IPv6 Opportunistic Encryption

Claude Castelluccia, Gabriel Montenegro

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

IPv6 Opportunistic Encryption

Claude Castelluccia — Gabriel Montenegro

N° 4568

Ocotober 2002

THÈME 1

ISSN 0249-6399 ISRN INRIA/RR--4568--FR+ENG

IPv6 Opportunistic Encryption

Claude Castelluccia*, Gabriel Montenegro †

Thème 1 — Réseaux et systèmes
Projets Planete

Rapport de recherche n° 4568 — Ocotober 2002 — 20 pages

Abstract: This paper presents an Opportunistic Encryption scheme for IPv6. Our proposal relies on IPv6 Anycast, Authorization certificates and Crypto-Based Identifiers (CBID) to provide secure and easily deployable Opportunistic Encryption in IPv6. Unlike existing schemes, our proposal does not rely on any global Third Trusted Party (such as DNSSEC or a PKI). Hence, we claim it is more secure, easier to deploy and more robust.

Key-words: Security, IPv6, Opportunistic Encryption, IPsec, CBID, delegation

* INRIA Rhône-Alpes, PLANETE group
† SUN Labs-Europe, 29, chemin du Vieux Chêne, 38240 Meylan, France

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38330 Montbonnot-St-Martin (France)
Téléphone : +33 4 76 61 52 00 — Télécopie +33 4 76 61 52 52

IPv6 Opportunistic Encryption

Résumé : Ce rapport presente un mécanisme de chiffrement opportunistic pour IPv6. Notre proposition utilise les adresses IPv6 Anycast, les certificats d'autorisation et les identifiants cryptographiques (CBID) afin de fournir un service de chiffrement opportunistic sure et facilement deployable pour IPv6. Contrairement aux solutions existantes, notre proposition n'utilise pas de serveurs de confiance (comme DNSSEC ou un PKI). Par conséquent, elle est plus sûre, plus facile à déployer et plus robuste.

Mots-clés : Securité, IPv6, Chiffrement Opportunistic, IPsec, CBID, delegation

1 Introduction

Because of its massive and widespread use, it is easy to overlook that the Internet remains a very hostile environment. Given that most of the packets are sent in the clear, there is a strong incentive both for legitimate as well as illegitimate reasons to install wiretaps [1] or to carry out passive eavesdropping. While end-to-end encryption is arguably the best solution for those concerned, currently it is not practical for several reasons: (1) most of the current hosts do not implement any encryption algorithms, (2) these can be quite expensive and prohibitive for constrained devices, and (3) end-to-end encryption requires a key management infrastructure which does not exist today.

Opportunistic encryption is a practical solution to this problem. It allows secure (encrypted, authenticated) communication without connection-by-connection pairwise pre-arrangement. To accomplish further ease-of use, instead of end-to-end encryption special security gateways can intercept packets and encrypt them for their traversal over the general Internet. The main idea is that the local security gateway intercepts an outgoing packet addressed to a remote host, and quickly negotiates an IPsec tunnel to that host's security gateway. As a result, packets sent by the hosts are encrypted as they traverse the Internet (i.e. between the security gateways). Although end-to-end encryption is preferable and more secure, this flavor of opportunistic encryption is easier to deploy as it requires modifying only the gateways, not the vastly more numerous end systems. The goal of opportunistic encryption is to increase the percentage of encrypted versus cleartext packets in the Internet. Security in existing schemes, such as the FreeSWAN system [2], relies on a *Trusted Third Party* (TTP), a globally-rooted security infrastructure such as DNSSEC [3] or a *Public Key Infrastructure* (PKI). As detailed in Section 4.1, relying on a TTP has major drawbacks in terms of security, deployment and robustness. In this paper we propose a solution for IPv6 that is opportunistic in a "gateway-to-gateway" manner, and that does not rely on any TTP. Our proposal relies on IPv6 Anycast, Authorization certificates and Crypto-Based Identifiers (CBID) to provide secure and easily deployable *Opportunistic Encryption* in IPv6.

The paper is structured as follows: Section 2 gives an overview of the opportunistic Encryption concept. Section 3 presents the related work and more specifically the FreeSWAN system. Section 4 discusses the motivations of our work. Section 5 details our proposal and its different components. Section 6 presents *Opportunistic JFK*, an opportunistic extension to the JFK protocol [4]. Section 7 assesses the security of our proposal. Finally, Section 9 concludes the paper.

2 Review of Opportunistic Encryption

The main idea of opportunistic encryption is to deploy security gateways that will sit between the border of *intranets* (private networks) and the Internet. A gateway intercepts an outgoing packet aimed at a remote host, and attempts to negotiate an IPsec tunnel to that host's security gateway. If the attempt succeeds, traffic can then be secured transparently (without changes to the end-host software). If the attempt fails, packets are sent through in the

clear or dropped, according to the local policy. Opportunistic encryption allows secure (encrypted, authenticated) communication via IPsec without connection-by-connection pairwise pre-arrangement. Each gateway administrator makes local arrangements to support opportunistic encryption. Once that is done, any two such gateways can communicate securely. Apart from careful attention to detail in various areas, there are three crucial design problems for opportunistic encryption.

