BLIND: A Complete Identity Protection Framework for End-points

Jukka Ylitalo and Pekka Nikander

Ericsson Research NomadicLab,
02420 Jorvas, Finland.
{Jukka.Ylitalo,Pekka.Nikander}@nomadiclab.com

Abstract. In this paper, we present a security framework that provides identity protection against active and passive attacks for end-points. The framework is based on a two-round-trip authenticated Diffie-Hellman key exchange protocol that identifies the end-points to each other and creates a security association between the peers. The protocol hides the public key based identifiers from attackers and eavesdroppers by blinding the identifiers. We complete the identity protection by offering location privacy with forwarding agents. To our knowledge, our privacy enhanced protocol is the first denial-of-service resistant two-round-trip key exchange protocol that offers identity protection for both communicating peers.

1 Introduction

The current structure of the Internet is very much the same as if a person's name would be defined by his or her current location. Let's say that a spy moves from Moonlight Street to Shadow Street. Since identification is bound to locations, his *identifier* changes due to movements, but his actual *identity* stays the same. M, who used to know his spy, James Blind[1], as Mr. Ten Moonlight Street, does not recognize him anymore. Now being identified as Mr. Twelve Shadow Street, he must convince M, in one way or another, about his actual identity, i.e, he still is the same person as he was before. Since there is no equivalent of the human face in the current Internet, convincing M is not a particularly easy task.

Several mobility protocols have solved the naming problem by using *home addresses*[2]. Each end-point is assigned a static address, its home address, which is used to identify the end-point independent of its location. This solves the basic naming problem; our spy is still known to his boss M as Mr. Ten Moonlight Street. However, even in the real world, the actual location is needed for reachability. It would be really hard for our spy, usually living in a hotel at Moonlight Street and currently walking at 12 Shadow Street, to prove to M that his name

[1] The name corresponds the Fully Qualified Domain Name (FQDN).

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

is Mr. Ten Moonlight Street c/o Twelve Shadow Street, without assurances. By sending a challenge message to a given location and waiting for a response, M can check if there is a spy named with the given location.[3]

Unfortunately, the existing naming convention causes many privacy problems that are related to location names. The presence of location information in a message reveals the location of its recipient and alledged sender[2]. At the same time, location names are used to identify end-point, and thereby allow an end-point's action to be traced. Our spy would definitely not want his identity nor location to be revealed to outsiders [4].

Our framework uses a cryptographic name space, based on the use of public keys, that separates the location and end-point identifier roles of location names. Continuing our analogy, the approach would bring our spy a genuine name, one cryptographically bound to his real identity. In a way, our Mr. Ten Moonlight Street would no longer be named after his location, but by a self-signed photograph of his face. However, the new naming convention causes privacy problems that are no longer related to location names, but the use of public keys[4][5]. In this paper, we focus on solving that problem [5].

We introduce a privacy enhanced authenticated Diffie-Hellman protocol. The protocol provides complete identity protection, requiring an initiating party only to possess a hash of the full public key of its peer at the start of the protocol run. Basically, the protocol scrambles the photograph of our spy in a way that only his old acquintances are able to recognize him. The protocol protects both parties from *passive and active Man-in-the-Middle and polling attacks*, unless the attacker is able to find the public key with the help of location information. Therefore, we complete our protocol by presenting forwarding agents that provide location privacy for end-points.

The rest of this paper is organized as follows. Section 2 defines the framework terminology. In Section 3, we define the privacy problems that we are addressing in this paper. This description is followed by a detailed problem statement in Section 4. In Section 5, we present the privacy protecting key exchange protocol. The forwarding agent and location privacy is discussed in Sections 6 and 7. Section 8 concludes the paper.

2 Framework Terminology

In our framework, a logical *end-point* is a participant in an end-to-end communication[6]. Each end-point generates a public key pair that works as a global

[3] In the Mobile IPv6[1] terminology, this is called Return Routability (RR) test.

[4] The current IP mobility practices[3][1] reveal both the end-point's identity (home address) and location (care-of-address) to outsiders. In the case of Mobile IPv6 route optimization, the identity and location also to the servers and peers the end-point is communicating with.

