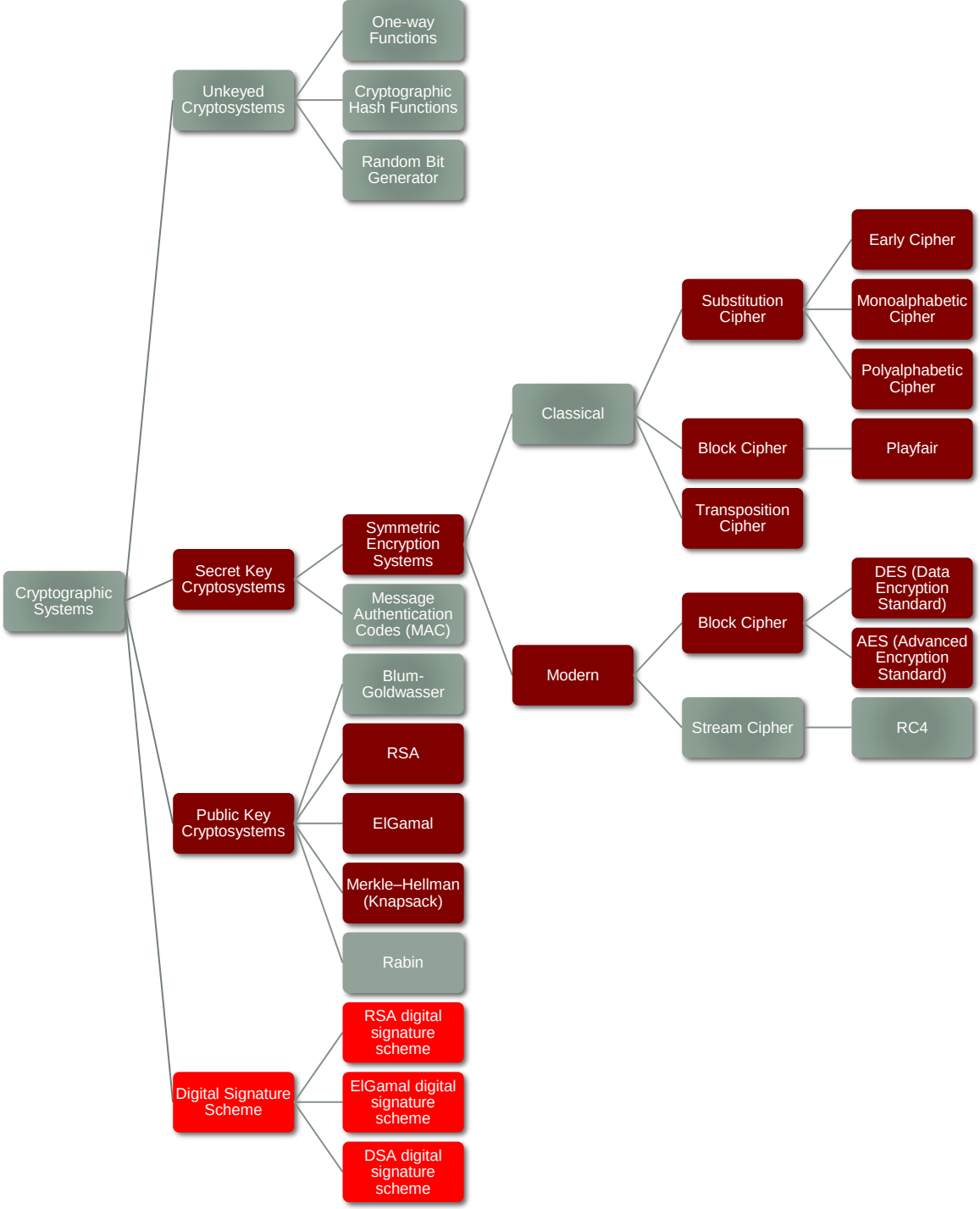
TUTORIAL 4

CSCI361 – Computer Security

Sionggo Japit

sjapit@uow.edu.au

12 February 2024


Cryptographic Systems
Unkeyed Cryptosystems
One-way Functions
Cryptographic Hash Functions
Random Bit Generator
Secret Key Cryptosystems
Symmetric Encryption Systems
Classical
Substitution Cipher
Early Cipher
Monoalphabetic Cipher
Polyalphabetic Cipher
Block Cipher
Playfair
Transposition Cipher
Modern
Block Cipher
DES (Data Encryption Standard)
AES (Advanced Encryption Standard)
Stream Cipher
RC4
Message Authentication Codes (MAC)
Public Key Cryptosystems
Blum-Goldwasser
RSA
ElGamal
Merkle–Hellman (Knapsack)
Rabin
Digital Signature Scheme
RSA digital signature scheme
ElGamal digital signature scheme
DSA digital signature scheme

Agenda

- Digital Signature Scheme
  - RSA digital signature scheme
  - ElGamal digital signature scheme
  - DSA signature scheme (Digital Signature Standard)

Digital signature scheme

The need to authenticate both the contents and origin of a message is crucial in any communications network.

The original concept of a digital signature based on public key cryptography was proposed by Diffie and Hellman (1976) and was shown to be practically viable by Rivest, Shamir and Adleman in the RSA paper (1978).

Digital signature scheme

Ideally a digital signature should provide the same guarantees as a handwritten signature, namely:

Unforgeability – Only the message sender should be able to sign his/her name to a message.

Undeniability – The message sender should not be able to deny he/she signed at a later stage.

Authentication – The signature should allow the contents of the message to be authenticated.

Digital signature scheme

A digital signature scheme consists of three efficiently computable algorithms:

Generate Generate public key pairs that consist of a signing key and a corresponding verification key.

Sign Generate digital signature

Verify or Recover Verify the digital signatures is one way or another.

Digital signature scheme

A digital signature scheme has the following components:

