MACs and Signature Schemes

Douglas Wikström
KTH Stockholm
dog@csc.kth.se

February 25

- MACs
- Signature Schemes

Message Authentication Code

- Message Authentication Codes (MACs) are used to ensure integrity and authenticity of messages.

Message Authentication Code

- Message Authentication Codes (MACs) are used to ensure integrity and authenticity of messages.
- Scenario:
  1. Alice and Bob share a common key k.
  2. Alice computes an authentication tag $\alpha = \mathrm{MAC}_k(m)$ and sends (m, α) to Bob.
  3. Bob receives (m, α') from Alice, but before accepting m as coming from Alice, Bob checks that $\mathrm{MAC}_k(m) = \alpha'$.

Security of a MAC

Definition. A message authentication code MAC is secure if for a random key k and every polynomial time algorithm A,

$$\Pr[A^{\mathsf{MAC}_k(\cdot)} = (m, \alpha) \wedge \mathsf{MAC}_k(m) = \alpha \wedge \forall i : m \neq m_i]$$

is negligible, where m_i is the ith query to the oracle $\mathsf{MAC}_k(\cdot)$.

Random Oracle As MAC

- Suppose that $H : \{0,1\}^* \rightarrow \{0,1\}^n$ is a random oracle.
- Then we can construct a MAC as $\mathrm{MAC}_\mathrm{k}(m) = H(\mathrm{k}, m)$.

Could we plug in an iterated hash function in place of the random oracle?

HMAC

- Let $H : \{0,1\}^* \to \{0,1\}^n$ be a "cryptographic hashfunction", e.g., SHA-256.
- $\mathrm{HMAC}_{k_1,k_2}(x) = H\big(k_2 \| H(k_1 \| x)\big)$
- This is provably secure under the assumption that
  - $H(k_1\|\cdot)$ is unknown-key collision resistant, and
  - $H(k_2\|\cdot)$ is a secure MAC.

CBC-MAC

Let E be a secure block-cipher, and $x = (x_1, \ldots, x_t)$ an input. The MAC-key is simply the block-cipher key.

1. $y_0 = 000 \ldots 0$
2. For $i = 1, \ldots, t$, $y_i = \mathrm{E_k}(y_{i-1} \oplus x_i)$
3. Return y_t.

Warning! Do not use without consulting literature.

Universal Hashfunction As MAC

Theorem. A t-universal hashfunction f_α for a randomly chosen secret α is an **unconditionally secure** MAC, provided that the number queries is smaller than t.

Digital Signature

- A digital signature is the **public-key** equivalent of a MAC; the receiver verifies the integrity and authenticity of a message.
- Does a digital signature replace a real handwritten one?

Textbook RSA Signature (1/2)

- Generate RSA keys $((N, e), (p, q, d))$.
- To sign a message $m \in \mathbb{Z}_N$, compute $\sigma = m^d \bmod N$.
- To verify a signature σ of a message m, verify that $\sigma^e = m \bmod N$.

Textbook RSA Signature (2/2)

- Are Textbook RSA Signatures any good?

Textbook RSA Signature (2/2)

- Are Textbook RSA Signatures any good?
- If σ is a signature of m, then σ^2 mod N is a signature of m^2 mod N.

Textbook RSA Signature (2/2)

- Are Textbook RSA Signatures any good?
- If σ is a signature of m, then σ^2 mod N is a signature of m^2 mod N.
- If σ_1 and σ_2 are signatures of m_1 and m_2, then $\sigma_1\sigma_2$ mod N is a signature of $m_1 m_2$ mod N

Textbook RSA Signature (2/2)

- Are Textbook RSA Signatures any good?
- If σ is a signature of m, then $\sigma^2 \bmod N$ is a signature of $m^2 \bmod N$.
- If σ_1 and σ_2 are signatures of m_1 and m_2, then $\sigma_1\sigma_2 \bmod N$ is a signature of $m_1 m_2 \bmod N$
- We can also pick a signature σ and compute the message it is a signature of by $m = \sigma^e \bmod N$.

Textbook RSA Signature (2/2)

- Are Textbook RSA Signatures any good?
- If σ is a signature of m, then $\sigma^2 \bmod N$ is a signature of $m^2 \bmod N$.
- If σ_1 and σ_2 are signatures of m_1 and m_2, then $\sigma_1\sigma_2 \bmod N$ is a signature of $m_1 m_2 \bmod N$
- We can also pick a signature σ and compute the message it is a signature of by $m = \sigma^e \bmod N$.

