Outline

Digital Signatures

More on random oracles
Digital Signatures
Digital Signatures

Public-key version of “MACs”

\[ \text{Gen}(1^k; r) \rightarrow (\text{vk}, \text{sign}_k) \]

\[ \text{Sign}(m, \text{sk}; r) \rightarrow \sigma \]

\[ \text{Verify}(m, \sigma, \text{vk}) \rightarrow 0/1 \]

Correctness: \( \forall m, \text{Verify}(m, \text{Sign}(m, \text{sk}; r), \text{vk}) = 1 \)
One-Time Signatures

- Definition of security

For a sig. scheme \((\text{Gen}, S, V)\) and adv. \(A\), define a game as:

\[
\begin{align*}
(pk, sk) & \leftarrow \text{Gen} \\
\sigma_1 & \leftarrow S(sk, m_1) \\
\end{align*}
\]

Adv. wins if \(V(pk, m, \sigma) = \text{`accept'}\) and \(m \neq m_1\)
Lamport One-Time Signatures
(from OWF)

- \( \text{vk} = \) \\

- \( \text{sk} = \)

\[ \text{Gen}(1^k; r) = (\ldots) \]

To sign \( m = m_1, m_2, \ldots, m_n \), the signer sends the openings of a few boxes.
Lamport One-Time Signatures

- $vk =$

- One-time security (definition):

$$Adv$$ is given $(m, c)$ for $m$, of his choice must not be able to output $(m, c)$ for $m \neq m'$. 

$m[j] \neq m'[j]$ : $Adv$ doesn't know how to open $j$th box from $m$. 
Lamport One-Time Signatures

\[ v_k = \begin{pmatrix} y_{10} & y_{20} & y_{30} & \square & \square \\ y_{11} & y_{21} & y_{31} & \square & \square \end{pmatrix} \]

- One-time security (proof):

\[ v_k = \sum y_{i0}, y_{i1} \in \mathbb{C}[n] \]

\[ \text{Adv} (v_k) \rightarrow m, \text{ gets } r_i \text{ on } m_i \]

\[ r_i \rightarrow \sum r_i(m_i) \in \mathbb{C}[n] \]

Must output \( \sigma = \sum r_i(m_i) \in \mathbb{C}[n] \) s.t. \( m_i \neq m_i') \text{ unless } \text{Adv} \text{ cannot find } r_i \text{ for } y_{i0} \]
Lamport One-Time Signatures

- What about two-time security?

Given \( m \) \( \in \{0,1\}^n \)

If Adv finds \( (m, \{ r_i \}_{i=1}^n) \)

Adversary given \( y_{10}, r_{10} \)

Can it find \( r'_1 \) s.t. \( f(r'_1) y_{10} \)

One-way permutation
Many-Time Signatures

For a sig. scheme \((\text{Gen}, S, V)\) and adv. A define a game as:

\[
\begin{align*}
(\text{pk}, \text{sk}) & \leftarrow \text{Gen} \\
(\text{pk}) & \leftarrow \text{Adv} \\
(\sigma_1) & \leftarrow S(\text{sk}, m_1) \\
(\sigma_2, \ldots, \sigma_q) & \leftarrow \text{Adv} \\
(\text{pk}, \text{sk}) & \leftarrow \text{Adversary}
\end{align*}
\]

Adv. wins if \(V(\text{pk}, m, \sigma) = \text{`accept'}\) and \(m \notin \{m_1, \ldots, m_q\}\)
Applications?

- Authenticating software updates

Server

\[ \text{Initial software, } \mathbf{vk} \]

update\(_1\), sign\(_1\)

update\(_2\), sign\(_2\)

\[
\text{Accepts update if } \text{Verify}(\text{update}_i, \text{sign}_i, \mathbf{vk}) = 1
\]
One-time => Many-time authenticated channel
Signing email: DKIM (domain key identified mail)

Problem: bad email claiming to be from someuser@gmail.com but in reality, mail is coming from domain baguy.com
⇒ Incorrectly makes gmail.com look like a bad source of email

Solution: gmail.com (and other sites) sign every outgoing mail
Important application: Certificates

Problem: browser needs server’s public-key to setup a session key

Solution: server asks trusted 3rd party (CA) to sign its public-key pk

Server uses Cert for an extended period (e.g. one year)
MACs versus signatures

- Use MAC when
  1. signer & verifier are same
  2. single signer & single verifier

- Use signature when
  * many verifiers for the same signer
  * "public" verification
Can you use MACs here?