1. *Remote Gateway Identification.* The local security gateway needs a way to quickly and securely identify the IP address of the remote Security Gateway for the packet that prompted the negotiation.

2. *Remote Gateway Authentication and Authorization.* The local security gateway needs to authenticate the other Security Gateway. This authentication needs to ensure that the other Security Gateway is who it claims to be and that it is authorized to represent the client for which it claims to be the gateway.

3. *Tunnel Establishment.* The security gateways need to establish a secure tunnel in a way that guarantees to reach agreement, without any explicit pre-arrangement or preliminary negotiation.

3 Related Work

3.1 FreeSWAN

The most recognizable Opportunistic Encryption system is certainly the one designed by the FreeSWAN project [2, 5]. This system heavily relies on DNSSEC to solve the *Remote Gateway Identification* and *Remote Gateway Authentication and Authorization* phase. It uses IKE [6] for the *Tunnel Establishment* phase. In the rest of this paper, the network is modeled as in Fig. 1.

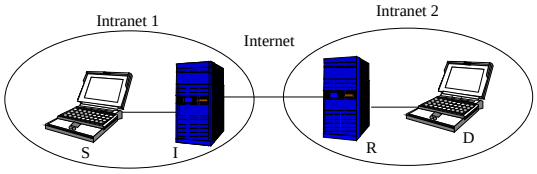

Figure 1: Network Model

The Initiator (I) is one of the Source's security gateways. The Responder (R) is one of the Destination's security gateways. The intercepted packet comes from the Source (S), addressed to the Destination (D), and is intercepted at the Initiator. The Initiator communicates over the insecure Internet to the Responder. The Source and the Initiator might be

the same host, or the Source might be an end-user host and the Initiator a security gateway (SG). Likewise for the Responder and the Destination.

FreeSWAN assumes that:

- Each security gateway publishes its public keys in their reverse DNS using DNSSEC KEY record.
- Each end-node publishes its authorized security gateways using a TXT record of the DNS. Optionally, for optimization, this record can also contain the gateways' public keys. This record is located in the reverse DNS (in-addr.arpa). The reverse DNS should be secured by DNSSEC, which is required to protect against active attacks.

The FreeSWAN scheme works as follows (see Fig. 2:

1. The application (at the Source) performs a DNS lookup to get the destination IP address. The resolver replies with the destination IP address.
2. The source initiates a connection (UDP or TCP) with the destination by sending a packet.
3. The Initiator intercepts the packet, buffers it and performs a reverse DNS request of the destination address. The DNS returns the TXT record that contains the security gateway(s) of the destination node and optionally their public keys (necessary for the IKE protocol). If the public keys are not contained in the TXT record, the Initiator performs a reverse DNS request of the selected remote gateway (Responder) to get its KEY record.
4. The Initiator and the Responder perform an IKE phase1 exchange (5 messages)
5. The Responder then performs a reverse DNS request of the Initiator address to get its KEY record (i.e. its public keys).
6. The Initiator and the Responder then perform a IKE phase2 exchange (3 messages).
7. The Responder needs to verify that the Initiator is actually a valid security gateway of the source node. It then performs a reverse DNS request of the source address and gets the TXT record. This record contains the list of the source node's authorized security gateway. The Responder then verifies that the Initiator belongs to it.
8. An (opportunistic) IPsec tunnel is then established between the Responder and the Initiator.

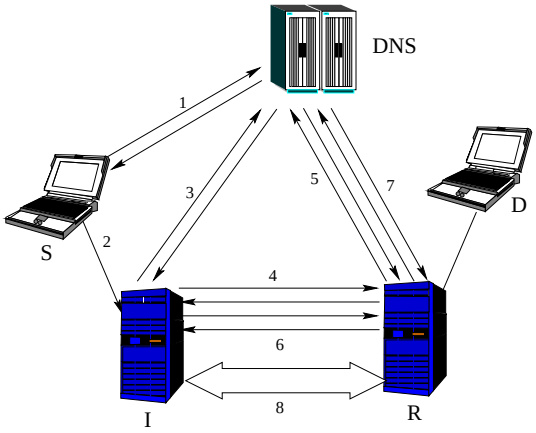

Figure 2: FreeSWAN Model

3.2 SSMAIL

The SSMAIL [7] system is also designed to counteract possible passive eavesdropping as user traffic traverses the internet. In particular, it secures email traffic exchanged between ESMPT mail agents. Typical email messages are relayed in a store-and-forward manner from agent to agent until it reaches the final destination, where the user fetches the messages. The objective of the SSMAIL system is to protect this traffic on those hops that traverse the internet. These ESMPT agents add a few steps to their session initialization. They engage in an unauthenticated Diffie-Hellman key exchange and use the resulting shared key to encrypt their traffic. Of course, this does not protect against active attackers in the middle, as its authors clarify. Neither does SSMAIL encrypt the traffic within the confines of a given administrative boundary, because they assume that within such an *intranet* the potential benefits of encrypting traffic are much less than for the outside Internet.

