Hash functions, certification and secure protocols

Contents

Hash functions

Signature can be used only for small sized messages.
Naive solution : cut the message to sign into fixed sized blocks ; then sign independently each block
Many problems

- the size of the signature becomes huge
- signing algorithms are pretty slow

Solution : use a hash function

Use a **cryptographic hash function**, quick to compute ; transforms a message of arbitrary length into a fingerprint of fixed size. Then, sign the fingerprint

message	x	arbitrary length
	$\downarrow$	
fingerprint	$z = h(x)$	160 bits
	$\downarrow$	
signature	$y = \text{sig}_{sk}(z)$	depends upon the signature

Principle : When Bob signs x he first computes the fingerprint $z = h(x)$, then he signs with $y = \text{sig}_{sk}(z)$ and sends the pair (x, y). Everyone can check the validity by

1. re-computing the fingerprint $\hat{z} = h(x)$
2. using the verification algorithm, $\text{ver}_{pk}(\hat{z}, y)$.

Conditions to fulfill

A hash function h computes

$$z = h(m)$$

for m a message of arbitrary length ; z is a fixed size fingerprint.
We require h to be **one way**, i.e.

- $h(m)$ must be easy to compute from m
- z must be hard to invert

Collision of h : pair of distinct words (x, x') st $h(x) = h(x')$.
h is **weak collision resistant** if, given a x, it is difficult to find a collision.
h is **strong collision resistant** if it is difficult to find any collision (x, x').

Birthday paradox

Given : $B = (b_1, \ldots, b_k) \in \{1, 2, \ldots, n\}^k$.
Problem : proba p to have at least 2 identical elements in B ?
Let us consider k messages m_i randomly chosen with $i \in [1, k]$ and we consider the proba that two m_i have the same image.
$z_i = h(m_i)$. Proba that all z_i are different :

$$1-p = q = \frac{1}{n^k}\prod_{i=0}^{k-1}(n-i) = \prod_{i=1}^{k-1}\left(1-\frac{i}{n}\right) = 1\left(1-\frac{1}{n}\right)\cdots\left(1-\frac{k-1}{n}\right)$$

Where 1 is the probability to choose z_1, $(1-\frac{1}{n})$ the probability to choose $z_2 \neq z_1$ (since there is one chance over n that $z_1 = z_2$),. . . , $(1-\frac{i}{n})$ the probability to choose $z_i \neq z_1, \ldots, z_{i-1}$.

We approximate $\prod_{i=1}^{k-1}\left(1-\frac{i}{n}\right)$ by $\prod_{i=1}^{k-1} e^{-\frac{i}{n}}$ since $e^{-x} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!}\cdots$, thus $e^{-x} \approx 1 - x$ for x small (which is our case). Thus, $1 - \frac{i}{n} \approx e^{-\frac{i}{n}}$.

$$q = \prod_{i=1}^{k-1}\left(1-\frac{i}{n}\right) \approx e^{-\frac{1}{n}\sum_{i=1}^{k-1} i} = e^{-\frac{(k-1)k}{2n}}$$
$$\ln q \approx -\frac{(k-1)k}{2n}$$
$$2n\ln\left(\frac{1}{q}\right) \approx k^2$$
$$\sqrt{2n\ln\left(\frac{1}{1-p}\right)} \approx k$$

$p = 1/2$, proba to have at least one collision for $k \approx \sqrt{2n\ln 2}$
Example : $n = 365$, $k = 23$ people ; we have more that proba 1/2 to have 2 people with the same birthday day.
Application : find the size n of the image by the hash function to avoid collisions. We have k in $O(\sqrt{n})$.

