Public Key Cryptography, Digital Certificates, Transport Layer Security and Internet encrypted services

Roberto Innocente

inno@sissa.it
Summary

• PKC (Public Key Cryptography)
  Introduction
• Digital Certificates
• SSL/TLS
• Use of SSL/TLS over Internet
• Encrypted services: pop, imap, smtp
Cryptographic systems taxonomy

- **Symmetric key cryptography**
  - same key for encryption and decryption
  - relatively fast
  - RC2, RC4, DES, triple DES

- **Asymmetric key cryptography**
  - different keys for encryption/decryption
  - slow
  - RSA, ElGamal, Elliptic curves
Symmetric Key Cryptography

Encryption

Plain text \( m \)

message

cipher text

K

key

Decryption

Cipher text \( c = K(m) \)

Plain text \( m = K(c) \)

K.K = 1
Asymmetric Key Cryptography

Encryption

\[ m \rightarrow c = E(m) \]

Decryption

\[ c \rightarrow m = D(c) \]

**m** message

**c** cipher text

**E** encryption key

**D** decryption key

\[ D \cdot E = 1 \]
Another classification

• Secret Key Cryptography
  • the key is kept secret
  • it requires a secure channel to be transmitted

• Public Key Cryptography
  • one key (the deciphering key) is kept secret
  • the other key is made public
Public-Key Cryptography
Diffie – Hellmann (1976)

• Each user generates a pair of inverse transformation $E$ and $D$.
• The deciphering key $D$ must be kept secret but need never be communicated on a channel.
• The enciphering key $E$ can be made public by placing it in a public directory (Public File).

The original idea here is that keys can be produced in pairs and that it can be very hard to generate a key from the other.
PKC algorithms

Since DH idea in 1976 many algorithms have been proposed, most were discovered insecure, of the remaining many are not feasible. Some of the algorithms are:

- *Knapsack algorithms* (later shown insecure)
- *RSA* (still considered secure)
- *El Gamal* (still considered secure)
Knapsack algorithms/1
(!!! insecure !!!)

First PKC algorithm proposed by Merckle and Hellmann in 1978.

• Given n integers $M_1, M_2, M_3, \ldots$, and a sum $S$, find a binary sequence $b_1, b_2, b_3 \ldots$ such that
  • $S = b_1 * M_1 + b_2 * M_2 + b_3 * M_3 + \ldots$

• where:
  • $M_1, M_2, M_3, \ldots$ is the public key
  • $b_1, b_2, b_3, \ldots$ are the bits of the plain message
  • $S$ is the ciphertext

In general it is an hard problem, but ...
Knapsack algorithms/2
(!!! insecure !!!)

A subclass of the general problem can be easily solved and mapped onto a more general one.

- A *superincreasing knapsack* is a knapsack in which every number in the ordered sequence is greater than the sum of the preceding numbers e.g. \{2,3,6,13,27\}
- Solving the problem for a *superincreasing knapsack* is quite easy. Starting from the greatest number, that will be an addend if it is less than the sum \(S\), and so on ...
- Now, choosing a number \(m\) (=55) greater than the sum of all numbers in the sequence, and a number \(n\) (=29) prime with \(m\), and taking the remainder module \(m\) of the numbers in the sequence multiplied by \(n\) e.g. \{2*29\mod 55 = 3, 3*29\mod 55 = 32,...\} we obtain a knapsack that is not *superincreasing*... if we take this sequence as the public key, and the underlaying *superincreasing* sequence as the private key ...
RSA/1

(Rivest, Shamir, Adleman 1978)

- Choose primes \( p, q \) \( n=p*q \)
- Choose encryption key \( e \) prime with \( (p-1)*(q-1) \)
- Compute the inverse \( d \) such that
  \[ e*d = 1 \mod n \]
- now for each message \( m \):
  \[ c = m^e \mod n \]
  \[ m = c^d \mod n \]
- \( n,e \) is the public key
- \( d \) is the private key
RSA/2

- **Fermat’s little theorem** (p prime, (p|a)=1):
  \[ a^{p-1} \equiv 1 \pmod{p} \]

- **Euler Totient function**:
  \[ \phi(n) = \# \text{ of integers less than } n \text{ primes with } n \]
  For p,q primes:
  \[ \phi(p) = p-1 \]
  \[ \phi(p*q) = (p-1)*(q-1) \]

- **Euler’s generalization of Fermat’s theorem**:
  \[ a^{\phi(n)} \equiv 1 \pmod{n} \]
  \[ a^{\phi(n)-1} \equiv a^{-1} \pmod{n} \text{ therefore: } e^{-1} \equiv e^{(p-1)(q-1)-1} \pmod{n} \]
RSA/3

• **Software speedups:**
  • RSA goes faster if you choose $e$ carefully
    • 3 (PEM), 65537 (X.509), 17 (PKCS#1) are good choices having only 2 bits set
    • in particular 65537 requires only 17 multiplications to exponentiate

