Digital Signatures
Digital Signatures

- Digital signature is the same as MAC except that the tag (signature) is produced using the secret key of a public-key cryptosystem.

<table>
<thead>
<tr>
<th>Message m</th>
<th>MAC&lt;sub&gt;k&lt;/sub&gt;(m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message m</td>
<td>Sign&lt;sub&gt;sk&lt;/sub&gt;(m)</td>
</tr>
</tbody>
</table>
• Digital signature:
  1. Bob has a key pair \((sk, pk)\).
  2. Bob sends \(m \| \text{Sign}_{sk}(m)\) to Alice.
  3. Alice verifies the received \(m' \| s'\) by checking if \(\text{Vrfy}_{pk}(m', s') = 1\)?

• \(\text{Sign}_{sk}(m)\) is called a signature for \(m\).

• Security requirement: infeasible to produce a valid pair \(\langle m, \text{Sign}_{sk}(m) \rangle\) without knowing \(sk\).
Encryption (using RSA):

\[ m \rightarrow E \rightarrow c \rightarrow D \rightarrow m \]

Alice \hspace{2cm} PK_{Bob} \hspace{2cm} SK_{Bob} \hspace{2cm} Bob

Signing (using RSA^{-1}):

E(s) = m? \leftarrow \ E \leftarrow \ s \leftarrow \ D \leftarrow m

Verify the signature \hspace{2cm} Sign

Alice \hspace{2cm} PK_{Bob} \hspace{2cm} SK_{Bob} \hspace{2cm} Bob
Basic RSA Signature

- **Keys** are generated as for RSA encryption:
  
  Public key: $pk = (N, e)$. Secret key: $sk = (N, d)$.

- **Signing** a message $m \in \mathbb{Z}_N^*$:
  
  $$\sigma = \text{Sign}_sk(m) = m^d \mod N.$$  

  That is, $\sigma = \text{RSA}^{-1}(m)$.

- **Verifying** a signature $(m, \sigma)$:
  
  $$\text{Vrfy}_{pk}(m, \sigma) = 1 \text{ if and only if } m = \sigma^e \mod N$$  

  or $m = \text{RSA}(\sigma)$. 
Correctness:

$$\text{RSA}_{pk} \left( \text{RSA}_{sk}^{-1}(m) \right) = m.$$ 

A message $m$ signed with $sk$ will be verified and accepted with the corresponding $pk$. 
Remarks:

- Basic RSA signature is the reverse of basic RSA encryption.
- Because of this, digital signatures are often mistakenly viewed as the reverse of public-key encryption.
- As will be seen, secure RSA signature is not the reverse of secure RSA encryption. Neither is ElGamal signature.
• **Existentially forgeable:**
  
  1. Every message $m$ is a valid signature of its ciphertext $c$, since $RSA^{-1}(c) = m$.
  
  2. If Bob signed $m_1$ and $m_2$, then the signature for $m_1m_2$ can be easily forged: $\sigma(m_1m_2) = \sigma(m_1)\sigma(m_2)$.

• **Remedy: hash then sign:**

\[
\sigma = \text{Sign}_{sk}(H(m)) = RSA_{sk}^{-1}(H(m)),
\]
using some hash function $H$. 

Question:
Does hash-then-sign make RSA signature secure against chosen-message attacks?

Answer:
Yes, if $H$ is a full-domain random oracle, i.e.,
- $H$ is a random oracle mapping $\{0,1\}^* \rightarrow \mathbb{Z}_N$
- $(\mathbb{Z}_N$ is the full domain of RSA $^{-1}$)

Theorem: Full-domain hash RSA signature is secure against any chosen-message attack under the random oracle model.
• **Problem with full-domain hash:**
  In practice, $H$ is **not** full-domain.
  For instance, the range of SHA-1 is $\{0,1\}^{160}$, while $\mathbb{Z}_N = \{0,1,...,N-1\} \approx \{0,1\}^{1024}$, if $\|N\| = 1024$.

• **Desired:** a secure signature scheme that does not require a full-domain hash.
Probabilistic signature scheme

- Hash function $H : \{0, 1\}^* \rightarrow \{0, 1\}^l \subset \mathbb{Z}_N$ (not full domain).

  $l \ll n = \|N\|$. (E.g., SHA-1, $l = 160$; RSA, $n = 1024$.)

- Idea: $m \xrightarrow{\text{pad}} m \| r$ $\in \{0, 1\}^*$

  $\xrightarrow{\text{hash}} x = H(m \| r) \in \{0, 1\}^l$

  $\xrightarrow{\text{expand}} y = x \| (r \| 0^{n-1-l-k}) \oplus G(x) \in \{0, 1\}^{n-1}$

  $\xrightarrow{\text{sign}} \sigma = \text{RSA}^{-1}(y) \in \mathbb{Z}_N$

where $r \in \{0, 1\}^k$ for some $k$

$G : \{0, 1\}^l \rightarrow \{0, 1\}^{n-1-l}$ (pseudorandom generator)
• **Signing** a message $m \in \{0,1\}^*$:

1. choose a random $r \in \{0,1\}^k$; compute $x = H(m \| r)$;
2. compute $y = x \| r \oplus G_1(x) \| G_2(x)$; \hspace{1cm} // $G = G_1 \| G_2$ //
3. The signature is $\sigma = RSA^{-1}(y)$.