Attack based on the paradox

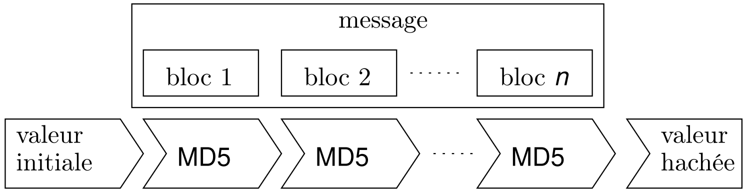
Compute and sort as many pairs $(x, h(x))$ as possible. Detect one (ore more) collisions.
There are 2^n values corresponding to the birthdays ; .
We assume that the images by h are uniformly distributed.
If we consider k inputs, we have more than 1/2 chance to find a collision when $k \approx 2^{\frac{n}{2}}$. Taking the logarithm,

n	50	100	150	200
$\log_2 k$	25	50	75	100

Thus, by computing a little bit more than $2^{n/2}$ images by h, we can find a collision with proba $> 1/2$.
For h strongly resistant, we choose n great enough to avoid that the computation of the $2^{n/2}$ images by h be feasible. Currently, $n \geq 160$.

One-way compression function

Like MD5, m is split into n blocs, each of fixed length and the following is applied :



Compression Function

From : $e_k : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$
one can design a compression function

$$g : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n \qquad \text{for } n \in \mathbb{N}$$

whose image size is n.
The cipher is used either directly if it is collision-resistant or by modifying it :

$$g(k,x) = e_k(x) \oplus x$$
$$g(k,x) = e_k(x) \oplus x \oplus k$$
$$g(k,x) = e_k(x \oplus k) \oplus x$$
$$g(k,x) = e_k(x \oplus k) \oplus x \oplus k$$

Classical construction

Start from your favorite e_k, and build a compression function :

$$g : \{0,1\}^m \to \{0,1\}^n \qquad \text{for } m,n \in \mathbb{N}, \quad m > n$$

Use function g to build a hash function :

$$h : \{0,1\}^\star \to \{0,1\}^n \qquad \text{for } n \in \mathbb{N}$$

Proposition
h collision resistant if g collision resistant

Hash Function

Merkle : constructs a hash function from a compression function $g : \{0,1\}^m \to \{0,1\}^n$.
Let $r = m - n > 1$. We want to build $h : \{0,1\}^\star \to \{0,1\}^n$.
Let $x \in \{0,1\}^\star$ and ℓ its length in binary

- fill x with "0" : $u = 0^i x$ st $|u| \equiv 0 \mod r$
- fill ℓ with "0" : $y = 0^j \ell$ st $|y| \equiv 0 \mod r-1$
- cut y in blocks of $r-1$ bits and add a "1" at the beginning of each block to form the word v
- build $w = u0^r v$ made of t blocks of size r.

Example : $r = 4$, $x = 11101$, $\ell = 101$. $u = \mathbf{000}1\,1101$, $v = \mathbf{1}101$.

$$w = 0001\,1101\,\mathbf{0000}\,1101 = w_1 w_2 w_3 w_4 \ (t = 4)$$

H inductively defined : $H_0 = 0^n$ et $H_i = g(H_{i-1} w_i)$, $1 \le i \le t$

$$h(x) = H_t$$

Modern hash functions

The hash functions which are commonly used are designed according to the previous construction.

name	bits	round×steps	relative speed
MD5	128	4×16	1
SHA	160	4×20	0,28

Application to DSA
Digital Signature Algorithm is a signature standard combining the use of a hash function (MD5 or SHA) and DSS, the latter being an improvement of El Gamal's signature scheme.

PK Certificate

A certificate of *B*'s PK contains *B*'s identity together with PK_B signed by a third party.
Usage : counter MIM attacks
A certificate contains

- the public key
- informations relative to *B*'s identity (name, e-mail. . .)
- the signature by a third party, Ivan

Ivan signs

- the key
- the informations relative to *B*

Ivan guarantees the correctness of those informations and that the public key corresponds to *B*'s identity.

Contents

How it works

Certification is done by the means of a **signature scheme**.
It consists in [2] :

- signing after hashing
- providing a verification algorithm

Example : if the contents of the certificate follows X509 norm, we provide a **digital id** like a numerical identity card.