We must be more careful!

Signature Scheme

- Gen **generates a key pair** $(\mathsf{pk}, \mathsf{sk})$.
- Sign takes a secret key sk and a message m and **computes a signature** σ.
- Verify takes a public key pk, a message m, and a candidate signature σ, **verifies the candidate signature**, and outputs a single-bit verdict.

Existential Unforgeability

Definition. A signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ is **secure against existential forgeries** if for every polynomial time algorithm and a random key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^n)$,

$$\Pr\left[A^{\mathsf{Sign}_{\mathsf{sk}}(\cdot)}(\mathsf{pk}) = (m, \sigma) \wedge \mathsf{Verify}_{\mathsf{pk}}(m, \sigma) = 1 \wedge \forall i : m \neq m_i\right]$$

is negligible where m_i is the ith query to $\mathsf{Sign}_{\mathsf{sk}}(\cdot)$.

Trapdoor One-Way Permutations

Let $f = \{f_\alpha\}$ be an ensemble of **permutations** (bijections).

- Gen **generates a random key pair** $\alpha = (\mathsf{pk}, \mathsf{sk})$.
- Eval takes pk and x as input and **efficiently evaluates** $f_\alpha(x)$.
- Invert takes sk and y as input and **efficiently evaluates the inverse** $f_\alpha^{-1}(y)$.

One-way if $\mathsf{Eval}_{\mathsf{pk}}(\cdot)$ is one-way for a random pk.

Trapdoor One-Way Permutations

Let $f = \{f_\alpha\}$ be an ensemble of **permutations** (bijections).

- Gen **generates a random key pair** $\alpha = (\mathsf{pk}, \mathsf{sk})$.
- Eval takes pk and x as input and **efficiently evaluates** $f_\alpha(x)$.
- Invert takes sk and y as input and **efficiently evaluates the inverse** $f_\alpha^{-1}(y)$.

One-way if $\mathsf{Eval}_{\mathsf{pk}}(\cdot)$ is one-way for a random pk.

RSA is a trap-door permutation over $\mathbb{Z}_N^*$.

Trapdoor One-Way Permutations (Less Formal)

Let $f = \{f_\alpha\}$ be an ensemble of **permutations** (bijections).

- Gen **generates a pair** $(f_\alpha, f_\alpha^{-1})$.

One-way if f_α is one-way when chosen randomly.

RSA is a trap-door permutation over $\mathbb{Z}_N^*$.

Full Domain Hash In ROM

Let $f = \{f_\alpha\}$ be a trapdoor permutation (family) and let $H : \{0,1\}^* \to \{0,1\}^n$ be a random oracle.

- Gen samples a pair $(f_\alpha, f_\alpha^{-1})$.
- Sign takes f_α^{-1} and a message m as input and outputs $f_\alpha^{-1}\big(H(m)\big)$.
- Verify takes f_α, a message m, and a candidate signature σ as input, and outputs 1 if $f_\alpha(\sigma) = H(m)$ and 0 otherwise.

Proof of Knowledge of Exponent

In an **identification scheme** one party convinces another that it holds some special token.

Proof of Knowledge of Exponent

In an **identification scheme** one party convinces another that it holds some special token.

- Let G_q be a group of prime order q with generator g.

Proof of Knowledge of Exponent

In an **identification scheme** one party convinces another that it holds some special token.

- Let G_q be a group of prime order q with generator g.
- Let $x \in \mathbb{Z}_q$ and define $y = g^x$.

Proof of Knowledge of Exponent

In an **identification scheme** one party convinces another that it holds some special token.

- Let G_q be a group of prime order q with generator g.
- Let $x \in \mathbb{Z}_q$ and define $y = g^x$.
- Can we prove knowledge of x without disclosing anything about x?

