04. Digital Signatures

이형태

2019학년도 2학기

Definitions of Digital Signatures

Overview of Digital Signatures

- **Message authentication**: Once Bob sends a message to Alice, he wants Alice to be certain that the message is indeed from him, though it is not important that the message be kept secret. $\Rightarrow$ Use digital signatures!
  1. Bob generates a signature using his secret key and sends the message with the signature attached to Alice.
  2. Alice verifies the received signature using Bob's public key and the message.
- **Security requirement**: No one can generate Bob's valid signature if he/she does not have Bob's secret key.
- The message is authenticated both in terms of source (Bob) and in terms of data integrity (message).

Digital Signatures

Definition (Digital Signatures)

A digital signature scheme consists of the following three polynomial-time algorithms:

- $\mathrm{KeyGen}(\lambda) \rightarrow (pk, sk)$: It takes a security parameter λ as an input and returns a public key pk and a secret key sk.
- $\mathrm{Sign}(sk, M) \rightarrow \sigma$: It takes the secret key sk and a message M as inputs and returns a signature σ.
- $\mathrm{Verify}(pk, \sigma) \rightarrow 1/0$: It takes the public key pk and a signature σ as inputs and returns 1(Accept)/0(Reject).

Correctness

A digital signature is correct if for any security parameter λ and any message M,

$$\mathrm{Verify}(pk, \mathrm{Sign}(sk, M)) = 1$$

where (pk, sk) is an output of $\mathrm{KeyGen}(\lambda)$.

Security Model for Digital Signatures

- Consider the following game between the challenger $\mathcal{C}$ and the adversary $\mathcal{A}$:
  1. **Setup**: $\mathcal{C}$ runs KeyGen(λ)$\rightarrow (pk, sk)$ and passes pk to $\mathcal{A}$.
  2. **Signing Queries**: $\mathcal{A}$ issues signing queries on messages M_i polynomially many times. For each M_i, $\mathcal{C}$ runs Sign(pk, M_i)$\rightarrow \sigma_i$ and returns σ_i to $\mathcal{A}$.
  3. **Output**: $\mathcal{A}$ outputs a pair (M, σ).
- The success probability of $\mathcal{A}$ in the above game is defined to

$$\Pr[\mathsf{Verify}(pk, (M, \sigma)) = 1]$$

  where (M, σ) is not generated in Step 2.
- A signature scheme is strongly unforgeable under adaptive chosen message attack if for any polynomial-time adversary $\mathcal{A}$ the success probability of the above game is negligible in the security parameter.

Signature Schemes

RSA Signatures

Key Generation

1. Choose two large primes p and q, and set $N = pq$.
2. Select a random public exponent $e \in \mathbb{Z}^*_{\phi(N)}$.
3. Compute d such that $d \cdot e \equiv 1 \pmod{\phi(N)}$.
4. Output a public key $pk = (N, e)$ and a secret key $sk = d$.

Sign

Given a message M, compute $s = M^d \pmod{N}$ and output $\sigma = (M, s)$.

Verify

Given $\sigma = (M, s)$, check whether $s^e \stackrel{?}{=} M \pmod{N}$. If it holds, return 1. Otherwise, return 0.

Correctness

$s^e = (M^d)^e = M^{ed} = M^{s\phi(N)+1} = M \pmod{N}$ by Euler Theorem

Security of RSA Signature

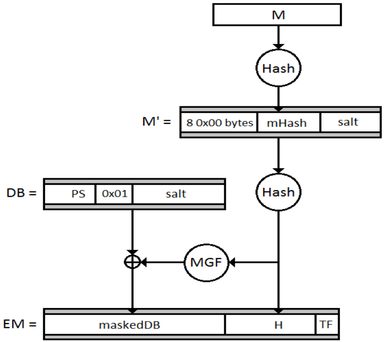
- Difficult to recover the secret key d if the factoring problem is hard
- Existential forgery attack:
    1. Choose a signature $s \in \mathbb{Z}_N$.
    2. Compute the message $M = s^e \pmod{N}$.
    3. Output $\sigma = (M, s)$.

  $\Rightarrow$ Then, it holds that $M = s^e \pmod{N}$ and the signature σ is valid.

