P7

Publication P7

Jukka Ylitalo, Patrik Salmela, and Hannes Tschofenig, "SPINAT: Integrating IPsec into Overlay Routing", in Proc. of the First International Conference on Security and Privacy for Emerging Areas in Communication Networks (SecureComm'05), pp. 315-326, Athens, Greece, September 5-9, 2005, ISBN 0-7695-2369-2, Publisher: IEEE Computer Society (Ed: JD Cantarella).

SPINAT: Integrating IPsec into Overlay Routing

Jukka Ylitalo and Patrik Salmela
Ericsson Research NomadicLab, Finland
{first-name.surname}@ericsson.com

Hannes Tschofenig
Siemens, Germany
Hannes.Tschofenig@siemens.com

Abstract

Tackling the major Internet security, scalability and mobility problems without essentially changing the existing Internet architecture has turned out to be a very challenging task. The overlay routing approaches fortunately seem to offer a sound way to mitigate most of these issues. Basically, they decouple the end-point identifiers from locators by defining a new namespace. Overlay routing is based on the dynamic binding, at middle-boxes, between the two namespaces. The approach is very close to Network Address Translation (NAT) principles. Therefore, the IPsec NAT traversal related problems apply also to overlay architectures. In this paper, we integrate IPsec into the overlay routing using Security Parameter Index (SPI) multiplexed NAT (SPINAT). Our approach reduces tunneling overhead and supports asymmetric communication paths. We believe that the SPINAT will be a key component in securing overlay routing infrastructures, like in the Internet Indirection Infrastructure (i^3).

1. Introduction

Overlay routing infrastructures (e.g. [34][2][26][37] [6][9][11][12]) introduce a new logical protocol layer that translates *end-point identifiers* into IP addresses. It must be noted that it is not universally agreed that a new layer is needed, nor that overlay routing infrastructures can or should be understood as a new layer. Thus, perhaps the idea of using a new namespace for end-point identifiers is much more important than the proposed logical layer. The new layer is just one particular way of implementing the new architectures. The end-point which holds a particular identifier is typically a host, but can basically be a smaller entity, like an application, or a larger entity, like a computer cluster[6][9].

In some overlay routing infrastructures[26], transport layer sockets are no longer bound to IP addresses, but to separate end-point identifiers[31]. This is also called *identifier-locator splitting*[19]. This dynamic one-to-many binding between end-point identifiers and locators results in *mobility*[44] and *multi-homing*[19]. Once the end-point identifier has cryptographical properties[24] [28], it is fairly easy to secure the mobility management messages needed to update this binding[4].

Overlay routing is based on the resulting end-point identifier namespace. Although, using a flat end-point identifier namespace for routing compels the overlay routers to establish a state for each listening end-point. This is the trade-off compared to stateless legacy routers[5] and structural IP addresses. However, the required state for IP address translation makes overlay routing and rendezvous servers even more similar to Network Address Translation(NAT)[32]. This easily involves problems with IPsec traversal through overlay routers. Instead of using UDP-tunneling[1], we have taken a fresh start and present Security Parameter Index (SPI) multiplexed NAT (SPINAT). SPINAT is an attempt to integrate IPsec into overlay routing infrastructures.

The tendency towards large scale Distributed Denial-of-Service (DDoS) attacks in the Internet has moved the focus on research from end-to-end control plane design to overall architectural design. This concerns also the IPsec design. Some IPsec control plane protocols are aimed to protect the responding parties from different kind of DoS attacks (e.g. [3][28]). However, protecting hosts from CPU and memory exhaustion attacks do not prevent malicious nodes to implement flooding attacks. In other words, an attacker can try to implement a DoS situation by causing payload traffic to flood the victim's local network. A variety of overlay routing architectures offer location privacy to end-points (e.g. [33]), which partially protects end-points from DDoS attacks. The end-points can control their incoming traffic flows (e.g. [37]) and mitigate the flooding effects. Thus, integrating IPsec into overlay routing indirectly increases the DDoS resistance of IPsec.

We present SPINAT as a building block for different overlay architectures, without binding it to any specific approach. Although, a reader who is, e.g., familiar with Internet Indirection Infrastructure (i^3)[34] can safely think that a SPINAT device corresponds to an i^3 node. Our SPINAT experiment results are based on a Host Identity Protocol

(HIP)[28] implementation.

The rest of this paper is organized as follows. In Section 2, we present related overlay routing work. Section 3 contains analysis about the security problems related to the current NAT practice. In Section 4, we present our SPINAT approach. The security implications of SPINAT are discussed in Section 5. Section 6 contains SPINAT experiment results. Finally, Section 7 concludes the paper.

2 Related Work

In a way, legacy IP routing and overlay routing are functionally similar, but at different layers in the stack. A legacy router[5] translates link layer addresses while an overlay router translates network layer addresses. A legacy router uses the IP addresses as routing table identifiers, used to guide link layer address translation. Using the same logic, the overlay router uses the end-point identifiers for network layer address translation (e.g. IPNL[13], DataRouter[36]). The situation is illustrated in Figure 1 a) and b).

Furthermore, port multiplexed NAT (NAPT)[32] devices use the transport layer identifiers as static identifiers in the network layer address translation. However, the situation is made a little bit untidy by the fact that current transport layer identifiers consist of both IP addresses and ports. In practice, the namespaces are not separated of each other, but they partially smear together (see Figure 1 c). The analogy between the overlay routing and NAT is evident. Both of them translate IP addresses, but use different namespace for mapping.

The common thing with IP routing, overlay routing and NAT is that they all use an upper layer namespace to guide lower layer address translation. To generalize this observation, we can say that the layer-n+1 namespace is used to guide layer-n identifier translation. Table 1 contains some commonly used definitions for identifier translation at different layers.

From the overlay routing infrastructure point of view one interesting remark is related to the layer-4 identifier translation. The layer-4 identifiers are used for end-point identifiers. Dynamically changing end-point identifiers results in authorization and delegation of signaling rights between

Ln identifier translation	definition
L1	switching
L2	routing
L3	NAT / overlay routing
L4	ALG / delegation

Table 1. Definition of identifier translation at different layers.

end-points (see [37][25]). However, layer-4 identifier translation is out of this paper's scope, but binds the overlay routing smoothly to delegation based architectures.

One frequently cited overlay routing architecture is the Internet Indirection Infrastructure (i^3)[34][44]. i^3 defines an overlay routing mechanism for multicast, anycast, and mobile communication. Packets are always routed from the sender to the receiver via rendezvous servers, called i^3 nodes. Overlay routing is based on a new end-point identifier namespace. The end-point identifiers can be any fixed length hash values having two kinds of semantics. They are used as global end-point identifiers and for table indices in overlay routing. However, the i^3 architecture uses the existing Internet routing infrastructure to deliver packets between the i^3 nodes.

