F7: Digital signature

tkkwon@snu.ac.kr

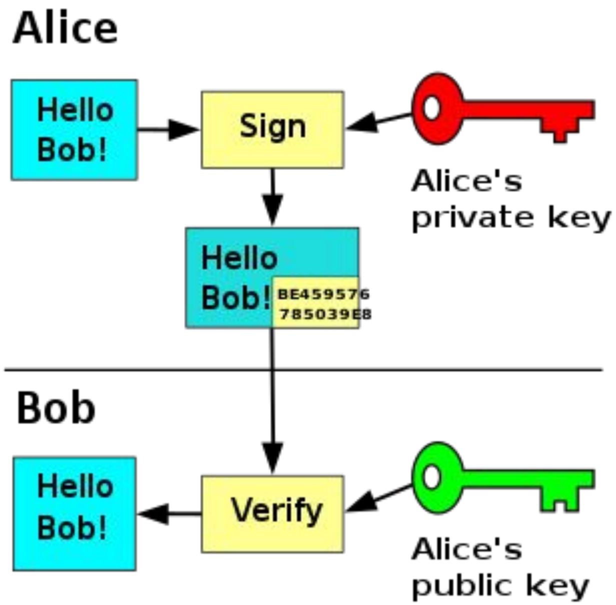
outline

- Intro
- RSA signature
- Blind signature
- ElGamal signature
- cyclic group vs cyclic subgroup
- Schnorr signature
- DSA signature
- Subliminal signature

Digital signature

- A digital signature is a number (or numbers) dependent on
  (i) some secret known only to the signer
  (ii) the signed message

- Digital signatures are implemented using public-key cryptography
  - the signer has a private key used for creating signatures
  - Anybody can know a public key used for signature verification



source: https://ko.wikipedia.org/wiki/디지털_서명

Digital Signatures

- data integrity, non-repudiation, authentication
- Basic idea
  - use private key on the message to generate a piece of information that can be generated only by yourself
    - because you are the only one who knows your private key
  - public key can be used to verify the signature
    - so everybody can verify
- Generally signatures are created and verified over the hash of the message
  - Not over the original message. Why?

Attack types

Key-Only Attack

the attacker is only given the public verification key.

Known-Message Attack

the attacker is given valid signatures for a variety of messages known by the attacker but not chosen by the attacker.

Chosen-Message Attack

the attacker first learns signatures on arbitrary messages of the attacker's choice (usually by tricking a signer). Then the attack forges a signature of a new message he chooses.

* Message usually means plaintext, but could be ciphertext

Forgery types

- Digital signature should be resistant to forgery
  - So called unforgeability

Existential Forgery

Existential forgery is the creation (by an adversary) of any message/signature pair (m,σ), where σ was not produced by the legitimate signer.

Selective Forgery

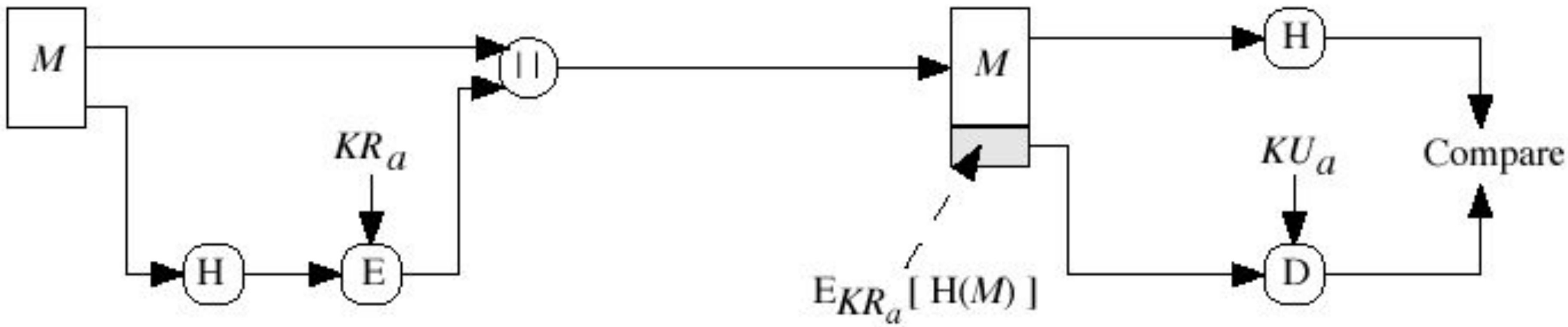
Selective forgery is the creation (by an adversary) of a message/signature pair (m,σ) where m has been *chosen* by the adversary prior to the attack.

