Digital signature schemes

Diego F. Aranha

Institute of Computing
UNICAMP

Introduction

Asymmetric cryptographic provides confidentiality, but how to provide integrity, authentication and non-repudiation?

Problem: How to protect against chosen-ciphertext attacks?

Solutions: Digital signatures!

Analogous to hand signatures:

- Entity S with public key b "signs" a message m in a way that allows anyone to verify the origin of S and that m was not modified.

Main applications:

- Secure distribution of software.
- Management of electronic documents.

Introduction

Advantages over MACs:

- No need to establish a shared key with each destination.
- Public and transferable verification.
- Non-repudiation.

Disadvantages compared to MACs:

- Message expansion.
- Lower performance.

Important: Signature operation is not necessarily an inverse of asymmetric encryption!

Digital signatures

Sets:

- Message space $\mathcal{P}$.
- Signature space $\mathcal{A}$.
- Key space $\mathcal{K}$.

Algorithms:

- Signature algorithm $S_K : \mathcal{P} \rightarrow \mathcal{A}$.
- Verification algorithm $V_K : \mathcal{P} \times \mathcal{A} \rightarrow \{0, 1\}$.
- Consistency:

$$\forall K \in \mathcal{K}, \forall m \in \mathcal{P}, V_K(m, s) = \begin{cases} 1 & \text{if } s = S_K(m) \\ 0 & \text{if } s \neq S_K(m). \end{cases}$$

Secure message authentication

Important: How to formalize security of digital signatures?

We need to define the adversary **strategy**!

Secure message authentication

Important: How to formalize security of digital signatures?

We need to define the adversary **strategy**!

Adversary strategy

Active adversary that intercepts and manipulates messages m and signatures n in transit.

Secure message authentication

Important: How to formalize security of digital signatures?

We need to define the adversary **strategy**!

Adversary strategy

Active adversary that intercepts and manipulates messages m and signatures n in transit.

Intuition

Notion that the adversary should not be capable of **forging** a signature for a message of his/her choice.

Secure message authentication

Important: How to formalize security of digital signatures?

We need to define the adversary **strategy**!

Adversary strategy

Active adversary that intercepts and manipulates messages m and signatures n in transit.

Intuition

Notion that the adversary should not be capable of **forging** a signature for a message of his/her choice.

Problem: Adversary can always **replay** previously captured (m, s).

Secure message authentication

Important: How to formalize security of digital signatures?

We need to define the adversary **strategy**!

Adversary strategy

Active adversary that intercepts and manipulates messages m and signatures n in transit.

Intuition

Notion that the adversary should not be capable of **forging** a signature for a message of his/her choice.

Problem: Adversary can always **replay** previously captured (m, s).

Solution: Prevent replays in the upper layer (application, transport protocol).

Forging attacks

Key-only attack:

- Adversary knows only the public verification key.

Known-message attack:

- Adversary has access to messages and corresponding signatures.

Chosen-message attack:

- Adversary chooses a message to be signed and receives the corresponding signature.

Adversary objectives

Existential forgery:

- Adversary is capable of creating a valid signature for at least one message, without knowing an authentic signature for that message. In other words, create a pair (m, s) such that $v_K(x, y) = 1$.

Selective forgery:

- Adversary is able to create a signature s valid for a message m chosen previously, without knowing an authentic signature for m.

Universal forgery:

- Adversary computes signing key and creates authentic signatures for any message $m \in \mathcal{M}$.

Important: Security under computational assumptions!

RSA signature (Rivest, Shamir, Adleman, 1977)

Key generation:

1. Generate primes p and q with $k/2$ bits.
2. Compute $N = pq$ and $\phi(N) = (p-1)(q-1)$.
3. Select b such that $gcd(b, \phi(N)) = 1$. (small prime?)
4. Compute a such that $a = b^{-1} \bmod \phi(N)$.
5. $\mathcal{M} = \mathcal{S} = \mathbb{Z}_N$.
6. $K = (N, p, q, a, b)$.
7. Public key is (b, N), private key is (a, N, p, q).

Signature: Compute $s = S_K(m) = m^a \bmod n$.

Verification: Compute $V_K(m, s) = 1 \leftrightarrow m \equiv s^b \bmod n$.

RSA signature

Security issues:

- Adversary can choose arbitrary $s \in \mathbb{Z}_N^*$ and obtain a message with valid signature $s^b \bmod N$.
  Important: Adversary does not have complete control over m.