The i^3 nodes act as overlay routers for the end-point identifiers, delivering packets to the listening receivers. In a typical case, a receiver registers *a trigger*, to an i^3 node, to listen to traffic. The trigger contains an end-point identifier and an IP address of the receiver. The identifier may belong to a single receiver or it may represent a group of receivers. The i^3 layer is a self-organizing network that applies Distributed Hash Tables (DHTs) for implementing a distributed directory service (see e.g.[35][43][15]). In addition, several DHT security issues are analyzed by Castro et. al. in [8].

i^3 suffers from the basic security vulnerabilities that are related to location updates and confidentiality protection of the traffic. However, Stoica et.al. [34] propose that the host 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 (HIP) offers [28].

Adkins et.al. solve several security vulnerabilities related to i^3 in their security enhanced approach, called Secure-i^3[2]. The Secure-i^3 defines a way to mitigate the most essential security issues in the overlay routing. However, Secure-i^3 is a framework that does not go into protocol level details. Nikander et.al. have continued the work by designing Host Identity Indirection Infrastructure (Hi^3)[26].

Hi^3 is an instantiation of Secure-i^3 architecture that implements the required security properties at the protocol level design. Hi^3 combines ideas from Secure-i^3[2] and HIP[28], producing an architecture that is more secure and efficient than either of the two approaches alone. Hi^3 is based on the observation that a DHT extended HIP rendezvous server and the basic Secure-i^3 infrastructure are fairly close to each other. In Hi^3, the rendezvous server forwards IPsec traffic between the end-points. In this paper, we have continued the work and we present an efficient way to integrate IPsec into Hi^3 kind of approaches. Basically, our approach can also be applied with other overlay routing infrastructures like DOA[37], a Layered Naming Architecture[6], FARA[9], PeerNet[11], and UIP[12].

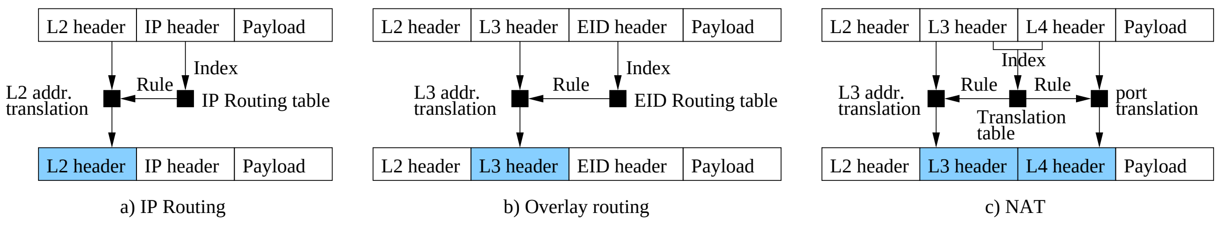

Figure 1. Analogy between IP routing, overlay routing and NAT.

3 Perils of Legacy NAT

To motivate our SPINAT work, we have to consider the problems related to the existing NAT devices. In the current Internet, IP addresses are used both for identifying hosts and naming their topological locations. This semantic overloading is deeply related to most of the well-known Network Address Translation (NAT) problems [16][23]. Since NAT changes the IP addresses, and the IP addresses are used as end-point identifiers, the end-host's identity appears to be changed at NAT traversal. In other words, the current NAT practice does not only translate locators, but it also changes the apparent identity of the communicating parties.

When a host *identifier* is only weakly bound to the *identity* [38], the host is vulnerable to an *identity theft*. The most prominent example of such IP level identity theft is the ease of IP address spoofing. In a way, NAT takes advantage of the weak binding between an end-point identifier and its locator. A NAT device transparently replaces the end-point identifiers with new ones during connection initialization, and becomes a Man-in-the-Middle (MitM). As a consequence, IP addresses cannot be used to identify hosts behind a NAT device. This turns out to be a problem when referrals are used at other layers in the protocol stack, for example with the SIP protocol.

There are also other problems related to identifiers. Typically, a NAT device maps several private addresses to a single public address using port information. This type of NAT is usually known as port-level multiplexed NAT, or sometimes as Network Address Port Translation (NAPT)[32]. One reason for including port numbers in the translation is the current shortage of public IP addresses. The IP address overloading in NAPT may cause misidentification. A peer node cannot find out whether it communicates with the single host or a number of different hosts behind a single NAPT device. Moreover, if a NAT device dynamically changes its address mapping policy, the end-point identifiers in the packets are changed, and all active connections will break. This situation is also known as site renumbering. The same problem appears if the NAT device is mobile or needs to change its IP address.

Dynamic NAT binding allocation and packet filtering make it impossible to contact end hosts behind NAT and NAPT devices, if no other third party entities (such as application servers or rendezvous servers) are used. Fortunately, a couple of approaches have been presented to solve the legacy NAT traversal problem using UDP tunneling, like STUN[30], TURN[29] and TEREDO[18]. STUN is an IETF standard, while TEREDO is implemented in the latest Windows XP operating systems to support NAT traversal for IPv4/IPv6 networks. NUTSS[14] and NATBLASTER[7] are proposals for establishing TCP connections through NAT devices [1].

The described approaches do not require modifications to the existing and largely deployed NAT devices. However, without incorporating additional logic into the middleboxes and support of third party entities, certain protocol solutions become tricky (e.g. hole punching) when both end hosts are behind a NAT device. From the protocol design point of view the UDP tunneling violates the TCP/IP layering, wastes bandwidth, and makes the protocol implementation gradually more complex.

Thus, the IETF MIDCOM working group has been working on protocols that allow the end host to directly interact with middleboxes in a path-decoupled fashion. More recently work on path-coupled NAT/Firewall signaling protocols has been started in the IETF NSIS working group. Path-coupled signaling aims to find middleboxes that are located along a particular data path. It is obvious that existing middleboxes need to be updated to support this functionality.

Furthermore, if a host wants to publish its identifier in the DNS, it must be the public IP address of the NAT device. This kind of separation between the actual end hosts and their public identifiers easily introduces security vulnerabilities. From a security and reliability point of view, an identity and the corresponding identifier should be always bound to the same end-point. IPsec is an example of a protocol that suffers from the related NAT traversal problems[1]. The current proposal is to use an UDP tunneling to pass NAT devices[20]. Although, the approach unnecessarily increases packet size, and may cause configuration difficulties, e.g., in firewalls.

[1]Please note that many of these proposals reuse techniques developed in the p2p filesharing and gaming community.

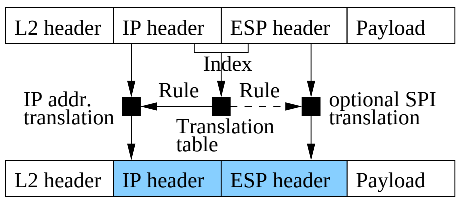

Figure 2. SPI multiplexed Network Address Translation (SPINAT)

4 SPI multiplexed NAT (SPINAT)

The new namespace used in overlay routing solves the previously mentioned problems in a sound way. From the logical point of view, the address translation in overlay routing is based on the global end-point identifiers. An end-point identifier enabled NAT device translates IP addresses; acting as an overlay router for the end-point identifiers. However, to avoid enhancing the IP and IPsec packet headers (e.g. by defining new IP options), the full end-point identifiers (or hashes of them) are not meant to be carried in the IP header. Instead, when IPsec is used the Security Parameter Index (SPI) value together with the destination IP address can work as *an index for end-point identifiers* (e.g. HIP[28]).