[5] The location and identity privacy problems can be solved by using privacy proxies, such as one provided by Zero Knowledge systems. Such usage necessitates that the proxy is trusted to keep the user's identity and location secret. However, using generic privacy proxies falls beyond the scope of this paper.

End-point Identifier (EID) [6]. The owner of the private key owns a specific *identity*, while the corresponding public key works as an identifier for the end-point. This defines the naming trust relationship between an identity and an identifier without name certificates. A cryptographic hash of the public key (EID) is called a *fingerprint*. The fingerprint represents a consistent format for protocols, independent of the whatever public key technology is used.

A *host* is an environment for an end-point(s). The end-points use an Application Programming Interface (API) to communicate with other end-points. *Transport Layer Identifiers (TLIs)* represent the EIDs in the communication API and at the transport layer. The local TLI presentation depends on the instantiation of our framework.

A *location name* (i.e. a locator) defines the topological point-of-attachment of an end-point in the network. A multi-homed host offers several topological point-of-attachments for end-points. The introduction of EIDs clarifies the role of locators. For example, IP addresses become pure topological labels, naming locations in the Internet, while the EID identify an end-point. The location names are bound dynamically to EIDs. Furthermore, a *connection* is a communication link between two end-points. It is bound to EIDs, instead of location names. An end-point may change its location without breaking connections.

The *key exchange protocol* uses the EIDs for mutual authentication and to generate end-to-end security associations (SAs) between two end-points. The end-point that initiates a protocol run is called the *initiator* and the responding end-point is called the *responder*. The security associations are used to protect the connections. In addition, the SAs are also used with *mobility management protocol* (out of the paper's scope). The mobility management protocol is used to update the binding, at middle boxes and peer nodes, between EIDs and location names. The key exchange must take place before end-points can update their address bindings.

Our framework contains a logical protocol layer between the OSI transport and networking layers. In the current Internet TCP/IP architecture, the processes are bound to transport layer sockets, and the sockets are identified using IP addresses and ports. In our framework the connections are no longer named with locators but with EIDs. The new abstract end-point identifier layer translates the TLIs to locators. The set of associated locators can belong the different families. This binding, between EIDs and locators, is simultaneously dynamic and one-to-many, providing for mobility and multi-homing, respectively. In the rest of this paper, we discuss the framework from the Internet architecture point of view.

[6] For example, Cryptographic Generated Address (CGA) [7], Host Identity Protocol (HIP) HI[8]

3 Privacy in IP Based Communications

In the current Internet, IP addresses are used to identify end-points and name their topological locations. IP addresses together with public keys, used in the key exchange protocols, reveal directly the location and identity of an end-point.

3.1 Public keys as a privacy problem

Using public keys as identifiers is a source of privacy problems. Firstly, a public key directly and strongly identifies an end-point[9]. Secondly, if an end-point has just a single public key, using it repeatedly, it is fairly easy to link together all the transactions made by the end-point. However, an end-point may have several public keys instead of just one. Some of the public keys can be used as more permanent identifiers, allowing the end-point to be recognized. At the same time, some other keys can be anonymous, being temporary and periodically replaced.

It is important to make a difference between anonymity and identity protection. If an end-point uses an unencrypted identifier, it deliberately reveals its identity to outsiders, breaking identity protection. On the other hand, one can openly use an anonymous public key and remain anonymous.

In identity protection, one of the goals is to prevent malicious nodes from tracing any identity. Therefore, if we are able to offer complete identity protection for any type of identities, public or anonymous, the role of anonymous identities is changed. They are no longer needed to protect from man-in-the-middle or eavesdropping attackers but from legitimate peers.

3.2 IP addresses as a privacy problem

To keep the size of routing tables small enough, the Internet addresses are distributed hierarchically [10]. That is, address prefixes and network topology are kept in rough synchrony, thereby allowing the routers to store less information than they otherwise would be forced to. At the same time, this practice binds the IP addresses to the topological locations in the network.

