

Topic 6. Digital Signatures and Identity Based Encryption

- 1 **Security** of public-key cryptography
- 2 **Diffie-Hellman (DH)** key exchange under authentic channels
- 3 **RSA** encryption and digital signature
- 4 **ElGamal** digital signature and Digital Signature Standard (DSS)
- 5 Elliptic curve digital signature algorithm (**ECDSA**)
- 6 **Identity** based cryptography (IBC) schemes from Bilinear Pairing
- 7 **LFSR** based DSA
- 8 **Fault** attacks

Properties of public and private key pairs

Each user has a key pair (E_x, D_x) which satisfy the following properties:

- E_x is an encryption key which needs to be certified, called a **public-key** of the user x , which can be made in a public domain.
- D_x is the decryption key, which should be kept privately, called a **private key** of the user. The private and public key pair satisfy the following conditions.
 - 1 $D_x(E_x) = E_x(D_x) = \text{identity map}$.
 - 2 From known E_x , it is **computational infeasible** to obtain D_x .

6.1. Security of public-key cryptosystems

- The **security** of public-key cryptosystems is based on the difficulty of some computational hard problems.
- The most important problems:
 - ▶ **factoring** large integers,
 - ▶ finite field **discrete logarithms**, and
 - ▶ **elliptic curve** discrete logarithms.

Design of Public-key Crypto Schemes

- **One-way function:** A one-way function is a function that is easy to compute but “hard to invert”:

a) Given the input x , there is some algorithm which can compute the function $f(x)$ in polynomial time (in the input size).

b) There is no probabilistic polynomial-time algorithm can compute a preimage of $f(x)$ with better than negligible probability, when x is chosen at random.

- **Trapdoor one-way function:**

It is easy to compute the image $f(x)$ for given x , but it is hard to find its inverse, i.e., given $y = f(x)$, it is hard to find the preimage x , without any special information which is called the “**trapdoor**”.

6.2 Diffie-Hellman (DH) key exchange

The system parameters:

- p : a prime number.
- g : a primitive element in $GF(p)$.

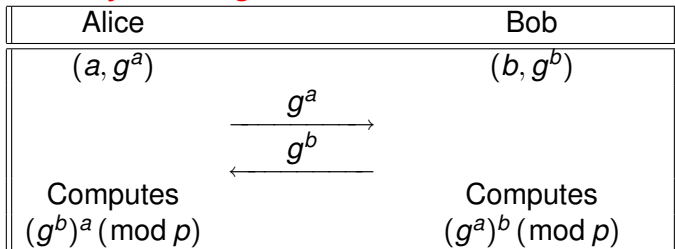
User Alice's private-key and public-key pair:

- **Private key**: $sk_A = a$, $0 < a < p$ and $\gcd(a, p - 1) = 1$.
- **Public key**: $pk_A = g^a$.

Bob's private-key pair and public-key pair:

- **Private key**: $sk_B = b$, $0 < b < p$ and $\gcd(b, p - 1) = 1$.
- **Public key**: $pk_B = g^b$.

DH Key Exchange Under an Authentic Channel



Security of DH key exchange protocol

Attacker: Known both public-keys, i.e., g^a and g^b , and the goal is to obtain the shared values g^{ab} .

Diffe-Hellman Problem: Given g^a and g^b , compute g^{ab}

- DH key exchange is secure if the Diffe-Hellman problem is computationally infeasible.
- **DH problem** is computational feasible if solving discrete logarithm in $GF(p)$ is computationally feasible.
- Thus, **the security** of the Diffe-Hellman key exchange scheme is based on the difficulty of solving discrete logarithm in the finite field $GF(p)$.

6.3 RSA encryption and digital signature

Key Generation: For each user, Bob:

- 1 **Generate** two large prime numbers p and q .
- 2 **Compute** $n = pq$, and $\phi(n) = (p - 1)(q - 1)$.
- 3 Choose a **random** number $e : 0 < e < \phi(n)$ such that $\gcd(e, \phi(n)) = 1$.
- 4 Compute $d = e^{-1} \pmod{\phi(n)}$.
- 5 **Public-key:** $pk_B = (n, e)$. Those keys need to be certified.
- 6 **Private key:** $sk_B = (d, p, q)$.

Encryption: For plaintext $m < n$, ciphertext c is computed as $c = m^e \pmod{n}$ using the public-key (n, e)

Decryption: For ciphertext c , the plaintext m is retrieved by computing $m = c^d \pmod{n}$ using the private-key (d, p, q) .

Security of RSA

- **Attacker**: known $(n, e) \implies d$?
- The security of RSA depends on the difficulty of **factorization** of a large integer n .