The address translation at a SPI multiplexed NAT (SPINAT) device is based on the SPI value and the destination IP address carried in the IPsec payload packets. This is illustrated in Figure 2. The SPINAT device establishes a state for translation during IPsec control plane signaling. In this paper, we extract the SPI exchange from the key-exchange context. The SPI exchange, like the Diffie-Hellman exchange, can be considered as a building block to be applied in different kinds of key-exchange protocols. Therefore, we do not focus on any specific protocol, like IKE, IKEv2 nor on HIP exchange. Instead, our purpose is to propose some design choices for the SPI exchange that can be implemented as part of IPsec control plane signaling.

4.1 State Establishment and Update

We propose that the IPsec control plane signaling is used to register *triggers* [34] and to establish *communication contexts* at SPINAT devices. A trigger consists of an identifier and an IP address of an end-point. The communication context stores the required information to implement the translation, like SPI values. Each receiver registers a trigger to receive IPsec control plane messages containing the end-point identifiers in the headers. An end-point can register a trigger in two ways:

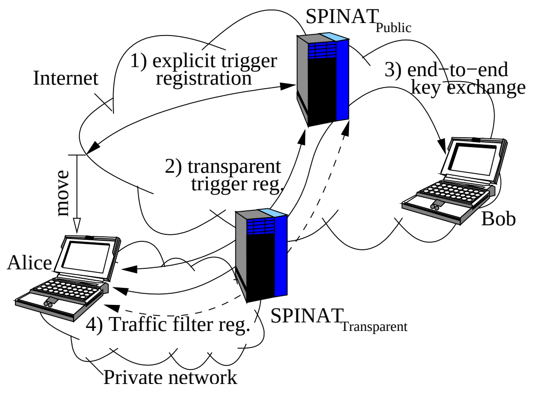

Figure 3. Trigger registration at public and transparent SPINAT devices.

1. it can *explicitly register a trigger at a public SPINAT device using Security Association (SA) establishment signaling*, i.e. the key exchange (Figure 3-1)

2. it can *transparently register a trigger at an on-path SPINAT device during end-to-end SA update signaling*, i.e. the mobility exchange (Figure 3-2).

We will discuss in Section 4.1.3 how the SA update and location update are strongly related to each other from the SPINAT's point of view. Once the receiver replies to the initial trigger message, the SPINAT devices on the path typically establish contexts for the communication session. However, the receiver may also explicitly establish a communication context at the SPINAT device where it has registered its trigger to. In other words, also the context can be established in two ways:

1. An on-path SPINAT device can transparently establish a context *during end-to-end SA establishment signaling* (Figure 3-3).

2. The receiver may *use an explicit protocol to register IPsec traffic filter* at a SPINAT device (Figure 3-4).

Figure 3 describes a situation where Bob wants to contact Alice who moves from the public network to a private network.

4.1.1 Explicit and Transparent Trigger Registration

Initially, (Figure 3-1) Alice establishes a security association with her trusted rendezvous server, i.e. SPINAT$_{Public}$, located in the public network. SPINAT$_{Public}$ learns the mapping between Alice's identifier and IP address during the exchange. Later (Figure 3-2), Alice wants also to be reachable at the private network. However, Alice and

$\text{SPINAT}_{Transparent}$ do not belong to the same administrative domain. The transparent $\text{SPINAT}_{Transparent}$ device is located between the private and public networks. In this case, the authentication is based on an assumption that $\text{SPINAT}_{Transparent}$ does not need to care about Alice's actual identity as long as the identity is the same during the communication context lifetime. To avoid identity theft and DoS attacks, Alice's identifier should be derived from a public-key, a hash chain, or a one-time random string (Sections 5.1 and 5.3).

The semantics in the transparent trigger registration is similar to STUN[30] and TEREDO[18]. Alice negotiates an SA update exchange with her trusted SPINAT_{Public} through the $\text{SPINAT}_{Transparent}$. The SPINAT_{Public} learns the public address of the $\text{SPINAT}_{Transparent}$ device, and updates the registered trigger. Further, the $\text{SPINAT}_{Transparent}$ transparently learns Alice's identifier and the private address. The mapping between the end-point identifier and the IP address, at SPINAT devices, makes it possible to *initialize connections* in both directions. This requires that public SPINAT devices implement DHT functionality (e.g. Chord[35] or Tapestry[43]).

When a public key pair is used as a host identifier (e.g.[28]), the host keeps the actual private key to itself, but is reachable via the IP address of the SPINAT device. From the security point of view, the situation is very different between a host inside a traditional NAT region and a host inside a SPINAT region. In the former case, the host implicitly trusts the legacy NAT device to represent the host through the public identifier, i.e., the public IP address. In the secure overlay routing case (e.g. Hi^3[26]), the identity is securely stored at the end-point; not at the SPINAT device. Therefore, the SPINAT device cannot misuse the identity of the host.

4.1.2 Communication Context Establishment

After Alice has registered her triggers to SPINAT devices she is able to receive traffic. Once Bob starts a key exchange with Alice, the SPINAT devices may either dynamically establish a communication context or require that Alice explicitly registers an IPsec traffic filter for the session (see Figure 3-4). In the former case the initiator and in the latter the responder establishes the context. The dynamic context establishment does not require extra round-trips, but does not either support traffic filtering. Therefore, if the responder explicitly establishes a context for a session it may protect itself from flooding. A more sophisticated approach is to move the decision point from the responder to the SPINAT device using authorization (see DOA[37]).

The overlay routing topology may change dynamically after the initial end-to-end SA establishment and packets may start flowing via different SPINAT devices during the same communication session. A change in the overlay routing topology and security association update must trigger an end-to-end SA update signaling to assure that the new transparent SPINAT devices on the path can establish states and avoid SPI collisions. It is obvious that the SPINAT based overlay routing comes closer to re-active ad-hoc routing [17].

4.1.3 Communication Context Update

The SPINAT device can map traffic between different address realms and even between address families [40]. Decoupling end-point identifiers from locators makes it possible to update SPI and IP address bindings dynamically without breaking the transport layer connections at end hosts. On the other hand, the address translation at SPINAT devices is based on the destination IP address and the SPI value. Therefore, the *re-keying* and *re-addressing* procedures are similar from the SPINAT point of view.

It is a logical design choice to integrate the SA update exchange with mobility management signaling. The SPI value is changed when a mobile node creates a new SA. On the other hand, the IP address is changed when a mobile node changes its topological location. It is good to notice that the two events, re-keying and re-addressing, do not typically happen simultaneously. The only static information related to a communication context at the SPINAT device is the end-point identifier-pair. The SPINAT device must authenticate each IP address and SPI binding updates to avoid re-direction and DoS related attacks (see Section 5.3).