While the primary purpose of IP addresses is to make packet delivery possible, they are also used directly by the transport layer protocols, including TCP, UDP, and SCTP. In all of these, IP addresses are used for naming the transport layer sockets. That is, each communication context is named by IP addresses together with protocol and port numbers. This necessitates using static IP addresses, or connections will break.

As long as a user is using a static IP address, it is possible to link her actions together and form a profile about her. With some little help, it is often even possible to link this profile to her real life identity[4][11][7].

The address tracking and profiling problem is slightly mitigated by the current practice of using dynamic IP addresses and especially Network Address

[7] Other techniques for user tracking, such as HTTP cookies, fall beyond the scope of this paper.

Translation (NAT)[12]. However, with IPv6, the privacy situation is likely to detoriate since NAT is less likely to be used. Furthermore, even if NAT is used in IPv6, the translation will typically be one-to-one, without multiplexing several hosts behind a single address.

3.3 Our position

The identity and location privacy problems are related in the sense that identity privacy does not protect a node against all kind of threats. In some cases, the location may reveal confidential information to a malicious person. E.g. someone is in the bank vault or someone is alone in the park. In such case, the attacker does not care of the actual identity of the end-point. On the other hand, IP addresses that are stored in the DNS together with public keys give information about end-points identity. Therefore, an end-point having a DNS record must protect its location together with its identity.

In our framework, cryptographic end-point identifiers are used to identify the communication end-points. They have no permanent relationship with locations or IP addresses. IP addresses, on the other hand, are used to identify only the topological locations, not end-points. We will show how it is possible to fully hide the used cryptographic identifiers from outsiders and integrate them with Network Address Translation (NAT). As a consequence we obtain a security framework where an end-point can control both its identity and location privacy.

4 Problem statement

As pointed out by Molina-Jimenez and Marshall[11], if it becomes possible to use random IP and link layer addresses, the problems related to IP address tracking more or less disappear. To be more precise, if an IP address no longer acts as an end-point identifier (see Section2), it is possible to take advantage of address translation at end-hosts and middle boxes (discussed later). As a consequence, the fact that it remains possible to keep track of IP addresses and find out their geographical location doesn't matter that much any more. The focus is moved to the end-point identifiers, public keys and fingerprints, that must be protected.

For the purposes of the rest of this paper, we define *end-to-end identity privacy* to denote the situation where any given two end-points are able to communicate, using their public identities, without having to disclose the identities to outsiders.

4.1 Identity protection

The identity protection problem arises in all two-round-trip Diffie-Hellman key-exchange protocols that use separate public-keys for mutual authentication. [8]

[8] Protocols based on public-key encryption (e.g.[15]) are vulnerable to CPU related DoS attacks and are therefore out of the paper's scope.

To obtain DoS protection, the responder must not generate the Diffie-Hellman shared secret before the initiator. Therefore, the responder must defer key generation until it has received two messages from the initiator. Consequently, if the responder sends its public key in the second protocol message, the public key must be transmitted in clear. Thus, the responder's public key can be safely transmitted only in the last (fourth) protocol message. As a result, the initiator can completely authenticate the responder only once it has received the last message.

The initiator, in turn, is able to generate the shared secret after receiving the second protocol message, i.e., the first message sent by the responder. Hence, the third protocol message may contain the initiator's public key in an encrypted form. In any typical Diffie-Hellman protocol, an active attacker can easily find out the initiator's public key, because the second message cannot be fully authenticated.

One way to solve the problem is to enhance the session key generation with a secret that is initially known only to the authentic end-points. Incidentally, the same method also protects the responder from active attacks. In other words, even an active attacker is not able to find out the responder's identity by sending spoofed first messages, i.e., by launching polling attacks [13]. In other words, all identifiers, fingerprints included, must be hidden in a cryptographically strong way to obtain identity privacy.

4.2 Privacy protection vs. denial-of-service protection

The design of any public key based key exchange protocol will eventually face a trade-off between denial-of-service and identity protection properties. For example, Bellovin et.al. [5] stated: *"We remark that it is essentially impossible, under current technology assumptions, to have a two-round-trip protocol that provides DoS protection for the responder, passive identity protection for both parties, and active identity protection for the initiator."*

In this paper, we present how the introduction of a new cryptographic name space can help to mitigate the problem. The new cryptographic end-point identifiers, together with the techniques discussed in this paper, make it possible to obtain complete privacy protection from both passive and active attacks for both end-points, in a two-round-trip DoS-resistant protocol (See Section 5.2).

