Digital Signatures

Cryptography and Protocols
Andrei Bulatov

Definition

- Similar to symmetric case we need to care about data integrity
- A triple $(\mathrm{Gen}, \mathrm{Sign}, \mathrm{Ver})$ is called a (T, ε)-secure signature scheme if

  validity for any pair (s, v) generated by Gen and every $P \in \{0,1\}^n$ we have $\mathrm{Ver}_v(P, \mathrm{Sign}_s(P)) = 1$

  security for any Eve with time complexity at most T in the following game:

  - Alice chooses (s, v)
  - Eve gets black box access to Sign_s (she has access to Ver_v)
  - Eve wins if in the end she produces a pair (P, σ) such that

    (a) P was not queried

    (b) $\mathrm{Ver}_v(P, \sigma) = 1$

Definition (cntd)

- The probability Eve wins

$$\Pr[\text{Eve wins}] < \varepsilon$$

- A scheme is secure if it is (T, ε)-secure for a superpolynomial pair (T, ε)

One-Time Signature Scheme

- Eve is allowed to make only one query and we certify only one bit
- Thus Eve's task is: Given $(b, \mathrm{Sign}_s b)$ find $(\bar{b}, \sigma)$ such that $\mathrm{Ver}_v(\bar{b}, \sigma) = 1$
- We use a one-way permutation $f\colon \{0,1\}^n \to \{0,1\}^n$ with
$$\Pr_{x \in \{0,1\}^n}\left[\mathrm{Eve}(f(x)) = x\right] < \varepsilon(n)$$
for any polynomial time Eve and some superpolynomial ε
- **Key generation**: Gen chooses $x^0, x^1 \in \{0,1\}^n$ and computes $y^0 = f(x^0), y^1 = f(x^1)$. Then set $s = (x^0, x^1)$ and $v = (y^0, y^1)$.
- **Signing**: $\mathrm{Sign}_s(b) = x^b$
- **Verification**: $\mathrm{Ver}_v(b, x) = 1 \Leftrightarrow f(x) = y^b$

Extending to Longer Messages

- Generate a pair for each bit of a message

Signing Messages Longer Than Key Length

- We use a hash function
- A collection of functions $\{h_k\}_{k\in\{0,1\}^*}$ with $h_k\colon\{0,1\}^{2n}\to\{0,1\}^n$ for $k\in\{0,1\}^n$ is called (T,ε)-collision resistant if the function $(k,x)\mapsto h_k(x)$ is polynomial time computable and for any Eve of time complexity at most T we have
  $$\Pr_k[\mathrm{Eve}(k)=(x,x') \text{ such that } h_k(x)=h_k(x')]<\varepsilon$$
- Having a hash function $h_k\colon\{0,1\}^{2n}\to\{0,1\}^n$ we can construct a function $h_k\colon\{0,1\}^{n^3}\to\{0,1\}^n$

Signing Messages Longer Than Key Length (cntd)

- We use a signature scheme $(\mathrm{Gen}', \mathrm{Sign}', \mathrm{Ver}')$ that signs n-bit messages with a key of n^2 bits long, and a hash function collection $\{h_k\}$ where for $k \in \{0,1\}^n$ we have $h_k: \{0,1\}^{n^3} \to \{0,1\}^n$

- **Key generation:** Gen uses Gen' to choose a pair (s', v') of the signature scheme for messages of length n, and a key $k \in \{0,1\}^n$ for the hash function.

  Public key: (v', k) Private key: s'

  Note that $n^2 + n \ll n^3$ where $n^2 + n$ is the key length

- **Singing**: To sign a message $P \in \{0,1\}^{n^3}$ Sign computes $P' = h_k(P)$ and then $\mathrm{Sign}'_{s'}(P')$
- **Verification:** $\mathrm{Ver}_v(P, \sigma) = 1 \Leftrightarrow \mathrm{Ver}'_{v'}(h_k(P), \sigma) = 1$

From One-Time to Many-Times Scheme

- Observe that a one-time scheme for signing messages of length m can be converted into a two-time scheme for messages of length $m/2$. Assume, we have a two-time scheme with n-bit public key for $2n$-bit messages
- **Key generation**: (s_0, v_0) initially, (s_i, v_i) at time i

  $v_0, \dots, v_i$ is the public key, $s_0, \dots, s_i$ the private key
- **Signing**: At time i to sign a message, first, generate (s_i, v_i)

  use s_{i-1} to sign v_i and obtain a signature σ_i

  sign P using s_i to obtain a signature σ

  the signature is the list $\sigma_1, \dots, \sigma_i, \sigma$
- **Verification**: Using v_{j-1} check for all $j > 0$ that σ_j is a signature for v_j and v_j to check that σ is a signature for P