4.2 SPI Translation

In large networks, SPI collisions at SPINAT devices are quite probable, compared to collision in large end-point identifier namespaces. Therefore, SPINAT may work in two different ways when an SPI collision happens. The device can either *a) drop the key-exchange message carrying the SPI value* or *b) translate the SPI value on the fly*. Both of the approaches are based on an assumption that SPI values, included in the IPsec control plane signaling *cannot be encrypted* [2]. In addition, in the latter case, the SPI values *cannot be signed* in the IPsec control plane messages *nor integrity protected* in IPsec payload packets. The security implications of these requirements are discussed in Section 5.

[2]Currently defined key-exchange protocols, IKE and IKEv2, encrypt the SPI values during the key exchange negotiation, and thereby make it impossible for the SPINAT devices to learn the correct association between SPI values and IP addresses.

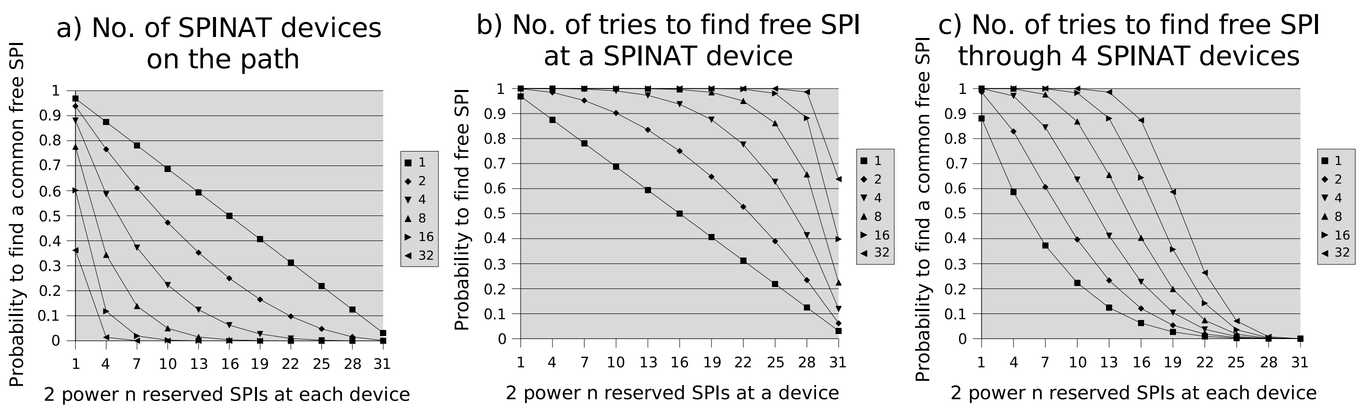

Figure 4. SPI collision probabilities.

4.2.1 Avoiding Changes to IPsec ESP

If the SPINAT device drops a key-exchange message due to an SPI collision, the initiator has to resend the message including another randomly selected SPI value after a resubmission timer goes off. In highly populated networks, the SPINAT device may need to drop SPI exchange packets a couple of times before the initiator manages to generate an SPI value that does not collide. Basically, the SPINAT device may send an ICMP packet to the initiator to inform about the collision and about available SPI values. However, the initiator should not directly trust an unsigned ICMP message as it may be generated by a malicious node. Instead, the initiator should first let the resubmission timer go off and then send the proposed SPI value in the new message. However, if the initiator receives several ICMP packets carrying different SPI values, it should generate a new random SPI value by itself.

The number of SPINAT devices on the path and number of existing contexts at the devices define the probability of finding an SPI value that is available at each device (see Figure 4 a) [3]. It is interesting to notice that the probability of finding a free SPI value at a single SPINAT device in Figure 4 b) increases controversaly in the same ratio as in Figure 4 a), but in function of retries. A SPINAT architecture without SPI translation support does not scale well, as illustrated in Figure 4 c).

4.2.2 Changing IPsec ESP Integrity Protection Computation

Scalable overlay routing should support SPI translation to avoid several extra round-trips caused by SPI collisions. In practice, SPINAT may work in the same way as NAPT[32]. This requires changes to the current IPsec ESP practice[21]; unlike the previous approach. If a specific SPI is already in use, the SPINAT device replaces the value with a new one carried in the IPsec control plane message and in the subsequent IPsec ESP headers. The most important difference between SPI and port based multiplexing is that an SPI value identifies a Security Association (SA) in the network layer, while the port number is used to identify a socket at the transport layer. The implications of changing SPI values are different than changing port values. If an attacker manages to traverse a NAT/firewall by spoofing a port number, the packet finishes up to a listening application. However, changing an SPI value does not bypass the IPsec handling.

For the purpose of SPINAT we present a variant of IPsec ESP transport mode, denoted as Stripped End-to-End Tunnel (SEET) mode. It is based on the initial Bound End-to-End Tunnel (BEET) mode proposal by Nikander et.al.[27]. The BEET mode is a combination of IPsec tunnel and transport modes, using the transport mode packet format but providing limited tunnel mode semantics. In particular, the mode takes care of the translation between IP addresses, used on the wire, and the end-point identifiers, used at the transport layer. The SEET mode does not change the IPsec ESP header structure, but the integrity computation and the details of the packet handling within the end-nodes. The SEET mode does not include the SPI value in the ESP header integrity protection computation (see Figure 5).

4.3 Decoupling IPsec Control Plane from Overlay Namespace

The main reason for introducing the IPsec SEET mode is to allow SPI translation at SPINAT devices. Alongside with the SEET mode we also propose another IPsec mode for overlay routing, called *Control Plane Header (CPH)*. According to its name, the IPsec control plane messages, carried typically in UDP datagrams, are additionally tunneled

[3] The probabilities are based on assumptions that the end-host selects randomly its SPI values and the contexts do not depend on each other at the different SPINAT devices.

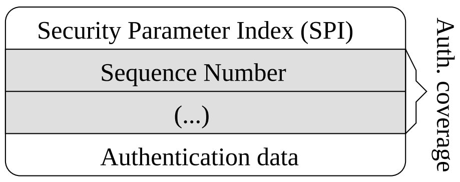

Figure 5. The SPI in the ESP header is not authenticated in the SEET mode.

over the IPsec CPH. The CPH contains only the end-point identifiers and optionally the related SPIs. The motivation to have CPH is to avoid the tight dependency between the IPsec control plane namespace and the related overlay routing namespace. The other reason is to support simple firewall configuration policies where the SPINAT firewalls pass only the IPsec protocol numbers.

It is good to notice that the semantics of the CPH and SEET modes are different. Although, it seems to be an attractive alternative to always replace the SEET header, used in IPsec payload packets, with the CPH header. In that case, the obtained header size optimization by tagging end point identifiers with SPIs would vanish. Also, if the IPsec ESP payload structure were to change, probably also the SPI namespace would be increased [4]. Furthermore, this would essentially change the semantics of SPI selection and require changes to Type-Length-Value (TLV) fields in the existing key-exchange protocols. Furthermore, SEET mode requires only small changes to integrity protection computation, while replacing the current ESP header structure with CPH would require more changes both to standards and existing implementations. Therefore, we propose that CPH is initially used only for tunneling the IPsec control plane signaling.

The semantics of the CPH is simple. Each IPsec daemon registers [5] itself to the IPsec module, e.g., using an extended (currently not standardized) PF_KEY[22] interface. Once a host receives an IPsec CPH message, the IPsec module verifies that the end-point identifier and the UDP destination port are registered to the IPsec module by the daemon. If the registration is missing IPsec drops the packet.