Two categories

- Digital signatures with appendix
  - require the original message (or its hash) as input to the verification algorithm;
  - use hash functions
  - Examples: ElGamal, DSA, Schnorr
- Digital signatures with message recovery
  - do not require the original message as input to the verification algorithm;
  - the original message (or its hash) is recovered from the signature itself;
  - Examples: RSA, Rabin,...

Digital Signature – message recovery

Sender a　　　　　　　　　　Receiver



M: message to be signed　　　H: Hash function

E: RSA Private Key Operation　KR_a: Sender's Private Key

D: RSA Public Key Operation　KU_a: Sender's Public Key

E_{KRa}[H(M)]　　Signature of sender a over hash of M

RSA Signatures

- Bob has a document m that Alice agrees to sign. Alice does the following.
- Alice chooses two primes: p, q and $n=pq$, makes (e,n) public with $\gcd(e,(p-1)(q-1))=1$
- $de \equiv 1 \pmod{\varphi(n)}$, she keeps p,q,d secret

- Alice's signature is $y \equiv m^d \pmod{n}$
  - m or H(m)
- Alice then makes the pair (y,m) public

How does Bob verify Alice's Signature

- Download Alice's (e,n)
- Compute $z \equiv y^e \pmod{n}$
- If z=m, then Bob accepts the signature as valid;
  - otherwise the signature is not valid

Chosen ciphertext attack in RSA

- Trent is a computer notary public. When Alice wants a document notarized, she sends it to Trent who signs it with an RSA digital signature.
- Mallory wants Trent to sign a bad message he otherwise wouldn't, call it b
- Mallory chooses arbitrary x and computes $y = x^{e} \bmod n$ (where e, n are Trent's public key).
- Then he computes $m = yb \bmod n$ and sends m to Trent to sign.
- Trent returns $m^{d} \bmod n = (yb)^{d} \bmod n = xb^{d} \bmod n$.
- Mallory calculates $(xb^{d} \bmod n)(x^{-1} \bmod n) = b^{d} \bmod n$, which is the signature of b.

- Lesson: don't sign message with unknown content

Blind Signatures (1/2)

- Alice chooses $n=pq$, finds e, and solves d as required in the RSA scheme, i.e., $ed \equiv 1 \pmod{(p-1)(q-1)}$
- Bob chooses a random k with $\gcd(k,n)=1$, computes $t \equiv k^e m \pmod{n}$ for message m, and sends t to Alice
  - Bob keeps k secret
  - Alice cannot read m; Alice trusts that Bob will not send bad messages
- Alice signs t by computing $s \equiv t^d \pmod{n}$. She returns s to Bob
- Bob computes $sk^{-1} \pmod{n}$ to get the signed message m^d

Blind Signatures (2/2)

- $sk^{-1} \equiv t^d k^{-1} \equiv (k^e m)^d k^{-1} \equiv m^d (k^{ed})\ k^{-1} \equiv m^d$
- Alice has never seen the message m
- $t \equiv k^e m$ and $s \equiv t^d$, then $sk^{-1} \equiv m^d \pmod{n}$
- The choice of k is random, therefore, $t \equiv k^e m \pmod{n}$ gives essentially no information about m. In this way, Alice knows nothing about the message m she is signing.

