Securing the Internet: A cryptographic viewpoint of IP security

Ran Canetti
IBM Research

canetti@watson.ibm.com

Today's Internet

- Susceptible to many types of attacks:
  - Eavesdropping
  - Impersonation
  - Malicious corruption of data
  - Denial of service, clogging
  - Sabotage

- Attacks may have many sources:
  - Other parties
  - The network:
    - Physical lines
    - Hosts on the local network
    - Routers
    - Wireless lines

⇒Want to engineer "secure Internet".

Requirements from a “secure Internet”

Many different (even contradictory) concerns:

Integrity and authentication: The ability to verify whether the received data:
— Originated with the ‘real’ sender.
— Was not corrupted on the way.

Confidentiality: The ability to send data so that only a predefined receiver(s) will know the contents.

Non-repudiation: The ability to commit to data or actions so that it will be impossible to later deny the commitment.

Repudiation: The ability to communicate (in an authenticated way) and later deny that the communication took place.

Requirements from a "secure Internet" (II)

Anonymity: The ability to hide the identity of the communicating parties from third parties. (As much as possible...)

Auditing and accounting: The ability to record relevant information about the communication.

Availability: Robustness to clogging and denial-of-service attacks.

Different trust models

- Internal corporate network: Central trusted server.

- Communication between corporate sites, across firewalls.

Different trust models (II)

- Communication between two home sites, with no commonly trusted entity.
- Communication between nomad/mobile user and base.
- Communication between a Web server and a browser.
- Multicast.

Why put security in IP?

Basic fact: No single solution can answer all the concerns in all scenarios.

— Still, want a small set of widely accessible solutions that will answer most concerns.

⇒IP layer: A good candidate for a unified solution.

- Secures even TCP/IP headers, firewalls
- Common to all Internet communication
- No need to modify application code
- Simple: Single type of communication (packets with known format).

(Disadvantages: No notions of session or user. Need to modify kernel.)

The proposed solution: high level view

The idea: Create a virtual “secure channel” between the endpoints:

Establishing a secure channel:

Step 1: Agree on cryptographic algorithms and on a secret key.

Step 2:
— Sender: Use the agreed algorithms and key to authenticate/encrypt each packet.
— Receiver: Use agreed info to verify/decrypt packets.
— Intermediate parties (routers, bridges, hosts etc) play no part in the process.

A basic concept: Security Associations [RFC 1825]

A Security Association (SA) is a record containing all security info on a connection required by the IP. Includes:

- Algorithm used for authentication
- Algorithm used for encryption
- Source address
- Lifetimes of keys

Secret info:

- Authentication key
- Encryption key

⇒Step 1 now means: Agree on a CA.

Who are the communicating principals?

- Primary intention:
  - Hosts
  - Firewalls

- Also possible: Users (somewhat problematic).

Nesting is also possible:

Basic Architecture

The key-determination protocol will:

— Negotiate the security parameters (algorithms, etc.)
— Agree on a secret key ("RKM")

- The protocol is above the IP layer and separate from it.

- The protocol establishes a SA, and generates a Security Parameters Index (SPI).

- Once the protocol is completed, IP needs only the SPI.

Basic Architecture

Step 2: After agreeing on a SA

Possible modes:

Authentication [RFC 1826]: For each outgoing packet, generate a Message Authentication Code (MAC). Put the outcome in a new header, called "Authentication Header" (AH).

Authentication + encryption [RFC 1827]: ("Encapsulating Security Payload - ESP".) Authenticate and Encrypt each packet.

Authentication

— Requirements from transform M:
(Let $a_i = M_k(p_i)$)

-Adversary sees pairs $(p_1, a_1), ...(p_n, a_n)$
-Cannot generate a valid new pair (p', a')

— Chosen paradigm for M: keyed hash functions.

Keyed hash for authentication

Cryptographic hash functions (MD5, SHA):

- Hash any input x to a short (128/ 160 bits) output $H(x)$
- Infeasible to find x, y such that $H(x) = H(y)$.

Are widely available, relatively fast, simple.

Q: How to use as a MAC?
A: incorporate the key.

A provably secure method — HMAC [BCK]:

$M_k(x) = H(k \oplus pad1(H(k \oplus pad2, x))$

— HMAC-SHA and HMAC-MD5 are proposed as mandatory transforms.

Replay prevention

Above scheme susceptible to replay attacks.

⇒ First solution: Use a counter.

Sender includes a count number in each packet.
Receiver accepts only if number is greater than last packet.

— But packets arrive out of order...

⇒ Receiver will allow a *window* of numbers.

Authentication + encryption

Encryption options:

- Encrypt only payload data:

- Encrypt also the IP header (tunneling):

But...

Simple encryption is not enough: encrypted packets can be maliciously corrupted en-route.

$\Rightarrow$Need authentication + encryption.

Chosen method:

$$ESP_{k_{auth},k_{enc}}(p) = E_{k_{enc}}(p, HMAC_{k_{auth}}(p))$$

— Optional replay prevention.

Mandatory encryption transformation

DES-CBC:

-Optional: 3DES-CBC.

Step 1: Agreeing on SA ("key agreement")

The problem: How to agree on a secret key?

- Solution 1: Manual keying.
  (Non scalable. Very limited applicability.)
- Solution 2: Key distribution centers (KERBEROS). Centralized. Requires trusted third parties. Vulnerable. (Good for medium size corporate sites.)

Solution 3: Use public key cryptography:

— Each party has a pair of keys, public and private.