To be more precise, if an attacker is able to learn the public key of the participants through some external means, e.g., if the public key can be resolved from reverse DNS, it remains impossible to provide active identity protection. However, whenever the attacker cannot use the IP addresses to find out the public keys, the public keys remain private even from active attackers.

4.3 Three scenarios

Identity privacy can be divided into three scenarios, which differ on the a priori knowledge of the participating parties. In the first scenario, both of the parties

have a priori knowledge about each other's identifiers, e.g. fingerprints, but possibly not the actual public keys themselves. For example, the parties have been in touch with each other earlier and remember the fingerprints but not the public keys. In the second scenario, the initiator knows the the responder's identity, but the responder may have no knowledge about the initiator's identity. In the third scenario, neither of the parties know each other's identity beforehand. In this case, the parties may use temporary identities, i.e., short term public key pairs, or public identities, e.g., published in a directory service. [9]

In the first and second scenarios, we are able to protect both identifiers from both passive and active attacks. Typically, the initiator learns the responder's host identity (public key or fingerprint) from a directory service. Additionally, in the first scenario, the responder has some configuration information that contains the fingerprints of potential initiating parties as well. However, neither of the parties may lack access to the full public keys, requiring them to learn the actual public key during the handshake.

In the second scenario, the initiator has a priori knowledge about the responder's identity, but the responder is oblivious of the initiator's identity. However, even in that case, the initiator's identity may be important *afterwards* (e.g., for auditing purposes), and therefore, we want to support the case where the initiator is able to use its public long-term identity without revealing it to anyone else but the right responder.

5 Key Exchange Protocol supporting Identity Protection

In this section, we present a two-round-trip authenticated Diffie-Hellman Key Exchange Protocol that protects the initiator's and responder's identity. Our solution is based on the idea of *blinding* the cryptographic hash (fingerprint) of the public key used in the exchange. That is, instead of directly using the hashes of the public keys to index the session, the parties create scrambled versions of the fingerprints and use each scrambled value only during one protocol run. This makes it impossible to correlate independent protocol runs.

Our blinding method is based on the assumption that at least the initiator has a priori knowledge about the actual or potential fingerprints of the responder (see Section 4.3). The parties must know their own fingerprints, of course.

Before starting our protocol, the initiator computes *blinded fingerprints* for both end-points, using a *nonce* to hide the actual *plain fingerprint*. The blinded fingerprints are generated using the following formula:

$$fingerprint_I^{Blinded} = SHA1(nonce_I || fingerprint_I^{Plain})$$

$$fingerprint_R^{Blinded} = SHA1(nonce_I || fingerprint_R^{Plain})$$

[9] The scenario where a responder would know who is going to contact it, but the parties taking contact do not know the identity of the responder, is not possible in practice.

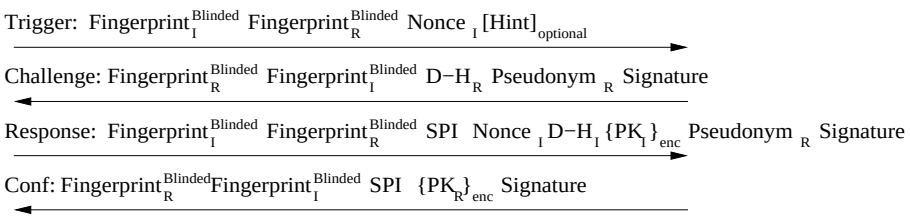

Fig. 1. The key exchange

The initiator generates a fresh nonce for every base exchange. The blinded fingerprints are changed for every key exchange, which makes it hard to trace the usage of plain fingerprints.