Schnorr's Signature Scheme (1/3)

1. The prover chooses $r \in \mathbb{Z}_q$ randomly and hands $\alpha = g^r$ to the verifier.

Schnorr's Signature Scheme (1/3)

1. The prover chooses $r \in \mathbb{Z}_q$ randomly and hands $\alpha = g^r$ to the verifier.

2. The verifier chooses $c \in \mathbb{Z}_q$ randomly and hands it to the prover.

Schnorr's Signature Scheme (1/3)

1. The prover chooses $r \in \mathbb{Z}_q$ randomly and hands $\alpha = g^r$ to the verifier.

2. The verifier chooses $c \in \mathbb{Z}_q$ randomly and hands it to the prover.

3. The prover computes $d = cx + r \bmod q$ and hands d to the verifier.

Schnorr's Signature Scheme (1/3)

1. The prover chooses $r \in \mathbb{Z}_q$ randomly and hands $\alpha = g^r$ to the verifier.
2. The verifier chooses $c \in \mathbb{Z}_q$ randomly and hands it to the prover.
3. The prover computes $d = cx + r \bmod q$ and hands d to the verifier.
4. The verifier accepts if $y^c \alpha = g^d$.

Schnorr's Signature Scheme (1/3)

1. The prover chooses $r \in \mathbb{Z}_q$ randomly and hands $\alpha = g^r$ to the verifier.
2. The verifier chooses $c \in \mathbb{Z}_q$ randomly and hands it to the prover.
3. The prover computes $d = cx + r \bmod q$ and hands d to the verifier.
4. The verifier accepts if $y^c \alpha = g^d$.

Suppose that a machine convinces us in the protocol with probability δ. Does it mean that it knows x such that $y = g^x$?

Schnorr's Signature Scheme (2/3)

Schnorr's Signature Scheme (2/3)

1. Run the machine to get α.

Schnorr's Signature Scheme (2/3)

1. Run the machine to get α.

2. Complete the interaction twice using **the same** α, once for a challenge c and once for a challenge c', where $c, c' \in \mathbb{Z}_q$ are chosen randomly.

Schnorr's Signature Scheme (2/3)

1. Run the machine to get α.

2. Complete the interaction twice using **the same** α, once for a challenge c and once for a challenge c', where $c, c' \in \mathbb{Z}_q$ are chosen randomly.

3. Repeat from (1) until the resulting interactions (α, c, d) and (α, c', d') are accepting and $c \neq c'$.

Schnorr's Signature Scheme (2/3)

1. Run the machine to get α.

2. Complete the interaction twice using **the same** α, once for a challenge c and once for a challenge c', where $c, c' \in \mathbb{Z}_q$ are chosen randomly.

3. Repeat from (1) until the resulting interactions (α, c, d) and (α, c', d') are accepting and $c \neq c'$.

4. Note that:

$$y^{c-c'} = \frac{y^c}{y^{c'}} = \frac{y^c \alpha}{y^{c'} \alpha} = \frac{g^d}{g^{d'}} = g^{d-d'}$$

which gives the logarithm $x = (d - d')(c - c')^{-1} \bmod q$ such that $y = g^x$.

Schnorr's Signature Scheme (3/3)

- Anybody can sample $c, d \in \mathbb{Z}_q$ randomly and compute $\alpha = g^d / y^c$.
- The resulting tuple (α, c, d) has **exactly** the same distribution as the transcript of an interaction!

Such protocols are called (honest verifier) **zero-knowledge**.

Schnorr's Signature Scheme In ROM

Let $H : \{0,1\}^* \to \mathbb{Z}_q$ be a random oracle.

- Gen chooses $x \in \mathbb{Z}_q$ randomly, computes $y = g^x$ and outputs $(\mathsf{pk}, \mathsf{sk}) = (y, x)$.
- Sign does the following on input x and m:
  1. it chooses $r \in \mathbb{Z}_q$ randomly and computes $\alpha = g^r$,
  2. it computes $c = H(y, \alpha, m)$,
  3. it computes $d = cx + r \bmod q$ and outputs (α, d).
- Verify takes the public key y, a message m, and a candidate signature (α, d), and accepts iff $y^{H(y,\alpha,m)}\alpha = g^d$.

Provably Secure Signature Schemes

Provably secure signature schemes exists if one-way functions exist (in plain model without ROM), but the construction is more involved and typically less efficient.

Provably secure signature schemes are rarely used in practice!

Standards used in practice: RSA Full Domain Hash, DSA, EC-DSA. The latter two may be viewed as variants of Schnorr signatures.