P8

Publication P8

Jukka Ylitalo, and Pekka Nikander, “A new Name Space for End-Points: Implementing secure Mobility and Multi-homing across the two versions of IP”, in Proc. of the Fifth European Wireless Conference, Mobile and Wireless Systems beyond 3G, pp. 435-441, Barcelona, Spain, February 24-27, 2004, ISBN 84-7653-846-4, Publisher: SCI UPC (Eds: Olga Casals, Jorge Carcia-Vidal, Jose M Barcelo, and Llorenc Cerda).

A new Name Space for End-Points: Implementing secure Mobility and Multi-homing across the two versions of IP

Jukka Ylitalo, Pekka Nikander

Ericsson Research Nomadiclab
e-mail: {jukka.ylitalo, pekka.nikander}@ericsson.com

***Abstract*: The current practice of naming Internet nodes with their IP address has turned out to be insufficient. We propose adding a new name space to the IP stack, and using cryptographic public keys as end-point names. It turns out that this allows us to provide end-node mobility and multi-homing, even between IPv4 and IPv6, in a fairly simple, secure, and elegant way.**

1. Introduction

In the early days of TCP/IP, computers had only one network interface and it was impossible to move them around without first turning the power off. Hence, any computer could be easily identified with its one and only Internet address (if it had any). In other words, the *location identified the node* in the Internet. As a consequence, it was a relatively easy design choice to identify transport layer connections with IP addresses.

Since then, the environment has changed, but the basic Internet architecture has not. Computers are being moved around without first turning them off, and more and more computers have more than one network interface, i.e., computers have become mobile and multi-homed. In contrast to this, the structure of the Internet is still very much the same as if a person's name would be defined by his or her current location. Let's say that a businessman moves from Oxford Street to Elm Street. Since identification is bound to locations, his *identifier* changes due to movements, but his actual *identity* stays the same. His old acquintances, who were used to knowing him as Mr. Ten Oxford Street, do not recognize him anymore. Now being identified as Mr. Twelve Elm Street, he must convince people, in one way or another, about his actual identity, that he still is the same person as he was before. Since there is no equivalent of the human face in the current Internet, convincing other people is not a particularly easy task.

Basically, a similar kind of historical load bothers also the existing Internet protocols. In Mobile IP [14][6], the problem has been solved by using so called *home addresses*[1]. Each node is assigned a static address, its home address, which is used to identify the node independent of its location. This solves the basic naming problem; our businessman is still known to his friends as Mr. Ten Oxford Street. However, even in the real world, the actual location is needed for reachability. It would be really hard for our businessman, usually living in Oxford Street and currently walking at 12 Elm Street, to prove to foreigners that his name is Mr. Ten Oxford Street c/o Twelve Elm Street, without credentials. Thus, using a static identifier, as such, does not solve the problem of convincing other people about the necessary name change caused by mobility. Firstly, most of the people in the world do not know the people living at Oxford Street. Secondly, people who are not at that specific moment at Elm Street cannot trust Mr. Ten Oxford Street's word that he really is walking at Elm Street without evidence.

The first problem (knowing who "lives" where) is called the IP address ownership problem [10]. Basically, it is very hard to prove, in any sense, that you "own" a particular IP address. For as far as your peer node knows, you might be an impostor that sends packets with a spoofed source address. The second problem (needing c/o addresses), in turn, is created by the Mobile IP requirement of being able to re-direct any node's traffic flow to anywhere. That is, since Mobile IP is implemented by re-directing traffic destined to a home address to another address (*care-of-address*), universal deployment of Mobile IP requires that any address can be dynamically re-directed to any other address. This creates a group of potential masquarade and denial-of-service attacks [12]. If there were no countermeasures, any malicious node would be able to spoof Mobile IP management messages, claiming to "own" another node's home address, and redirecting all traffic destined to that address to somewhere else. [12]

Instead of relying on the Mobile IP ideas [2], we have taken a fresh start and experimented with the Host Identity Protocol (HIP) [8][13], a proposal to enhance the current Internet architecture by introducing a new name space, *Host Identity* name space, between the internetworking and transport layers. Continuing our analogy, the approach would bring our businessman a genuine name, one cryptographically bound to his real identity. In a way, our Mr. Ten Oxford Street would no longer be named after his location, but by a self-signed photograph of his face.