Unlike our work (and FreeSWAN) which provide opportunistic encryption for IPsec (thus benefitting most protocols layered over IP), SSMAIL only encrypts mail (ESMPT) traffic. They recognize this disadvantage, but argue that SSMAIL is still justified because it is much more easily deployable than FreeSWAN. In effect, SSMAIL only requires simple patches to ESMPT in order to work, instead of the major overhaul required by FreeSWAN. Nevertheless, the comparison is not quite fair, because a large part of the complexity in FreeSWAN comes from the defense against active attacks (their use of DNSSEC). Were FreeSWAN to be deployed without regard to active attacks, it would be much simpler.

On the other hand, FreeSWAN, being based on opportunistic IPsec between the security gateways, protects packets against all intermediate hops. SSMAIL, on the other hand, only protects traffic in a hop-by-hop fashion.

4 Motivations

4.1 The FreeSWAN model limitations

While end-to-end encryption is preferable, using opportunistic encryption in security gateways is very attractive for primitive or constrained devices that do not have the CPU capacities to perform expensive cryptographic operations. However Opportunistic Encryption, as specified in [2], relies on secure DNS as the mechanism to discover the remote gateway and to obtain its authentication key. This solution has the following limitations:

- The availability of the Opportunistic Encryption service depends on the availability of the DNS service. If the DNS service becomes unavailable, opportunistic encryption becomes impossible.
- The security of the system depends on DNS security. If the DNS is broken or is under DoS attacks, the opportunistic encryption system is unable to operate securely. Considering current DNS security [8], we believe this is a strong limitation of the proposed architecture.
- It assumes that each host has a DNS entry and has control over it (and on the reverse DNS). Some devices (e.g. constrained devices, sensors) may not have a DNS entry. In some environments (i.e. adhoc environments), hosts might not have a DNS entry (and neither access to a DNS) whereas Opportunistic Encryption might be very valuable. Furthermore many users do not have control over their DNS entries.
- Its assumes that secure DNS is deployed. This is far from being a reality. A variant of Opportunistic Encryption without secure DNS is proposed but any spoofed DNS reply can compromise the security of the whole system. This variant considerably weakens the overall security. Additionally, it is not clear if DNSSEC will not bring the same trouble as a large scale PKI would [9].
- It does not support mobile hosts. A mobile host that is visiting a foreign network might use a Care-of address that is not registered in its DNS.
- It introduces significant latencies. In fact, a security gateway must process few secure DNS requests and replies (i.e. performs few signature verifications) even before initializing its tunnel establishment with the remote gateway.
- It creates several new opportunities for DoS attacks. For example, a Bad Guy could send packets with forged source address. For each packet, the security gateway would perform a secure DNS lookup [10].

4.2 Problem statement

The motivation and objective of our work is to solve the limitations of the current proposal for Opportunistic Encryption for IPv6 while not overly relying on higher level services that are difficult and error-prone to configure or simply out of control of most users.

We aim to develop a system that allows a security gateway to intercept a packet from a local host, addressed to a destination host, and to establish an IPsec tunnel securely to the destination host's security gateway. By securely, we mean: (1) The local gateway must be able to identify one of the gateways associated with the remote host, authenticate it, and, further, it must also be able to verify that it has been authorized to act as a gateway for the remote host. (2) The identity of the communicating hosts must be protected over the (insecure) Internet, i.e. between the Initiator and the Responder. (3) The proposed system must not create new DoS attacks opportunities. We set the additional requirement that the gateways must be able to establish the opportunistic tunnel without relying on any kind of infrastructure nor any higher level services (such as DNS or PKI). Strictly speaking, we do rely on a distributed system, that of the routing infrastructure. But this requirement is common with all the other proposed solutions, not one we impose over and beyond existing requirements.

Finally we make the assumption that the path between the source and the initiator, being within an intranet, is much more secure than the outside segment between initiator and responder (e.g. there is a pre-existing tunnel or the source and initiator belong to the same organization). This is the "hard outside shell, soft interior" security model. We believe that while this is not always true, the risk of eavesdropping on outside packets is so much larger that it deserves more immediate attention. Finally, in a security-conscious intranet, the existence of a homogeneous administrative domain makes it operationally much more possible for local systems (e.g. source and initiator) to be able to secure their traffic. In such a situation it is much more straightforward to obtain a security association using more traditional IPsec and key exchange mechanisms.

5 IPv6 Opportunistic Encryption

5.1 Proposal Overview

Our proposal relies on three mechanisms: *anycast addresses*, *Cryptographically Based Identifiers (CBID)* and *authorization certificates* to solve the problems described in Section 2 at the IP layer without relying on higher layers' support (such as DNS support). Anycast is used to identify the remote security gateway. CBIDs are used for authentication and authorization certificates are used by the remote gateway to prove that it has been authorized by the destination host to act as a security gateway on its behalf. As described below, by using these three mechanisms together with a key establishment protocol, such as IKE [6] or JFK [4], we are able to propose an Opportunistic Encryption system that is able to establish IPsec tunnels between two security gateways securely and without relying on higher layer support. This system is also very easily deployable because all of these mechanisms are already (almost) available and our system does not require any changes in the existing Internet architecture.

The rest of this section describes these three basic entities and then presents our proposal in more details.