RSA Padding: Probabilistic Signature Standard (PSS)

- Prevent the previous attack by allowing only certain message formats
- Probabilistic Signature Standard (PSS)

1. Generate a random value *salt*.
2. Form a string M' by concatenating a fixed padding 8 0x00's, the hash value $mHash = Hash(M)$, and *salt*.
3. Compute $H = Hash(M')$.
4. Form a string DB by concatenating a fixed padding PS, 0x01 and *salt*.
5. Compute $MGF(H)$.
6. Compute $maskedDB = MGF(H) \oplus DB$.
7. The encoded message $EM = maskedDB \| H \| TF$ for the fixed padding TF.



ElGamal Signatures

Key Generation

1. Choose a large prime p and select a generator g of a large subgroup of $\mathbb{Z}_p^*$.
2. Choose a random integer $x \in \{2, 3, \ldots, p-2\}$.
3. Compute $X = g^x \pmod{p}$.
4. Output a public key $pk = (p, g, X)$ and a secret key $sk = x$.

Sign

Given the secret key $sk = x$ and a message M,

1. Choose a random element k from $\mathbb{Z}_{p-1}^*$.
2. Compute $r = g^k \mod p$ and $s = (M - rx)k^{-1} \pmod{p-1}$.
3. Output $\sigma = (M, (r, s))$.

ElGamal Signature (Cont.)

Verify

Given the public key $pk = (p, g, X)$ and a signature $\sigma = (M, (r, s))$,

1. Compute $t = X^r \cdot r^s \pmod{p}$.
2. Check whether $t \stackrel{?}{=} g^M \pmod{p}$. If it holds, return 1. Otherwise, return 0.

Correctness

$$\begin{aligned} t &= X^r \cdot r^s = (g^x)^r (g^k)^s \\ &= (g^x)^r (g^k)^{(M-rx)^{k^{-1}}} = g^{xr+M-rx} = g^M \pmod{p} \end{aligned}$$

Security of ElGamal Signature

- Difficult to recover the secret key x if the discrete logarithm problem is hard
- Existential forgery attack:
  1. Choose integers i, j where $\gcd(j, p-1) = 1$.
  2. Compute $r = g^i X^j \pmod{p}$.
  3. Compute $s = -rj^{-1} \pmod{p-1}$.
  4. Compute $M = si \pmod{p-1}$.
  5. Output $\sigma = (M, (r, s))$.

  $\Rightarrow$ Then, it holds that $t = X^r \cdot r^s \pmod{p}$ where $t = g^M \pmod{p}$ since

$$
\begin{aligned}
t &= X^r \cdot r^s = (g^x)^r (g^i g^{xj})^s = g^{xr+si+sxj} \\
&= g^{xr+si-rj^{-1}xj} = g^{si} = g^M \pmod{p}
\end{aligned}
$$

  and thus $\sigma = (M, (r, s))$ is valid.

- To prevent the above attack use $H(M)$ instead of M itself where H is a hash function. If H is hard to find an inverse, then it is secure.

DSA Signatures

Key Generation

1. Generate a cyclic subgroup $\mathbb{G}$ with order q of $\mathbb{Z}_p^*$.
2. Select a random generator g of $\mathbb{G}$.
3. Choose a random integer x from $\mathbb{Z}_q^*$ and compute $X = g^x$ in $\mathbb{G}$.
4. Output a public key $pk = (p, q, g, X)$ and a secret key x.

Sign

Given the public key $pk = (p, q, g, X)$ and a message M,

1. Choose a random integer k from $\mathbb{Z}_q^*$.
2. Compute $r = (g^k \bmod p) \bmod q$.
3. Compute $s = (H(M) + xr)k^{-1} \bmod q$.
4. Output $\sigma = (M, (r, s))$.

DSA Signature (Cont.)

Verify

Given the secret key $sk = x$ and a signature $\sigma = (M, (r, s))$,

1. Compute $w = s^{-1} \pmod q$.
2. Compute $u_1 = w \cdot H(M) \pmod q$.
3. Compute $u_2 = w \cdot r \pmod q$.
4. Compute $\nu = (g^{u_1} X^{u_2} \pmod p) \pmod q$.
5. Check whether $\nu \stackrel{?}{=} r \mod q$. If it holds, return 1. Otherwise, return 0.