Before considering why, in our opinion, such a new name space is essential for the future of the Internet, and how it solves the security problems related to node mobility and multi-homing in an elegant and almost effortless way, we have to consider a little bit more background.

2. Binding the Name to the Identity

The mobility security problems, described above, arise when a name is not directly bound to the identity of the communicating node. In the digital world, such a binding could be created with a little help of cryptography and/or by relying on an external, trusted infrastructure. However, in the current Internet neither of these are used particularly often. Instead, most of the Internet

[1] In a way, the Mobile IP naming convention resembles human naming conventions in the medieval times, when people were named after their home town, e.g., William of Ockham.

protocols are still relying on a simpler assumption, the integrity of the routing system.

The Internet is based on stateless routers, collectively maintaining a topological map of the network. That is, given an IP address, the routers know how to pass packets to the given address. In practice, if you send a packet with a given destination IP address, you can be fairly confident that the packet either reaches the location named by that address (and the node at the location, if there is one), or the packet is dropped on the way. Thus, the current situation where the nodes are named by their addresses creates a weak (but sufficient) form of security: by sending a packet to a given address and waiting for a response, one can check if there is a node named with the given address. In the Mobile IPv6 terminology, this is called Return Routability (RR) [12].

As we saw above, mobility breaks this security property. In Mobile IP, with the mapping between home addresses and care-of-addresses, it is not sufficient to simply rely on the routing system. The same applies to multi-homing. A *multi-homed* node is a computer that has more than one IP address. Our real world analogy of this is that our poor businessman is known simultaneously by multiple names; he is not just Mr. Ten Oxford Street but also Mr. Two Cedar Street. (Perhaps he has a large house that has doors on both streets.) Naturally, he wants to take advantage of this position, and make sure that his deliveries can be directed to the (perhaps less used) Cedar Street door whenever there is congestion at Oxford Street. Thus, he has to have a reliable way of conveying this information to other people, lest a thief could re-direct our businessman's deliveries to the thief's newly rented apartment at Mossland Street. Again, since all we have is location names and the ability to send packets to them, securely conveying multihoming information is not particularly easy.

2.1. Names and name spaces

Getting back to the Internet, we end up with a dilemma: *what is the relationship between an end-point's identity, its static identifier, and location names?*

The previous examples described an existing name space problem in the Internet. The IP addresses are currently used both as end-point identifiers and as topological location names. That is, the logical end-point of communication, typically an application hosted by a network node, is named by tagging the node's location, the IP address, with a port number. As a side effect, the IP address becomes the name of the node, and a part of the name of all end-points residing in that particular node.

This semantic overloading of IP addresses is causing a number of problems related to mobility, multi-homing, and security (see e.g. [7][12]). The main problems are related to maintaining the communication context active during topological movements of the nodes [2], and to protect the integrity of the management packets that are used to create the mappings between end-point identifiers and location names.

In the current Internet, there is another important name space, the Domain Name System (DNS). The DNS suffers from a different but related set of problems. However, it is not particularly relevant to our discussion, since Domain Names are used at the application layer, and they are always resolved to IP addresses before the actual communications can take place.

In the present approach, we try to use each name space separately, for semantically different cases. Basically, non-cryptographic names, like IP addresses or DNS names, are not suitable as secure identifiers in an open environment, like in the Internet. The only way to "own" something in the Internet is to keep it secret. The same principle concerns identities. An end-point must be able to prove to a peer that it has a specific identity without losing it, i.e., suffering an identity theft.

2.2. A new name space

There are two ways of proving the possession of a secret without actually revealing the secret: zero-knowledge protocols and public key cryptography. Either of these could be used to solve our identification problem. In practice, however, public-key cryptography is particularly suitable for identifying end-points in open environments [1][8][16].

The sole holder of the private key "owns" a specific *identity*. The corresponding public key, or any cryptographical derivative of it, works as an *identifier* for the *id*entity. This construction defines a naming trust relationship between an identity and an identifier without name certificates or other external infrastructure. *An identifier* is bound in a cryptographically strong way to *an identity*, while the trust in an *noncryptographic name* is more or less a matter of faith.

If we consider, once more, our Mr. Ten Oxford Street, he now gets a face. In a way, the private key is like a person's face in the real world. The public key, in turn, is like a photograph of that face. In the ideal case, a photograph identifies the person uniquely, and allows anyone holding a copy of the photo to recognize the person.