The CPH implements a combination of the tunnel and transport mode, like SEET and BEET[27] mode do (Section 4.2.2). It replaces the locators in the IP header with the end-point identifiers before passing the packet to upper layers. In this way, also the transport layer connections used by the key-exchange protocols can be bound to end-point identifiers. Thus, new identifier-locator splitting protocols can be implemented on UDP, instead of requiring IANA to appoint unique protocol numbers for them, e.g., for HIP[28] or for other IETF [19] protocols.

[4]Enlarging the size of the SPI value to something like 128, 196 or 256 bits would make collisions at middleboxes quite unlikely, if the values are selected randomly. This would allow state identification only based on the SPI without the need for using an IP address in combination with the SPI. A similar effect could be accomplished when combining the SPI with the end-point identifier that were to be carried inside the header.

[5]Only an administrator, or a corresponding user with equal rights, is able to register a daemon. If an attacker manages to act as an administrator, it can modify the Security Policy and Security Association Database (SPD and SAD) in an arbitrary way. Thus, our approach does not change the current IPsec security level at the end host.

4.4 Overlay Routing Topology

The overlay routing topology can be divided into *symmetric* and *asymmetric* communication path cases. The symmetric communication paths support transparent trigger registration (Section 4.1.1). The asymmetric communication paths require explicit trigger registration.

4.4.1 Symmetric Communication Path

Figure 6 illustrates a partial key-exchange through a SPINAT device located on the symmetric communication path [6]. The figure contains only the SPI exchange part. Each message includes Alice's and Bob's end-point identifiers (EIDs).

1. Alice selects an SPI$_{Alice}$ value for Bob which is subsequently used by Bob for creating the IPsec protected packets towards Alice[21]. Alice establishes an SA for incoming traffic using this SPI value. The SPINAT device establishes a state for the traffic flow between Alice and Bob to allow the SPINAT to perform proper packet forwarding:

$$< EID_{Alice}, SPI_{Alice}, IP_{Alice} >$$

2. Once Bob receives the message, he establishes SAs for both incoming and outgoing traffic. He uses the received SPI$_{Alice}$ for outgoing traffic and selects an SPI$_{Bob}$ for the incoming traffic. Bob sends a reply containing the SPI$_{Bob}$ value to the SPINAT device.

3. The SPINAT device finalizes its association and forwards the message to Alice.

4. Finally, Alice names her outgoing SA with the SPI$_{Bob}$.

5. Shortly, Jill initializes another key-exchange with Bob. Unfortunately, Jill selects the same SPI value for Bob as Alice did (SPI$_{Alice}$ = SPI$_{Jill}$). The SPINAT device has to replace the value with SPI$_{Jill\#2}$ value and establishes the following three-tuple:

[6]Please note that the symmetric communication path refers to the forwarding path related to the SPINAT devices.

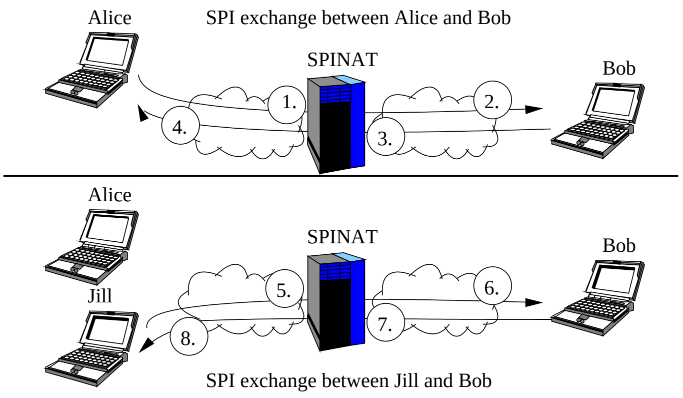
Alice
SPI exchange between Alice and Bob
SPINAT
Bob
1.
2.
3.
4.
Alice
SPINAT
Bob
Jill
5.
6.
7.
8.
SPI exchange between Jill and Bob

Figure 6. SPI translation walk through in the *symmetric* communication path case.

$$< EID_{Jill}, SPI_{Jill\#2}, IP_{Jill} >$$

Now, Jill's incoming SA contains different SPI$_{Jill}$ value than the corresponding outgoing SA at Bob's host.

6. Bob names the outgoing SA with SPI$_{Jill\#2}$ and selects an unique SPI$_{Bob\#2}$ value for Jill.

7. Now, the SPINAT device is able to multiplex traffic by replacing the incoming SPI$_{Jill\#2}$ with SPI$_{Jill}$ before forwarding the IPsec packets to Jill.

8. Finally, Jill and Alice use the same SPI value to name their incoming SAs. However, Bob has different SPI values for the outgoing SAs.

4.4.2 Asymmetric Communication Path

The asymmetric communication paths involve changes to the previous symmetric SPI exchange design. The SPINAT devices are not able to learn the incoming SPI values during one round-trip, because each host selects an SPI value for its peer. Basically, there are three different design choices to solve the problem from the SPINAT point of view.

The first alternative is to enlarge the size of the SPI namespace to avoid collisions at SPINAT devices. The drawback is that the approach essentially changes the semantics of the current IPsec and enlarges the packet size (Section 4.3). The second way to solve the problem is to design a protocol for exchanging end-point identifier and SPI information between SPINAT devices inside an administrative domain. Once a SPINAT device receives an outgoing key exchange message, it propagates the host identifier and SPI information to other SPINAT devices at the edge of the same domain. However, this kind of approach does not scale well in an overlay routing infrastructure where SPINAT devices belong to different administrative domains.

The third alternative is to include a two-round trip SPI exchange in the IPsec control plane signaling. Figure 7 illustrates a case, when Alice and Bob communicates via asymmetric paths. Before the communication happens, Alice and Bob have registered their triggers to SPINAT$_{Alice}$ and SPINAT$_{Bob}$ to receive traffic. Additionally, Alice and Bob may be in private networks behind transparent SPINAT devices. However, to keep the Figure 7 clean we haven't included the transparent SPINAT devices in the picture.

1. Once Bob receives an SPI$_{A1}$ value from Alice, he selects SPI$_{B1}$ value for Alice. Bob includes both of the SPI values to the reply message.

2. The SPINAT$_{Alice}$ device establishes a state and optionally translates the incoming SPI$_{A1}$ to SPI$_{A2}$.

3. Alice includes the translated SPI$_{A2}$ and the received SPI$_{B1}$ to the third message.

4. SPINAT$_{Bob}$ optionally translates SPI$_{B1}$ to SPI$_{B2}$ and establishes a state for the translation.

5. Bob establishes Security Associations using the SPI$_{B1}$ and SPI$_{A2}$ values. He also forwards the translated SPI$_{B2}$ value to Alice.

6. Finally, Alice names her incoming SA with SPI$_{A1}$ and outgoing SA with SPI$_{B2}$. Section 5.2 contains security considerations concerning the protocol design issues.