1. P is a finite set of possible messages.
2. A is a finite set of possible signatures.
3. $\mathcal{K}$, the keyspace, is a finite set of possible keys.

And satisfies the following condition:

DIGITAL SIGNATURE SCHEME

1. For each $K \in \mathcal{K}$, there is a signing algorithm $sig_K \in S$ and a corresponding verification algorithm $ver_K \in V$.
2. Each $sig_K : P \to A$ and $ver_K : P \times A \to \{true, false\}$ are functions such that the following equation is satisfied for every message $x \in P$ and for every signature $y \in A$:

$$ver(x, y) = \begin{cases} true & if\ y = sig(x) \\ false & if\ y \neq sig(x). \end{cases}$$

DIGITAL SIGNATURE SCHEME

- For every $K \in \mathcal{K}$, the function sig_K and ver_K should be polynomial-time functions.
- ver_K will be a public function and sig_K will be secret.
- Digital signature scheme should be computationally infeasible for the adversary to "forge" the sender's signature on a message x. That is, given x, only the sender should be able to compute the signature y such that $ver(x, y) = true$.
- A digital signature scheme cannot be unconditionally secure, since the adversary can test all possible signatures y for a message x using the public algorithm *ver*, until he/she finds the right signature.

TUTORIAL 4

RSA Digital Signature Scheme

RSA

- RSA is a public-key cryptosystem that offers both encryption and digital signatures (authentication).
- RSA's strength lies in the tremendous difficulty in factorization of large numbers.

RSA

- RSA works as follows:
  - Take two large primes , *p* and *q*, and compute their product *n=pq*; *n* is called the modulus.
  - Choose a number, *e*, less than *n* and relatively prime to *(p-1)(q-1)*; i.e., have no common factors except 1.
  - Find another number *d* such that *(ed – 1)* is divisible by *(p-1)(q-1)*.
  - The values *e* and *d* are called the public and private exponents, respectively.

RSA

- The public key is the pair *(n,e)*;
- The secrete key is *(n,d)*.
- The factors *p* and *q* may be kept with the private key, or destroyed.

