A Silicon Valley Insider

Internet Protocol Security (IPSec)

Authentication Header (AH)
Encapsulating Security Payload (ESP)
Internet Key Management (IKE)

Technology White Paper

Serge-Paul Carrasco

Basic Cryptography

Cryptography is the science of writing and reading coded messages. An original message is called a **plaintext** or cleartext. An encrypted message is called a **ciphertext**. Cryptographers invent secret codes. Cryptanalysts attempt to break secret codes. A **key** is a digital secret that can be used to encrypt, decrypt and sign information.

Four concepts are required for securing data communication:

- **Confidentiality** (or secrecy);
- **Authentication**;
- **Integrity**;
- **Non-repudiation** (non-denying that data was sent).

Symmetric Key Encryption (Private Key Encryption)

Symmetric encryption uses a common key, for both the sender and the receiver, called a **private key** and the same algorithm to encrypt and decrypt the message. Both sender and receiver have to agree on the same secret key and algorithm.

Present private key algorithms include:

- **DES (Data Encryption Standard)** operating on 64-bit message with a 56-bit key and today outdated;
- **3DES** alternative to DES and more secured. It operates like DES on 64-bit message but can use 1, 2 or 3 keys on 64-bits messages;
- **AES (Advanced Encryption Standard)** the most secured algorithm presently. It operates on 128-bit message using 3 different key lengths: 128, 192 and 256 bits.

When messages are not 64 or 128-bits lengths, a number of schemes have been developed such as **Electronic Code Book (ECB)** and **Cipher Block Chaining (CBC).**

Private key algorithms are often used for encrypted data since they can be implemented in hardware and have been optimized for encrypting large amount of data at one time.

The challenges inherent to the use of a private key are:

- Changing the secret key frequently;
- Generating and distributing the secret keys.

A common used algorithm to derive and exchange secret keys securely is the **Diffie-Hellman** algorithm.

Asymmetric Encryption (Public Key Encryption)

Asymmetric encryption uses a **public key** that is a key available to anyone, to encrypt and a private key to decrypt the message. The sender encrypts his message with the public key of the receiver. The receiver decrypts the message with his private key.

The most used public key algorithm is:

- **RSA (Rivest Shamir Adleman)** that generates two keys that are large prime numbers of at least 1,024 bits!

Unfortunately, the RSA algorithm to generate the public/private pairs is fairly processor intensive with keys of 1,024 bits.

So public key encryption is typically used in the beginning of a communication for authentication and for establishing a temporary shared secret key. Authentication is performed by the sender signing with his private key; verification of the signature is done by the receiver with the sender's public key. The secret key is used to encrypt the reminder of the message using private key encryption.

Hash Functions

One-way hash function takes an input message of arbitrary length and outputs a fixed length bit string. The hash function or message digest (MD) has four important properties:

- Given x, it easy to compute MD (x);
- Given MD (x), it is impossible to find x;
- No one can find x' such as: MD (x) = MD (x');
- A change to the input of even 1 bit produces a very different output.

Common hash functions include:

- RSA's **MD4** and **MD5 (Message Digest 4 and 5)**;
- US Government's **SHA (Secure Hash Algorithm)**.

Hash functions are typically used for fingerprinting or **digitally signed messages**.

Digital Signatures

A digital signature is an encrypted **message digest** appended to a message. Digital signature enables authentication of the sender and integrity of the message.

Digital signatures are based on a combination of public key encryption and hash functions.

A digital signature is generated with the following steps (authentication):

- The sender sends its public key to his or her receiver;
- The sender hashes the message and encrypts the message digest with his or her private key;
- The digital signature of the sender is the encrypted hash.

And verified with the following steps (integrity):

- The receiver decrypts the digital signature using the public key;
- The receiver one-way hashes the message received;
- The decrypted digital signature and the hash of the original message must matches.

This assumes that the initial exchange of the public key is performed in a trusted manner.

Keyed-Hash Message Authentication Code (HMAC)

HMAC (also called **keyed message digest)** is a function that authenticates both the source of the message and its integrity. HMAC used hash functions (either SHA or MD5) and takes two inputs: the message and the private key known by the two parties.

The sender produces a value, the MAC, from the message and the key. The receiver computes the MAC on the received message using the same key and HMAC function used by the sender. If the two values match, the sender is authenticated and the message has not been changed.

Creating and Distributing Private Keys

The Diffie-Hellman algorithm provides a way for two parties to establish a shared secret key that only the two parties know even though they are communicating over an insecure channel.