5 Security Considerations of SPINAT

If we do not protect the SPI values with signatures (section 4.2.2), a MitM attacker may change the SPIs on the fly. However, the SPI is only an index to a specific Security Association at the receiving party. The actual security is based on the shared session keys. All that is needed is that the

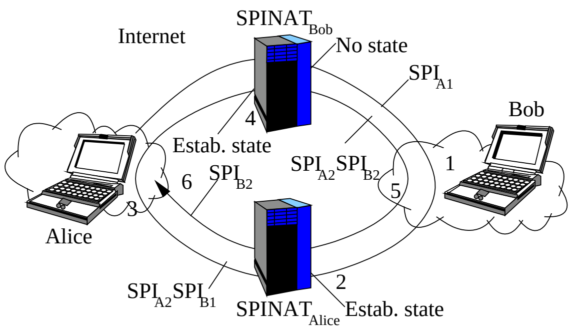

Figure 7. SPI exchange in the *asymmetric* communication path case.

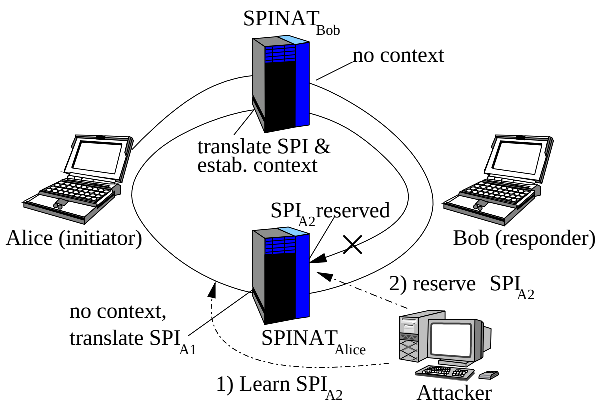

Figure 8. If the initiator's SPINAT device does not establish a context during the 1^{st} round-trip the exchange is vulnerable for the illustrated DoS attack.

peers are able to map an SPI value to the correct IPsec SA. Hence, an SPI changing attack does not affect the confidentiality or integrity properties of the protocol.

It must be noted that changing SPI values is only possible for an on-path attacker that is able to modify packets on the fly. Such an attacker is not only able to change the SPI values, but he can block all communications between the parties. Hence, having unsigned and changeable SPIs does not really introduce any new security vulnerabilities. A host trusts a SPINAT device to change the SPI values in the same way it trusts the NAT device to change the IP addresses.

5.1 Transparent Trigger Registration

A potential attack that we have to consider is a so called *running ahead attack*. To launch this attack, an attacker has to first listen to traffic and find out the end-point identifier a mobile host is using for some of its connections. Once the attacker has acquired the end-point identifiers, it anticipates the movements of the mobile host, and moves ahead of it. Using the known end-point identifiers, the attacker can try to register a trigger at a SPINAT using the the sniffed information.

To protect from this kind of attack the on-path SPINAT device may release the created soft state if the responder does not reply to the attacker's trigger message. Typically, the responder silently drops incorrect IPsec control messages. Another way to solve the identity theft problem is to use public key based end point identifiers. The transparent SPINAT device verifies the ownership of the identifiers with signatures.

However, when the key exchange protocol supports identity protection it is impossible for a MitM, like for a SPINAT device, to learn public keys and verify signatures. In such a case, the mobile host should use one-time random end-point identifiers (see BLIND[41]) [7]. This prevents the eavesdropper to implement the running ahead attack, because the mobile host changes its end-point identifier every time it negotiates a new SA with some of its peers. However, random end-point identifiers cannot be used to update a state without binding them to a cryptographical namespace, e.g. to hash chains [42] or to anonymous (temporary) public keys[41].

5.2 Communication Context establishment

Another problem related to the communication context establishment is to bind SPI values to correct end-point identifiers. Depending on the SPI exchange design, the end-point SPI values may be revealed to outsiders before the final context establishment. This opens a possibility for MitM attackers to reserve SPI values allocated to other hosts if subsequent SPI exchange messages are not cryptographically bound together. Fundamentally, the security issues related to the communication context establishment are similar to the ones in micro-mobility architectures[10].

To protect the responder from DoS attacks and avoid eavesdroppers to reserve SPI values at SPINAT devices, we have to consider the actual protocol design (Figure 8). The SPI exchange for asymmetric communication paths was presented in Section 4.4.2. Bob's SPINAT$_{Bob}$ device should not establish a context during the first round-trip, before Bob has accepted to receive traffic from Alice. Because of Alice initializes the communication, she is obviously willing to receive packets from Bob. Thus, she may directly reveal her private trigger [34], i.e. registered to SPINAT$_{Alice}$, to Bob. In the initiator's case, the SPINAT$_{Alice}$ device

[7]BLIND is an example of a protocol that offers mutual identity protection for end-points, and allows them still to authenticate each other using their public identities.

should establish a context during the first round-trip to prevent an attacker to learn and establish a context using the translated SPI$_{A2}$ value (see Section 4.4.2). Otherwise, the attacker may cause a DoS situation as illustrated in Figure 8.

5.3 Communication Context Update

The SPINAT device takes care of SPI collisions and prevents allocating the same SPI value to two hosts. SPI values, like port values are local identifiers at end-host. If an attacker manages to update other host's SPI binding at a SPINAT device, only the target host suffers from the DoS situation. However, unverified address binding updates open several security vulnerabilities. A malicious node can cause packets to be delivered to a wrong address. This can cause DoS both at the communicating parties and at the address that receives the unwanted packets [4]. Thus, the SPINAT device must verify that the binding updates come from the authentic mobile host. However, the SPINAT device cannot verify the validity of the updates without a secure binding between the IP addresses and the host. In other words, the SPINAT device and the peer nodes need evidence that the IP address belongs to the specific mobile node. On the other hand, mobile hosts have to verify messages sent by SPINAT devices to protect from MitM attackers. Moreover, the peers cannot trust SA update messages without making an end-to-end reachability test whenever a mobile host arrives to a new SPINAT region.

Mobile hosts may use existing security associations with their trusted SPINAT devices to update bindings. However, if an end-host does not have an explicitly established SA with a transparent SPINAT device, the device must verify the bindings updates with signatures or using weak authentication techniques.

The identity protection is not the only reason to replace the public key authentication with weak authentication techniques. Since, signature verification is a time consuming operation, it is a security problem in heavily loaded SPINAT devices. An attacker may send a storm of address and SPI binding update packets and cause a DoS attack by increasing the CPU load at a SPINAT device. This favours lightweight Lamport one-way hash chains and secret splitting authentication techniques to update the bindings at SPINATs.

For example, Ylitalo et.al. present an appropriate trust model in [39]. In their approach, the trust between a middle-box and an end-host inherits from the existing trust relationship between the end host and its trusted rendezvous server. The presented protocol allows dynamic binding updates at transparent middle-boxes using relatively weak, but adequate authentication techniques.

6 Experiment Results

We have implemented the transparent SPINAT device on the FreeBSD 5.2 operating system. Host Identity Protocol (HIP)[28] was selected for the IPsec control plane signaling protocol for two reasons. First, HIP is an identifier-locator splitting architecture consisting of SA establishment and update signaling. Second, the SA establishment signaling, also known as HIP base exchange, contains end-point identifiers and plain SPI values.