• **Hardware chips:**
  • it’s about 1000 times slower than DES
  • 1 Mb/s using a 512 bits modulus (GEC Marconi)
The RSA patent was valid only for the US because it was requested after publication.
In any case the patent expired on September 20, 2000
and from then on RSA it’s now free everywhere
El Gamal (T.ElGamal 1984)

- Choose a prime \( p \) and two random numbers \( g(<p), x (<p) \) and compute
  \[ y = g^{**x} \mod p \]
- public key is \( y, g, p \)
- private key is \( x \)
- To encrypt a message \( M \) choose a random \( k, (p-1|k)=1 \) and compute
  - \( a = g^{**k} \mod p \)
  - \( b = (y^{**k}) * M \mod p \)
- \( a, b \) is the ciphertext, to decrypt:
  - \( M = b / a^{**x} \mod p \)
Message digests or Hash or fingerprints

- One way function that maps a file on a fixed length key. As with real fingerprints one hopes that no 2 msgs have the same fingerprint.
  - collision free
  - un-reversible
- e.g:
  - Unix sum is a bad example (16 bits) (Unix sum,cksum)
  - MD5(128 bits) invented by RSA (Unix md5sum)
  - SHA1(160 bits) Secure Hash Algorithm-1
Uses of PKC

Pics from M.Branchaud

Encrypted Message

Signed Message

Signed and Encrypted Message

Pic from M.Branchaud
Digital Signature

Message

MD5

Message Digest

RSA
with private key

Signature
Pitney-Bowes Veritas system

- Uses digital signatures to authenticate info stored on physical documents (including the digital encoding of photographs)
- It’s been used successfully at the Olympic World Games in New Haven in 1995
- On the back of a badge a high density bar code encoded a photograph, biographical data and medical data of the athletes
- A Veritas reader can scan the bar code, verify the digital signature and then display a copy of the photograph
Digital Certificates
L. Kohnfelder (1978)

In an effort to overcome performance problems related to the use of a single Public File, Kohnfelder proposed a digitally signed data record containing a name and a public key called a CERTIFICATE.
Digital Certificates/2

- Name
- Public key
- Digital signature
Certificates/3

- Binary format of certificates is defined using ASN.1 (x.208)
- Binary encoding is defined using DER (Distinguished Encoding Rules) which is based on BER (Basic Encoding Rules)
- Binary format can be translated into ASCII using Base64 encoding, this form is called PEM encoding
ASN.1 (X.208 1988)

Certificate ::=SEQUENCE {
  tbsCertificate TBSCertificate,
  signatureAlgorithm AlgorithmIdentifier,
  signature  BIT STRING
}

TBSCertificate ::=SEQUENCE {
  version [0] EXPLICIT Version DEFAULT v1,
  serialNumber CertificateSerialNumber,
  signatureAlgorithm AlgorithmIdentifier,
  issuer    Name,
  validity  Validity,
  subject   Name,
  subjectPucliKeyInfo,
  issuerUniqueID [1] IMPLICIT UniqueIdentifier OPTIONAL,
  subjectUniqueID [1] IMPLICIT UniqueIdentifier OPTIONAL,
  extensions [3] EXPLICIT Extensions OPTIONAL
}

UniqueIdentifier ::= BITSTRING

Extensions ::=SEQUENCE OF Extension
Private-Key: (1024 bit)

modulus:

publicExponent: 65537 (0x10001)

privateExponent: 
9d:5e:01:01:93:81:23:09:45

-----BEGIN RSA PRIVATE KEY-----
MIIE9wIBAAKBgQCgEaYBxtZVIwYSr3YElF1qlGf3Am5MG5A5uG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont
mI/7fZVwVw3Ys3JVEF1q1Q3Am5H55AuG2mAgFXhwtX7ont

2001/03/26
r.innocente
24
ASN.1/3

0:d=0  hl=4 l= 603 cons: SEQUENCE
4:d=1  hl=2 l=  1 prim:  INTEGER           :00
7:d=1  hl=3 l= 129 prim:  INTEGER           :

2001/03/26 r.innocente 25
OID (object identifiers)

- *Object identifiers* are unique numbers assigned to objects. They identify a node in a global tree.

  0  1  2
  ITU-T  ISO  joint ISO/ITU

- e.g. 1.2.840.113549.1.7.2 is an OID, it means SignedData which is defined by RSADSI
OID global tree

Pic from M. Branchaud
RSA keys according to PKCS#.. 

Private-Key: (1024 bit)
modulus:
  b9:26:b1:fe:93:15:f0:04:75:03:1c:e7:a9:3a:cf:
  9d:5e:01:01:93:81:23:09:45
publicExponent: 65537 (0x10001)
privateExponent:
  cd:b0:f0:5b:cc:58:f6:fd:1d:0a:93:01:58:83:79:
X.500 Directory Services
X.500

Pic from M. Branchaud

Attributes

<table>
<thead>
<tr>
<th>Common Name</th>
<th>Louis Riel</th>
<th>Tel.</th>
<th>1 514 987-6543</th>
</tr>
</thead>
<tbody>
<tr>
<td>E-mail</td>
<td><a href="mailto:Iriel@bombardier.com">Iriel@bombardier.com</a></td>
<td>Title</td>
<td>VP Marketing</td>
</tr>
<tr>
<td>DN:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{ C=CA,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O=Bombardier Inc.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CN=Louis Riel }</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Distinguished Names (DN) fields