Practical Signature Schemes

- Idea: Use a trapdoor permutation

  To sign a message `decrypt' it

Practical Signature Schemes: Rabin Signatures

- **Key generation**: Choose random $p, q \equiv 3 (\mathrm{mod}\ 4)$.
  p, q is a private / signing key
  $n = p \cdot q$ is a public key
- **Signing**: To sign P compute $\sigma = \sqrt{P}$ (how?)
  there are 4 square roots of P, choose any.
- **Verification**: Check that $\sigma^2 = P$

- This scheme is not secure
- Eve: choose a random $X \in \mathbb{Z}_n^*$ and $P = X^2$
  given $\sigma = \sqrt{P}$ with probability 1/2 $\sigma \neq \pm X$
  then $\gcd(\sigma - X, n)$ is a nontrivial divisor of n

Rabin Scheme (cntd)

- A fix: Before signing apply a hash function
- Hope: It is difficult to find P such that $X^2 \equiv h(P) \pmod{n}$

- It does not suffice in general. Collision resistance does not guarantee this property.

  Indeed, if h is the identity function, it is collision resistant, but does not work
- Or it may be that $h(c \cdot P) = c \cdot h(P)$ for some quadratic residue c

  Indeed, Eve can ask for a signature σ of P, and then output $\sqrt{c} \cdot \sigma$ as a signature for $c \cdot P$

Practical Signature Schemes: RSA Scheme

- **Key generation**: choose random primes p, q of length k
  $n = p \cdot q$. Note that $\varphi(n) = (p-1)(q-1)$
  choose e at random from $\mathbb{Z}^*_{\varphi(n)}$
  private key $d \equiv e^{-1}(mod\ \varphi(n))$ public key n, e
- **Signing**: $\sigma \equiv P^d (\mathrm{mod}\, n)$
- **Verification**: check if $\sigma^e \equiv P(\mathrm{mod}\, n)$

- Not secure!!!
- Eve can ask for signatures σ_1 and σ_2 for P_1 and P_2
  Then return $(P_1 \cdot P_2, \sigma_1 \cdot \sigma_2)$

Digital Signature Standard

- The Digital Signature Algorithm (DSA) is a United States Federal Government standard or FIPS for digital signatures
- It was proposed by the NIST 1991 for use in their Digital Signature Standard (DSS), specified in FIPS 186
- Revised in 1993, 1996, 2000, and 2009
- Uses a cryptographic hash function, SHA-1 or SHA-2
- Uses the discrete logarithm problem as the basis of security

- Key: q is an N-bit prime ($N = 160, 224, 256$)

  p is an L-bit prime ($L = 1024, 2048, 3072$) such that $q | p - 1$

  g a residue of order q modulo p

Digital Signature Standard (cntd)

- X is random with $0 < X < q$

  $Y = g^X (\mathrm{mod}\, p)$
- Public key: (p, q, g, Y)

  Private key: X
- Signing:

  - generate random k, $0 < k < q$
  - calculate $r \equiv (g^k (\mathrm{mod}\, p))(\mathrm{mod}\, q)$
  - calculate $s \equiv k^{-1}(H(P) + Xr)(\mathrm{mod}\, q)$
  - signature is (r, s)

Digital Signature Standard (cntd)

- Verification:
  - calculate $w \equiv s^{-1} (\mathrm{mod}\ q)$
  - calculate $u_1 \equiv H(P) \cdot w\ (\mathrm{mod}\ q)$
  - calculate $u_2 \equiv r \cdot w\ (\mathrm{mod}\ q)$
  - calculate $v \equiv \big((g^{u_1} Y^{u_2}) (\mathrm{mod}\ p)\big) (\mathrm{mod}\ q)$
  - the signature is valid if $v = r$
- Correctness

How to Prove Security

- The security of practical schemes is not proved
- How could we prove security of such a scheme with a hash function:
  - define sufficiently `crazy' hash functions
  - using factoring / DDH / PRG / OWP construct a sufficiently crazy hash function
  - prove that Rabin / RSA is secure when using this construction
- Cannot make even the first step

Random Oracle Model

- Is our signature scheme secure if we use a random function instead of a hash function?
- It is called the Random Oracle Model
- Then we hopefully can replace a random function with a PRF
- Problem: with PRF we know that Eve cannot distinguish it from a random function when given as a black box.

  But in this case Eve has the seed / key / description of the PRF
- The Random Oracle Thesis:

  If a protocol is secure in the random oracle model, then it is secure when instantiated with a "sufficiently crazy" hash function
- Looks true in practical cases
- False in general