Using public keys as *primary* identifiers is similar to using photographs as the primary means of recognizing people. If you have a photo of a person, you don't necessarily know his or her name. You can still easily deal with him or her, and you can rely on the stability of the identity. Unfortunately, at this point we must gradually depart from our analogy, since the differences between the real world and the IP networks starts to grow larger than the similarities. For example, in an IP network, a node can easily have several private-public key pairs, and even create new ones on demand, while it is relatively hard to create a new face for yourself on demand. However, what is important here is how the new name space makes it possible to identify nodes based on some abstract identity of them instead of relying on topological locations for identification purposes.

Now, once we make the semantic separation of location names and end-point identifiers, and introduce a name space for end-point identifiers, the role of the IP address becomes clear. IP addresses are used purely for naming topological locations in the Internet. An end-point may change its attachment to the network without losing its name, or identity. The interrelationship between location names and end-points identifiers be-

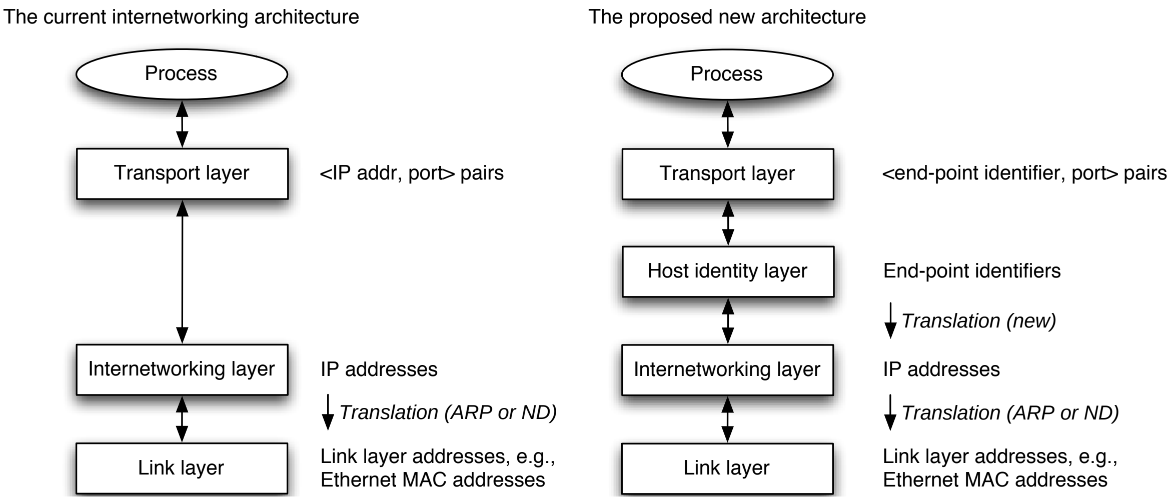

Figure 1: The current Internetworking and the proposed new architectures

comes dynamic. The new binding between IP addresses and end-point identifiers requires a new logical layer. [4]

3. A New Logical Layer

The Host Identity Protocol architecture [8] introduces a new protocol layer between the transport and internetworking (IP) layers[2]. Figure 1 describes the difference between the current and the new architectures. The left hand side of the figure describes the current architecture. There, processes are bound to transport layer sockets, and the sockets are identified with IP addresses and ports. As a result, this structure binds the processes to a specific topological location, thereby making process migration, end-host mobility, and multi-homing difficult.

The new structure is described on right hand side. In the new architecture, the transport layer sockets are no longer named with IP addresses but with separate end-point identifiers, i.e., public keys. The end-point which holds a particular private key is typically a host, but can basically be a smaller entity, like an application, or a larger entity, like a computer cluster.

The new identity layer translates the end-point identifiers into IP addresses. This is achieved by dynamically binding an end-point identifier to one or more IP addresses. This binding is a dynamic relationship, resulting in easy mobility, and simultaneously a one-to-many relationship, providing support for multi-homing. Due to the cryptographic nature of the end-point identifiers, it is fairly easy to secure the management messages needed to update this binding.