5.1.1 IPv6 Anycast Review

An IPv6 Anycast address is an address that is assigned to more than one interface. Thus an IPv6 Anycast address defines a group but as opposed to multicast group a packet sent to an Anycast address is not routed to all members of the group but only to the source's "nearest" one [11]. All interfaces belonging to an Anycast address usually reside within a topological region defined by an address prefix, P. Within this region, each member must be advertised as a separate "host route" entry in the routing system. A *router* that is member of an Anycast group will advertise its membership using the routing protocol (RIP, OSPF, BGP, etc). A *host* that wants to join an Anycast group will have to use a group membership protocol, such as MLD [12], to register with the local router(s) that will then propagate this registration to the region using the routing protocol. From outside the region, such a reserved subnet anycast address can be aggregated into the routing entry for prefix P.

5.1.2 Cryptographically-Based Identifiers (CBID)

Cryptographically Generated Identifiers and Addresses [13, 14], otherwise known as Crypto-Based Identifiers (CBID's), are identifiers derived from the hash of a public key.

We use the term *CBID* to refer to either of the two following entities derived from a host's public key as follows:

- Crypto-Based Address (CBA): an IPv6 address whose leftmost 64 bits are set to a valid prefix (as per normal IPv6 usage), and whose rightmost 64 bits (interface identifier) are set to a 64-bit entity obtained as follows: *hmac_64(imprint, PK)* (see Fig. 3).
- Crypto-Based Identifier (CBI): a fixed length entity obtained as follows: *hmac_x(imprint, PK)*, where x is the size of the identifier (typically 128 bits for IPv6).

Where *imprint* is a 64-bit field and PK is the host's public key.

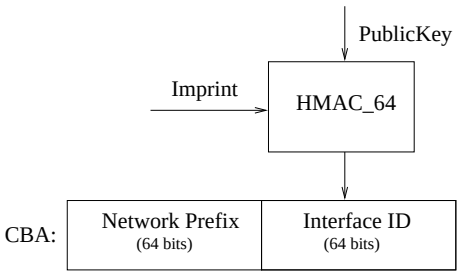

Figure 3: IPv6 Crypto-Based Address

These identifiers have two very important properties [13]:

- They are *statistically unique*, because of the collision-resistance property of the cryptographic hash function used to generate them.
- They are *securely bound to a given node*, because a node, N, can prove ownership of its CBID, $CBID\_N$ by revealing the public key, PK_N, and the imprint value, $imprint\_N$, used to generate the CBID and by proving that it knows the corresponding private key, SK_N. This can be performed by signing a message, *msg*. We refer to the signature of *msg* as sig_msg.

  Any other node can verify that node N owns $CBID\_N$ by running the following algorithm:

```
input: CBID_N, PK_N, imprint_N,
       sig_msg
output: accept or reject message CBID_N.

if (CBID_N == hash_x(PK_N, imprint_N)){
  if (VERIFY(sig_msg,PK_N)) {
/* CBID_N is owned by node N */
     accept CBID_N;
   }} else
/* CBID_N is not owned by node N */
     reject CBID_N;
```

  where VERIFY(sig, pkey) is a function that verifies, using the public key *pkey*, the signature *sig*.

  Note that this verification does not rely on any centralized security service such as a PKI or Key Distribution Center.

5.1.3 Review of Authorization Certificates

Authorization certificates are used to express delegation. There are several options available. The SPKI [15] (Simple Public Key Infrastructure) IETF working group has developed digital certificates whose main purpose is *authorization* rather than authentication. The relevant specifications have been published as *experimental* RFC's. Keynote2 [16], while not the product of an IETF working group, is defined in *informational* RFC's. It defines a semi-formal language and provides generous support to applications. In contrast, SPKI only defines the certificate format and leaves their actual definition to the applications. As a result, SPKI is conceptually much simpler. More recently, the PKIX working group has released *standards-track* RFCs specifying X.509 profiles for *Public Key Certificates* (PKC) [17] and *Attribute Certificates* for Authorization (AC) [18]. In particular, the latter provide a profile of X.509 for authorization purposes. This profile allows for more traditional *attribute certificates* in which the "holder" or beneficiary of the authorization is an identity (subsequently tied to a public key by a PKC), or *authorization certificates* which authorize a public

key directly. Even though the latter are close to the SPKI model, this X.509 profile is new and not yet widely supported by existing software.

For the sake of simplicity, we choose to use SPKI in this paper even though Keynote2 or potentially X.509 Attribute Certificates for Authorization could also be used. The main principles of SPKI can be summarized as follows:

- a certificate has 5 fields: (1) issuer (who is giving the authorization), (2) subject (who is acquiring the permission, (3) delegation (set if the subject can delegate the permission), (4) authorization (specifies the permission being communicated) and (5) validity.
- SPKI is *key-oriented*. No (name, key) binding, and therefore no CA, is necessary. The entities possessing, delegating and receiving access rights are cryptographic key pairs. A certificate can in short be written as: Sk(K' has the right R., t) (K gives the right R to K' and the validity period is t), where K and K' are two public key.
- A certificate has a validity period.
- Delegation certificates differ from traditional access control schemes in that any key may issue certificates. There is no central or trusted authority.
- A key may delegate rights to services it controls, it may also re-delegate rights it received by delegation from other keys.