RSA as digital signature

RSA Digital Signature

Suppose Alice wants to send a message *m* to Bob in such a way that Bob is assured the message is both authentic, has not been tampered with, and from Alice. Alice creates a digital signature *s* by exponentiating: $s=m^d \bmod n$, here *d* and *n* are Alice's private key. She sends *m* and *s* to Bob. To verify the signature, Bob exponentiates and checks that the message *m* is recovered: $m = s^e \bmod n$, where *e* and *n* are Alice's public key.

RSA as digital signature

Thus encryption and authentication take place without any sharing of private keys: each person uses only another's public key or their own private key. Anyone can send an encrypted message or verify a signed message, but only someone in possession of the correct private key can decrypt or sign a message.

RSA as digital signature

Example: Suppose Alice wants to sent message 28 to Bob using her private key pair (13, 143) and public key (37, 143).

Key generation:

- Alice chose p = 11 and q = 13, and compute the modulus; that is

  $n = p \times q = 11 \times 13 = 143$

- Next Alice chooses e = 37 such that $gcd(e,(p-1)(q-1)) = 1$

  $(p-1)(q-1) = (11 - 1)(13 - 1) = (10 * 12) = 120$

RSA as digital signature

- Check if gcd(37, 120) = 1?

n1	n2	r	q	a1	b1	a2	b2
120	37	9	3	1	0	0	1
37	9	1	4	0	1	1	-3
9	1	0	9	1	-3	-4	13

Thus gcd(37,120) = 1

- Next Alice finds another number *d* such that *(ed – 1)* is divisible by *(p-1)(q-1)*.

RSA as digital signature

- Using the gcd algorithm, Alice can find d such that gcd(ed, (p-1)(q-1)) = 1, in other words, ed = 1 mod (p-1)(q-1). *Thus d can be found using the extended Euclidean algorithm since d is the multiplicative inverse of e moduloΦ(n).*

n1	n2	r	q	a1	b1	a2	b2
120	37	9	3	1	0	0	1
37	9	1	4	0	1	1	-3
9	1	0	9	1	-3	-4	13

gcd(120,37) = a2 (n1) + b2(n2) = (-4) (120) + (13) (37)

Thus d = 13.

RSA as digital signature

- (d, n) pair is Alice private key; i.e., (13, 143) is Alice private key.
- (e, n) pair is Alice public key; i.e., (37, 143) is Alice public key.

RSA as digital signature

Message Signing:

Alice creates her digital signature S:

$$S = m^d \bmod n$$

Where
m: The message
(d, n) pair: Alice private key

$$S = 28^{13} \bmod 143$$

RSA as digital signature

- Calculate 28^{13} mod 143 using fast exponentiation:

$2^0 = 1$	28^1 mod 143	28 mod 143
$2^1 = 2$	28^2 mod 143	28 x 28 mod 143 = 69
$2^2 = 4$	28^4 mod 143	69 x 69 mod 143 = 42
$2^3 = 8$	28^8 mod 143	42 x 42 mod 143 = 48
28 x 42 x 48 mod 143 = 106 mod 143		

Thus the digital signature S = 106.

RSA as digital signature

Message Verification:

- Bob receives *m (28)* and *s (106)* from Alice.
- Bob to verify that the message is indeed from Alice:

$m = s^e \bmod n$

where
s: Alice digital signature
(e, n) pair: Alice public key

$m = 106^{37} \bmod 143$

RSA as digital signature

- Calculate 106^{37} mod 143 using fast exponentiation:

$2^0 = 1$	106^1 mod 143	106 mod 143 = 106
$2^1 = 2$	106^2 mod 143	106 x 106 mod 143 = 82
$2^2 = 4$	106^4 mod 143	82 x 82 mod 143 = 3
$2^3 = 8$	106^8 mod 143	3 x 3 mod 143 = 9
$2^4 = 16$	106^{16} mod 143	9 x 9 mod 143 = 81
$2^5 = 32$	106^{32} mod 143	81 x 81 mod 143 = 126
106 x 3 x 48 mod 126 = 28 mod 143 = 28		

Thus the message = 28.