Thus, by breaking the tight connection between the transport and internetworking layers, we are able to ease the current technical problems hampering Internet mobility and multi-homing. By using public keys as primary identifiers we are able to easily solve the related security problems, at the same time slightly raising the general level of security in the Internet. Furthermore, the approach allows us to make applications and nodes to communicate with each other even across the chasm between the current version of IP, IP version 4, and the new IP, IP version 6.

4. Bridging the two versions of IP

The Internet is facing a painful growth process. The current version of the Internet Protocol, IPv4, is being replaced with IPv6. This transition has already taken several years, and is likely to take a few decades. During this time, there are network nodes that are able to communicate only with IPv4, only with IPv6, or both, and there are applications that know only about IPv4 address, only about IPv6 addresses, or about both. During the long transition period, getting all these to communicate with each other is a demanding task.

Since the new Host Identity name space separates the internetworking layer from the transport layer (and hence from the applications), we can solve the problems of node interoperability and application interoperability separately. We first focus on application level backward compatibility.

4.1. Application level backward compatibility

One of the design principles in the HIP based IPv4 to IPv6 interoperability has been backwards compatibility. Its implementation does not require changes to most existing IPv4 or IPv6 protocol specifications or existing applications. It supports both IPv4 and IPv6 sockets, which means we are able to run IPv4 applications over IPv6 protocol, and vice versa. The principle is the same in both cases, only the *handle* that is used to denote the actual end-point identifier is different.

A single end-point identifier, i.e., a public key, may have several handles denoting it. In the IPv4 socket API the handle is a 32 bit long datum, called Local Scope Identifier (LSI), and with the IPv6 socket API it is a 128 bit long globally scoped name, called Host Identity Tag

[2] It must be noted that not all people agree that a new layer is needed, nor that HIP can or should be understood as a new layer. For example, the Secure Shell protocol (ssh) already provides a similar kind of public key based identifiers, and it is being used to provide for limited mobility, like running cvs and rsync from mobile hosts. Thus, perhaps the idea of using public keys as host names is much more important than the proposed new layer. The new layer is just one particular way of implementing the new architecture.

(HIT). Handles replace the IP address(es) in the socket API; when making DNS queries, the DNS library returns handles instead of actual IP addresses, whenever appropriate.

In practice, when an application starts to communicate, it resolves a DNS name into an IP address. If public key end-point identifiers are used, the DNS contains the public key in addition to the usual IP address(es) for the peer host. The DNS library fetches the public key, and creates a corresponding handle. Before the resolver library returns the handle to the application, the operating system in the initiating node stores the received IPv4/IPv6 address(es) and the handle. This allows the operating system to map the handle to the IP address(es).

4.2. Cross-version Mobility and Multi-Homing

In the Internet, *end-node mobility* means that a mobile node changes the topological location of its interfaces in the network. A *hand-off* occurs whenever a host moves and its address is changed. *Mobility management* includes any mechanism where the mobile end-node keeps its communication contexts, i.e., connections, active during movement. Fundamentally, Internet communication is based on stateless packet exchange, relying on the network's ability to pass packets to their destination addresses. Therefore, in order to continue to communicate, a mobile node must be able to signal the changes in its addresses to its peer nodes. Furthermore, this signaling must be secure lest unsecured signaling can lead to unauthorized traffic diversion and denial-of-service [12].

Packets are normally sent serially, one after the other, via one topological path between the end-points. This nature of the packet data flow defines the nature of the hand-off procedure. The hand-off happens serially, between two different IP addresses, which each may belong to either IPv4 or IPv6 address families. From this point of view, the typical mobility hand-off within a single address family is just a special case of the more generic cross family hand-off. Replacing an IPv6 address with an IPv4 address, or vice versa, is just as simple. Furthermore, it does not matter whether the addresses are bound to one or different interfaces.

The magic behind the address family hand-off is that the static end-point identifier, the public key, is not used for routing. Unlike in Mobile IP or Mobile IPv6, we do not bind the end-point identifiers to address families.

The hand-off between IPv4 and IPv6 networks is based on separating the transport layer from the internetworking layer. Consider a legacy IPv4 application in a host that is initially connected to an IPv4 network, but later moves to a network that provides only IPv6 connectivity (see Figure 2). The application socket is initially bound to an IPv4 address, according to the new bindings architecture, as depicted in Figure 3. During the hand-off, the IPv4 address binding is replaced with an IPv6 address.