Correctness

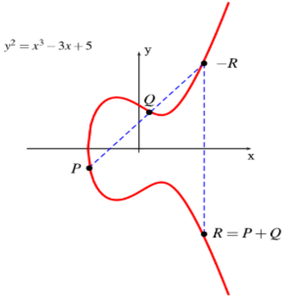
$$
\begin{aligned}
g^{u_1} X^{u_2} &= g^{u_1 + xu_2} = g^{w \cdot (H(M)+xr)} \\
&= g^{k(H(M)+xr)^{-1}(H(M)+xr)} = g^k \ (\because \ s = k^{-1}(H(M) + xr))
\end{aligned}
$$

$\Rightarrow r \pmod q = g^k \pmod q = g^{u_1} X^{u_2} \pmod q = \nu$

Elliptic Curve Cryptography

Elliptic Curves (EC) in Cryptography

- Use the set of points on elliptic curves, e.g., $E : Y^2 = X^3 + aX + b$ over a field $\mathbb{F}$
  $\Rightarrow$ additive (cyclic) groups
- Based on the hardness of Discrete Logarithm Problem
- Pros: Short parameter sizes (160 bits (EC) vs. 1024 bits (RSA) for 80-bit security)
- Cons: Complicated computations for addition
- Standards such as IEEE P1363



Picture from http://www.purplealienplanet.com/node/27

ECDSA

Key Generation

1. Use an elliptic curve E with
   - modulus p
   - coefficient a and b
   - a point G which generates a cyclic group of prime order q (i.e. G: a generator).
2. Choose a random integer x with $0 < x < q$ and compute $X = xG$.
3. Output a public key $pk = (p, a, b, q, G, X)$ and a secret key $sk = x$.

Sign

Given the public key $pk = (p, a, b, q, G, X)$ and a message M

1. Choose a random integer k with $0 < k < q$.
2. Compute $K = kG$ and let $r = x_K$ where x_K is the x-coordinate of the point K.
3. Compute $s = (H(M) + xr)k^{-1} \pmod{q}$.
4. Output $(M, (r, s))$.

ECDSA (Cont.)

Verify

Given the secret key $sk = x$ and a signature $(M, (r, s))$,

1. Compute $w = s^{-1} \pmod{q}$.
2. Compute $u_1 = w \cdot H(M) \pmod{q}$.
3. Compute $u_2 = w \cdot r \pmod{q}$.
4. Compute $P = u_1 G + u_2 X$.
5. Check whether $x_P \stackrel{?}{=} r \pmod{q}$. If it holds, return 1. Otherwise, return 0.

Correctness

$$\begin{aligned} P = u_1 G + u_2 X &= (u_1 + x u_2) G = (w(H(M) + xr))G \\ &= k(H(M) + xr)^{-1}(H(M) + xr)G = kG \end{aligned}$$

$\Rightarrow r =$ the x-coordinate of $kG =$ the x-coordinate of $P \pmod{q}$

cf. Elliptic Curve ElGamal

Key Generation

1. Use an elliptic curve E with
   - modulus p
   - coefficient a and b
   - a point G which generates a cyclic group of prime order q (i.e. G: a generator).
2. Choose a random integer x with $0 < x < q$ and compute $X = xG$.
3. Output a public key $pk = (p, a, b, q, G, X)$ and a secret key $sk = x$.

Encryption

Given the public key $pk = (p, a, b, q, G, X)$ and a message M,

1. Choose a random element r from $\mathbb{Z}_q^*$.
2. Compute $C_1 = rG$ and $C_2 = M + rX$ and output $C = (C_1, C_2)$.

Decryption

Given the secret key $sk = x$ and a ciphertext $C = (C_1, C_2)$, compute and output

$$C_2 - xC_1 (= M + rX - xrG = M + rX - rX).$$

References

PP10 C. Paar and J. Pelzl, Understanding Cryptography, Springer, 2010

Sho08 V. Shoup, A Computational Introduction to Number Theory and Algebra, 2nd ed., Cambridge University Press, 2008. (Chapter 2)