In order to derive a plain fingerprint from a blinded one and the nonce, the plain fingerprint must be already known. That is, if a party that knows the nonce has a list of possible plain fingerprints, it can test these, one by one, to see if any of them matches with the blinded fingerprint. This allows the initiator to use the same plain fingerprints all the time between a given pair of end-points, while the blinded fingerprints, used in the key exchange packets on the wire, will be stored only for the life time of one connection. On the other hand, if only the nonce and the blinded fingerprint are known, it remains computationally infeasible to find the plain fingerprint.

5.1 Sending trigger packet

The first packet contains the nonce and corresponding blinded fingerprints. The packet structure is shown in Figure 1. In addition to the nonce, the packet may also contain an optional *hint* that makes it easier for the responder to obtain a correct plain responder fingerprint than by trying out them all. The hint discloses k lowest bits of the plain responder fingerprint.

The idea behind our approach is that the responder finds out its plain fingerprint by repeatedly attempting to generate blinded fingerprints from the plain fingerprints it knows, using the given nonce. A typical responder has only a few public keys at most, and it is able to easily find its own plain fingerprint even without a hint. Thus, in most cases, the initiator does not have to include the hint, and therefore the responder obtains complete identity privacy.

Basically, the hint protects the responder from DoS attacks. A busy server (e.g., a proxy) with thousands of own fingerprints may drop the 1st packet if it does not contain a large enough hint. However, the k value must be so small that the hint does not statistically give enough information to bind it to the plain fingerprint [10].

[10] The framework is based on that assumption that most of the end-points in the Internet will store their own public key identifiers into the directory services. Therefore, once there is any substantial number of public key owners, it is statistically really

The responder does not need to know the actual identifier of the initiator before it receives the 2nd packet from the initiator. Thus, there is no need to use a hint for the plain initiator fingerprint in the trigger packet.

5.2 Sending challenge packet

Upon receiving a trigger packet, the responder sends a temporary pseudonym and starts the Diffie-Hellman exchange in the challenge packet (see Figure 1). The pseudonym is implicitly bound to responder's public key, because the challenge packets are created and signed beforehand. To select a suitable challenge packet, the responder needs to be able to find out the correct plain fingerprint from the blinded one. The responder does not send the public key in the challenge packet, but signs the packet with its private key.

Later when the initiator sends a response packet, it will contain the responder's pseudonym. The pseudonym acts as an index for the responder's public key. In this way, the responder does not need to resolve its plain fingerprint twice. The pseudonym is a local random number. It does not give any information about the responder's actual identity.

It is possible to include a challenge puzzle (e.g. HIP[8]) to the challenge packet. After solving the puzzle, the initiator sends the result back to the responder in the third message. The puzzle protects the responder from DoS attacks.

5.3 Forming the session keys

Since the challenge packet does not contain the responder's public key, the initiator may not be able to verify at this stage that the packet is signed by the correct responder [11]. In this case, the initiator defers the verification of the signature until it receives the 2^{nd} packet from the responder. If the initiator finds out later that the signature in the challenge packet was invalid, the initiator's identity is still not revealed to passive or active attackers, as we will shortly see.

Basically, it is possible that a malicious node sends a spoofed challenge message to the initiator in trying to disclose the initiator's identity. The initiator may not have the public key of the responder at this point, therefore being unable to verify the signature in the challenge packet, but it knows the responder's correct plain fingerprint. The malicious node, in turn, does not know either of the plain fingerprints. To take advantage of this, the initiator generates the required key material using its own blinded fingerprint and the responder's plain fingerprint, known only to the correct responder:

$$KEY_1 = SHA1(KEY_{DH}||fingerprint_I^{Blinded}||fingerprint_R^{Plain}||1)$$
$$KEY_n = SHA1(KEY_{DH}||KEY_{n-1}||n)$$
$$KEY_{MATERIAL} = KEY_1||...||KEY_n$$

hard to use a small size hint to find the right fingerprint among all of the worlds fingerprints.

[11] The initiator may only have the responder's fingerprint, and not the public key itself.

While any eavesdropper learns the blinded fingerprint, only the correct responder knows the responder's plain fingerprint. Therefore, a malicious node is not able to form the session key even if it has gained access to the Diffie-Hellman key through an active attack.