- Adversary can forge a signature s over message $m = m_1 m_2$ if capable of obtaining signatures s_1, s_2 for $m_1 \in \mathbb{Z}_N^*$ and $m_2 = m/m_1 \bmod N$:

$$s^b = (s_1 \cdot s_2)^b = (m_1^a \cdot m_2^a)^b = m_1 \cdot m_2 = m \bmod N.$$

  Important: Adversary must convince signer to sign random-looking messages.

RSA signature

Security issues:

- Adversary can choose arbitrary $s \in \mathbb{Z}_N^*$ and obtain a message with valid signature $s^b \bmod N$.
  Important: Adversary does not have complete control over m.

- Adversary can forge a signature s over message $m = m_1 m_2$ if capable of obtaining signatures s_1, s_2 for $m_1 \in \mathbb{Z}_N^*$ and $m_2 = m/m_1 \bmod N$:

$$s^b = (s_1 \cdot s_2)^b = (m_1^a \cdot m_2^a)^b = m_1 \cdot m_2 = m \bmod N.$$

  Important: Adversary must convince signer to sign random-looking messages.

Solution: Employ a hash function and compute $S_K(H(m))$!

Digital signatures and hash functions

Key-only attack:

- Ineffective if hash function is preimage resistant.

Known-message attack:

- Ineffective if hash function is second preimage resistant.

Chosen-message attack:

- Ineffective if hash function is collision resistant.

Other signature schemes

It is possible to instantiate digital signatures from other security assumptions:

- Discrete logarithm (ElGamal, Schnorr, DSA).
- Discrete logarithm in elliptic curves (ECDSA).
- Cryptographic hash functions alone (Merkle, Lamport).

Digital signatures based on the discrete logarithm are usually probabilistic:

- Many signatures are valid for the same message.
- Verification needs to accept all of these signatures.

ElGamal signature (ElGamal, 1985)

Key generation:

1. Choose prime p such that $p - 1$ has a big factor and primitive element $\alpha \in \mathbb{Z}_p^*$.
2. $\mathcal{M} = \mathbb{Z}_p^*, \mathcal{S} = \mathbb{Z}_p^* \times \mathbb{Z}_{p-1}$.
3. $\mathcal{K} = \{(p, \alpha, a, \beta) : \beta \equiv \alpha^a \pmod{p}$.
4. Public key is $b = \langle p, \alpha, \beta \rangle$, private key is a.
5. Let $H : \{0, 1\}^* \rightarrow \mathbb{Z}_{p-1}$ a cryptographic hash function.

Signature:

1. Choose integer k uniformly at random from $\mathbb{Z}_{p-1}^*$.
2. Compute $S_a(m, k) = (\gamma, \delta)$, where $\gamma = \alpha^k \bmod p$ and $\delta = (H(m) - a\gamma)k^{-1} \bmod (p - 1)$.

Verification:

1. Compute $V_b(m, (\gamma, \delta)) = 1 \leftrightarrow \beta^\gamma \gamma^\delta \equiv \alpha^{H(m)} \pmod{p}$.

Important: Verify consistency!

ElGamal signature (ElGamal, 1985)

If signature was correctly constructed, verification will accept it:

$$
\begin{aligned}
\beta^{\gamma}\gamma^{\delta} &\equiv \alpha^{a\gamma}\alpha^{k\delta} \pmod{p} \\
&\equiv \alpha^{H(m)} \pmod{p},
\end{aligned}
$$

because we have that $a\gamma + k\delta \equiv H(m) \pmod{p-1}$.

ElGamal signature (ElGamal, 1985)

If signature was correctly constructed, verification will accept it:

$$\begin{aligned} \beta^\gamma \gamma^\delta &\equiv \alpha^{a\gamma}\alpha^{k\delta} \pmod{p} \\ &\equiv \alpha^{H(m)} \pmod{p}, \end{aligned}$$

because we have that $a\gamma + k\delta \equiv H(m) \pmod{p-1}$.

Intuition from the verification:

$$\alpha^{H(m)} \equiv \beta^\gamma \gamma^\delta \equiv \alpha^{a\gamma + k\delta} \pmod{p}.$$

Since α is primitive element modulo p, the congruence is valid iff the exponents are congruent modulo $\phi(p) = p - 1$. Solving for δ, We obtain the signature equation:

$$\delta = (H(m) - a\gamma)k^{-1} \pmod{p-1}.$$

ElGamal signatures (ElGamal, 1985)

Security issues:

- If attacker chooses γ, needs to solve $\delta = \log_\gamma(\alpha^{H(m)}\beta^{-\gamma})$.
- If attacker chooses δ, needs to solve $\beta^\gamma\gamma^\delta \equiv \alpha^{H(m)} \pmod{p}$.
- Recovering the private key from public key amounts to computing $a = \log_\alpha \beta$.
- Hash function prevents existential forgery.

