Hindering Eavesdropping via IPv6 Opportunistic Encryption

Claude Castelluccia[1], Gabriel Montenegro[2], Julien Laganier[2,3], and Christoph Neumann[1]

[1] INRIA Rhône-Alpes, 655 Avenue de l'Europe, 38334 Saint Ismier CEDEX, France
{claude.castelluccia,christoph.neumann}@inrialpes.fr
[2] Sun Labs, Europe, 180 Avenue de l'Europe, 38334 Saint Ismier CEDEX, France
{gab,ju}@sun.com
[3] LIP (UMR #5668 CNRS/ENS Lyon/INRIA/UCB Lyon)
46, Allée d'Italie, 69007 Lyon, France

Abstract. This paper presents an opportunistic encryption scheme strictly layered on top of IPv6. Assuming that a node needs to send data toward another node, our proposal enables the dynamic configuration of an encrypted tunnel between the two nodes' IPsec gateways. The main contribution of this paper is to propose a solution that is fully distributed and does not rely on any global Trusted Third Party (such as DNSSEC or a PKI). The IPsec gateways are discovered using IPv6 anycast, and they derive authorization from authorization certificates and Crypto-Based Identifiers (CBIDs). The result is a robust and easily deployable opportunistic encryption service for IPv6.

Keywords: Security, IPv6, Opportunistic Encryption, IPsec, CBID, delegation, IKE.

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

P. Samarati et al. (Eds.): ESORICS 2004, LNCS 3193, pp. 309–321, 2004.

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 2, relying on a TTP has major drawbacks in terms of security, deployment and robustness. The main contribution of this paper is to 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 presents the concept of opportunistic encryption and provides an overview of the related work. Section 3 discusses the motivations of our work. Section 4 details our proposal and its different components. Section 5 presents the *Opportunistic SUCV* protocol, an opportunistic extension to the SUCV protocol [4]. Section 6 assesses the security of our proposal. Section 7 concludes the paper. Finally, we include implementation details and a description of how to integrate our scheme in IKEv2 [5] in the appendix.

2 Opportunistic Encryption: Concept and Related Work

Concept Overview: The main idea of opportunistic encryption is to deploy IPsec security gateways at site borders such that they are dynamically discoverable and usable by remote security gateways instead of requiring pre-configuration between specific sites. Such gateways (1) intercept an outgoing packet from a *source* aimed at a remote host (the *destination*), (2) dynamically discover the destination's security gateway, and (3) negotiate an IPsec tunnel with the destination's gateway (the *responder*).

Once an administrator configures its site's gateway(s) to support opportunistic encryption, the security services afforded by such an arrangement are (1) encrypted and authenticated communication *between the gateways* via IPsec without site-by-site pair-wise pre-arrangement, and, (2) protection from eavesdroppers (in particular, from the Internet at large).

In this paper, opportunistic encryption does not necessarily provide end-to-end security. For example, opportunistic encryption does not provide end-to-end authentication: A node that receives packets from a given IP address does not have the guarantee that these packets were actually sent by this IP address or even that the packets have not been modified on their way. The main goal of opportunistic encryption is to improve privacy on the Internet by enabling message privacy as a default. It aims at hindering eavesdropping on the Internet by encrypting packets at intermediate gateways.

The Design Challenges: Apart from careful attention to detail in various areas, there are three crucial design challenges for opportunistic encryption:

1. *Remote Gateway Discovery.* The local security gateway needs a way to quickly and securely discover 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. Without this authorization phase, a malicious host could pretend to be the gateway of a node and eavesdrop on its packets.
3. *Tunnel Establishment.* The security gateways need to establish a secure tunnel in a way that guarantees to reach agreement, without any explicit prearrangement or preliminary negotiation.

FreeSWAN System: The most recognizable opportunistic encryption system is certainly the one designed by the FreeSWAN project [2]. This system heavily relies on DNSSEC to solve the *Remote Gateway Discovery* and *Remote Gateway Authentication and Authorization* phase. It uses IKE [5] for the *Tunnel Establishment* phase. FreeSWAN assumes that each end-node publishes in the reverse DNS tree its authorized security gateway(s), and their respective public key(s).