Note that a full certificate is composed of a sequence of three objects [19]: the *public-key object* that contains the issuer public key, the *certificate object* that defines the authorization and *a signature object* that contains the signature.

5.2 Proposal Description

5.2.1 System Configuration

In our proposal:

- each host is configured with a Crypto-Based Address (CBA) as one of its IPv6 unicast addresses.
- each security gateway is configured with a Crypto-Based Identifier (CBI).

Additionally, each security gateway of a given network is reachable by a reserved IPv6 subnet anycast address, the OEGW (OE Gateway) Anycast address to be defined by the IANA [20]. This address must be configured and each authorized security gateway must join it.

Each security gateway must also be authorized by the hosts that it is serving as a security gateway for them. For this, each host issues a SPKI certificate to each security gateway it wants to authorize to act as a security gateway. This certificate specifies that the host,

identified by its CBA address, authorizes the security gateway, identified by its CBI to act as a security gateway. This certificate is signed by the host private key[1]. The format of the authorization certificate (actually of the certificate object) is the following:

```
(cert
  (issuer  (addr <host_cba>)
  (subject (addr <GW_cbi>)
  (tag ( OEauthorization)
  (not-before <date1>)
  (not-after  <date2>)
)
```

This certificate authorizes the security gateway, defined by its CBI, *GW_cbi*, to act as a security gateway for the host defined by its CBA, *host_cga_addr*. This certificate is only valid after *date1* and before *date2*. It is signed with the host private key. *Note that since the issuer's address is derived from its public key, the certificate does not need to be signed by a Certification Authority. It only needs to be self-signed.* This certificate does not bind a public key to an identity.

5.2.2 Protocol Overview

This section describes the message exchange of the proposed protocol. Note that this protocol has been intentionally simplified for readability reason. The complete protocol is described in the following section.
Our protocol works as follows:

1. The application (at the Source) performs a DNS lookup to get the destination IP address. The resolver replies with the destination IP address. This is normal DNS usage unrelated to opportunistic encryption.

2. The source initiates communications (UDP, ICMP, TCP, etc) with the destination by sending a packet.

3. The Initiator intercepts the packet, buffers it and sends a "OEGW request" (*OEGW_REQ*) message to the reserved subnet OEGW anycast address which corresponds to the packet's destination address. This packet contains the Source's CBA (CBA_S), the Source's Public Key (PK_S), the Destination's CBA (CBA_D), the Initiator's CBI (CBI_I), the Initiator's IP address (IP_I), the Initiator's Public Key (PK_I), the imprint value used by the Source to generate its CBA (imprint_S), the imprint value used by the Initiator to generate its CBI (imprint_I) and the SPKI certificate issued by the Source to the Initiator's CBI (SPKI_S). This message is signed with the Initiator's private key (SK_I). The signature is defined as (sig_request).

[1] The gateway needs to keep one certificate per host. Note however that the certificates do not need to be stored locally but can be stored on a local server. This is just a *storage* server, not a TTP since it does not need to be trusted.

4. Upon reception of this message, the Responder (1) verifies that the Initiator owns its CBI (i.e. the Initiator's CBI was generated from its public key and imprint and *OEGW_REQ*'s signature is correct), and (2) that the SPKI certificate is valid (it is signed by the source's private key and it does authorize the Initiator's CBI to act as a gateway). The algorithm executed by the Responder upon the reception of a *OEGW_REQ* is the following:

```
input:  sig_request, CBA_S, CBA_D, CBI_I,
        PK_I, PK_S, imprint_S,
        imprint_I, SPKI_S.
output: accept or reject OEGW_REQ

if (CBI_I == hash_x(PK_I,imprint_I)) {
 if (VERIFY(sig_request, PK_I)) {
  /* Initiator owns CBI_I */
  if ((issuer==CBA_S) and
      (subject==CBI_I)and
      (CBA_S==hash_64(PK_S,imprint_S))
   if (VERIFY(SPKI_S, PK_S))
     /* SPKI is valid */
      accept OEGW_REQ;
   else
      reject OEGW_REQ;
 }}
```

where VERIFY(sig, pkey) is a function that verifies, using the public key *pkey*, the signature *sig*.

Upon this verification, the Responder has the assurance that it is talking with a legitimate and authorized security gateway. It then replies to the Initiator with a "OEGW reply" (*OEGW_REP*) message that contains its CBI (CBI_R), its IP address (IP_R), its public key (PK_R), its imprint (imprint_R), and the SPKI certificate signed by the destination host (SPKI_D). This message is signed with the Responder's private key (SK_R).

5. Upon reception of the *OEGW_REP*, the initiator (1) verifies that the responding Gateway owns its CBI (i.e. the *OEGW_REP*'s signature is correct and the responder's CBI was generated from its public key and imprint) and (2) that the SPKI certificate is valid (it is signed by the destination address's private key and that it actually authorizes the responder to act as a gateway). The signature is defined as (sig_reply).