Before DLP-based signatures with appendix

- Signer has her private key x, its public key y
  - $y = g^x \bmod p$

(mod p) is omitted below

- How an appendix is generated only from who holds x?
  - Only the signer can make (Eve can't); Verifier can verify the signature
  - Signature should be dependent on message
    - msg or its hash is known to everyone
- Message is m = H(M), hash of original message M
- How about my or mx?
- How about g^{m+x} or g^{mx}?
- Pick random r, then how about g^{m+rx} or g^{mr+x}?
  - Everyone receives r as well
- Signer picks a random secret k, $g^{k * \boxed{k^{-1}(m+rx)}} = (g^k)^{k^{-1}(m+rx)}$
  - Everyone receives r, g^k, $s=k^{-1}(m+rx)$
  - More efficiently, $r=g^k$, $s=k^{-1}(m+rx)$

ElGamal Digital Signatures

- signature variant of ElGamal
  - so uses exponentiation in a finite (Galois) field
  - with security based on the difficulty of computing discrete logarithm problem (DLP)
- A signer uses private key for encryption (signing)
- A verifier uses public key for decryption (verification)
- A signer (say, Alice) generates her key
  - chooses a private key $1 < x < p-1$
  - Computes her public key: $y = a^x \bmod p$

ElGamal Digital Signature

- Alice signs a message M to Bob by computing
  - the hash $m = H(M)$, $0 \leq m \leq (p-1)$
  - choose random secret k s.t. $1 \leq k < (p-1)$ and $\gcd(k,p-1)=1$
    - k should not be reused
  - compute temporary key: $r = a^k \bmod p$
  - compute $k^{-1} \bmod (p-1)$: the inverse of k
  - compute the value: $s = k^{-1}(m-xr) \bmod (p-1)$
  - signature is: (r,s)
- Bob (i.e., verifier) can verify the signature as:
  - $V_1 = a^m \bmod p$
  - $V_2 = y^r r^s \bmod p$
  - signature is valid if $V_1 = V_2$

What if $s = k^{-1}(m-x)$ in ElGamal?

- An attacker can forge a signature for msg m
- Pick arbitrary s in Z_p^*
- Let's choose r s.t. $a^m = yr^s \bmod p$
  - $r = (a^m y^{-1})^{s-1}$
- signature is: (r,s)
- Bob (i.e., verifier) deems the signature valid:
  - $V_1 = a^m \bmod p$
  - $V_2 = y\ r^s \bmod p$
  - signature is valid if $V_1 = V_2$

Revisit Group Theory (mod p case)

- group: a set of elements with a binary operation
  - the operation should satisfy four properties below
- The group of positive integers modulo a prime *p*

  $Z_p = \{0, 1, 2, 3, \ldots, p-1\}$

  $Z_p^* = \{1, 2, 3, \ldots, p-1\} = \langle g \rangle = \{g^1, g^2, g^3, \ldots, g^{p-1}\}$ (g is a generator)

  $*_p$ = multiplication modulo p

  Denoted as: $(Z_p^*, *_p)$
- **Required properties**
  1. Closure. Yes.
  2. Associativity. Yes.
  3. Identity. 1.
  4. Inverse. Yes.
- **Example:** $Z_7^* = \{1,2,3,4,5,6\}$

  $1^{-1} = 1,\ 2^{-1} = 4,\ 3^{-1} = 5,\ 6^{-1} = 6$

Background for finite cyclic groups

- Order
  - order of a set S: |S|, # of elements in S
  - Order of an element x: ord(x), the least $n \geq 1$ s.t. $x^n \equiv 1 \pmod{p}$
- Cyclic group G
  - A group that can be generated by exponentiating a single element g (generator or primitive element)
    - G = <g>
  - Group order $|G| = n$, s.t. $g^n \equiv 1 \pmod{p}$
- Subgroup (≠ subset) of a cyclic group G
  - Every subgroup of G is cyclic, and satisfies all the properties of group
  - if order of G is |G|, each subgroup S has the form $< g^d >$, where d is a positive divisor of |G|
    - each subgroup has size |S| that divides the size of the group |G|
    - $|S|*k=|G|, k \in N$

A cyclic group (from modular exponentiation)

- mod 7 case, $Z_7^* = \{1,2,3,4,5,6\}$

Look at rows this time!