These two pieces of information are combined into a single IPSECKEY DNS Resource Record [6], stored in the reverse DNS tree (in-addr.arpa, or ip6.arpa). Lookups in the reverse DNS tree should be secured by DNSSEC in order to protect against active attacks. Note that a single node might publish several such IPSECKEY RRs with different precedence values for failover or load-balancing (similarly to what is done for mail servers with MX records). This solution has the following limitations:

- The availability of the opportunistic encryption service depends on the availability of the DNS service.
- The security of the system depends on DNS security [7] (DNSSEC) and its deployment, currently, a significant limitation. Using FreeSWAN without DNSSEC, while possible, renders the whole system vulnerable to spoofed DNS replies. Additionally, the full-scale deployment of DNSSEC may be as troublesome as that of a large-scale PKI [8].
- It assumes that each host has an entry in the DNS reverse tree, and has control over it. This is a very heavy constraint, in particular, for dynamic environments and the associated devices (e.g., a mobile node visiting a foreign network might use a care-of address that is not registered in its DNS).
- It introduces significant latencies. A security gateway must process some secure DNS requests and replies (i.e., perform some signature verifications) before establishing a tunnel with the remote gateway.
- It creates several new opportunities for DoS attacks. For example, a malicious host could send packets with forged source address. For each packet, the Responder security gateway would perform a secure DNS lookup.

3 Motivations

The motivation and objective of our work is to propose a model for IPv6 opportunistic encryption that is fully distributed and does not rely on higher level services. We aim to develop a *secure* opportunistic encryption system. By *secure*, we mean that the local gateway must be able to *discover* one of the gateways associated with the remote host, *authenticate* it, and verify that it has been *authorized* to act as a gateway for the remote host. The identity of the communicating hosts must be protected over the (insecure) Internet. *We impose 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).*

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 locally associated systems (e.g., a source and its initiator gateway) 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.

4 IPv6 Opportunistic Encryption

4.1 Proposal Overview

Our proposal relies on three mechanisms: *anycast addresses*, *Crypto-Based Identifiers (CBID)* and *authorization certificates*. 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.

The contribution of our work is to effectively combine these three mechanisms together with a key establishment protocol, such as IKE [5] or sucvP [4], 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.

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 [9]. 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 [10], 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.

Crypto-Based Identifiers (CBID): Crypto-Based Identifiers and Addresses [4, 11, 12], 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-Generated Address (CGA): 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)*.
- Crypto-Based Identifier (CBI): a fixed length cryptographic token obtained as follows: *hmac_x(imprint, PK)*, where x is the size of the identifier.

Where *imprint* is a 64-bit field and PK is the host's public key. The imprint is a quantity used to limit certain types of brute-force attacks [4]. In this work, it is assumed to be equal to the IPv6 64-bit network prefix for CGA (in agreement with [12]), and ignored (e.g., set to 0) for CBI.

These identifiers have two very important properties [4]:

- 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

A node can prove ownership of its *CBID* by revealing the public key, *PK*, and the imprint value, *imprint*, used to generate the CBID and by proving that it knows the corresponding private key, *SK*. This can be performed by signing a message.

Any other node can verify that a given node owns its *CBID* by recomputing it and verifying the signature. Note that this verification does not rely on any centralized security service such as a PKI or Key Distribution Center.

Review of Authorization Certificates: Authorization certificates are used to express delegation. We choose to use SPKI [13] in this paper even though Keynote2 [14] or potentially X.509 Attribute Certificates for Authorization [15] 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 Certification Authority (CA), is necessary. The entities possessing, delegating and receiving access rights are cryptographic key pairs. A certificate can in short be written as: $(PK', R, t, PK)_{SK}$: PK gives the right R to PK' and the validity period is t, where PK and PK' are two public key. The certificate is signed with SK, where SK is the private key corresponding to PK.
- 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 [16]: 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.

4.2 Proposal Description

System Configuration: In our proposal each host is configured with a Cryptographically Generated Address (CGA) as its default IPv6 unicast address. 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 (Opportunistic Encryption Gateway) Anycast address to be defined by the IANA [17]. 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 CGA address, authorizes the security gateway, identified by its CBI to act as a security gateway. This certificate is signed by the host[1]. The format of the authorization certificate (actually of the certificate object) is the following:

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

[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.