   The algorithm executed by the Responder upon the reception of a *OEGW_REP* message is the following:

```
input:  sig_reply, CBA_S, CBA_D, CBI_R,
        PK_D, PK_R, imprint_D,
        imprint_R, SPKI_D.
output: accept or reject OEGW_REP

if (CBI_R == hash_x(PK_R,imprint_R)) {
 if (VERIFY(sig_reply, PK_R)) {
  /* R owns CBI_R */
  if ((issuer==CBA_D) and
      (subject==CBI_R)and
      (CBA_D==hash_64(PK_D,imprint_D))
   if (VERIFY(SPKI_D, PK_D))
     /* SPKI is valid */
      accept OEGW_REP;
   else
      reject OEGW_REP;
 }}
```

Upon this verification, the Initiator has the assurance that it is talking with a legitimate and authorized security gateway.

6. The Responder and the Initiator engage into a key establishment exchange (such as IKE or JFK) to establish an IPsec Security Association.

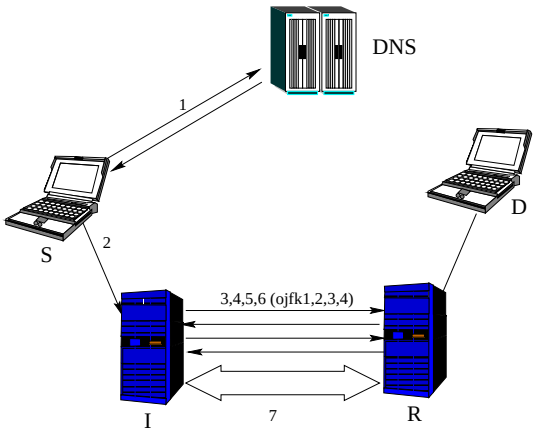

Figure 4: OE6 Model

We are aware that the simplified message exchange described above is vulnerable to several DoS attacks. However the above protocol should not be used as it is but must

be integrated within a key establishment protocol, such as IKE or JFK, as described in Section 6. The following section describes the integration of these messages into the $JFKr$ variant of the protocol. We have selected JFK because we believe that JFK design, based on simplicity and limited negotiation capabilities in order to facilitate interoperability, fits very well to the requirements of our system. Furthermore, the JFK specification is easy to understand and therefore to analyze. Recently, the $IKEv2$ [21] protocol (as opposed to JFK) has been selected as the replacement for the current IKE (v1) standard. Future work includes efforts to reformulate our opportunistic key exchange using $IKEv2$.

6 OJFK: Opportunistic Just Fast Keying

Using the same notation as in [4], our protocol, that we call OJFK (Opportunistic Just Fast Keying) is:

- *ojfk1:* $I \rightarrow \underline{OEGWAnycastAddress}$:
  Ni, g^i
- *ojfk2:* $R \rightarrow I$:
  Ni, Nr, g^r, $GRPINFOr$, $HMAC\{HKr\}$ (g^r, Nr, Ni, IPi)
- *ojfk3:* $I \rightarrow R$:
  Ni, Nr, g^i, g^r, $\underline{CBAd}$,$\underline{CBAs}$, $HMAC\{HKr\}(g^r, Nr, Ni, IPi, \underline{CBAd}, \underline{CBAs})$,
  $E\{Ke\}(IDi, IDr', \underline{CBI_I}, \underline{Imprint_I}, \underline{SPKI_{S(I)}}, sa, SIG\{i\}(Ni, Nr, g^i, g^r, GRPINFOr))$,
  $HMAC\{Ka\}('I', E\{Ke\}(IDi, IDr', sa, SIG\{i\}(Ni, Nr, g^i, g^r, GRPINFOr)))$
- *ojfk4:* $R \rightarrow I$:
  $E\{Ke\}(IDr, sa', SIG\{r\}(g^r, Nr, g^i, Ni))$, $HMAC\{Ka\}('R', E\{Ke\}(IDr, sa',$
  $\underline{CBI_R}, \underline{Imprint_R}, \underline{SPKI_{D(R)}}, SIG\{r\}(g^r, Nr, g^i, Ni)))$

The OJFK additions to the basic $JFKr$ protocol are underlined. They are the following:

- Three fields, CBI_I, $Imprint_I$ and $SPKI_{S(I)}$, have been added in *ojfk3*. CBI_I is the Initiator's CBI. $Imprint_I$ is the imprint used by the Initiator to generate its CBI together with its public key. This information is used by the Responder to verify that the Initiator owns its CBI. Note that the Initiator public key is in the ID_i field, which is a certificate. However as opposed to the regular JFK protocol, ID_i is a self-signed certificate. It is here only to carry the public key and does not need to be signed by a CA. $SPKI_{S(I)}$ is the SPKI certificate issued by the Source authorizing the Initiator I (actually CBI_I) to act as its security gateway. It is signed by the Source node's private key (of which the corresponding public key is in the SPKI certificate). These three fields were added in *ojfk3*, instead of *ojfk1* because (1) in JFK the Responder does not commit any resource before message3 (as a DoS protection) so it does not make sense to send information before *ojfk3* and (2) by adding these fields in *ojfk3* they can be encrypted which provides further privacy.