$1^1 \equiv 1$	$1^2 \equiv 1$	$1^3 \equiv 1$	$1^4 \equiv 1$	$1^5 \equiv 1$	$1^6 \equiv 1$
$2^1 \equiv 2$	$2^2 \equiv 4$	$2^3 \equiv 1$	$2^4 \equiv 2$	$2^5 \equiv 4$	$2^6 \equiv 1$
$3^1 \equiv 3$	$3^2 \equiv 2$	$3^3 \equiv 6$	$3^4 \equiv 4$	$3^5 \equiv 5$	$3^6 \equiv 1$
$4^1 \equiv 4$	$4^2 \equiv 2$	$4^3 \equiv 1$	$4^4 \equiv 4$	$4^5 \equiv 2$	$4^6 \equiv 1$
$5^1 \equiv 5$	$5^2 \equiv 4$	$5^3 \equiv 6$	$5^4 \equiv 2$	$5^5 \equiv 3$	$5^6 \equiv 1$
$6^1 \equiv 6$	$6^2 \equiv 1$	$6^3 \equiv 6$	$6^4 \equiv 1$	$6^5 \equiv 6$	$6^6 \equiv 1$

Group Z_7^* or G has two generators: $G = <3> = <5>$

$|G| = 6$, its unique divisors are $\{1,2,3,6\}$,

for each unique divisor d, $<g^d>$ generates a subgroup

$< 3^1 > = \{1,2,3,4,5,6\} = G$

$< 3^2 > = <2> = \{1,2,4\}$

$< 3^3 > = <6> = \{1,6\}$

$< 3^6 > = <1> = \{1\}$

Zoom in to a subgroup

- $<\cancel{3^2}> \; = \; <2> \; = \{1,2,4\} = S$
- 2 is the generator g of a subgroup S; S is cyclic
- the order of g is the size of the subgroup $<g> = S$,
  $2^{|S|} \equiv 1 \pmod{p}$
- $|S| = 3 = q$
- For $\forall x \in S$, $x^q \equiv 1 \pmod{p}$
  - Then x is the q-th root of 1 (mod p)
  - How many roots for $x^q = 1 \pmod{p}$?
    - q
  - S consists of the q-th roots of 1 (mod p)

Schnorr group

- Pick a very large prime *p,* and a large prime *q*
  - non-zero elements form a group of order *p*-1 under multiplication
  - $(Z_p^*, *_p)$ is a group of size p-1
  - *p-1* is composite, and factored into qr where q is prime, *p=qr+1*
- Pick any *h* s.t. $h^r \neq 1$ (mod p)
- $g = h^r$ is the generator of Schnorr group (subgroup of Z_p^*)
  - *q* is the order of Schnorr group
- $g^q \equiv h^{qr} \equiv h^{p-1} \equiv 1$ (mod p)
- <*g*> = $\{1, g^1, g^2, g^3, \ldots, g^{q-1}\}$ is Schnorr group
- Order of <g> is q

Schnorr Signatures

- Patented until 2008
- Uses exponentiation in a finite (Galois) field
  - Security based on discrete logarithms
- Minimizes message dependent computation
- Main work can be done in idle time
- Picks a prime modulus p
  - $p-1$ has a prime factor q of appropriate size
  - Typically p 1024-bit and q 160-bit numbers
  - p is chosen to be large enough to resist Index calculus algorithm
  - q is large enough to resist the general algorithm for DLP

Schnorr Key Setup

- Choose suitable primes p, q
  - $p = qr+1$
- Choose a such that $a^q \equiv 1 \pmod p$
  - $a = h^r$ is the generator of Schnorr group
- (a, p, q) are global parameters for all
- A signer generates a private key s
  - Chooses a private key: $0 < s < q$
  - Compute a public key: $v = a^{-s} \bmod q$

Schnorr Signature

- User signs message by
  - Chooses random k with $0<k<q$ and computes $x=a^k \bmod p$
    - Keeps k secret, does not reuse k
  - Concatenate msg with x, which is hashed as: $e=H(M||x)$
  - Computes: $y = (k+se) \bmod q$
  - Signature is pair (e, y)
- Any other user can verify the signature as follows:
  - Computing: $x' = a^y v^e \bmod p$
  - Verifying that: $e=H(M||x')$

Note that message dependent part is small!