- Authenticating software updates

No.
Signatures from Trapdoor Permutations

**Trapdoor Permutations**

Three algorithms: \((G, F, F^{-1})\)

- **G**: outputs \(\langle pk, sk \rangle\) \(pk\) defines a function \(F(pk, \cdot): X \rightarrow X\)

- **F\((pk, x)\)**: evaluates the function at \(x = y\)

- **\(F^{-1}(sk, y)\)**: inverts the function at \(y\) using \(sk = x\)

**Secure** trapdoor permutation:

The function \(F(pk, \cdot)\) is one-way without the trapdoor \(sk\)
Signatures from Trapdoor Permutations

• First stab:
  • Gen outputs (vk = pk, sign key = sk)

  • Sign (m, sign key) = $F^{-1}(sk, m) = \sigma$

  • Verify ($\sigma$, vk, m): Check if $F(pk, \sigma)^{\frac{1}{2}} = m$.

• Is this secure?

Zero-time setting: \text{ Adv } : (pk)

$F(pk, x) \xrightarrow{m \times r \text{ of its choice}} f(pk, x) = y \xrightarrow{\text{ Set } (m^2, y, \frac{1}{\sqrt{2}} x)}$
Recall:

For a sig. scheme \((\text{Gen}, S, V)\) and adv. A define a game as:

\[(pk, sk) \leftarrow \text{Gen} \rightarrow \text{Adv.} \]

\[m_1 \in M \quad m_2, \ldots, m_q \]

\[\sigma_1 \leftarrow S(sk, m_1) \quad \sigma_2, \ldots, \sigma_q \]

Adv. wins if \(V(pk, m, \sigma) = \text{`accept'}\) and \(m \notin \{m_1, \ldots, m_q\}\)
So how should we modify it?

\[ S(sk, \text{msg}) : \]

\[ \text{msg} \]

\[ H \]

\[ F^{-1}(sk, \cdot) \]

\[ \text{sig} \]
So how should we modify it?

\[ S(sk, msg): \]

\[ \text{msg} \]

\[ H \]

\[ F^{-1}(sk, \cdot) \]

\[ \text{sig} \]

\[ \text{Sign}(sk, m) = F^{-1}(sk, H(m)) \]

\[ o = F^{-1}(sk, H(m)) \]

\[ V(pk, msg, sig): \]

\[ \text{msg} \]

\[ H \]

\[ \text{sig} \]

\[ F(pk, \cdot) \]

\[ \text{Verify}(m, o, pk) = 1 \text{ if } H(m) = F(pk, o) \]

\[ 0 \text{ otherwise} \]
Full-Domain Hash

- Gen outputs (vk = pk, sign key = sk)

- Sign (m, sign key) = F^{-1} (sk, H(m)) = \sigma

- Verify (\sigma, vk, m) : Check if F (pk, \sigma) = H(m)

- Is this secure? Yes, assuming H is a random oracle
Many-time Signature Security of Full-Domain Hash: Proof of Security

\[ \text{Sign}(sk, m) = F^{-1}(sk, H(m)) \]

\[ \text{Verify}(\sigma, m, pk) = F(pk, \sigma) \]

Adv. wins if \( V(pk, m, \sigma) = \text{`accept'} \) and \( m \notin \{m_1, \ldots, m_q\} \)

Chall.

\[ \sigma_1 = F^{-1}(sk, H(m_1)) \]

Adv

\[ \sigma \]

\[ (m, \sigma) \]
[Step 1: Generate \( \sigma_1 \) without access to sk at all)

\[ \sigma_1 = x_1 \]

Step 2 : Generate \((\sigma_1, \sigma_2)\) without access to sk...
Step n : Generate \((\sigma_1, \ldots, \sigma_n)\) without access to sk...
(m, σ) s.t.
H(m) = F(pk, σ).

Case 1: Adv never queried R.O. on m!
⇒ Adv gave us σ s.t. F(pk, σ) = H(m)
⇒ contradicts "programmability"

⇒ Build wrapper B around Adv
that outputs (m, F(pk, σ)) = (m, H(m)) without querying R.O. on m!
Case 2: Adv queried R.O. on $m$.

Then outputs $(m, \sigma)$ s.t. $H(m) = F(pk, \sigma) = y$.

Attacker outputs $\sigma = F^{-1}(sk, y)$.

Take Adv that breaks MT Sign Sec. game.

Build wrapper around Adv, that breaks trapdoor permutation.
Signature lengths

Goal: best existential forgery attack time $\geq 2^{128}$

<table>
<thead>
<tr>
<th>algorithm</th>
<th>signature size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSA</td>
<td>2048-3072 bits</td>
</tr>
<tr>
<td>EC-DSA</td>
<td>512 bits</td>
</tr>
<tr>
<td>Schnorr</td>
<td>384 bits</td>
</tr>
<tr>
<td>BLS</td>
<td>256 bits</td>
</tr>
</tbody>
</table>

Open problem: practical 128-bit signatures

- Domain Extension
- Aggregatable Signatures