Topic 6

Digital Signature Algorithms

Understand the techniques and standards of digital signatures by explaining the two mostly used signature algorithms

Overview

- ❑ What is a Digital Signature?
- ❑ Digital Signature using RSA
- ❑ DSS (Digital Signature Standard, also called DSA - digital signature algorithm)
- ❑ DSS vs RSA
- ❑ Conclusion

Next Topic: PKI

source: Stalling's book:

Cryptography and Network Security - chapter 13

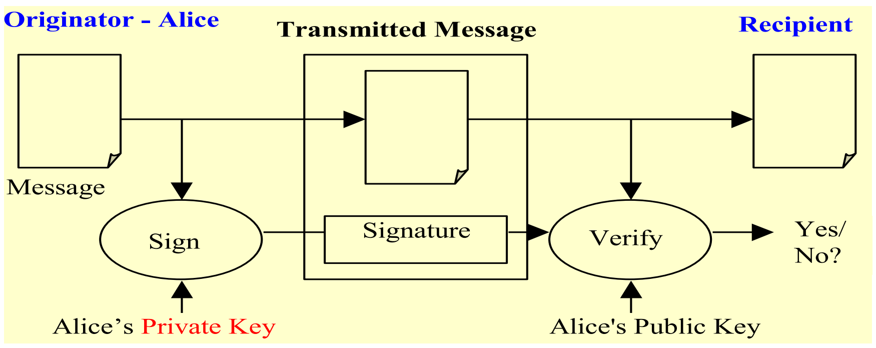
What is a Digital Signature? (1)

- A digital signature associates a mark unique to an individual with a body of text. It should be
  - message-dependent
    - unreusable
    - ensures integrity
  - signer-dependent
    - unforgeable
    - ensures authenticity
  - verifiable, others should be able to verify that the signature is indeed from the originator.

Both integrity and authenticity are necessary to ensure **non-repudiation**, i.e. the signer can not falsely deny that he/she has generated the signature.

What is a Digital Signature? (2)

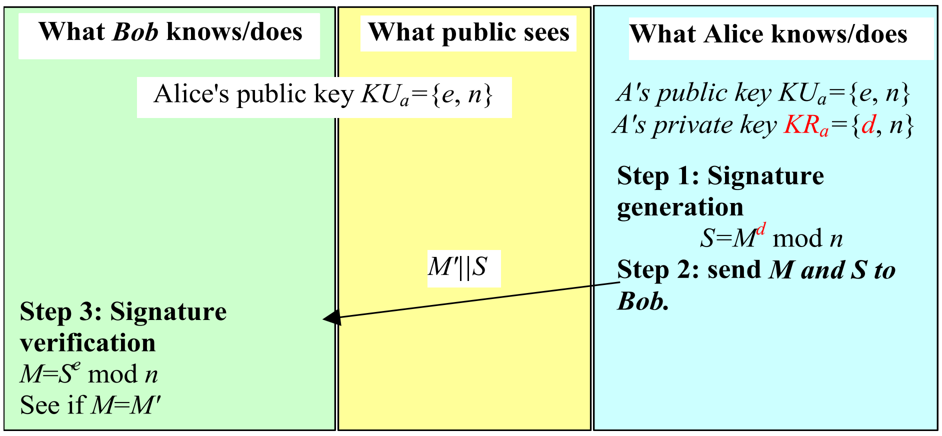
- A digital signature scheme uses public-key cryptography, typically, RSA or DSA.



What is a Digital Signature? (3)

- The main idea is
  - only *A* can sign a message, since only *A* has access to the private key.
  - anyone can verify *A*' s signature, since everyone has access to her public key.
- A digital signature scheme consists of:
  - A key generation algorithm
  - A signature (generation) algorithm
  - A signature verification algorithm