This certificate authorizes the security gateway, defined by its CBI, *GW_cbi*, to act as a security gateway for the host defined by its CGA, *host_cga*. This certificate is only valid after *date1* and before *date2*, and is signed by the host.

Protocol Overview: This section describes the message exchange of the proposed protocol. We assume that the application (at the Source) knows the destination IP CGA. The specific means of obtaining this destination IP address are not specific to (and out of scope of) our proposal, and may use any of various methods including: lookups (DNS/DNSSEC, LDAP, NIS, etc.), manual configuration, dynamic referrals and redirects, etc. Also possible are user-friendly exchanges using secure side channels such as SCEAU [4].

Our protocol works as follows[2].

1. The Source initiates a packet exchange (TCP, SCTP, DCCP, UDP, ICMP, etc) with the Destination.
2. The Source's security gateway, referred as the *Initiator*, intercepts the packet, and verifies that it is authorized to act as gateway by the Source's address CGA_S (by matching CGA_S against its available list of authorization certificates). If an adequate certificate is found, based upon the packet's destination address prefix, the Initiator gateway calculates the reserved subnet OEGW anycast address according to normal IPv6 usage [17]. The Initiator gateway then sends an "OEGW request" (*OEGW_REQ*) to that anycast address. This packet contains the Source's CGA (CGA_S), the Source's Public Key (PK_S), the Destination's CGA (CGA_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 CGA ($imprint_S$), 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 ($SPKI_S(I)$). This message is signed by the Initiator.
3. Upon reception of this message, one of the destination node's security gateway, 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 and it does authorize the Initiator's CBI to act as a gateway). 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 ($SPKI_D(R)$). This message is signed by the Responder.
4. 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 host and that it actually authorizes the responder to act as a gateway).

[2] This protocol has been intentionally simplified for clarity.

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

5. The Responder and the Initiator engage into a key establishment exchange (such as IKE [5] or sucvP [4]) to establish an IPsec Security Association.

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 sucvP, as described in Section 5.

5 osucvP: Opportunistic Statistically Unique and Verifiable Protocol

This section presents *osucvP* (Opportunistic Statistically Unique and Verifiable Protocol), the protocol that is used between security gateways to establish secure channels. This protocol relies on the *sucvP* protocol described in [4]. We have selected *sucvP* because we believe that its design, based on simplicity and limited negotiation capabilities in order to facilitate interoperability, fits very well to the requirements of our system. *OsucvP* is very similar to the ISO protocol described and analysed in [18]. The security on this protocol can be based on the analytical work of [19] where it is shown that the ISO protocol is a secure key establishment protocol. *OsucvP* provides perfect forward secrecy via a Diffie-Hellman exchange authenticated with digital signatures.

Recently, the $IKEv2$ [20] protocol has been selected as the replacement for the current IKE (v1) standard. We describe in the Appendix how $IKEv2$ could be used with our OE scheme.

Using the same notation as in [4], our protocol is defined by the four following messages (illustrated by Fig. 1):

- *osucvP1* ($I \rightarrow OEGW Anycast Address$):
  N1, CGA_D, CGA_S
- *osucvP2* ($R \rightarrow I$):
  N1, *puzzle request*, CBI_R, g^r
- *osucvP3* ($I \rightarrow R$):
  N1, *puzzle reply*, CBI_I, g^i, CGA_S, CGA_D, $SPKI_S(I)$, PK_I, SPI, $lifetime_I$, SIG_{SK_I}(N1, *puzzle reply*, CBI_R, CBI_I, $SPKI_S(I)$, PK_I, g^i, g^r, CGA_S, CGA_D, SPI, $lifetime_I$)
- *osucvP4* ($R \rightarrow I$):
  $N1$, $SPKI_D(R)$, PK_R, SPI, $lifetime_R$, SIG_{SK_R}(N1, *puzzle reply*, CBI_R, CBI_I, $SPKI_S(I)$, $SPKI_D(R)$, PK_I, PK_R, g^i, g^r, CGA_S, CGA_D, SPI, $lifetime_R$)

The first message (*osucvP1*) is the *OEGW_REQ* request that is sent by the source's security gateway (Initiator I) to the destination node's OEGW anycast address. Upon reception of this message, one security gateway (Responder R) serving the destination node replies with a *osucvP2* message. This message