(X.509) Certificate

Associates a public key to the identity of a subject ; it contains :

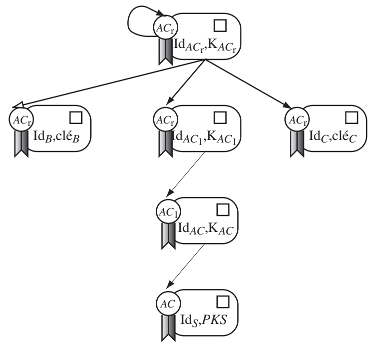
- **Subject** : Distinguished name ; public key
- **Issuer** : Distinguished name, signature
- **Period of Validity** : not before, not after
- **Administrative Information** : version, serial number
- **Extended Information** :

The information « Distinguised name » contains :

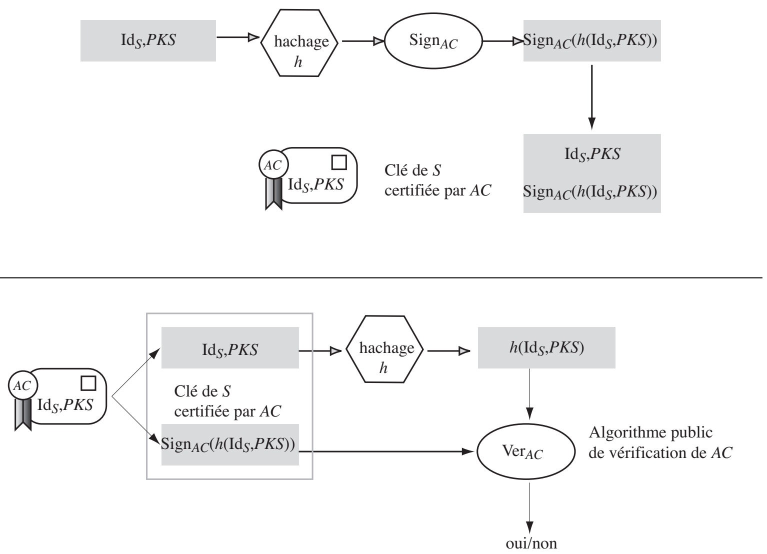
- **Common Name** : name to be certified *Bruno Martin*
- **Organization| Company** : context *UNS*
- **Organizational Unit** : more specific *Deptinfo*
- **City/Locality** : town *Sophia Antipolis*
- **State/Province** : for US *PACA*
- **Country** : country code *fr*

Certification Chain

CA also provides a certificate to another CA.
Alice can traverse the certification chain until she finds a CA she trusts to check the validity of the relation (Id_S, *PKS*)



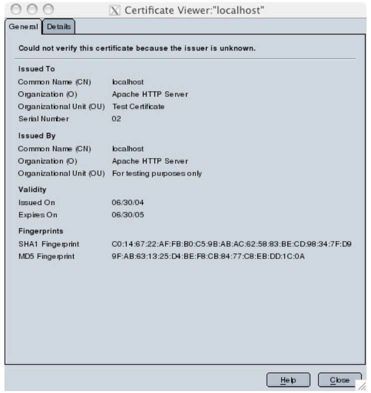
Certification & Verification



Root CA creation

Problem of the certification chains : we need a root CA.
This root CA cannot be certified : its certificate is self-signed.
The trust relies on a wide distribution of the root CA's public key.
Clients and servers are configured to trust some root CA by default like `CertiSign` or `VeriSign`.
Those firms propose techniques to request for signatures, have procedures for verifying the information and they sell, provide and manage certificates.
Note that, by default, `openssl` is not preconfigured with any trusted root certificates. They're provided by the OS vendor or embedded in software applications (firefox).

With no trusted CA...