RSA as digital signature

- The verification matches, that is, the message received (m=28) and the verified message ($m=106^{37} \bmod 142$) are the same, thus Bob is assured that the message and signature received from Alice are valid (authentic).

RSA as digital signature

RSA DSS - Summary

- Supports confidentiality and authentication
- Encrypt and decrypt are the same function
  - E(D(M, privK), pubK) = M
  - Use as basis for authentication
    - For authentication, user sign message using private key
    - Receiver encrypt signature to verify is the message is the same
    - Inefficient

RSA as digital signature

- Use different key for providing confidentiality and authentication
  - Sign then encrypt
  - Provide non-repudiation
- Only sender can generate D(M, privK)
- Weakness
  - Plaintext same as ciphertext
    - Not so important if n>200
  - Selection of e is important
    - Need safe prime
      - p = 2q + 1, where q is also prime

TUTORIAL 4

ElGamal Digital Signature Scheme

Elgamal as digital signature

ElGamal Digital Signature

Key generation:

- The key generation algorithm of ElGamal digital signature system is the same as the one employed by the ElGamal asymmetric encryption system.
- For every user, it generates a public ElGamal verification key (p, g, y) and a corresponding private ElGamal signing key x.
- All users may be using the same p and g.

Elgamal as digital signature

Message signing:

- Select a number *k* from Z_p^* such that *gcd(k, p-1)=1.*
- Compute the first element of the signature $r = g^k \bmod p$.
- Hash the message *m*, and the result *h(m)* is used to compute the second element of the signature $s = k^{-1} (h(m) - xr) \bmod (p-1)$

Elgamal as digital signature

(p, g, x, m)

$$k \hat{\imath}_R Z_p^*$$

$$r \circ g^k \bmod p$$

$$s \circ (k^{-1}(h(m) - xr) \bmod (p - 1)$$

(r, s)

The algorithm takes as input a private ElGamal signing key x together with the modulus p, generator g, and a message m.

Generates as output the digital signature for m, that consists of two numbers, r and s, that are both element of Z_p^*

Elgamal as digital signature

Signature verification:

- Verify that
  1. $1 \leq r \leq p-1$
  2. $g^{h(m)} \bmod p = y^r r^s \bmod p$
- The signature is valid only if both the verification checks are positive.

(Note: it is important to verify that $1 \leq r \leq p-1$ because ElGamal signing algorithm is probabilistic which means there may be many valid signatures for any given message.)

Elgamal as digital signature

Example: Suppose that Alice wants to send to Bob a message m=1463.

Key generation:

1. Alice chooses a prime number $p = 2357$ and a generator $g = 2$ of Z^*_{2357}
2. Alice chooses a private key $x = 1751$ and compute

$$y = g^x \bmod 2357$$

$$= 2^{1751} \bmod 2357$$

$$= 1185$$

As an exercise, compute 2^{1751} mode 2357.

Elgamal as digital signature

- Alice's public key is thus (2357, 2, 1185).

Message signing:

- For simplicity, I choose an identity hash function to hash the message, thus

$$h(m) = h(1463) = 1463.$$

- To sign the hashed message *h(1463)*, Alice selects a random integer k = 1529 and compute $r = g^k \bmod p$.

Elgamal as digital signature

$$r = g^k \bmod p$$

$$= 2^{1529} \bmod 2357$$

$$= 1490$$

- Next Alice compute $s = k^{-1} (h(m) – xr) \bmod (p-1)$.

$$k^{-1} \bmod (p - 1)$$

$$1529^{-1} \bmod 2356 = 245$$

Thus, s = 245 (1463 – (1751 x 1490)) mod 2356 = 1777.

Elgamal as digital signature

- Alice's signature for m is thus the pair (r = 1490, s = 1777).