5.4 Sending and receiving response packet

The initiator uses the key material to encrypt its public key in the response packet (see Figure 1). The response packet includes the same nonce that was sent in the trigger packet and the pseudonym that was sent in the challenge packet.

When the responder receives the response packet, it uses the pseudonym to find out its own public key. The responder generates the key-material in the same way the initiator did earlier, and uses the key-material to decrypt the initiator's public key.

As an additional verification step, the responder computes the initiator's plain fingerprint from the decrypted public key and verifies that the blinded fingerprint is a correct one.

5.5 Sending confirmation packet

At the final stage, the responder sends its encrypted public key in the confirmation packet to the initiator (see Figure 1). The initiator has a state related to the blinded fingerprints, and therefore it is able to easily look up related plain fingerprints and key-material. The initiator decrypts the public key and verifies that the expected responder's plain fingerprint corresponds to the received public key. Finally, the initiator verifies the signatures of the challenge and confirmation packets.

In addition to the presented first and second scenarios, it is possible that neither of the end-points know their peer's public key or fingerprint beforehand. The initiator knows an IP address of the responder, but nothing else. This kind of scenario is vulnerable to certain man-in-the-middle attacks, since there is no security relevant information at the start of the protocol run.

5.6 Security analysis

The first and second scenarios correspond to one-way and two-way authenticated Diffie-Hellman key exchanges. The parties do not need to know the full public keys of their peers before the protocol starts; hashes or other one-way derivatives of the public keys are enough. This is different from earlier identity protecting protocols that require the parties to know the full public keys before the protocol starts.

The blinded fingerprint protects the responder from the polling attack[13], where an attacker impersonates an initiator, since only an initiator knowing the identity of the responder may compute the blinded fingerprint. The only efficient

way for an attacker to learn the responder's identity is to find out the mapping between the IP address and public key. If an attacker is not able to map IP addresses to identifiers the end-points obtain full identity privacy. In Section 7, we present a mechanism that hides the end-point's actual location; offering location privacy.

In the first and second scenarios, a Man-in-the-Middle (MitM) knowing the Identity of the actual responder can solve the responder's plain fingerprint from the blinded one. However, without such a priori knowledge, an attacker must guess the correct fingerprint, which is statistically very difficult. Therefore, the only way for an active MitM attacker to succeed is to map the recipient IP address, in the key exchange packets sent by the initiator, to the identity of the responder. Since the protocol protects the responder from polling attacks, such identity knowledge must be obtained in another way, e.g. from DNS. Therefore, the DNS should not contain reverse look-up table for identifiers.

In the second and third scenarios, the question is about partially or completely *opportunistic* identification, where the initiator or both of the end-points learn their peer's identity during the protocol run. That is, one or both of the parties do not have any a priori knowledge about the identity of their peers. Naturally, the third scenario, where both parties are initially oblivious about their peer's identity, is vulnerable to active attacks.

6 EID enabled Network Address Translation

An EID enabled NAT device translates IP addresses, acting as a router for end-point identifiers. The NAT device associates the connection state with the EIDs. It is able to multiplex several connections on a single address based on the end-point identifiers.

In a preferable instantiation of our framework the existing IP or IPSec packet structures are not changed. As a consequence, the EIDs are not present in the regular traffic between two hosts. They appear only in the key exchange messages. However, at the logical level, the end-point identifier name space imposes changes to the logical packet structure. That is, each packet must logically include both the end-point identifiers and IP addresses of the sender and recipient. If IPSec is used, the Security Parameter Index (SPI) values can be used as *indices*

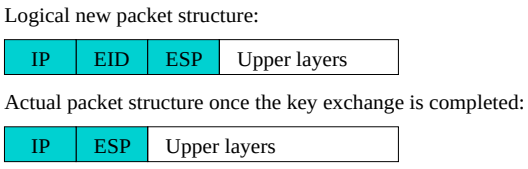

Fig. 2. The packet structures

for end-point identifiers [12] , resulting in packets that are syntactically similar to those used today. This is illustrated in Figure 2.