Properties of Schnorr Signature

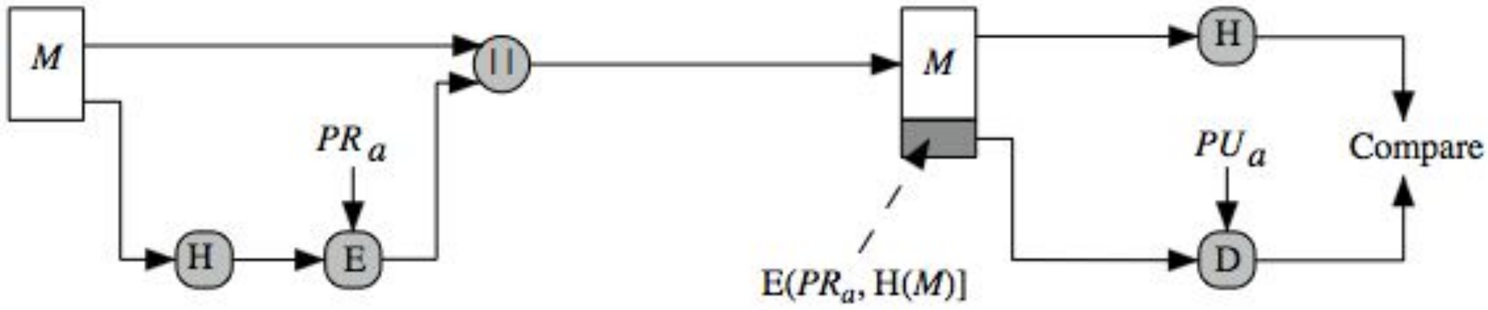
- Working in the order-q subgroup of Z_p^*
- The signature size is much shorter than that of a signature in ElGamal
  - Schnorr: 2|q|
  - ElGamal: 2|p|
- Fewer operations in signature generation and verification
- Patented until 2008

Digital Signature Algorithm (DSA)

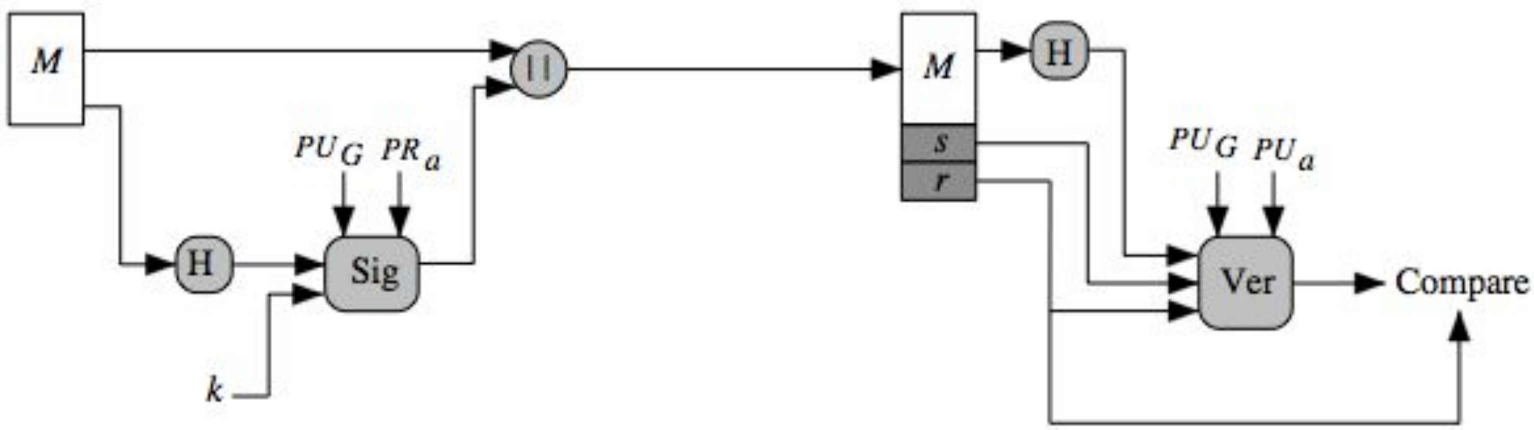
- US Gov't approved signature scheme
- designed by NIST & NSA in early 90's
  - Patent-free
- To avoid the patent of Schnorr signature
- DSS is the digital signature standard
- DSA is its algorithm