- Common name CN e.g. CN=Joe Wells
- Organizational unit OU e.g OU=Sales
- Organization O e.g. O=Heaven,Inc.
- City/Locality L e.g. L=Tampa
- State/Province ST e.g. ST=Florida
- Country C e.g. C=US

/ 
C=US/ST=Florida/L=Tampa/O=Heaven,Inc./OU=Sales/CN=Joe Wells
Digital Certificates/3
X.509 cert v1-2

Pic from M.Brancaud
CRL(CertificateRevocationList)
CRLv2

<table>
<thead>
<tr>
<th>CRL version</th>
<th>Issuer's signature algorithm ID</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Issuer's X.500 name</td>
</tr>
<tr>
<td></td>
<td>Date &amp; time of this update</td>
</tr>
<tr>
<td></td>
<td>Date &amp; time of next update</td>
</tr>
<tr>
<td>Certificate serial number</td>
<td>Revocation date</td>
</tr>
<tr>
<td>Certificate serial number</td>
<td>Revocation date</td>
</tr>
<tr>
<td>Certificate serial number</td>
<td>Revocation date</td>
</tr>
</tbody>
</table>

Revoked certificates

Issuer's private key

Pic from M. Branchaud
SSL/1

The SSL (Secure Socket Layer) protocol was designed by Netscape to be used with its browser.

- SSL v.1 was used only internally.
- SSL v.2 was incorporated in Navigator v1 and v2.
- Microsoft created a similar protocol called PCT which overcame some problems of SSL.
- SSL v.3 incorporated PCT enhancements.
SSL/2

• The first implementation of SSL was available only in Netscape browsers and servers
• SSLRef is a reference implementation in C that Netscape made available in source code (does’nt include RC2 or RC4 encryption algorithms)
• SSLeay is an independent implementation of SSLv.3 made by Eric A. Young a programmer in Australia
• OpenSSL is based on SSLeay
TLS/SSL Layers

TLS layers

TLS
Handshake protocol

TLS
Record protocol

Transport protocol
e.g. TCP

2001/03/26  r.innocente  38
SSL record protocol

• Each SSL record contains:
  • content type
  • proto version
  • length
  • payload
  • Message authentication (Changed in TLS to HMAC), it contains a sequence number to be hashed together with data
SSL/TLS handshake

• ClientHello (version, random, session, cipherhs)
  – Server hello(version, random, session, cipher)
  – [server may send its certificate]
  – [server may send a KeyExchange]
  – [server may send a CertReq]

• [Client sends its certificate]
• client sends a KeyExchange
• [client sends a cert verify]
• both send a Change Cipher
SSL cert accept

Pic
From netscape

2001/03/26  r.innocente  41
Key Exchange

• SSL v.2 uses RSA key exchange only
• SSL v.3 supports:
  • RSA key exchange when certificates are used
  • DH (Diffie-Hellmann) for exchanging keys w/o certificates or prior communication
Diffie-Hellmann Key Exchange

- Given a large prime n and a primitive g
- A chooses a random x and sends to B
  - $X = g^x \mod n$
- B choses a random y and sends to A
  - $Y = g^y \mod n$
- A and B can compute
  - $k = Y^x \mod n = g^{(y \times x)} \mod n = X^y \mod n$
- The patent held by PKP expired in 1997
PKI (Public Key Infrastructure)

• It is a practical and viable way of publishing public keys on the Internet
• PGP, PEM, PKIX, SPKI and SDSI are different proposals
PEM CA model

Pic from M. Branchaud
PKI

Pic from M. Branchaud
STARTTLS (RFC 2487)

- a server announce its support of TLS
  - ehlo heaven.org
  - 250 inferno.org
  - 250 starttls
- the client then can switch to TLS
  - starttls
  - 220 ready to start tls
- STARTTLS is supported in sendmail 8.11
Microsoft Authenticode

- Announced in 1996 by Microsoft as part of IE3.0 and ActiveX (A system for downloading programs from web pages)
- It describes some file formats to sign Microsoft 32bit EXEs, DLLs and OCXs
- The signed file contains:
  - original file
  - digital signature
  - an X.509 certificate for the public key needed to verify the authenticode signature
- The tools needed are in the ActiveX software developer’s Kit (CSW Code Signing Wizard)
Java signed applets

- Java too can use X.509 certificates to sign the code in a jar file (keytool and jarsign utilities)
- The idea is similar to that of Microsoft, the code signed can obtain better trust according to user chosen confidence in signing publishers
Encrypted services

Note that recently the name of encrypted services has changed from an initial s to a final s (simap to imaps)

- **https** 443/tcp  #http over ssl
- **telnets** 992/tcp  #telnet over ssl
- **pop3s** 995/tcp
- **imaps** 993/tcp
- **smtps** 465/tcp
- **sshell** 614/tcp  #SSLshell
- **nsiiops** 261/tcp  #IIOP name service over ssl