- Three fields, CBI_R, $Imprint_R$ and $SPKI_{D(R)}$ have been added in *ojfk4*. CBI_R is the Responder's CBI. $Imprint_R$ is the imprint that was used by the Responder to generate its CBI together with its public key. This information is used by the Initiator to verify that the Responder owns its address. Note that the Responder's public key is in the ID_r field, which is a self-signed certificate. It is here only to carry the public key and does not need to be signed by a CA. $SPKI_{D(R)}$ is the SPKI certificate issued by D to the gateway R (actually to CBI_R) that authorizes CBI_R to act as a security gateway. It is signed by the destination node's private key whose public key is the SPKI certificate. These three fields have been added in *ojfk4* instead of *ojfk2* because: (1) In JFK the Responder does not commit any resource before message3 (as a DoS protection). The Responder could actually send them in *ojfk2* but since this message is not signed, the Initiator will not be able to make use of them before *ojfk4*. (2) By adding these fields in *ojfk4* they can be encrypted which provides further privacy.

The keys used to protect *ojfk3* and *ojfk4*, Ke and Ka, are computed as $HMAC\{g^{ir}\}(Ni, Nr, 1)$ and $HMAC\{g^{ir}\}(Ni, Nr, 2)$ respectively. The session key used by IPsec (or any other application), Kir, is $HMAC\{g^{ir}\}(Ni, Nr, 0)$.

7 Security Analysis

The security analysis of the JFK protocol is detailled in [4]. In this section we access the security of the extension that we added in the JFK protocol.

7.1 Impersonation Attacks

A malicious host can attack a host, defined by a CBA, if it can find a public/private pair whose public key hashes to the target's CBA. It can then issue fake SPKI certificates and impersonate the target host's gateway. As a result of this attack, the malicious host can then wiretap the target's packets. To complete this attack the malicious host must attempt 2^{62} (i.e. approximately $4.8 * 10^{18}$) tries to find a public key that hashes to the CBA. If the attacker can do 1 million hashes per second it needs 142,235 years. If the attacker can hash 1 billion hashes per second it still needs 142 years.

An attacker can attack a security gateway, defined by a CBI, if it can find a public/private pair whose public key hashes to the target's CBI. If it succeeds, the malicious host can then wiretap the traffic of all hosts supported by the target security gateway. This attack is therefore more severe than the previous one. Fortunatly this attack is much more difficult to perform. In fact, in order to complete it the attacker must attempt 2^{128} (i.e. approximately $3.4 * 10^{38}$) tries to find a public key that hashes to the CBI. If the attacker can do 1 million hashes per second it needs 10^{25} years. If the attacker can hash 1 billion hashes per second it still needs 10^{22} years. Brute-force attacks are nearly impossible.

7.2 DoS Attacks

Fake (or malicious) Initiator: a malicious host (or a set of malicious hosts) could attack a Responder by bombing it with fake OJFK messages. In OJFK (as in JFK), a Responder does not commit any resource before *ojfk3*. This is done in order to detect Initiator that uses spoofed address (in this case they won't receive *ojfk2*). So this attack is not very severe and probably not worse than just bombing the Initiator with regular packets. If it was, a Puzzle mechanism [22] could be added in *ojfk2* and *ojfk3*.
Note that a set of Initiators (using a DDoS type of attack) could attack a target Responder by establishing a lot of opportunistic tunnels with it just for the sake of exhausting its resource. There is not much that can be done here because by nature the Responder will accept any Opportunistic Tunnels. Note that the same DDoS attack exists with SSL. In fact a set of attackers could establish a lot of connections with a server. The solution is, for OJFK, to increase the number of security gateways and perform some load-balancing. If the number of tunnels is too large (or increase too rapidly), a puzzle mechanism could be executed.

Fake (or malicious) Responder: A malicious host could reply to a valid *ojfk1* by (possibly a lot of) fake *ojfk2* in order to overload Initiator (upon reception of a *ojfk2*, the Initiator performs a DH exponentiation, generates a signature and encrypts two messages). This attack requires the malicious host to be on the path between the Initiator and the legitimate Responder. Note that this attack is not specific to OJFK but also exists in JFK.

Fake (or malicious) Destination: A malicious host can attack an Initiator by sending packets to a lot of different destinations through it. For each of this packet, the Initiator will establish an IPsec tunnel and consume a lot of resource. To prevent this attack the Initiator must only establish tunnels for trusted sources. How this trust relationship is established is out of the scope of this paper. Ingress filtering might be enough in most of the cases. An Initiator will only establish tunnels for packets that come from the internal network.

7.3 Privacy Considerations

In OJFK, the destination host (in *ojfk3*) and the SPKI certificate (in *ojfk4*) are never sent in the clear on the Internet. As a result, the Privacy of the communicating hosts is assured since a host snooping the packets between the Initiator and the Responder will not be able to identify the communicating nodes. This also makes traffic analysis more difficult to perform.

8 Implementation Considerations

We have begun an implementation of our opportunistic encryption scheme. An opportunistic encryption scheme requires a policy module that holds the rules regarding the proper treatment according to a packet's signature (protocol, addresses, etc): (1) should the packet be dropped, (2) allowed in the clear or (3) if an opportunistic encryption key exchange must be initiated in order to encrypt it. So far we have implemented the opportunistic encryption key exchange by modifying the JFK implementation from Columbia University. Future

work involves implementing the required policy module such that our Opportunistic JFK gets triggered appropriately by the corresponding packets[2].