When the application sends a packet, the packet is passed through the new host identity layer. Since the host identity is currently bound to an IPv6 address, the host identity layer strips off the IPv4 header and replaces it with an IPv6 header. The packet is passed through the IPsec stack, where the IP payload is encrypted and an Encapsulated Security Payload (ESP) header added to the packet. Finally, the packet is sent out via the IPv6 stack.

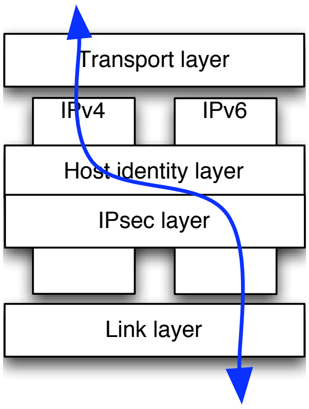

Figure 2: The actual packet data flow in IPv4 to IPv6 hand-off

When the IP packet arrives at the peer node, the ESP payload is verified, decrypted, and the ESP header is stripped off. The new semantics take place after this. Since the application is a legacy IPv4 application, the received IPv6 header is replaced, conceptually, by a freshly created IPv4 header, containing the LSIs in the place of IPv4 source and destination addresses, and the packet is sent into the IPv4 stack for further input processing. The IP layer finds the right TCP or UDP socket for the packet on account of the LSI handles, and passes the packet to the transport layer. For the IPv6 socket API the scenario is a mirror-image of the previous one.

4.3. Communicating across IP versions

As we saw above, the transport layer sockets are no more bound to IP addresses but to end-point identifiers. This makes it possible to make legacy IPv4 applications talk directly to IPv6 applications, and vice versa. The identifiers passed in legacy IPv4 and IPv6 APIs are handles to end-point identifiers. The only difference is that we use 32-bit LSIs in the IPv4 API and 128-bit HITs in the IPv6 API. However, at the logical level both of these identifiers are handles to the single identifiers in the host identity name space.

To facilitate communication across the APIs be-

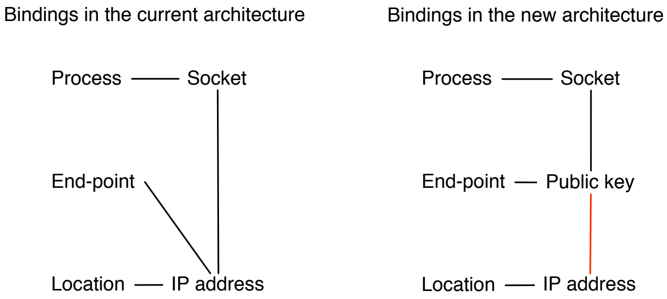

Figure 3: Bindings

tween different IP versions, it is necessary to make the TCP/UDP pseudo header checksums compatible. Since the sockets are bound to the host identifiers, it is natural to use the end-point identifiers in the pseudo header checksums. In practice, the 128-bit HITs are used underneath both the IPv4 and IPv6 APIs, and always use the IPv6 pseudo header format to compute the checksum. In our implementation, this is accomplished by "stealing" IPv4 sockets from the IPv4 stack to the IPv6 stack, at the system call level, and by making the conversion between HITs and LSIs at the appropriate system calls. Thus, in our implementation all packets are always passed through the IPv6 versions of TCP and UDP, and the IPv6 system call interface is enhanced to support IPv4 legacy applications.

There are two classes of applications that are affected, though. Applications that either inspect IP addresses directly or pass IP addresses explicitly in application layer payloads may need modifications. However, the first class of applications are typically diagnostic in nature, and most probably need to continue working with IP addresses. In the latter class, there are no difficulties if both of the peers use the same version of IP. If a legacy IPv4 application passes an IP address to its legacy IPv4 peer, an LSI will be passed. Since the LSIs are negotiated during the HIP protocol setup (see below), it designates the same end-point identifier at both ends. The same applies to IPv6 applications, the only difference being that a HIT is passed instead of an LSI. However, if an IPv6 application passes an IP address to a legacy IPv4 application, a 128 bits long HIT, in IPv6 address format, will be passed. Since most IPv4 applications cannot cope with IPv6 addresses, communication will fail. The reverse case, an IPv4 application passing an LSI to an IPv6 application, is likely to work, since most IPv6 applications are capable of handling IPv4 addresses.