Signature verification:

- Bob verifies that

1. $1 \leq r \leq p-1;$

   $1 \leq 1490 \leq 2356$ positive!

Elgamal as digital signature

2. Bob verifies that

$$g^{h(m)} \bmod p = y^r r^s \bmod p$$

$$2^{1463} \bmod 2357 = 1185^{1490} \cdot 1490^{1777} \bmod 2357$$

$$1072 = 1072 \qquad \text{positive!}$$

Both verifications are positive, thus Bob is assured that the message and signature are authentic.

Elgamal as digital signature

ElGamal DSS – Summary

Public paramters:

- A large prime number p, a generator g of Z_p^*.

Key generation:

- Generate a random x Î Z_{p-1} and compute

$y = g^x \bmod p$.

$Secret\ key = x$.

$Public\ key = y$.

Elgamal as digital signature

Signing the message:

- Hashed message $h(m) \in Z_{p-1}$.
- Pick a random $k \in Z^*_{p-1}$, compute $r = g^k \bmod p$ and $s = k^{-1}(h(m) - xr) \bmod (p-1)$.

The signature is (r, s).

Verifying the message:

- Check that
  1. $1 \le r \le p-1$
  2. $g^{h(m)} \bmod p \equiv y^r r^s \bmod p$

TUTORIAL 4

DSA Digital Scheme (Digital Signature System – DSS)

DSA

- Unlike RSA and ElGamal, DSA can be used to provide digital signatures only.
- A variation of ElGamal that does the modular arithmetic not in a group of order *p-1*, e.g., Z_p^*, but in a much smaller subgroup of prime order *q* with *q*|*p-1*.

DSA

Key Generation:

- To generate keys, the algorithm uses the following parameters:
  - A prime modulus *p* that is L bits long, where L ranges from 512 to 1024 and is a multiple of 64.
  - A prime modulus *q* that divides *p-1* and that is 160 bits long.
  - A generator $g = h^{(p-1)/q} \bmod p$. where h is any number less than *p-1* such that $h^{(p-1)/q} \bmod p$ is greater than 1.

DSA

- The number *p*, *q*, and *g* can then be considered as system parameters that are shared and can be the same for all users.
- Select a private DSA signing key $x \in Z_q$
- Compute a corresponding public DSA verification key $y = g^x \bmod p$.

DSA

Message Signing :

- The algorithm consists of three steps:
  - First, a number k must be randomly selected from Z_p^*.
  - Second, k must be used to compute $r = (g^k \bmod p) \bmod q$.
  - Third, the message m must be hashed with h, and the result $h(m)$ must be used to compute $s = (k^{-1}(h(m) + xr)) \bmod q$.
- The pair *(r, s)* with *r* and *s* elements of Z_q represents the digital signature for *m* or *h(m)*.
- Note that *r* and *s* are both 160-bit numbers.

DSA

Signature verification:

- Verify that $r, s \in Z_q$, that is, $0 < r' < q$ and $0 < s' < q$. If r or s is not in Z_q, then the signature is considered to be invalid and must be rejected accordingly.
- If, however, the two conditions are satisfied, then the algorithm must compute the following values:

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

$$u_1 = h(m)w \bmod q$$

$$u_2 = rw \bmod q$$

$$v = (g^{u_1} y^{u_2} \bmod p) \bmod q$$

DSA

- The signature is valid if and only if $v = r$.

DSA

Example: Alice wants to sent Bob message m with a hash value $h(m) = 6$.

Key generation:

- Alice chose p = 23, q = 11, and g = 2.

  (Note that q = 11 is prime and divides p – 1 = 22.)

  (Note that $g = 5^2 \bmod 23 = 2$)

- Alice chose x = 3 and compute

  $y = g^x \bmod p = 2^3 \bmod 23 = 8$

- Thus, Alice's signing key = 3, and verification key (public key) is 8.