A NAT device learns the SPIs together with the EIDs during a key exchange. The SPIs and the IP addresses are used together to act as shorthands for the EIDs. The NAT device needs the SPIs to properly demultiplex any packets arriving to a shared IP address. Whenever the packets are integrity protected with ESP, the recipient is always able to verify that a received packet was sent by the peer no matter what the source and destination addresses are.

Basically, SPI multiplexed NAT (SPINAT) works in the same way as port multiplexed NAT (NAPT). If a specific SPI is already in use, the SPINAT device replaces the value with a new one. When there are several NAT devices on the path, all of them may not have the initially assigned SPI value available to be used with the connection. Therefore, it may become necessary to change the SPIs several times along the way. Thus, the SPI values in key exchange messages cannot be encrypted or included into the signature. The SPINAT technique does not require any tunneling headers.

7 Location Privacy with Forwarding Agents

A Forwarding Agent (FA)[14] is functionally similar to an EID enabled NAT device. In fact, functionally the two different devices are equal for all practical purposes. However, a FA is not necessarily located between two different addressing domains (such as the public Internet and a privately addressed intranet), but it may just be conveniently located almost anywhere, even with a single interface[13].

The forwarding agents in our framework, take advantage of the end-point's multi-homing properties. A multi-homed end-point having several IP addresses is considered to be present at several locations at the same time. In functional terms, the end-point is able to receive packets sent to several different IP addresses. On the other hand, if an end-point has leased an IP address from a forwarding agent, the end-point is also able to receive packets sent to the forwarded address. Thus, in a sense, the packet forwarding agent can be considered to dynamically provide a virtual interface to the end-point, and that the end-point is *virtually present* at the location of the forwarding agent. The situation is illustrated in Figure 3.

It must be noted that a forwarding agent always translates only one IP address, and never both the source and destination addresses. When a forwarding agent is passing a packet to an end-point that has leased a virtual interface, only the destination address is changed. What was previously the virtual address now becomes the real address. For packets sent by the leasing end-point, the situation is reversed.

[12] It is also possible to use, e.g., flowid for the same purpose with IPv6.

[13] The FA discovery protocol is out of the paper's scope. However, the discovery can be based on anycast addresses or DNS queries.

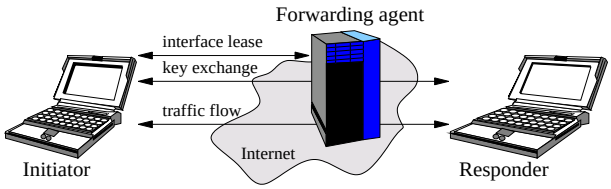

Fig. 3. The virtual interface model

The FA assigns an IP address from a pool of addresses for each virtual interface lease. In the IPv4 world the shortage of public IP addresses forces the FAs to overload IP addresses, just like SPINAT devices do. An end-point must negotiate a key exchange with its peer via the forwarding agent to obtain location privacy. Therefore, an end-point must make a bi-directional lease. The lease consists of following information:

$$(fingerprint_{dst}^{Blinded}, addr_{dst}, fingerprint_{src}^{Blinded}, addr_{src}, lifetime)$$

The forwarding agent provides location privacy by hiding the real location of the node. The peers are able to see only the virtual address, not the real address(es) of the end-point.

7.1 Complete Privacy

From the privacy point of view there are trusted and untrustworthy forwarding agents. Untrustworthy FAs may allow anonymous leases, while trusted FAs may require the initiator to identify itself during the lease. The responder typically has a trust relationship with its long-lived forwarding agent. In such a case, the forwarding agent trusts the responder not to establish extra states and the responder trusts forwarding agent not to disclose the responder's identity.

The initiator has to acquire a virtual interface from a forwarding agent to obtain complete identity privacy. The initiator may negotiate a lease with an untrustworthy forwarding agent using some temporary identifier during the lease. The lease contains a pair of blinded fingerprints that are used in the communication with the actual responder. The communication between the initiator and the responder goes via the forwarding agent. The identity privacy protects the end-points against Man-in-the-Middle attackers including the forwarding agent. Furthermore, the responder or a man-in-the-middle, on the responder's side of the forwarding agent, are not able to learn the topological location of the initiator.