As we have seen, the introduction of the new host identity layer neatly separates the transport and internetworking layers from each other, making mobility and multi-homing easy, even across the two versions of IP. Since HIP integrates security with mobility and multihoming, HIP establishes a security context before any communication takes place between the nodes[3]. This is the task of the actual Host Identity Protocol.

5. Host Identity Protocol (HIP)

The Host Identity Protocol (HIP) is the end-point to end-point signalling protocol [9]. Most importantly, it implements a key exchange protocol, using the public key end-point identifiers to authenticate a Diffie-Hellman exchange. After exchanging the initial messages, both communicating nodes know that at the other end-point there indeed is an entity that possesses the private key that corresponds to the claimed public key, i.e., its end-point identifier. Additionally, the exchange creates a pair of IPsec Encapsulated Security Payload (ESP) security associations, one in each direction. The nodes then use the ESP security associations to protect the integrity and confidentiality of the packets flowing between them.

In addition to protecting the network layer integrity of the payload traffic with ESP, HIP is used to secure the management messages exchanged between the end-points. The end-points inform their peers about the interfaces they have and the current IP addresses assigned to the interfaces. In effect, this shares information about the current *multi-homing situation* of the end-points. Each end-point has complete freedom to select which interfaces and which IP addresses to announce to each peer node.

To the peer node, it is immaterial whether the announced interfaces are real or virtual[13]. All it needs to know is to make sure that the announcing end-point is indeed reachable through the claimed IP addresses. The reachability needs to be checked, or otherwise the mechanism may be used to launch denial-of-service attacks.[8]

Once the security associations and the multi-homing situation are established and verified, the end-points may communicate in an secure and resilient way. As the connectivity status of the end-points change, they may signal the changes in the situation as needed. That is, if an end-node looses connectivity on an interface, acquires a new interface, or moves an interface from one location to another, it typically wants to signal the change to its peers. The HIP protocol includes the Readdressing Packet (REA) for this purpose. Naturally, all the REA packets must be secured; this is accomplished by signing them by the node's private key corresponding to its end-point identifier.

5.1. A new IPsec mode

In IPsec, *transport mode* is used to establish a secure communications directly between any two communication end-points. In the *tunnel mode*, in turn, the end-points of a tunnel are typically not the same as the final communication end-points. In other words, the tunnel mode security associations are bound to different IP addresses than the sockets.

For the purposes of HIP, we have proposed a new IPsec mode, denoted as Bound End-to-End Tunnel (BEET) mode[11]. This new mode is a combination of the tunnel and transport mode, using the transport mode packet format but providing limited tunnel mode semantics. In particular, the new mode takes care of the translation between IP addresses, used on the wire, and the end-point identifiers, used at the transport layer.[1]

It is important to notice that in practice we do not change the IP or IPsec header (ESP and AH) structures, but just the details of the packet handling within the end-nodes. However, at the logical level, the new name space architecture imposes changes to the logical packet structure. That is, each packet must logically include the end-point identifiers of the sender and recipient. However, as IPsec is used, the IPsec Security Parameter Index (SPI) in ESP packet can be used as tags for end-point identifiers, resulting in packets that are syntactically similar to those used today. This is illustrated in Figure 4.

[3] The security context is needed for HIP enabled communications. Before the context is established, the nodes can communicate using either IPv4 or IPv6, just like today.

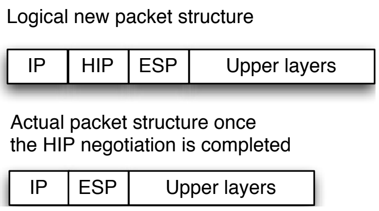
Logical new packet structure
IP
HIP
ESP
Upper layers
Actual packet structure once
the HIP negotiation is completed
IP
ESP
Upper layers

Figure 4: The packet structure

As the packets are integrity protected with ESP, the recipient is always able to verify that a received packet was sent by the alleged peer no matter what the source and destination addresses are. Thus, by binding the IPsec Security Associations to end-point identifiers instead of IP addresses, the destination address becomes pure routing information, and the source address becomes almost obsolete [3]. Only during connection setup, when the nodes have not authenticated each other, does the source address play any substantial role. Once the peer nodes have secure bindings between the end-point identifiers and addresses, the source address is not needed any more by the nodes, and its only function becomes to carry information about the topological path the packet has taken [3].