Key exchange

All ciphers require the keys to be securely exchanged.
Obvious with symmetrical ciphers and PKC to counter MIM.

What are the solutions ?

Some solutions

1. Fix a meeting to exchange the keys
2. Sending the key by surface mail
3. Use a key previously shared by both parties and compute a new key

First two cases : not always possible ; if two army corps are isolated.

Contents

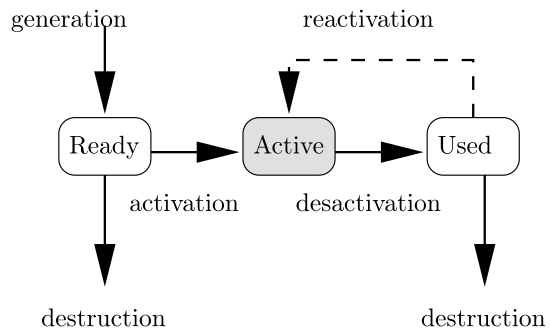
Key management techniques [1]

Make use of

- enciphering mechanisms
- key usage
- security policy

counter :

- modification
- unintended disclosure
- relay
- modification



During the lifetime of the keys, we need to ensure :

secure generation	suppression	revocation	certification
storing	distribution	destruction	installation

Models for key establishment

Process which makes a key available to one or several entities ; covers :

- key agreement
- key transportation (public, secret)
- key update
- key derivation

Key agreement

Imagine a solution based on the problem hardness (complexity) which is easy to compute for legitimate users and hard for an attacker.
We use a one-way function.
A good candidate is the discrete log. problem.

Diffie Hellman key agreement protocol

Let q be a big prime and a, $1 < a < q$.
Each user U

- randomly selects a secret value X_U, $1 < X_U < q$
- publishes $Y_U = a^{X_U} \mod q$

A and B build a shared key only known by them :

- A computes $K = (Y_B)^{X_A} \mod q$
- B computes $K = (Y_A)^{X_B} \mod q$

A and B share the key K :

$$\begin{aligned} Y_B^{X_A} &\equiv (a^{X_B})^{X_A} \equiv a^{X_B X_A} \equiv \\ &\equiv a^{X_A X_B} \equiv (a^{X_A})^{X_B} \equiv Y_A^{X_B} \mod q \end{aligned}$$

Security

1. Shared keys are secure : if an attacker is able to compute the key, X_A of A from $Y_A = a^{X_A} \mod q$, he must solve DLP
2. Is it possible to find the shared key from the published information ? It is known as hard as solving DLP.

Secret keys transport mechanism

Process which allows to transfer a secret key by an entity to another entity.
By using ciphers either asymmetrical or symmetrical.
ISO/IEC 11770-2 and 3 define 18 mechanisms, 5 are point to point, the remaining ones use a trusted third party as key distribution center. For short : distribution

- in the same domain
- between domains

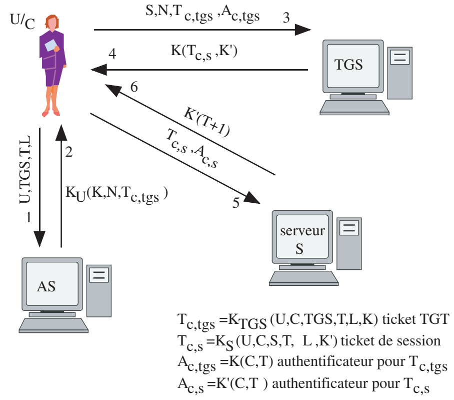
Other examples, see `http://www.microsoft.com/technet/prodtechnol/windows2000serv/reskit/distrib/dsch_key_xihm.mspx?mfr=true`

Kerberos

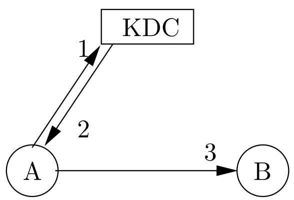
Allow a user connected on a client to prove his identity to a service or an application server without transmitting its credentials over the network.
Requires a trusted third party acting as a **key distribution center** (KDC) for the domain ; it is made of :