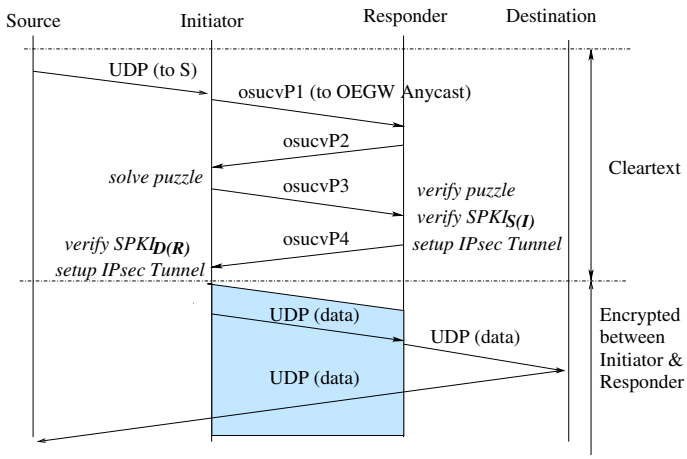

Fig. 1. Opportunistic SUCV Protocol

contains the nonce $N1$ that was sent in *osucvP1*, a client puzzle request, a Diffie-Hellman component and a session key lifetime. Upon reception of this message, I solves the puzzle and sends *osucvP3* back to R. This message is signed with SK_I, the private key whose corresponding public key, PK_I, was used to generate CBI_I. The four fields, CBI_I, CGA_D, CGA_S and $SPKI_S(I)$, have been added in *osucvP3* instead of *osucvP1* because in *sucvP* R does not commit any resource before *osucvP3* (as a DoS protection). Hence, it is useless to send this information before *osucvP3*. Upon reception of *osucvP3*, R verifies the signature and $SPKI_S(I)$, as described in the previous section, computes the IPsec session key, *Skey_ipsec*, using Diffie-Hellman key exchange protocol and establishes an IPsec tunnel with I. It then replies with *osucvP4*. This message is signed with SK_R, the private key whose corresponding public key, PK_R, was used to generate CBI_R. The messages *osucvP3* and *osucvP4* are signed to authenticate the DH exchange and to solve the key-identity binding problem described in [18]. The three fields, CBI_R, PK_R and $SPKI_D(R)$ are sent in *osucvP4* instead of *osucvP2* because R does not commit any resource until it verified the puzzle reply contained in *osucvP3*, to protect against DoS attacks. R could actually send them in *osucvP2* but since this message is not signed, I will not be able to make use of them before *osucvP4*. Upon reception of *osucvP4*, I verifies the signature and $SPKI_S(R)$, as described in the previous section, computes the IPsec session key, *Skey_ipsec*, using Diffie-Hellman key exchange protocol and establishes an IPsec tunnel with R.

Thereafter, the packets are encrypted between I and R, i.e. when they traverse the Internet.

6 Security Analysis

The security analysis of the sucvP protocol is detailed in [4]. In this section we assess the security of the extension that we added in the sucvP protocol to support opportunistic encryption.

6.1 Impersonation Attacks

CGA impersonation: A malicious host can attack a host, defined by a CGA, if it can find a public/private pair whose public key hashes to the target's CGA. 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 CGA. If the malicious host can do 1 million hashes per second it needs 142,235 years. If the malicious host can hash 1 billion hashes per second it still needs 142 years[3].

CBI impersonation: A malicious host 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. Fortunately this attack is much more difficult to perform. In fact, in order to complete it the malicious host must attempt 2^{128} (i.e. approximately 3.4×10^{38}) tries to find a public key that hashes to the CBI. If the malicious host can perform 1 million hashes per second it needs 10^{25} years. If the malicious host can hash 1 billion hashes per second it still needs 10^{22} years. Brute-force attacks are nearly impossible.

6.2 DoS Attacks

Fake (or malicious) Initiator: a malicious host (or a set of malicious hosts) could attack a Responder by bombing it with fake *osucvP* messages. In *osucvP* (as in sucvP), a Responder does not commit any resource before *osucvP3*. This is done in order to detect Initiator that uses spoofed address (in this case they won't receive *osucvP2*). So this attack is not very severe and probably not worse than just bombing the Initiator with regular packets.