The algorithm has two public system parameters:

- P = large prime number;
- G =generator, an integer lower than p;

The exchange requires:

- Both sender and receiver generates two private numbers: a and b;
- Both sender and receiver generates two public numbers:
    - Sender: X = ga mod(p) ; Receiver: Y = gb mod (p)
- The sender send X to the receiver and the receiver Y to the sender;
- The secret key is gab mod (p) = gba mod (p).

Distributing Public Keys

Public keys are usually distributed through **digital certificates** using a **public key infrastructure (PKI)**. A digital certificate is a digitally signed message that attests the validity of a public key belonging to someone. Digital certificates require the use of a trusted third party called the **certificate authority (CA)**. CA vouches the validity of the certificate. CAs are based on ITU-T X509 standard.

Perfect Forward Secrecy (PFS)

PFS is a property where the compromise of a single key will not compromise subsequent future keys. PFS provides ephemeral single-use keys. One way to do that (as use in IPSec IKE) is to repeat the Diffie-Hellman exchanges.

IPSec Overview

IP packets have no inherent security. There is no guarantee that IP datagrams received are:

- From the source address in the header (avoiding IP spoofing);
- That they contained the original payload (guaranteeing data integrity);
- That the original data was not inspected by a third-party while the packet was being transmitted (avoiding eavesdropping).

IPSec provides **network-layer security services** for IP upper-layer protocols (TCP and UDP):

- **Data origin authentication**: guarantee that the message actually was sent by the apparent originator of the message;
- **Connectionless data integrity**: guarantee that the message that is received is the exact one that was sent;
- **Data content confidentiality**: guarantee that the message cannot be read by a third-party; that is loss of privacy.

IPSec can protect packets between hosts (clients and servers), between security gateways such as firewalls, routers and IPSec specialized encryption devices and between hosts and security gateways.

IPSec is built on a framework of specifications for two reasons. First, not all applications require encryption, which is by nature computationally expensive. Second, the framework can survive even if one piece is found later unsecured.

IPSec Protocols

IPSec' suite of protocols includes:

- **AH (Authentication Header)** which provides:
  - Data origin authentication;
  - Connectionless integrity;
  - Anti-replay protection: the same message is not delivered multiple times and is not delivered grossly out of order.
- **ESP (Encapsulating Security Payload)** which provides:
  - AH capabilities (but with a different scope of authentication coverage);
  - Plus optional data confidentiality e.g. data encryption.
- **IKE (Internet Key Management)** which provides:
  - Dynamic authentication of IPSec peers;
  - Negotiation of security services;
  - Generating shared keys (but keys can be manually set-up).

AH goal is to provide authentication while ESP provides authentication and confidentiality. ESP functionality overlapped with AH (still a topic of discussion). AH and ESP are optional for IPv4 but required for IPv6.

IPSec Architecture

IPSec protocols AH and ESP can be used in two different ways or "modes"

- **Transport mode**: to protect only the upper-layer protocols of an IP payload. IPSec is used between the IP header and the upper-layer protocol payload.
- **Tunnel mode**: to protect the entire IP payload. IPSec encapsulates the whole IP packet and a new IP header is created to transmit the packet.

Because of its construction, transport mode can only be used between end-point communication e.g. the hosts have cryptographic capabilities whereas tunnel mode is typically used by IPSec gateways.

IPSec security services are created between two end-points through a **Security Association (SA)**. SA makes IPSec connection-oriented when IP is connectionless-oriented. Typically, SA exists in pairs, one in each direction. SAs are used for negotiating the encryption algorithms applied to the IP traffic between the peers, based on **IPSec Policy** that define the type of traffic to be protected.

SAs are identified by its **Security Parameter Index (SPI)** and the destination of to which the SA applies. When created manually, SA has not lifetime until it is deleted. When created dynamically, an SA may have a lifetime associated with it.

IPSec Policy defines which traffic to be protected, how to protect it and with whom the protection is shared. IPSec policy is maintained in the **Security Policy Database (SPD)**. A SPD entry may define one of the three actions to take upon traffic match:

- Discard: do not let this packet in or out;
- Bypass: do not apply security services to an outbound packet and do not expect security services on an inbound packet;
- Protect: apply security services on outbound packets and require inbound packets to have security services applied.

IP traffic is mapped to SPD by a **selector** that includes:

- A user ID or a system name identified by DNS or X500;
- Source and destination address (IPv4 or IPv6);
- Source and destination port;
- Transport layer TCP or UDP.

AH and ESP Protocols

Authentication Header (AH)

AH does not provide confidentiality so all the fields are in the clear.

AH includes the following fields:

- Next header: describes the type of the next header (ESP, TCP/UDP, ICMP);
- Payload length: number of 32-bit words in the AH minus 2;
- Reserved: reserved;
- Security Parameter Index (SPI): into to the receiver's SA database (SPD);
- Sequence data: number of messages sent from the sender to the receiver using the current SA to perform replay protection;
- Authentication data (HMAC): a digital signature for that packet that can used either MD5 or SHA.