- authentication server (AS)
- ticket granting service (TGS)

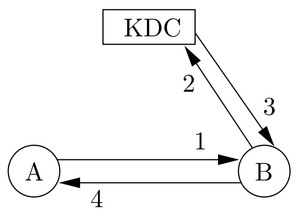
which are both secured



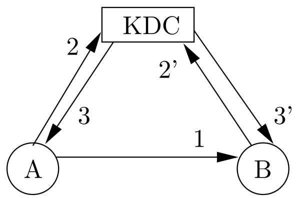
Models for key distribution



Model pull



Model push



Mix Model

Keys update

Let the key evolve session after session : **key update**. *Process which allows to share keys previously constructed by updating them by the means of a session parameter.*
A new session key K is defined from :

- a shared key K_{AB}
- a parameter F (random, time stamp, sequence number)
- a key updating function f

Works in two steps :

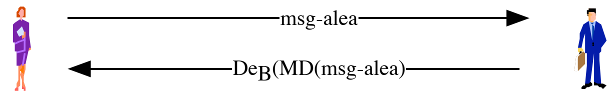
1. the initiator A chooses a derivation parameter F which is transmitted to B.
2. A and B compute the new key K by f st

$$K = f(K_{AB}, F)$$

Example of function f **:** crypto hash function h applied to the data concatenation : $K = h(K_{AB}; F)$

Contents

Example of asymmetrical authentication



If we do not use a fingerprint (MD) of the random message, this protocol can be attacked by a known plaintext/ciphertext attack : (msg-alea/De_B(msg-alea)).
Requires that Alice knows Bob's public key prior the use of the protocol.

Identification & auth

Authentication : process of determining whether someone or something is, in fact, who or what it is declared to be (I'm Bruno and here's the proof of my identity)

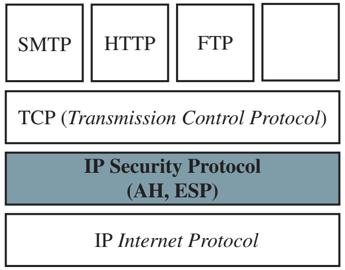
Identification : permits to prove the identity of an entity by means of its identifier (I'm Bruno)

Identification gives the entity's identity and authentication to check its validity.

Contents

IP Security

Add crypto techniques (IPSec working groups) to the Internet standard protocols. The IP security architecture provides security mechanisms (described in RFC1825) which provide authentication, integrity, access control and confidentiality services.



SSL & TLS

SSL provides *authentication, compression, integrity, confidentiality.*
allows several auth. or confidentiality mechanisms and secures all applicative protocol.
SSL becomes TLS, a standard, by IETF. It contains two layers :

- **Agreement** or *Handshake Protocol*
- **Communication** or *Record Protocol*

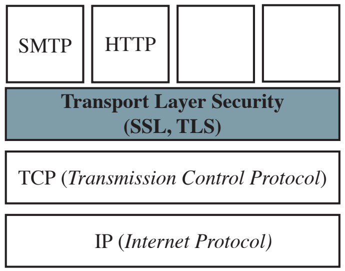
which provide the following services :

- **connection confidentiality** by AES, Camellia, DES, 3DES
- **connection integrity** by a MAC using a non-zero IV (SHA-1 or SHA256 or SHA384)

TCP security

Protocols used to secure TCP :

- **Secure Socket Layer** used by netscape
- **Private Communication Technology** by Microsoft (stopped with SSL3)
- **Transport Layer Security** IETF standard



Current libraries for TLS : **BoringSSL** designed by Google (2015) **OpenSSL**, **LibreSSL** coming from OpenBSD and **GnuTLS**.