A set of Initiators (using a DDoS type of attack) could also attack a target Responder by establishing a lot of opportunistic tunnels with it just for the sake of exhausting its resource. Notice that the responder cannot be forced to participate in the tunnel creation unless the responder had a certificate from the packet destination. The same family of attacks exists with SSL, the main difference being that the target of the attack is the destination in the SSL case, whereas it is authorized by the destination in the OE case. The solution is, for *osucvP*, to increase the number of security gateways and perform some load-balancing.

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.

[3] As shown in [12], the security of a CGA address can easily be increased if needed.

7 Conclusion

The main contribution of this paper is to propose a solution for IPv6 that is opportunistic in a "gateway-to-gateway" manner, and that does not rely on any Trusted Third Party (such as a Public Key Infrastructure or DNSSEC). We have obtained a solution that is fully distributed thanks to a judicious combination of IPv6 anycast, authorization certificates (or delegation) and Crypto-Based Identifiers (CBIDs), which rely on an inherent cryptographic binding between the identity of the entities and their public keys. This efficient combination is scalable, robust, efficient and easier to deploy.

Acknowledgements

We thank the anonymous reviewers for their valuable comments. Special thanks to Alberto Escudero-Pascual for his valuable help and many constructive suggestions.

References

1. Bellovin Steve, "Wiretapping the net," The Bridge, National Academy of Engineering., 2000.
2. M. Richardson and R. Redelmeier, *Opportunistic Encryption unsing the Internet Key Exchange (IKE)*, IETF, draft-richardson-ipsec-opportunistic-13.txt, February 2004.
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. Gabriel Montenegro and Claude Castelluccia, "Crypto-based identifiers (CBIDs): Concepts and applications," *ACM Transactions on Information and Systems Security (TISSEC)*, vol. 7, no. 1, February 2004.
5. D. Harkins and D. Carrel, *The Internet Key Exchange (IKE)*, IETF, RFC2409, November 1998.
6. Michael Richardson, *A Method for Storing IPsec Keying Material in DNS*, IETF, draft-ietf-ipseckey-rr-09.txt, Feb 2004.
7. Steven Bellovin, "Using the domain name system for system break-ins," in *Fifth Usenix UNIX Security Symposium*, July 1995.
8. 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.
9. B. Hinden and S. Deering, *IP Version6 Addressing Architecture*, IETF, RFC3513, April 2003.
10. B. Haberman and D. Thaler, *Host-based Anycast using MLD*, IETF, draft-haberman-ipngwg-host-anycast-00.txt, February 2001.
11. Greg O'Shea and Michael Roe, ""Child-proof Authentication for MIPv6 (CAM)," *ACM Computer Communications Review*, April 2001.
12. Tuomas Aura, "Cryptographically generated addresses (CGA)," in *6th Information Security Conference (ISC'03*, Bristol, UK, October 2003, vol. 2851, pp. 29–43, LNCS.
13. C. Ellison and etc., *SPKI Certificate Theory*, IETF, RFC 2693, September 1999.

14. M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis, *The KeyNote Trust-Management System Version 2*, IETF, RFC2704, September 1999.
15. S. Farrel and R. Housley, *An Internet Attribute Certificate Profile for Authorization*, IETF, RFC3281, April 2002.
16. C. Ellison and etc., *SPKI Examples*, IETF Internet Draft, Internet Draft, Available at http://world.std.com/cme/examples.txt, March 1998.
17. D. Johnson and S. Deering, *Reserved IPv6 Subnet Anycast Addresses*, IETF, RFC2526, March 1999.
18. Hugo Krawczyk, "SIGMA: The SIG-and-MAc approach to authenticated diffie-hellman and its use in the ike protocols," in *CRYPTO*, Santa Barbara, August 2003.
19. R. Canetti and H. Krawczyk, "Analysis of key-exchange protocols anf their use for building secure channels," in *Eurocrypt*, 2001, vol. 2045.
20. C. Kaufman, *Internet Key Exchange IKEv2 Protocol*, IETF, draft-ietf-ipsec-ikev2-13.txt, March 2004.
21. D. McDonald, C. Metz, and B. Phan, *PF_KEY Key Management API, Version 2*, IETF, RFC2367, July 1998.

Appendix A: Implementation

We implemented IPv6 opportunistic encryption on the FreeBSD-4.7 platform, which incorporates most of the KAME IPv6 stack.