The actual SPINAT functionality was implemented as a user space daemon. The daemon dynamically establishes a communication context for end-point identifiers during the base exchange. When an IPsec protected payload packet arrives to the SPINAT device, a firewall rule diverts the packet to the SPINAT daemon. The daemon makes the required translation and forwards the packet to the destination. Although, the current version of our implementation does not support SPI translation. The performance results of our SPINAT implementation, compared to legacy Freebsd IPv6 router, are presented in Figure 9 [8].

We measured the HIP key exchange delay, end-to-end throughput and Round-Trip Time (RTT) through the FreeBSD IPv6 router and our SPINAT device [9]. The SPINAT context establishment delay was really small and therefore it is not visible in Figure 9 c). Negotiating the HIP key exchange through IPv6 router and SPINAT device took in average 0.2878 seconds in both cases.

Figure 9 a) presents average end-to-end RTTs measured with plain ICMP packets through a router and with IPsec ESP protected ICMP packets through a router and a SPINAT device. The IPsec handling at end-hosts caused a 0.24 ms (52%) increase in RTT compared to RTT of the plain traffic. Furthermore, the SPINAT device caused a 0.07 ms (10%) increase in RTT compared to the legacy routed IPsec ESP protected traffic.

Figure 9 b) illustrates average end-to-end throughput via the router and the SPINAT device. The IPSec handling at end-hosts decreases the throughput 32.86 Mbits/sec (52%), i.e., roughly half of the plain traffic throughput. However, in the SPINAT device case the IPsec ESP traffic throughput is only 0.01 Mbits/sec (0.03%) slower than in the router case. Based on the presented experiment results we can argue that SPINAT devices are not bottle-necks in the architecture.

[8]The initiator was running the HIP protocol on a laptop equipped with Intel Pentium M 1.6GHz processor, while the responder was running HIP on AMD Athlon XP 2200+ (1.8GHz) processor. Each middle-box was equipped with a Mobile P4 2.26 Ghz processor. All computers contained 512 MB RAM and the 100Mbits/sec ethernet interfaces were directly connected to each other with cross switched Cat-5 cables.

[9]As the efficiency of a context state search algorithm has a strong effect on results, we decided to use a simple O(1) hash table search algorithm. The rehashing events are not included into results.

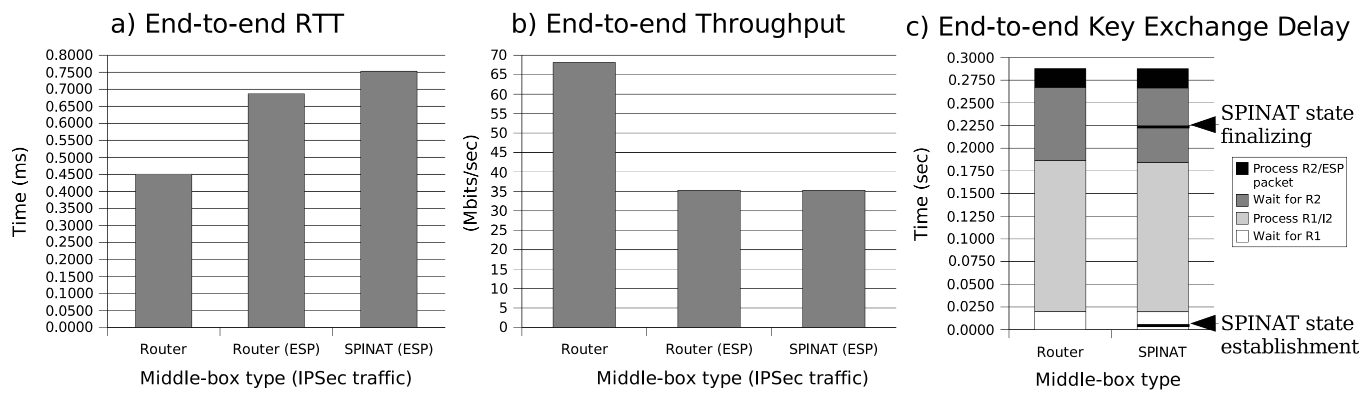

Figure 9. End-to-end round-trip time, throughput and HIP key-exchange delays through a IPv6 router and a SPINAT device.

7 Conclusions

In this paper, we have presented an approach that integrates IPsec into overlay routing. The solution requires small changes to key exchange protocols and optionally also to IPsec ESP integrity protection computation. The new functionalities are not currently supported in the IETF standards. On the other hand, the presented overlay architectures are neither yet standardized. Therefore, the required changes in IPsec can be gradually applied in the new overlay network infrastuctures.

The presented changes do not alter the existing security level of IPsec, but they make it possible to implement simultaneously scalable and secure overlay routing architectures. Our solution is based on the observation that middle-boxes in the overlay routing implement Network Address Translation (NAT) functionality. Therefore, to mitigate the problems related to the existing NAT practice we have presented SPI multiplexed NAT (SPINAT). The SPINAT device supports address and SPI translation for IPsec protected payload traffic even in the asymmetric communication path case. The SPINAT functionality can be integrated into middle-boxes in any overlay routing architecture, including the i^3 nodes in the Internet Indirection Infrastructure.

References

[1] B. Aboba and W. Dixon. IPsec-Network Address Translation (NAT) Compatibility Requirements. RFC 3715, Mar. 2004.

[2] D. Adkins, K. Lakshminarayanan, A. Perrig, and I. Stoica. Towards a More Functional and Secure Network Infrastructure. Technical Report UCB Tech. Rep. UCB/CSD-03-1242, Univ. California, Berkeley, 2003.

[3] W. Aiello, S. Bellovin, M. Blaze, J. Ioannidis, O. Reingold, R. Canetti, and A. Keromytis. Efficient, DoS-resistant, secure key exchange for internet protocols. In *Proc. of the 9th ACM conference on Computer and communications security*, pages 48–58, Washington, DC, USA, 2002.

[4] T. Aura, M. Roe, and J. Arkko. Security of Internet Location Management. In *Proc. of the 18th Annual Computer Security Applications Conference*, Las Vegas, USA, Dec. 2002.

[5] F. Baker. Requirements for IP Version 4 Routers. RFC 1812, June 1995.

[6] H. Balakrishnan, K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica, and M. Walfish. A Layered Naming Architecture for the Internet. In *Proc. of the ACM SIGCOMM'04 Conference*, Portland, OR, USA, Sept. 2004.

[7] A. Biggadike, D. Ferullo, G. Wilson, and A. Perrig. NATBLASTER: Establishing TCP Connections Between Hosts Behind NATs. In *Proc. of the SIGCOMM'05 Asia Workshop*, Beijing, China, 2005.

[8] M. Castro, P. Druschel, A. J. Ganesh, A. I. T. Rowstron, and D. S. Wallach. Secure Routing for Structured Peer-to-Peer Overlay Networks. In *Proc. of the 5th Symposium on Operating Systems Design and Implementation (OSDI'02)*, Boston, MA, USA, Dec. 2002.