Important: The scheme does not have a known security reduction.

ElGamal signatures (ElGamal, 1985)

Protocol failures:

1. Leaking k with $mdc(\gamma, p-1) = 1$ allows to recover the private key $a = (H(m) - k\delta)\gamma^{-1} \pmod{p-1}$.
2. The same k used in two signatures $(m_1, (\gamma, \delta_1))$ e $(m_2, (\gamma, \delta_2))$:

$$\beta^{\gamma}\gamma^{\delta_1} \equiv \alpha^{m_1}, \quad \beta^{\gamma}\gamma^{\delta_2} \equiv \alpha^{m_2} \pmod{p}.$$

   Dividing the two equations above:

$$\begin{aligned} \alpha^{m_1 - m_2} &\equiv \gamma^{\delta_1 - \delta_2} \equiv \alpha^{k(\delta_1 - \delta_2)} \pmod{p} \\ m_1 - m_2 &\equiv k(\delta_1 - \delta_2) \pmod{p-1}. \end{aligned}$$

   Let $d = mdc(\delta_1 - \delta_2, p-1)$, $x' = (m_1 - m_2)/d, \delta' = (\delta_1 - \delta_2)/d$:

$$x' \equiv k\delta' \pmod{(p-1)/d} \rightarrow k = x'\delta'^{-1} \pmod{(p-1)/d}.$$

Correct: One of the two d values of k modulo $(p-1)$ with $\gamma \equiv \alpha^k \pmod{p}$.

Schnorr signature (Schnorr, 1989)

Key generation:

1. Choose primes p, q with $q|(p-1)$ and q much smaller than p.
2. Let $\alpha = \alpha_0^{(p-1)/q} \in \mathbb{Z}_p^*$ the q-th root of 1 modulo p, with α_0 primitive element modulo p.
3. $\mathcal{M} = \{0,1\}^*, \mathcal{S} = \mathbb{Z}_q \times \mathbb{Z}_q$.
4. $\mathcal{K} = \{(p, q, \alpha, a, \beta) : \beta \equiv \alpha^a \pmod{p}$.
5. The public key is $b = \langle p, q, \alpha, \beta \rangle$, the private key is a.
6. Let $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$ a cryptographic hash function.

Signature:

1. Choose integer k uniformly at random from $\mathbb{Z}_q^*$.
2. Compute $S_a(m, k) = (\gamma, \delta)$, where $\gamma = H(m||\alpha^k \bmod p)$ and $\delta = k + a\gamma \bmod q$.

Verification: $V_b(m, (\gamma, \delta)) = 1 \leftrightarrow H(m||\alpha^\delta \beta^{-\gamma} \bmod p) = \gamma$.

Important: Shorter signatures and formal security under ideal H!

Digital Signature Algorithm (NIST, 1991)

Small modification to ElGamal signature:

$$\delta = (H(m) + a\gamma)k^{-1} \pmod{p-1}.$$

The verification equation changes to $\alpha^{H(m)}\beta^{\gamma} \equiv \gamma^{\delta} \pmod{p}$.

Supposing $q|(p-1)$ like in Schnorr, α, β, γ have order q. Reducing exponents modulo q:

$$\delta = (H(m) + a\gamma)k^{-1} \pmod{q} \tag{1}$$

Now define $\gamma' = \gamma \bmod q = (\alpha^k \bmod p) \bmod q$. We can replace γ by γ' in the previous equation and the verification equation changes to $\alpha^{H(m)}\beta^{\gamma'} \equiv \gamma^{\delta} \pmod{p}$.

Multiplying (1) by $\delta' = \delta^{-1} \bmod q, \delta \neq 0$, we obtain:

$$\alpha^{H(m)\delta'}\beta^{\gamma'\delta'} \bmod p = \gamma \rightarrow (\alpha^{H(m)\delta'}\beta^{\gamma'\delta'} \bmod p) \bmod q = \gamma'.$$

Digital Signature Algorithm (NIST, 1991)

Key generation:

1. Choose primes p, q with $q|(p-1)$ and q much smaller than p.
2. Let $\alpha = \alpha_0^{(p-1)/q} \in \mathbb{Z}_p^*$ the q-th root of 1 modulo p, with α_0 primitive element modulo p.
3. $\mathcal{M} = \{0,1\}^*, \mathcal{S} = \mathbb{Z}_q \times \mathbb{Z}_q$.
4. $\mathcal{K} = \{(p, q, \alpha, a, \beta) : \beta \equiv \alpha^a \pmod{p}$.
5. The public key is $b = \langle p, q, \alpha, \beta \rangle$, the private key is a.
6. Let $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$ a cryptographic hash function.