Authentication

This is how Alice verifies Bob's identity.
Let us call *SKB* Bob's private key and *PKB* its public key

$$\begin{array}{c|l} A \rightarrow B & r = \text{a random message} \\ B \rightarrow A & c = \{r\}_{SKB} \end{array}$$

But signing a random message r given by someone and sending the signature can be dangerous.
An idea would be to use a hash function h : Bob signs $h(r)$ but the danger remains.

Authentication

It's better if Bob signs a message he has chosen provided he avoids sending m and its signature together :

$A \to B$	*"Hi, are you Bob ?"*
$B \to A$	$m =$ *"Alice, I'm Bob"*
	$c = \{h(m)\}_{SKB}$

Identification

Alice does not know Bob's PK in advance. How to securely send his PK ?

$A \to B$	*"Hi"*
$B \to A$	*"Hi, I'm Bob. Here's my PK" PKB*
$A \to B$	*"Prove it."*
$B \to A$	$m =$ *"Alice, I'm Bob"*
	$c = \{h(m)\}_{SKB}$

Anybody can usurp Bob's identity for Alice by giving his own PK (MIM).

Transmit a certificate

A certificate provides evidence between an identity and the corresponding PK.

$A \to B$	*"Hi"*
$B \to A$	*"Hi, I'm Bob. Here's my certificate"* $cert_B$
$A \to B$	*"Prove it."*
$B \to A$	$m =$ *"Alice, I'm Bob"*
	$c = \{h(m)\}_{SKB}$

Marjorie could usurp Bob's identity during the 3 first exchanges but it would fail after. (Tell when it might not be the case)

Exchange a secret

Securing communications with public key crypto is costly. Once the authentication step is completed, it's better to share a secret key to use a symmetrical cipher.

$A \to B$	*"Hi"*
$B \to A$	*"Hi, I'm Bob. Here's my certificate"* $cert_B$
$A \to B$	*"Prove it".*
$B \to A$	$m =$ *"Alice, I'm Bob"*
	$c = \{h(m)\}_{SKB}$
$A \to B$	*"Ok Bob, here's our secret :"*
	$s = \{\text{secret}\}_{PKB}$
$B \to A$	$m' = \{\text{message from Bob}\}_{\text{secret}}$

Attack

Melchior, the man in the middle can be active during the 5 first steps. At step 6, he can scramble Bob's message and Alice receives an un-readable message :

$$
\begin{array}{ll}
B \rightarrow M & m' = \{\text{message from Bob}\}_{\text{secret}} \\
M \rightarrow A & m' \text{ changed}
\end{array}
$$

Alice has no proof of Melchior's existence, even if she finds suspicious Bob's last message.

Communication

This protocol allows to send messages of arbitrary size. It splits it into blocks, eventually compresses, adds a MAC, enciphers and adds a sequence number to ensure integrity.

SSL

To counter this attack, it's better to use a MAC :

$$M = h(\text{ message from Bob}||\text{secret})$$

$$
\begin{array}{l|l}
A \rightarrow B & \text{"Hi"} \\
B \rightarrow A & \text{"Hi, I'm Bob. Here's my certificate" } cert_B \\
A \rightarrow B & \text{"Prove it".} \\
B \rightarrow A & m = \text{"Alice, I'm Bob"} \\
 & c = \{h(m)\}_{SKB} \\
A \rightarrow B & \text{"Ok Bob, here's our secret :"} \\
 & s = \{\text{secret}\}_{PKB} \\
B \rightarrow A & m' = \{\text{message from Bob}\}_{\text{secret}}||h(\text{message from Bob}||\text{secret})
\end{array}
$$

Melchior can scramble everything, but Alice will be warned of Melchior's existence.

References

W. Fumy.
Key management techniques.
In *State of the art in applied cryptography*, number 1528 in LNCS, pages 209–223. Springer Verlag, 1997.

RSA Laboratories.
PKCS ♯1 v2.0, RSA cryptography standard.
Technical report, RSA Data Security, 1998.