DSA

Message signing:

- To sign the message, Alice chooses k = 7 and compute

$$r = (g^k \bmod p) \bmod q$$
$$= (2^7 \bmod 23) \bmod 11$$
$$= 13 \bmod 11$$
$$= 2$$

- Next Alice determines $k^{-1} \bmod q$

$$7^{-1} \bmod 11 = 8$$

DSA

- Alice computes

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

  $s = (8 (6 + 3 \times 2)) \bmod 11$

  $= 8.$

  Thus Alice signature for the message is (2, 8).

DSA

Signature verification:

- To verify, Bob verifies that
  - 0 < 2 < 11, that is 0 < r < q Positive!
  - 0 < 8 < 11, that is 0 < s < q Positive!

- Bob computes:
  - $w = 8^{-1} \bmod 11 = 7$
  - $u_1 = 6 \times 7 \bmod 11 = 42 \bmod 11 = 9$
  - $u_2 = 2 \times 7 \bmod 11 = 14 \bmod 11 = 3$
  - $v = (2^{u1} \times 8^{u2} \bmod 23) \bmod 11 = 2^9 \times 8^3 \bmod 23 \bmod 11$

    $= 512 \times 512 \bmod 23 \bmod 11 = 2$

DSA

- V = 2 and r = 2, thus Bob is assured that the signature is authentic.

DSA

DSA Summary

Key generation :

Public key:

p 512-bit to 1024-bit prime. (Can be shared among a group of users)

q 160-bit prime factor of p-1, that it, q that divides p-1. (Can be shared among a group of users)

$g = h^{(p-1)/q} \bmod p$, where h is less than p-1 and $h^{(p-1)/q} \bmod p > 1$. (Can be shared among a group of users)

$y = g^x \bmod p$ (a p-bit number)

Private key:

x < q (a 160-bit number)

DSA

Message signing:

k choose at random, less than q

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

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

Signature verification:

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

$u_1 = (H(m) \times w) \bmod q$

$u_2 = (rw) \bmod q$

$v = ((g^{u_1} \times y^{u_2}) \bmod p) \bmod q$

If $v = r$, then the signature is verified.

DSA

- Show that DSA works.

In Digital Signature Algorithm (DSA), the private signing key is $u \in Z_q$, and the verification key is $y = g^u \ mod \ p$.

The message m is signed as followed:

$r = \left(g^k mod \ p\right) mod \ q$

$s = k^{-1}(h(m) + ur) \ mod \ q$

$where \ u \in z_q \ is \ the \ private \ signing \ key.$

The signed message is $\left(h(m), (r, s)\right)$

DSA

- To verify, the message recipient computes $\frac{r \times h(m)}{s}$ in the following three steps:
  i. $w = s^{-1} mod\ q$
  ii. $u_1 = h(m)w\ mod\ q$
  iii. $u_2 = rw\ mod\ q$

DSA

- Correctness of the algorithm:

$$r =? \ (g^{u_1} y^{u_2} \ mod \ p) \ mod \ q$$

$$=? \ \left(g^{h(m)w} \ y^{rw} \ mod \ p\right) mod \ q,$$

$$since \ y = g^{u} \ mod \ p$$

$$=? \ \left(g^{h(m)w} \ (g^{u})^{rw} \ mod \ p\right) mod \ q$$

$$=? \ \left(g^{h(m)w} \ g^{urw} \ mod \ p\right) mod \ q$$

$$=? \ \left(g^{h(m)w+urw} \ mod \ p\right) mod \ q$$

$$=? \ \left(g^{w(h(m)+ur)} \ mod \ p\right) mod \ q$$

$$=? \ \left(g^{s^{-1}(h(m)+ur)} \ mod \ p\right) mod \ q$$

$$=? \ \left(g^{\left(k(h(m)+ur)^{-1}\right)(h(m)+ur)} \ mod \ p\right) mod \ q$$

$$=? \ \left(g^{k} \ mod \ p\right) mod \ q \quad since \ \ r = \left(g^{k} mod \ p\right) mod \ q$$

$$r = r \ mod \ q$$