Signature:

1. Choose integer k uniformly at random from $\mathbb{Z}_q^*$.
2. Compute $S_a(m, k) = (\gamma, \delta)$, with $\gamma, \delta \neq 0$, where $\gamma = (\alpha^k \bmod p) \bmod q$ and $\delta = (H(m) + a\gamma)k^{-1} \bmod q$.

1. **Verification:** $V_b(m, (\gamma, \delta)) = 1 \leftrightarrow (\alpha^{H(m)\delta'} \beta^{\gamma\delta'} \bmod p) \bmod q = \gamma$.

Elliptic Curve DSA (NIST, 2000)

Key generation:

1. Choose curve $E(\mathbb{F}_p)$ and let A be a point or prime order q.
2. $\mathcal{M} = \{0,1\}^*, \mathcal{S} = \mathbb{Z}_q^* \times \mathbb{Z}_q^*$.
3. $\mathcal{K} = \{(p, q, E, A, a, B) : B = aA$.
4. Public key is $b = \langle p, q, E, A, B\rangle$, private key is a.
5. Let $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$ a cryptographic hash function.

Signature:

1. Choose integer k uniformly at random from $\mathbb{Z}_q^*$.
2. Compute $S_a(m, k) = (r, s)$, with $r, s \neq 0$, where $kA = (u, v), r = u \bmod q, s = (H(m) + ar)k^{-1} \bmod q$.

Verification: Let $s' = s^{-1} \bmod q$ and $(u, v) = (H(m)s')A + (rs')B$. We have that $V_b(m, (r, s)) = 1 \leftrightarrow u \bmod q = r$.

Hash-based one-time signatures (Lamport, 1979)

Key generation:

1. Let H a cryptographic hash function at the security level n.
2. For $i \in \{1, \ldots, \ell(n)\}$ choose random $y_{i,0}, y_{i,1} \leftarrow \{0,1\}^n$.
3. Compute $x_{i,0} = H(y_{i,0})$ and $x_{i,1} = H(y_{i,1})$.
4. The public key b is composed by the values $x_{i,j}$ and the private key a by the values $y_{i,j}$.

Signature:

1. On message $m = m_1 \cdots m_{\ell(n)} \in \{0,1\}^{\ell(n)}$ and private key a, compute signature $s = (y_{1,m_1}, \ldots, y_{\ell(n),m_{\ell(n)}})$.

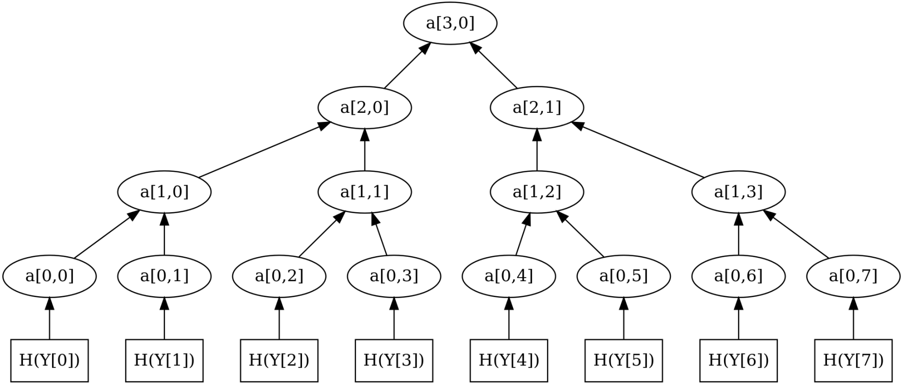
Verification:

1. On message $m = m_1 \cdots m_{\ell(n)} \in \{0,1\}^{\ell(n)}$, signature $s = (s_1 \cdots s_{\ell(n)})$, and public key b, check if $H(s_i) = x_{i,m_i}$.

Important: Security of the signature scheme can be reduced to security of H. Keys can never be repeated.

Merkle hash-based signatures (Merkle, 1979)

Define (x_j, y_j) to be the j-th one-time key pair. Compute inner nodes by applying H recursively. The public key is the root of the tree. A signature can be computed by traversing the tree to select a one-time key pair.



Merkle hash-based signatures (Merkle, 1979)

Verification involves traversing the tree upwards and checking if the last hash matches the public key.