- creates a 320 bit signature
- a digital signature scheme only
- security depends on difficulty of computing DLP
- variant of ElGamal & Schnorr schemes

RSA vs DSA Signatures (recovery vs appendix)



(a) RSA Approach



(b) DSS Approach

Source: William Stallings

DSA Key Generation

- Need global public key values (p, q, g):
  - chooses 160-bit prime number q
  - chooses a large prime p with $2^{L-1} < p < 2^{L}$
    - where L= 512 to 1024 bits and is a multiple of 64
    - such that q is a 160 bit prime divisor of $(p-1)$
    - $p = qr+1$
  - chooses $g = h^{(p-1)/q} = h^{r}$
    - where $1<h<p-1$ and $h^{(p-1)/q} \bmod p > 1$
- signer chooses private key & computes public key:
  - chooses random private key: $0 < x < q$
  - computes public key: $y = g^{x} \bmod p$

DSA Signature Creation

- to sign a message M, the signer:
  - generates a random signature key k, $0<k<q$
  - k must be random, be destroyed after use, and never be reused
- then computes signature pair:

$$r = (g^k \bmod p) \bmod q$$

$$s = [k^{-1}(H(M)+xr)] \bmod q$$

- sends signature (r,s) with message M

- r is a "masked" k
- s is a "clue" to help the verifier to cancel out k at exponent; it also hides x

DSA Signature Verification

- having received `M` and signature `(r,s)`
- to verify a signature, the verifier computes:

  $w = s^{-1} \bmod q$

  $u1 = [H(M)w] \bmod q$

  $u2 = (rw) \bmod q$

  $v = [(g^{u1}\ y^{u2}) \bmod p] \bmod q$

- if ($v = r$) then signature is verified

mod q can be other modulus.
mod q is needed to counteract sub-exponential DLP algorithm

- $r = (g^{k} \bmod p) \bmod q$

  $= (g^{H(M)s-1+\ xrs-1} \bmod p) \bmod q$

  $= (g^{H(M)s-1}\ g^{xrs-1} \bmod p) \bmod q$

  $= (g^{H(M)s-1}\ y^{rs-1} \bmod p) \bmod q$

- $v = [(g^{u1}\ y^{u2}) \bmod p] \bmod q$

Subliminal channel

- One of covert channels
- A subliminal channel is a way of embedding information in public communications in an undetectable way.
- Some sort of shared secret (a key, knowledge of what to look for) is needed to reconstruct the subliminal information.

terminology

- *Covert channel*: Intentionally used to communicate
  - An attack that creates a capability to transfer information objects between processes that are not supposed to be allowed
- *Side channel*: Unintentionally reveals information
  - An attack based on information gained from the physical implementation of a computer system, rather than weaknesses in the implemented algorithm itself
- *Steganography*: Techniques for hiding the very presence of communication
  - the practice of concealing a file, message, image, or video within another file, message, image, or video
- *Subliminal channel*: Covert channel with mathematically proven steganographic properties

Subliminal channel

- Two prisoner problem
  - Two prisoners want to escape jail.
  - They are in separate cells and need to communicate.
  - All messages between the prisoners must go through the warden.

  - What should they do?

Toy example of a subliminal channel

- Two prisoners may do the following:
  - They communicate their plans to each other in code.
  - Each word with an even number of letters represents a 1.
  - Each word with an odd number of letters represents a 0.
  - The actual message is the reassembled binary string.

“Hey, what are you up to?”
3 4 3 3 2 2

“010011”

DSA: subliminal channel

- Two prisoners share x in $y = g^x \bmod p$
- Use k as the subliminal message
- Sender sends any message M with DSA signature

$$r = (g^k \bmod p) \bmod q$$

$$s = [k^{-1}(H(M) + xr)] \bmod q$$

- Receiver gets k as follows

$$k = s^{-1}(H(M) + xr) \bmod q$$