6. Standardization and implementation status

At the 58th IETF meeting, which was held at Minneapolis Hilton in November 2003, there was a HIP BOF meeting. In the meeting, we gave a public demonstration of HIP based IPv4 and IPv6 application interoperability and mobility, using our BSD Unix based prototype implementation. The decision at the meeting was to form a HIP working group. At the time this paper is being written (December 2003), the HIP community is waiting for the final desicion from the Internet Engineering Steering Group (IESG) to form a a new IETF working group for HIP.

At the time of this writing, the first release of our prototype implementation is publicly available in source code format. The implementation is distributed as a set of patches against a current version of the FreeBSD operating system. More information and download instructions are available at `http://www.hip4inter.net`

7. Related Work

7.1. i^3: Internet Indirection Infrastructure

The Internet Indirection Infrastructure (i^3)[15] defines a new overlay routing mechanism for multicast, anycast and mobile communication. The architecture decouples the sender and receiver of each other using rendevouz servers, called i^3 nodes. Packets are routed from the sender to the receiver always via these i^3 nodes. Therefore, the architecture uses triangle routing and does not offer optimal packet delivery.

In the i^3 architecture, routing is based on a new end-point identifier name space. An end-point identifier can be any fixed length hash value, like a hash of a DNS name, a hash of a web address, or a hash of a public key. Basically, the new identifiers have two kinds of semantics. Firstly, they are used as end-point identifiers. Secondly, they are used for i^3 level packet routing. That is, the i^3 layer is a self-organizing network and does not use DNS for address resolution. Therefore, the i^3 architecture does not separate the location names from end-point identifiers in a clean enough way; the end-point identifiers are still used for routing.

The architecture uses the existing Internet routing infrastructure to deliver packets between the i^3 nodes. Furthermore, the i^3 nodes work like routers for the end-point identifiers, delivering packets to the listening receivers.

i^3 suffers from the basic security vulnerabilities that are related to location updates. Unverified address binding updates cause easily several difficult Man-in-the-Middle, masquerade, or Denial-of-Service attacks. However, Stoica et.al.[15] propose that the end-point identifiers can be generated from public keys and public key cryptography can be used to secure the i^3 architecture. That is very similar to what the Host Identity Protocol offers.

7.2. FARA

FARA[5] (Forwarding directive, Association, and Rendevouz Architecture) defines an abstract network architecture model. The model decouples of identity from entity's location in a clean way. An *entity* can be a single process, a group of processes or even a computer cluster. The communication connections between entities are identified with *associations*.

The architecture avoids the introduction of a new global name space for the identity. In FARA, the identity of an entity is an abstraction without a real world correspondence. An association is not used to name an entity. Basically, the definition of the identity is left open. However, the paper mentions that an entity can be discovered by using some global mechanism. They define a rendezvous mechanism and a FARA Directory System to discover an entity. The rendezvous mechanism discovers the location of an entity and initiates an association. The directory system is basically used to contact a rendezvous point.

FARA leaves open the end-to-end authentication model. To be globally scalable, any authentication protocol must define an identifier name space. In other words, the FARA architecture hides, more or less indirectly, the global identity name space behind the security properties.

To facilitate mobility, the associations are dynamically bound to a set of locations. An entity may have several associations, each of them having their own local index. However, the security aspects requires an explicit mapping between the associations and the entity. Therefore, the architecture requires a scalable entity name space, which was bypassed in the paper. However, the paper mentions that HIP could be used to authenticate the entities to each other. Basically, HIP is an instantiation of FARA, when the identifiers in HIP are used to identify smaller entities than hosts. The main difference is that

HIP requires global identifier name space.

8. Conclusions

In this paper, we have briefly described the Host Identity Protocol (HIP), and discussed how it can be used to securely implement end-host mobility and multihoming, even across IPv4 and IPv6. In HIP, the key idea is to introduce a new name space, based on cryptographical key pairs, into the IP stack. With the new name space, communication end-points are no longer named with IP addresses but with public keys. At the implementation level, transport layer sockets are no more bound to IP addresses but to these new end-point identifiers. Being public keys, the new identifiers can be securely mapped to IP addresses. The mapping is dynamic and one-tomany, providing for mobility and multi-homing.

