks and Defenses," draft-aura-mipv6-bu-attacks-01, work in progress, February 2002.

[5] Kempf, J., and E. Nordmark, "Threat Analysis for IPv6 Public Multi-Access Links," draft-kempf-ipng-netaccess-threats-00, work in progress, October 2001.

[6] Montenegro, G., and C. Castelluccia, "SUCV Identifiers and Addresses," draft-montenegro-sucv-02, work in progress, November 2001.

[7] Mankin, A., et al., "Threat Models Introduced by Mobile IPv6 and Requirements for Security in Mobile IPv6," draft-ietf-mobileip-mipv6-scrty-reqts-02, work in progress, May 2001.

[8] Nikander, P., E. Nordmark, G. Montenegro, and J. Arkko, “Mobile IPv6 Security Design Rationale,” draft-nikander-mipv6-design-rationale-00, work in progress, February 2003.

[9] Nikander, P., “Address Ownership Problem in IPv6,” draft-nikander-ipng-address-ownership-00, work in progress, February 2001.

[10] Nordmark, E., “Securing MIPv6 BUs Using Return Routability (BU3WAY),” draft-nordmark-mobileip-bu3way-00, work in progress, November 2001.

[11] O’Shea, G., and M. Roe, “Child-proof Authentication for MIPv6 (CAM),” *Computer Communications Review*, April 2001.

[12] Roe, M., T. Aura, G. O’Shea, and J. Arkko, “Authentication of Mobile IPv6 Binding Updates and Acknowledgments,” draft-roe-mobileip-updateauth-02, work in progress, March 2002.

[13] Savola, P., “Security for IPv6 Routing Header and Home Address Option,” draft-savola-ipv6-rh-ha-security-03, work in progress, December 2002.

[14] Thomas, M., “Binding Updates Security,” draft-thomas-mobileip-bu-sec-00, work in progress, November 2001.