Encapsulation Security Payload (ESP)

ESP provides authentication like AH and confidentiality. SA for ESP defines one for authentication called authenticator and another one for confidentiality called authenticator?. It is possible to use ESP authentication without confidentiality and the reverse but ESP cannot be used without both.

ESP contains a header not encrypted and a trailer that is partially encrypted.

ESP includes the following fields:

- ESP header:
    - SPI (same as for AH);
    - Sequence number (same as for SPI);
- ESP payload data: subjects to the normal limits of IP;
- ESP trailer:
    - Padding (optional): to ensure that the data will be of the correct length for particular types of encryption;
    - Padding length: total number of bytes of padding;
    - Next header: TCP/UDP or ICMP;
    - Authentication data: either MD5 or SHA.

Internet Key Management (IKE)

IPSec supports manual and automatic distribution of keys. IKE is the default key IPSec key management protocol.

IKE is a hybrid of **Oakley** and **SKEME (Secure Key Exchange Mechanism for the Internet)** protocols and operates inside a framework defined by **ISAKMP (Internet Security Associations and Key Management Protocol)**:

- ISAKMP (developed by NSA): provides a framework for authentication and key exchange but does not define them. ISAKMP is designed to be key exchange independent.
- Oakley (developed by Hilarie Oakley): defines a series of key exchanges called modes and the services provided by them.
- SKEME (developed by Hugo Krawcyk): defines a versatile key exchange technique.

IKE is a hybrid protocol that uses the foundation of ISAKMP, the modes of Oakley and the share and rekeying techniques of SKEME to define its own unique way of deriving authenticated keys and negotiating shared policy.

IKE can be used for other Internet protocols such as SNMPv3 and OSPFv2. The specification of what IKE is being used for is done in a **Domain of Interpretation (DOI)**. For IPSec, DOI is defined in RFC 2407.

IKE Overview

IKE authenticates each peer involved in IPSec, negotiates the security policy and handles the secure exchange of session keys. IKE is a request-response protocol with an **initiator** and a **responder**.

IKE creates first an authenticated secure tunnel between two entities called **IKE phase I** and then negotiates the IPSec SA, called **IKE phase II**. These two phases, IKE phase I and phase II are performed with the following steps:

- Traffic is generated or received by one of the IPSec peers that is identified to require IPSec protection to its destination;
- IKE phase I creates an IKE SA between the two IPSec peers that enable a secure communication channel;
- IKE phase II results in the creation of two IPSec SAs between the two IPSec peers. This pair of unidirectional SAs creates the secure IPSec tunnel;
- Data starts passing between the IPSec peers over the established secure IPSec tunnel.

IKE SAs are different from IPSec SA but both stored in the SADB. IKE SAs are used to negotiate the algorithms to encrypt IKE traffic and authenticate the peers. There is only one IKE SA between peers. More than one pair of IPSec SAs may be created at once using a single IKE exchange, and a number of such exchanges may be performed by a single IKE SA.

IKE Phase I

In an IKE phase I exchange, the two ISAKMP peers establish an ISAKMP or IKE SA that is a secure, authenticated channel with which to communicate. There are two ways in which this IKE SA can be created in phase I: **main mode** and **aggressive mode** but not both.

- Main mode: consists of six messages, three in each direction, exchanged between the initiator and the responder.
- Aggressive mode: consists of three messages speeding up the IKE negotiation but providing lower security.

Most IPSec implementations use IPSec main mode.

The IKE SA has various parameters that are negotiated between two peers. Mandatory parameters include:

- Encryption algorithm (and key length): DES, 3DES and AES;
- Hash algorithm: MD5 or SHA-1;
- Authentication method:
    - Pre-shared keys using an out-band method (mandatory);
    - Digital signature using RSA;
    - Other methods:
        - Public key original mode;
        - Public key revised mode.
- Diffie-Hellman exchange parameters.

Optional parameter includes:

- Lifetime: determines the life of the IKE SA, configured either in seconds or kilobytes.

Messages in phase I exchange **cookies** prior to performing the Diffie-Hellman calculation in order to verify that the peer exists and interested in conducting IKE exchange. All messages are identified by a **message ID**.

IKE Phase II

IKE Phase II called quick mode establishes the IPSec SA in each direction. Quick mode involves three exchanges.

New shared keys are generated for encrypting the traffic across the IPSec tunnel. These keys can be derived by using a new Diffie-Hellman exchange or by refreshing the shared secret derived from the IKE phase I exchange by hashing it with a nonce. The first method is slower but provides greater security through **perfect forward secrecy (PFS)**.