[9] D. Clark, R. Braden, A. Falk, and V. Pingali. FARA: Reorganizing the Addressing Architecture. In *Proc. of the ACM SIGCOMM'03 Workshop on Future directions in network architecture*, pages 313–321, Karlsruhe, Germany, Aug. 2003.

[10] P. Eardley, M. Mihailovic, and T. Suihko. A Framework for the Evaluation of IP Mobility Protocols. In *Proc. of the 11th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'00)*, London, UK, Sept. 2000.

[11] J. Eriksson, M. Faloutsos, and S. Krishnamurthy. PeerNet: Pushing Peer-to-Peer Down the Stack. In *Proc. of the Peer-to-Peer Systems II, Second International Workshop (IPTPS'03)*, pages 268–277, Berkeley, CA, USA, Feb. 2003.

[12] B. Ford. Unmanaged Internet Protocol: taming the edge network management crisis. *Computer Communication Review*, 34(1):93–98, 2004.

[13] P. Francis and R. Gummadi. IPNL: A NAT-extended Internet architecture. *ACM SIGCOMM Computer Communication Review*, 31(4):69–80, Oct. 2001.

[14] S. Guha, Y. Takeda, and P. Francis. NUTSS: A SIP based Approach to UDP and TCP Network Connectivity. In *Proc. of the SIGCOMM'04 Workshops*, Portland, Oregon, USA, Aug. 2004.

[15] K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica. The Impact of DHT Routing Geometry on Resilience and Proximity. In *Proc. of the SIGCOMM'03*, Karlsruhe, Germany, Aug. 2003.

[16] M. Holdrege and P. Srisures. Protocol Complications with the IP Network Address Translator. RFC 3027, Jan. 2001.

[17] Y. C. Hu, S. M. Das, and H. Pucha. Exploiting the Synergy between Peer-to-Peer and Mobile Ad Hoc Networks. In *Proc. of the Ninth Workshop on Hot Topics in Operating Systems (HotOS-IX)*, Lihue, Kauai, Hawaii, USA, May 2003.

[18] C. Huitema. Teredo: Tunneling IPv6 over UDP through NATs. Internet-Draft, work in progress, Jan. 2005.

[19] G. Huston. Architectural Approaches to Multi-Homing for IPv6. Internet-Draft, work in progress, Feb. 2005.

[20] A. Huttunen, B. Swander, V. Volpe, L. DiBurro, and M. Stenberg. UDP Encapsulation of IPsec ESP Packets. RFC 3948, Jan. 2005.

[21] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. RFC 2401, Nov. 1998.

[22] D. McDonald, C. Metz, and B. Phan. PF_KEY Key Management API, Version 2. RFC 2367, July 1998.

[23] K. Moore. Things that NATs break. Unpublished note, Oct. 2003.

[24] P. Nikander. Denial-of-Service, Address Ownership, and Early Authentication in the IPv6 World. In *Proc. of the 9th International Workshop on Security Protocols*, pages 12–26, Cambridge, UK, Apr. 2001.

[25] P. Nikander and J. Arkko. Delegation of Signalling Rights. In *Proc. of the 10th International Workshop on Security Protocols*, pages 203–212, Cambridge, UK, Apr. 2002.

[26] P. Nikander, J. Arkko, and B. Ohlman. Host Indentity Indirection Infrastructure (Hi3). In *Proc. of the Second Swedish National Computer Networking Workshop (SNCNW'04)*, Karlstad, Sweden, Nov. 2004.

[27] P. Nikander and J. Melen. A Bound End-to-End Tunnel (BEET) mode for ESP. (Expired) Internet-Draft, work in progress, June 2004.

[28] P. Nikander, J. Ylitalo, and J. Wall. Integrating security, mobility, and multi-homing in a hip way. In *Proc. of the NDSS'03*, pages 87–99, San Diego, CA, USA, Feb. 2003.

[29] J. Rosenberg, R. Mahy, and C. Huitema. Traversal Using Relay NAT (TURN). Internet-Draft, work in progress, Feb. 2005.

[30] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs). RFC 3489, Mar. 2003.

[31] J. Saltzer, D. Reed, and D. Clark. End-To-End Arguments in System Design. *ACM Transactions on Computer Systems*, 2(4):277–288, Nov. 1984.

[32] P. Srisuresh and M. Holdrege. IP Network Address Translator (NAT) Terminology and Considerations. RFC 2663, Aug. 1999.

[33] A. Stavrou, A. D. Keromytis, J. Nieh, V. Misra, and D. Rubenstein. MOVE: An End-to-End Solution to Network Denial of Service. In *Proc. of the NDSS'05*, pages 81–95, San Diego, CA, USA, Feb. 2005.

[34] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. Internet Indirection Infrastructure. In *Proc. of the ACM SIGCOMM'02*, pages 73–88, Pittsburgh, PA, USA, Aug. 2002.

[35] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. In *Proc. of the ACM SIGCOMM'01*, pages 149–160, San Diego, CA, USA, Aug. 2001.

[36] J. D. Touch and V. K. Pingali. DataRouter: A Network-Layer Service for Application-Layer Forwarding. In *Proc. of the International Workshop on Active Networks (IWAN'03)*, Kyoto, Japan, Dec. 2003.

[37] M. Walfish, J. Stribling, M. Krohn, H. Balakrishnan, R. Morris, and S. Shenker. Middleboxes no longer considered harmful. In *Proc. of the USENIX OSDI*, San Francisco, CA, USA, Dec. 2004.

[38] J. Ylitalo, P. Jokela, J. Wall, and P. Nikander. End-point Identifiers in Secure Multi-Homed Mobility. In *Proc. of the 6th International Conference On Principles Of DIstributed Systems (OPODIS'02)*, pages 17–28, France, Dec. 2002.

[39] J. Ylitalo, J. Melén, P. Nikander, and V. Torvinen. Rethinking Security in IP based Micro-Mobility. In *Proc. of the 7th Information Security Conference (ICS'04)*, pages 318–329, Palo Alto, CA, USA, Sept. 2004.

[40] J. Ylitalo and P. 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*, pages 435–441, Barcelona, Spain, Feb. 2004.

[41] J. Ylitalo and P. Nikander. BLIND: A Complete Identity Protection Framework for End-points. In *Proc. of the Twelfth International Workshop on Security Protocols*, Cambridge, UK, Apr. 2004.

[42] J. Ylitalo, V. Torvinen, and E. Nordmark. Weak Identifier Multihoming Protocol (WIMP). (expired) Internet-Draft, work in progress, Jan. 2004.

[43] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz. Tapestry: A Resilient Global-Scale Overlay for Service Deployment. *IEEE Jornal on selected areas in communications*, 22(1):41–53, Jan. 2004.

[44] S. Zhuang, K. Lai, I. Stoica, R. H. Katz, and S. Shenker. Host Mobility Using an Internet Indirection Infrastructure. In *Proc. of the First International Conference on Mobile Systems, Applications, and Services (MobiSys'03)*, San Francisco, CA, USA, May 2003.