9 Conclusions

This paper presents a secure IPv6 Opportunistic Encryption scheme that, in contrast to the FreeSWAN project that relies on DNSSEC, does not need the support of any TTP.

We have seen another proposal, SSMAIL which limits itself to protecting only email (SMTP) traffic against passive eavesdropping, while ignoring active attacks, in order to obtain a more readily deployable solutions.

We feel that our proposal obtains the best of both worlds: we protect against active attacks without relying on DNSSEC to provide the certificates for gateway authentication. Instead, it relies on an inherent cryptographic binding between the identity of the gateways and their public keys. We handle authorization by using proper cryptographic tools instead of (as in FreeSWAN) by forcing this role onto DNS TXT records.

Furthermore, we provide for opportunistic encryption at the IPsec layer, so the greatest number of applications are transparently protected.

Our solution uses IPv6 Anycast, Authorization certificates (or delegation) and Crypto-Based Identifiers (CBID) to provide secure and easily deployable Opportunistic Encryption in IPv6. Security gateways establish security tunnels among themselves without having to contact any TTP. This provides a solution which is more scalable, robust, easier to deploy, and as a result of the above, more secure.

References

[1] Bellovin Steve, "Wiretapping the net," The Bridge, National Academy of Engineering., 2000.

[2] Henry Spencer and D. Hugh Redelmeier, *Opportunistic Encryption*, Linux FreeSWAN Project, http://liberty.freeswan.org/freeswan_trees/freeswan-1.91/doc/opportunism.spec, May 2001.

[3] R. Arends, M. Larson, D. Massey, and S. Rose, *DNS Security Introduction and Requirements*, IETF, draft-ietf-dnsext-dnssec-intro-02, July 2002.

[4] W. Aiello and S. et al. Bellovin, *Just Fast Keying (JFK)*, IETF, draft-ietf-ipsec-jfk-04.txt, July 2002.

[5] M. Richardson, R. Redelmeier, and H. Spencer, *Opportunistic Encryption unsing the Internet Key Exchange (IKE)*, IETF, draft-richardson-ipsec-ipsec-opportunistic-09.txt, April 2002.

[2] Note to reviewers: we will endeavor to complete this section for the final version of our paper in the event it gets accepted for the conference.

[6] D. Harkins and D. Carrel, *The Internet Key Exchange (IKE)*, IETF, RFC2409, November 1998.

[7] Damien Bentley, Greg Rose, and Tara Whalen, "SSMAIL: Opportunistic Enncryption in Sendmail," in *13th Usenix LISA 99*, Seattle,Washington, November 1999, Usenix.

[8] Steven Bellovin, "Using the domain name system for system break-ins," in *Fifth Usenix UNIX Security Symposium*, July 1995.

[9] C. Ellison and B. Schneier, "Ten risks of PKI: What you're not being told about public key infrastructure," *Computer Security Journal*, vol. 16, no. 1, pp. 1–7, 2000.

[10] D. Hugh Redelmeier, *FreeS/WAN Opportunistic HowTo*, Linux FreeSWAN Project, http://liberty.freeswan.org/freeswan_trees/freeswan-1.91/doc/opportunism.howto, June 2001.

[11] B. Hinden and S. Deering, *IP Version6 Addressing Architecture*, IETF, RFC2373, July 1998.

[12] B. Haberman and D. Thaler, *Host-based Anycast using MLD*, IETF, draft-haberman-ipngwg-host-anycast-00.txt, February 2001.

[13] G. Montenegro and C. Castelluccia, "Statistically Unique and Cryptographically Verifiable (SUCV) identifiers and addresses," in *NDSS'02*, February 2002.

[14] Greg O'Shea and Michael Roe, ""Child-proof Authentication for MIPv6 (CAM)," *ACM Computer Communications Review*, April 2001.

[15] C.and al. Ellison, *SPKI Certificate Theory*, IETF, RFC 2693, September 1999.

[16] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis, *The KeyNote Trust-Management System Version 2*, IETF, RFC2704, September 1999.

[17] R. Housley, W. Polk, W. Ford, and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) profile," April 2002.

[18] S. Farrel and R. Housley, "An Internet Attribute Certificate Profile for Authorization," April 2002.

[19] C. et al. Ellison, *SPKI Examples*, IETF Internet Draft, Internet Draft, Available at http://world.std.com/cme/examples.txt, March 1998.

[20] D. Johnson and S. Deering, *Reserved IPv6 Subnet Anycast Addresses*, IETF, RFC2526, March 1999.

[21] D. Harkins, C. Kaufman, S. Kent, T. Kivinen, and R. Perlman, *Proposal for the IKEv2 Protocol*, IETF, draft-ietf-ipsec-ikev2-02.txt, April 2002.

[22] T. Aura, P. Nikander, and J. Leiwo, "DOS-resistant authentication with client puzzles," in *8th International Workshop on Security Protocols*, April 2000.

Contents

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399