Our implementation consists of two parts:

- a user level daemon (*sucvPd*) for (1) proof of ownership and verification of CBID's (CBI's and CGA's), (2) exchange and verification of SPKI delegation certificates and (3) key exchange via the sucvP ([4]) protocol.
- kernel level modification of the IPv6 outbound IPsec tunnel mode routine (*ipsec6_output_tunnel*), which does not interfere with regular IPsec behavior.

Prior to the sucvP transaction, the local security gateway is unaware of the remote OEGW's address. Accordingly, an opportunistic encryption SPD entry has the unspecified address (i.e., *::0*) as its remote tunnel endpoint.

The sucvP daemon communicates with the IPsec stack through the *PFKEY_V2* API [21]. It listens for an *SADB_ACQUIRE* message with the unspecified address as its remote tunnel endpoint, indicating that an OEGW needs to be discovered. To better understand this mechanism, let's look at the normal tunnel mode processing. Usually when a packet matches an SPD entry specifying that IPsec tunnel mode is required, the stack sends an *SADB_ACQUIRE* message to the key management daemon, thus requesting a key exchange between the tunnel endpoints (indeed, the two OEGW's). However, with opportunistic encryption the remote tunnel endpoint is not known at this moment. It needs to be discovered.

We defined a new kernel variable (*net.inet6.ipsec6.sucv_oe*) whose intent is to provide a user level means to modify the behavior of the IPsec stack when dealing with OE SPD entries.

When this variable is set to 0, the IPsec stack behaves like a regular FreeBSD implementation, i.e., a packet matching an opportunistic encryption SPD entry

would trigger a regular *SADB_ACQUIRE* with the unspecified address as its remote tunnel endpoint (in the destination *SADB_ADDRESS* payload). Obviously, this will not progress any further, since there is not enough information for the key management daemon to know with which entity it should negotiate.

When this variable is set to 1, a packet matching an SPD entry will trigger a special *SADB_ACQUIRE* message in which the two *SADB_ADDRESS* payloads contain (1) the source, and (2) the destination addresses of the original packet (instead of the IPsec tunnel endpoints as in the previous case). Thus, the key management daemon knows the real destination of the packet, and uses it to derive the anycast address of the remote security gateway *OEGW*.

The key management daemon can identify this *SADB_ACQUIRE* message as one triggered by opportunistic encryption (as opposed to a regular one), because the first *SADB_ADDRESS* payload does not contain a source address assigned to the gateway itself (indeed, it contains the source address of the packet which triggered this message). This causes discovery and authorization verification of the remote *OEGW*, and, finally, key-exchange. If all the previous steps succeed, the key exchange daemon sends an *SADB_X_POLICY* message to the IPsec subsystem. This requests protection of subsequent communications between this source and destination via IPsec tunnel mode using the previously discovered OEGW as its tunnel remote endpoint.

Appendix B: Integration with IKE

No changes to the usual IKEv2 message exchanges are required. Rather, when configuring an opportunistic tunnel via IKEv2, an OEGW must use ISAKMP payloads in a specific manner to achieve CBI proof-of-ownership during the initial IKE_AUTH, and to prove, during the subsequent $CREATE\_CHILD\_SA$ exchange, that the OEGW is authorized to provide opportunistic gateway service on behalf of a CGA.

An ID payload of type ID_IPV6_ADDR would not trigger any verification by the peer of the binding between the public key and the CBI. Hence, a new ID type of ID_CBID_128 is needed.

When performing an opportunistic IKE exchange, two certificates needs to be carried, in two separate $CERT$ payloads: (1) An OEGW wanting to prove CBI ownership must send a $CERT$ payload (e.g., X.509) that contains the public key used when generating its CBI, and (2) an OEGW wanting to prove that it is authorized to act as such for a given CGA must send the corresponding $CERT$ payload (e.g., SPKI)

The Traffic Selector (TS) Payload contains headers used to identify IP packet flows which need IPsec processing. In the case of opportunistic encryption, those flows will fly between two CGA's. Hence, we require that the TS payloads used contain CGA's. This implies that the TS Type is set to TS_IPV6_CGADDR, causing the CGAs used as traffic selectors to be validated against the CGAs which issue (and sign) the SPKI authorization certificates contained in the exchanged CERT payloads.