Digital Signature using RSA - Not a so good way



What *Bob* knows/does

Alice's public key $KU_a=\{e, n\}$

Step 3: Signature verification
$M=S^e \bmod n$
See if $M=M'$

What public sees

$M'||S$

What Alice knows/does

A's public key $KU_a=\{e, n\}$
A's private key $KR_a=\{d, n\}$

Step 1: Signature generation
$S=M^d \bmod n$

Step 2: send *M and S to Bob.*

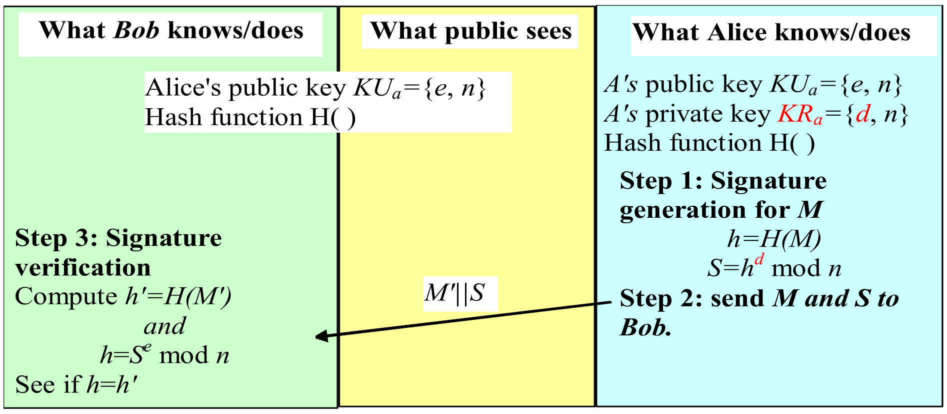
Digital Signature using RSA - Problems

- Security loopholes (some examples)
  - *Example 1*
    - Let *s* be a random value, apply the public key (e, n) *to s: P= s^e (mod n) = m*
    - Then (*m, s) is a valid message-signature pair.*
  - *Example 2*
    - Let *m* be the message of which you want to forge a signature.
    - Choose two messages *x and y* such that *xy = m (mod n)*
    - If you could obtain the signatures of *x and y, then,* you can easily forge a signature of *m* by *S(m) = S(y)S(x) (mod n)*

Digital Signature using RSA - Problems

- In addition,
  - Computation and communication costs – PKC is a time consuming process and signing long messages is costly.
- A solution to these problems
  - Use “hash-and-sign” paradigm: sign the hash value of the message. - *See next page….*

Digital Signature using RSA - A proper way



What *Bob* knows/does

What public sees

What Alice knows/does

Alice's public key $KU_a=\{e, n\}$
Hash function H( )

A's public key $KU_a=\{e, n\}$
A's private key $KR_a=\{d, n\}$
Hash function H( )

Step 1: Signature generation for *M*
$h=H(M)$
$S=h^d \bmod n$

Step 2: send *M and S to Bob.*

$M'||S$

Step 3: Signature verification
Compute $h'=H(M')$
and
$h=S^e \bmod n$
See if $h=h'$

Usually, we write in this format: $S=E_{KRa}[H(M)]$ for signature generation, and $h= D_{KUa}[S]=D_{KUa}[E_{KRa}[H(M)]]$ for signature verification.

Digital Signature using RSA - Problem remained

- ❑ The remaining problem now is how to ensure public keys (i.e. signature verification keys) are trust-worthy?
  - ❍ How can you be certain a given public key is indeed associated to a claimed entity?
  - ❍ You may be told a public key belongs to Alice, but it is actually Eve's key - Eve impersonates Alice and signs …. - masquerade or impersonation attack.

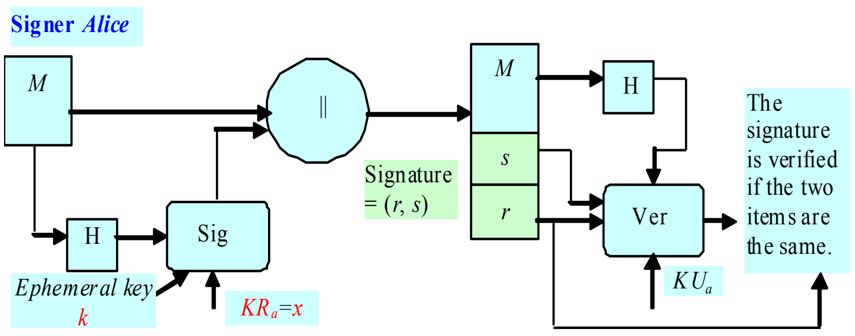
  - ❍ *The solution to this problem is in Topic - PKI.*

DSS/DSA - Background

- In 1991, the NIST (National Institute of Standards and Technology) proposed the **DSA** (Digital Signature Algorithm) for use in their **DSS** (Digital Signature Standard).
- DSA is a variant of the ElGamal signature algorithm.

DSS/DSA - The algorithm overview

- k is a random number generated for this signature.
- KR_a sender's private key, and KU_a sender's public key.



DSS/DSA - Key generation

- Key Generation
  - Generate global public-key components (can be shared among a group of users):
    - p - 512-bits to 2048-bits prime number.
    - q - 160-bit prime number & divides (p-1), i.e. q is a prime divisor of (p-1).
    - $g = h^{(p-1)/q} \bmod p$, where h is any integer with $1<h<(p-1)$ such that $h^{(p-1)/q} \bmod p > 1$.