Selection of p and q

- 1 p and q should differ in length only a few digits.
- 2 Both $p - 1$ and $q - 1$ should contain a **large prime** factor.
- 3 $\gcd(p - 1, q - 1)$ should be **small**.
- 4 d should not be small: $d > n^{0.292}$.

RSA Digital Signature Algorithm (RSA-DSA)

Requirements for digital signatures:

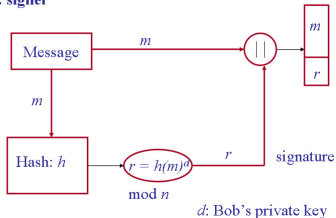
- 1 Everyone can **verify** a digital signature of some message.
- 2 Only the **signer** can sign, and no one can forge the signer's signature (this prevents forgery and denial attacks).
- 3 Once a dispute occurs, a **third party** can resolve it.

Signing Process:

Bob: signer

Signing Message m
Compute: $h(m)$ and $r = h(m)^d \pmod{n}$
r is a digital signature of the message m

Bob: signer

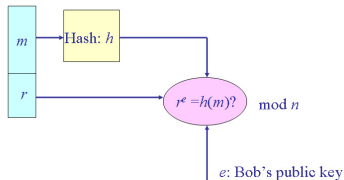


Verifying Process:

Verifier:

Verifying the Digital Signature r of Message m
Compute: $r^e \pmod{n}$
Check whether $r^e = h(m)$
If it is true, accept it as a valid signature; otherwise, reject it.

Alice: verifier



6.4 ElGamal digital signature and Digital Signature Standard (DSS)

System Parameters

- 1 p : a prime number of 1024 bits.
- 2 q : a **prime factor** of $(p - 1)$ with $2^{159} < q < 2^{160}$, i.e., the bit length of q is 160.
- 3 $g \in GF(p)$ with **order** q .
- 4 h : a cryptographic **hash function** h . In the original DSS, h was selected as SHA 1.
- 5 **PRNG is** not specified in the DSS.

The signer, say Bob:

Private key: $sk_B = x, 0 < x < q$;

Public key: $pk_B = y = g^x$.

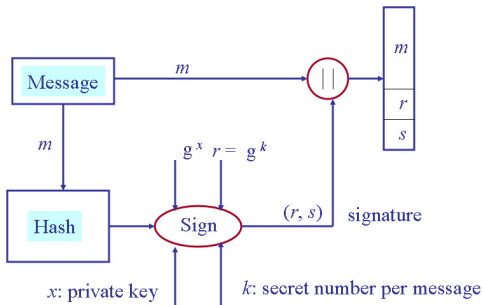
DSS Signature Generation

- a) Randomly pick or generated by PRNG $k, 0 < k < q$ (**per message**), and compute $r = g^k$.
- b) Solve for s in the equation: $h(m) \equiv xr + ks \pmod{q}$.

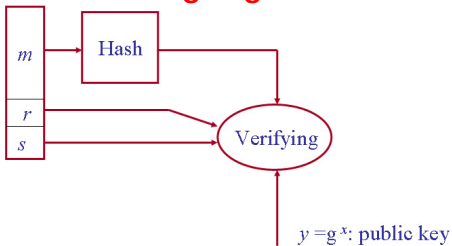
The pair (r, s) is a **digital signature** of the message m (both r and s are 160-bit numbers).

DSS Verification Process

- a) **Reject** the signature if either $0 < r < q$ or $0 < s < q$ is not satisfied.
- b) **Set** $u = h(m)s^{-1} \pmod{q}$, and $v = -rs^{-1} \pmod{q}$.
- c) **Compute** $w = g^u y^v$.
- d) Check whether $w = r$. If it is true, accept it as a valid signature; otherwise, reject it.



DSS Signing Process



DSS Verifying Process

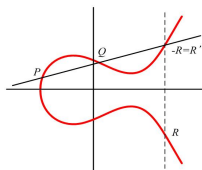
Security of DSS

The security of DSS depends on the following four factors.

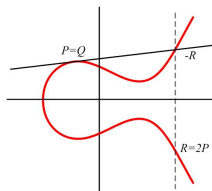
- 1 **Finite field** DL,
- 2 **Collision** resistance of hash function,
- 3 **Randomness** of PRNG, and
- 4 Solving an **equation** with two unknowns.

6.5 Elliptic Curve Cryptography (ECC)

- F : a finite field.
- An elliptic curve E over F is **a set of points** (x, y) with $x, y \in F$ which satisfy some cubic equation.
- **Operation on E** : addition of two points. The E , the elliptic curve $y^2 = x^3 - x$ in xy - real number plane, is sketched below for two different points $R = P + Q$, and for two same points $R = 2P$.