— Private key used for signing/decrypting (by the party).

— Public key used for verifying/encrypting (by other parties).

The problem: How to correctly obtain the public key of partner.

Solutions: Public key infrastructure:

— Certificate authorities.

— Secure DNS.

- Public key operations are expensive. Should minimize usage.

A basic technique:
Diffie-Hellman key exchange [DH].

Common knowledge: a large field F (say, Z_p, for a prime p) and a generator g of F.

- Directory version:

- Key exchange version:

Perfect Forward Secrecy (PFS)

— A protocol offers PFS if an adversary that finds the current secret keys still cannot decrypt information encrypted in the future.

— Standard way to achieve PFS:
Ephemeral [DH] (Key-exchange version).

ISAKMP/Oakley key agreement protocols [O,MSST,HC]

Modes of operation:

- Key agreement with PFS and non-repudiation. (DH+signatures.)
- Key agreement with PFS and repudiation. (DH+ PK-encryption.)
- Key derivation with PFS.
- Fast key derivation without PFS.
- Fast(er) key agreement without PFS.

First mode: a la STS [DVW].

Other modes: a la SKEME [K].

Key agreement with PFS and non-repudiation

```
Initiator                             Responder
----------                           ------------
HDR, cookie, SA               -->
                              <--  HDR,  cookie, SA
HDR, g^xi, NONCE (Ni)         -->
                              <--  HDR, g^xr, NONCE(Nr)
E(HDR|SIG(g^{xi*xr}|IDi)) -->
                              <--  E(HDR|SIG(g^{xi*xr}|IDr))


        KEY = prf(Ni|Nr , g^{xi*xr}|...)
```

Cookies: mechanism for preventing clogging attacks. (Hash of sender's IP address and data.)

prf: pseudorandom function. For "mixing" data. (e.g., HMAC,DES)

— Uses signatures ⇒non-repudiation.

— Ephemeral DH exchange ⇒PFS.

Key agreement with PFS and repudiation

```
Initiator                               Responder
---------                              -----------
HDR, cookie, SA           -->
                          <--    HDR, cookie, SA
HDR, g^xi, E_Kr[Ni|ID]    -->
                                 K0 = H(Ni|Nr)

                          <--    HDR, g^xr, E_Ki[Nr|IDr],
                                 prf(K0 , g^xr|g^xi)
K0 = H(Ni|Nr)

HDR, prf(K0 , g^xi|g^xr) -->

         KEY = prf(K0 , g^{xi*xr}|...)
```

— No signatures ⇒repudiation.

— Ephemeral DH exchange ⇒PFS.

Fast key agreement without PFS

```
Initiator                                  Responder
---------                                 -----------
HDR, cookie, SA              -->
                             <--   HDR, cookie, SA
HDR, Ri E_Kr[Ni|ID]          -->
                                   K0 = H(Ni|Nr)

                             <--   HDR, Rr, E_Ki[Nr|IDr],
                                   prf(K0 , Rr|Ri)
K0 = H(Ni|Nr)

HDR, prf(K0 , Ri|Rr)         -->

         KEY = prf(K0 , IDi |IDr | ...)
```

— No signatures ⇒repudiation.

— No ephemeral DH exchange ⇒no PFS.

Key derivation with PFS

```
Initiator                          Responder
---------                         -----------

    K                                  K

HDR, cookie, SA           -->
                          <--   HDR, cookie, SA
HDR,  g^xi, MAC_K(g^xi)   -->
                          <--   HDR,  g^xr, MAC_K(g^xr)

         KEY = prf(K , g^{xi*xr} | ...)
```

— Ephemeral DH exchange $\Rightarrow$PFS.

Fast key derivation without PFS

```
Initiator                               Responder
---------                              -----------

    K                                       K

HDR, cookie, SA           -->
                          <--   HDR, cookie, SA
HDR,  Ni,                 -->
                          <--   HDR,  prf(K , Ni), Nr
prf(K , N_i|Nr)           -->
         KEY = prf(K , prf(K , Ni|Nr))
```

— No ephemeral DH exchange $\Rightarrow$no PFS.

In-line keying: the SKIP protocol [AMP]

Idea: use the Directory mode of [DH].

- Each party I will have secret x_i, and g^{x_i} in the DNS.

-When party I sets up a connection with J, it:

- Obtains g^{x_j} from DNS, and computes: $K_{ij} = g^{x_i x_j}$, $K_{ijn} = H(K_{ij}$,counter,time)
- Chooses a random key K_s and computes $[K_s]$ =Encrypt$_{K_{ijn}}(K_s)$.
- Prepends $[K_s]$ to each packet sent. (in a special SKIP header).
- Encrypt/authenticate each packet using K_s.

Packet format under SKIP:

The SKIP protocol

Big advantage: Simple, no two-way interaction needed. (True to the concept of connection-less IP layer.)

Drawbacks:

- Offers limited security:
  - x_i is a vulnerable, long-lived, very attractive secret.
  - "Single point of failure" (no Perfect Forward Secrecy).
  - Receiver not assured of freshness of K_s.
  - Limited anonymity.
- No negotiation of SA.
- Per-packet overhead.
- Does not comply with the architechture.

Conclusion

- Overviewed security threats and concerns in today's Internet.
- Discussed the reasoning behind placing security mechanisms in the IP layer.
- Sketched the proposed solutions:
  - Architechtural concepts.
  - Algorithms for authentication and encryption: DES-CBC, HMAC.
  - Key exchange mechanisms: SKIP, ISAKMP/Oakley.