DSS/DSA - Key generation continued

- Choose a (long-term) private key x, i.e.
  - $KR_a = x$, with $0< x <q$, chosen randomly.
- Compute a (long-term) public key:
  - $y = g^x \bmod p$,
  - $KU_a = \{p, q, g, y\}$.

DSS/DSA - Signature generation

- Signing
  - Chooses a random number (the ephemeral key), k with $0 < k < q$; k must be destroyed after use, and must never be reused.
  - Computes:
    - $r = (g^k \bmod p) \bmod q$.
    - $s = [k^{-1} (H(M)+xr)] \bmod q$.
    - $Signature = (r, s)$.
- Sends $M||Signature$ to the receiver.

DSS/DSA - Signature verification

- Verifying
  - The receiver has got $KU_a = \{p, q, g, y\}$, and $\{M', r', s'\}$.
  - Compute
    - message hash $H(M')$.
    - mod q inverse of s': $w=(s')^{-1} \bmod q$.
    - $u_1 = [H(M')w] \bmod q$.
    - $u_2 = (r')w \bmod q$.
    - $v = [(g^{u1} y^{u2}) \bmod p] \bmod q$.
  - Test: $v = r'$, if true, then the signature is verified (U prove!).
- Here, M = message to be signed; H(M) = hash of M using SHA-1; M', r', s' = received versions of M, r, s.

DSS/DSA - a baby example

- Key generation
  - Generate q, p and g
    - q=13; p=4q+1=53; g=16;
  - Generate private key: x=3;
  - Compute public key: y=g^3 (mod p) =15;
- Signature Signing
  - assuming H(M)=5;
  - choose k=2;
  - r = (g^k mod p) mod q= (16^2 mod 53) mod 13 = 5;
  - s = [($H(M)$+xr)*k^{-1}] mod q = [(5+3*5)*2^{-1}] mod 13 = 10.

DSS/DSA - a baby example

- Signature Verification
  - computes
    - message hash $H(M')$ =5.
    - mod q inverse of s':
      $w=(s')^{-1}$ mod q = $(10)^{-1}$ mod 13 = 4.
    - u_1 = [$H(M')w$] mod q =5*4 mod 13= 7.
    - u_2 = (r')w mod q = 5*4 mod 13 = 7.
    - v = [(g^{u1} y^{u2}) mod p] mod q
      = [(16^7*15^7) mod 53] mod 13 = 5.
  - Note v = r', so the signature is verified.

DSA vs RSA

RSA	DSA
Security is based on difficulty of factoring large numbers.	Security is based on difficulty of taking discrete logarithms.
Can encrypt and sign.	Can only sign messages.
Faster than DSA in signature verification, and about the same in signature generation.	Some signature computation can be computed a priori.
Can recover the message digest from the signature.	Cannot recover the message digest from the signature.
	Need to choose a unique secret number *k* for each message.

Exercise 6 (a)

- ❑ Use the signature signing and verification facilities in CrypTool to contrast the two signature schemes, RSA and DSA, in terms of key generation times, signature signing times and signature verification times ...
- ❑ The signing and verification facilities are available via Menu: “Digital Signatures/PKI” \“Sign Document” and “Digital Signatures/PKI” \“Verify Signature”, respectively.
- ❑ The key generation facility is available via Menu: “Digital Signatures/PKI” \“Generate/Import Keys”.
- ❑ The CrypTool’s Help Menu provides lots of useful information to guide you through the e-learning process; try to make the best use of the tool to familiarize yourself with the two signature algorithms.

Exercise 6 (b)

❑ Exercise 6 (b)

(i) Design a digital signature protocol using symmetric encryption and an arbiter, but do not expose the content of a message to be signed to the arbiter.

(ii) Compare the signature protocol designed in (i) with the RSA based signature scheme.

Conclusion (1) - Signatures

❑ Digital signatures provide message authentication (integrity & origin authentication) and non-repudiation security services.

❑ There are *two* well-known signature schemes

  ❍ RSA encryption algorithm can be used in reverse to produce a signature.

  ❍ DSA is a signature algorithm based on discrete logarithms.

❑ A signature scheme should be used in conjunction with a hash algorithm to obtain security in an efficient way.

Conclusion (2) – A quick summary of key points

- Confidentiality – cryptography allows encryption, and therefore protection against unauthorised disclosure of info.
- Signatures and integrity – cryptography can be used to generate a signed token for a message so that the recipient of the message can determine if the message is authentic, or has been modified.
- Signatures and non-repudiation – cryptographic signatures can be used to determine if a message has indeed been sent by a particular sender.
- Identification and authentication – cryptography can be used to determine an identity of a user and a service.
- Timestamping - cryptography can be used to generate a time stamp to certify that a particular message has indeed been sent at a particular time and date.