Point Addition



Point Doubling

Public-key and Private Keys in ECC

- **System parameters**: G is a subgroup of the elliptic curve E with order q where q is a prime, and P , an arbitrary generator of G .
- **Key pairs** for Alice and Bob:

	Alice	Bob
Private key:	$x_A, 1 < x_A < q$	$x_B, 1 < x_B < q$
Public key:	$Q_A = x_A P$	$Q_B = x_B P$
Key pairs	(x_A, Q_A)	(x_B, Q_B)

- Shared Diffie-Hellman Key:

$$x_B Q_A = (x_A x_B) P = x_A Q_B$$

- **Security**: Given Q_A , compute x_A is hard, i.e., compute discrete logarithm in E is hard.

6.6 Identity-based Cryptography (IBC) from Bilinear Pairing

Pre-shared Secret Keys in IBC

A. Bilinear Maps

\mathbb{G}_1 : a group of points on an elliptic curve; and P , an arbitrary generator of \mathbb{G}_1 . \mathbb{G}_2 : a multiplicative subgroup of a finite field. Both has order q .

A **bilinear mapping**:

$$\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \mapsto \mathbb{G}_2$$

which satisfy the following properties:

- **Bilinearity:** $\hat{e}(aP, bQ) = \hat{e}(P, Q)ab$ for all $P, Q \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_q^*$.
- **Non-degeneracy:** If P is a generator of \mathbb{G}_1 , then $\hat{e}(P, P)$ is a generator of \mathbb{G}_2 . In other words, $\hat{e}(P, P) \neq 1$.
- **Computable:** There exists an efficient algorithm to compute $\hat{e}(P, Q)$ for all $P, Q \in \mathbb{G}_1$.

System Setup

In any IBC scheme, a key generation center (KGC) is needed.

KGS conducts the following process.

- 1 **Chooses** q , generates, two groups \mathbb{G}_1 and \mathbb{G}_2 of order q , and a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \mapsto \mathbb{G}_2$. The KGC chooses a random generator $P \in \mathbb{G}_1$.
- 2 **The master private key:** $s \in \mathbb{Z}_q^*$
master public key: $P_{pub} = sP$.
- 3 A hash function H_1 that maps an arbitrarily long binary string to an element in group \mathbb{G}_1 .

The KGC publishes

$$param = \langle \mathbb{G}_1, \mathbb{G}_2, q, P, \hat{e}, P_{pub}, H_1 \rangle$$

as public parameters and keeps s secret.

The KGC's private and public key pair is (s, P_{pub}) .

Extract process: key generation

Node i : submits his ID_i to the KGC.

KGC conducts:

- The **public key** $Q_i \in \mathbb{G}_1^*$ of a node i with identity ID_i is as

$$Q_i = H_1(ID_i), \quad (1)$$

where ID_i is an arbitrarily long binary string $\in \{0, 1\}^*$.

- The KGC derives the private keys d_i for node i as

$$d_i = sQ_i.$$

- **Securely** distributes this key pair (d_i, Q_i) to the node i .

Observations

- We can observe that instead of directly using identities as public keys, as done in Shamir's scheme, here the **identity string** is first mapped to a point on an elliptic curve using hash function H_1 .
- Note that **public keys** can be computed from publicly available information, whereas private keys can only be computed by the KGC because the computation requires the KGC's private key s as input.
- The key distribution channel between KGC and nodes needs to be **authentic and confidential**.

Extension of ID-based Public Key

- To **limit the validity** period of an ID-based public key, an expiry date can be embedded in the key itself. This can be done by concatenating an expiry date t_x to the public key with

$$Q_i = H_1(ID_i || t_x)$$

for the public key Q_i of user i .

- Only if user i is in **possession of the matching private key** that corresponds to the correct date, i.e. $d_i = sH_1(ID_i || t_x)$, he can sign or decrypt messages.
- The **granularity** of the validity period is a system parameter that describes a security/efficiency trade-off.
- For instance, a **shorter period** reduces the risk of key compromise but induces more overhead because users need to frequently obtain fresh private keys from the KGC.

Non-interactive Pre-shared Secret Key

- **Each pair** of users i and j in a pairing-based IBC scheme is able to compute a pairwise pre-shared secret key K_{ij} with

$$\begin{aligned}K_{ij} &= \hat{e}(d_i, Q_j) = \hat{e}(Q_j, d_i) \\ &= \hat{e}(Q_i, Q_j)^s\end{aligned}$$

- **This key** is shared between users i and j in a **non-interactive fashion**, since both parties compute the bilinear mapping $\hat{e}(\cdot)$ over their own private key d_i and the public key Q_j of the desired communication partner.
- **Note** that the KGC is able to compute all pre-shared keys.