Basically, both peers may obtain complete identity privacy without knowing the other's topological location. In a typical case, the responder leases a virtual interface, and publishes the virtual address in the DNS. The lease may have long lifetime, e.g., months. Rendezvous servers in different mobility architectures have this kind of forwarding agent role.

An initiator can make a lease from a suitable forwarding agent and hide the actual destination of the responder. As a result, the initiator obtains identity protection against active attacks, i.e., impersonating the responder.

8 Conclusions

In this paper, we have presented a framework offering identity and location privacy for end-points. The main focus has been on making the guessing of identities as difficult as possible by blinding the public keys and hiding the topological locations of the end-points using forwarding agents. The presented key exchange protocol provides passive and active identity protection for both peers unless an attacker is able to find out the public keys, utilizing location information. Thus, if the attacker can learn the identity with the help of IP addresses, no protocol can provide identity protection.

Our protocol does not require the parties to initially know the public keys of each other. It is sufficient that only the initiator knows a hash of the public key of its peer. The actual public keys are transmitted as a part of the protocol, but in such a way that even an active attacker is not able to learn them. To our knowledge, this is the first time this level of combined end-to-end security and privacy has been achieved in IP networks.

Acknowledgments

We would like to thank Gonzalo Camarillo, Petri Jokela, Tony Jokikyyny, Miika Komu, Goran Schultz, Vesa Lehtovirta, Vesa Torvinen, and Jari Arkko, for their valuable comments.

References

1. Johnson, D., Perkins, C., Arkko, J.: Mobility Support in IPv6. Internet Draft, work in progress. June, 2003.
2. Lamm, S.E., Reed, D.A., Scullin, W.H.: Real-time geographic visualization of world wide web traffic. World Wide Web Journal, The Web After Five Years, Summer 1996.
3. Perkins, C.: IP Mobility Support. RFC 2002. 1996.
4. Escudero-Pascual, A.: Privacy in the next generation internet: Data protection in the context of the european union policy. Ph.D. dissertation, Royal Institute of Technology, Stockholm, Dec. 2002. [Online]. Available: http://www.imit.kth.se/~ aep/PhD/
5. Aiello, W., Bellovin, S.M., Blaze, M., Canetti, R., Ionnadis, J., Keromytis, A., Reingold, O.: Efficient, dos-resistant, secure key exchange for internet protocols. ACM Computer Communications Review, Nov. 2002.
6. Saltzer, J., Reed, D., Clark, D.: End-To-End Arguments in System Design. ACM Transactions on Computer Systems, vol.2, Nov. 1984.
7. Shea, G., Roe, M.: Child-proof Authentication for MIPv6 (CAM). ACM Computer Communications Review, vol.31, Apr. 2001.
8. Moskowitz, R., Nikander, P., Jokela, P., Henderson, T.: Host Identity Protocol. Internet Draft, work in progress. Feb. 2004.

9. Nikander, P.: An architecture for authorization and delegation in distributed object-oriented agent systems. Ph.D. dissertation, Helsinki University of Technology, Helsinki, Mar. 1999. [Online]. Available: http://www.tml.hut.fi/~ pnr/publications/PhDThesis.pdf.
10. Fuller, V., Li, T., Yu, J., Varadhan, K.: Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy. RFC 1519. Sept. 1993.
11. Molina-Jimenez, C., Marshall, L.: True anonymity without mixes. In Proc. IEEE Workshop on Internet Applications '01, San Jose, CA, July, 2001.
12. Srisuresh, P., Holdrege, M.: IP Network Address Translator (NAT) Terminology and Considerations. RFC 2663. 1999.
13. Perlman. R.: Understanding IKEv2: Tutorial, and rationale for decisions. Internet Draft, work in progress. Feb. 2003.
14. Nikander, P., Ylitalo, J., Wall, J.: Integrating Security, Mobility, and Multi-Homing in a HIP Way. In Proc. Network and Distributed Systems Security Symposium, NDSS'03, San Diego, CA, Feb. 2003.
15. Abadi, M.: Private Authentication. In Proc. 2002 Workshop on Privacy Enhancing Technologies, Springer-Verlag, pp. 27–40. 2003.