We believe that an architectural change of this nature is essential for the future of the Internet. It is no longer sufficient to name end-points based on their location, i.e., their IP address. Either a new name space is needed, or some other existing name space (such as the DNS) must be converted to fulfil the need. HIP is an attractive attempt to provide such a new name space, characterized by its tight integration with IPsec, reliance on public key cryptography, backward compatibility with almost all existing protocols and applications, and relatively lightweight implementation[13]. To our knowledge, HIP is the only proposal that is able to securely provide for mobility and multi-homing across the two versions of the Internet Protocol.

Acknowledgments

Developing HIP has largely been a community effort. The basic ideas were put together by Robert Moskowitz, and he acted as the primus motor for a long time, before the baton was mostly transferred to the present authors. The team of early implementors, including but not limited to Tom Hendersson, Andrew McGregor, and Tim Shepard, was invaluable during the development of the ideas into their present stage. For a more complete list of the people involved, see the acknowledgements sections in the relevant Internet Drafts[8][9].

The authors would like to thank Tom Henderson, Miika Komu, Martti Mantyla, Kenneth Oksanen, Matti Rantanen, Goran Schultz and Timo Ylitalo for their valuable comments on various versions of this paper. Our special thanks go to Tim Shepard who gave us numerous very valuable comments.

REFERENCES

[1] S. Bellovin. EIDs, IPsec and HostNAT. A presentation give at 41st IETF in Los Angeles, California, March 1999.

[2] P. Bhagwat, C. Perkins, and S. Tripathi. Network Layer Mobility: an Architecture and Survey. *IEEE Personal Communications Magazine*, June 1996.

[3] C. Candolin and P. Nikander. IPv6 Source Addresses Considered Harmful. In *Proc. NordSec 2001*, November 2001. Sixth Nordoc Workshop on Secure IT Systems, Lyngby, Denmark.

[4] I. Castineyra, N. Chiappa, and M. Steenstrup. The Nimrod Routing Architecture. RFC 1992, August 1996.

[5] D. Clark, R. Braden, A. Falk, and V. Pingali. FARA: Reorganizing the Addressing Architecture. In *Proc. ACM SIGCOMM'03*, August 2003. ACM SIGCOMM 2003 Workshops, August 25-27, Karlsruhe, Germany.

[6] D. Johnson, C. Perkins, and J. Arkko. Mobility Support in IPv6. Internet Draft, work in progress, June 2003.

[7] A. Mankin and et.al. Threat Models introduced by Mobile IPv6 and Requirements for Security in Mobile IPv6. Internet Draft (expired), in Proc. 35th IETF meeting, Minneapolis, March 2002.

[8] R. Moskowitz and P. Nikander. Host Identity Protocol Architecture. Internet Draft, work in progress, September 2003.

[9] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson. Host Identity Protocol. Internet Draft, work in progress, October 2003.

[10] P. Nikander. Denial-of-service, address ownership, and early authentication in the ipv6 world. In *Security Protocols*, number 2467 in LNCS, pages 12–21, 2002. Cambridge Security Protocols Workshop, April 2001.

[11] P. Nikander. A Bound End-to-End Tunnel (BEET) mode for ESP. Internet Draft, work in progress, October 2003.

[12] P. Nikander, T. Aura, J. Arkko, G. Montenegro, and E. Nordmark. Mobile IP version 6 Route Optimization Security Design Background. Internet Draft, work in progress, April 2003.

[13] P. Nikander, J. Ylitalo, and J. Wall. Integrating Security, Mobility, and Multi-Homing in a HIP Way. In *Proc. Network and Distributed Systems Security Symposium*, February 2003. NDSS'03, San Diego, CA, USA.

[14] C. Perkins. IP Mobility Support. RFC 2002, 1996.

[15] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. Internet Indirection Infrastructure. In *Proc. ACM SIGCOMM 2002*, August 2002. Pittsburgh, PA, USA.

[16] J. Ylitalo, P. Jokela, J. Wall, and P. Nikander. Endpoint Identifiers in Secure Multi-homed Mobility. In *Proc. OPODIS'02*, December 2002. 6th International Conference On Principles Of DIstributed Systems OPODIS'02, Reims, France.