• **Verifying** a signature $(m, \sigma)$: compute $RSA(\sigma) = x \| t \| u$; check if $u = G_2(x)$ and $x = H(m \| t \oplus G_1(x))$. 
Remarks

- PSS is secure against chosen-message attacks in the random oracle model (i.e., if $H$ and $G$ are random oracles).
- PSS is adopted in PKCS #1 v.2.1.
- Hash functions such as SHA-1 are used for $H$ and $G$.
- For instance,
  
  let $n = 1024$, and $l = k = 160$
  
  let $H = $SHA-1
  
  $G(x) = H(x \parallel 0) \parallel H(x \parallel 1) \parallel H(x \parallel 2), \ldots$
DLP-based Digital Signatures
Ideas behind DLP-based signature

- \( G = \{g^0, g^1, g^2, ..., g^x, ..., g^{q-1}\} \), a cyclic group of order \( q \).
- \( \mathbb{Z}_q = \{0, 1, 2, ..., x, ..., q-1\} \).
- \( sk = (G, g, q, x) \), \( pk = (G, g, q, h) \) where \( h = g^x \).

- To sign a message \( m \), Alice needs to show that she knows the secret key \( x \). Besides, non-deterministic signature is desired. So, the signature should be a function of \((m, x, k)\), where \( k \) is random.
- We have \( x \in \mathbb{Z}_q \), suggesting that \( m, k \in \mathbb{Z}_q \).
- So, let the signature \( s \) be a function of \((m, x, k)\) whose validity can be verified using \( g^m, g^x, g^k \).
- The signer needs to send \( r := g^k \) along with \( s \).
The signature $s$ is a function of $(m, x, k)$.

- $s = km + k'x \mod q$ where $k, k' \leftarrow_u \mathbb{Z}_q$
- $s = km + F(r)x \mod q$ where $r = g^k$, $F: G \rightarrow \mathbb{Z}_q$
- $s = (m - F(r))k^{-1} \mod q$ // $m = ks + F(r)x \mod q$ //

ElGamal signature: $(r, s) \in G \times \mathbb{Z}_q$, $s = (m - F(r)x)k^{-1} \mod q$

To verify a signature $(r, s)$, the verifier checks if $g^m = r^s h^{F(r)}$ without knowing $x$ and $k$:

\[
\begin{align*}
    s &= (m - F(r)x)k^{-1} \mod q \\
    \iff m &= ks + F(r)x \mod q \\
    \iff g^m &= g^{ks} g^{F(r)x} \quad \text{in } G \quad \text{//} \\
    \iff g^m &= r^s h^{F(r)}
\end{align*}
\]
• Shnorr signature: \( (F(r), s) \in \mathbb{Z}_q \times \mathbb{Z}_q \)

\[
s = (m + F(r))k^{-1} \mod q
\]

• To verify a signature \( (F(r), s) \),

the verifier checks if \( F(r) = F\left(\left( g^m h^{F(r)} \right)^t \right) \):

\[
s = (m + F(r)x)k^{-1} \mod q
\]

\[
\Rightarrow k = (m + F(r)x)t \mod q \quad // t = s^{-1} \mod q //
\]

\[
\Rightarrow g^k = \left( g^m g^{F(r)x} \right)^t \quad // \text{in } G //
\]

\[
\Rightarrow r = \left( g^m h^{F(r)} \right)^t \quad // \text{in } G //
\]

\[
\Rightarrow F(r) = F\left(\left( g^m h^{F(r)} \right)^t \right) \quad // \text{in } \mathbb{Z}_q //
\]
ElGamal signature in $\mathbb{Z}_p^*$

1. Key generation: same as in ElGamal encryption.
   - a large prime $p$ and a generator $g \in \mathbb{Z}_p^*$.
   - a randomly chosen number $x \in \mathbb{Z}_{p-1}$ and $h = g^x \mod p$;
   - $sk = (p, g, x)$ and $pk = (p, g, h)$.

2. To sign a message $m \in \mathbb{Z}_{p-1}$,
   - randomly choose $k \in \mathbb{Z}_{p-1}^*$;
   - compute $r = g^k \mod p$ and $s = (m - rx)k^{-1} \mod (p - 1)$;
   - the signature is $\text{Sign}_{sk}(m) = (r, s) \in \mathbb{Z}_p^* \times \mathbb{Z}_{p-1}$.

3. Verification: $\text{Vrfy}_{pk}(m, r, s) = \text{true}$ if and only if
   $$(m, r, s) \in \mathbb{Z}_{p-1} \times \mathbb{Z}_p^* \times \mathbb{Z}_{p-1}$$ and $g^m \equiv r^s h^r \mod p$. 
Security of ElGamal signature

- Based on the assumed intractability of discrete logarithm.
- Should use a new $k$ for each signing, or the adversary can compute $k$ from two signatures
  
  $s = (m - rx)k^{-1}$ and $s' = (m' - rx)k^{-1}$

  $\Rightarrow s - s' \equiv (m - m')k^{-1} \mod(p - 1)$

  $\Rightarrow k = (m - m')(s - s')^{-1} \mod(p - 1)$

- Knowing $k$, the adversary can compute $x$ with high probability:
  
  $s = (m - rx)k^{-1} \mod(p - 1)$

  $\Rightarrow x = (m - sk)r^{-1} \mod(p - 1)$, if $r^{-1} \mod(p - 1)$ exists.
Security of ElGamal signature (cont'd)

• Existential forgery. Construct a message $m$ and a valid signature $(r, s)$ as follows.
  a) choose $k, c \in \mathbb{Z}_{p-1}^*$.
  b) set $r = g^k h^c \mod p, s = -rc^{-1} \mod (p - 1)$, and
     $m = -rkc^{-1} \mod (p - 1)$.

• Countermeasure: hash then sign.
Digital Signature Algorithm (DSA) - an NIST standard

0. Shnorr's idea: working in a subgroup of $\mathbb{Z}_p^*$ of prime order $q \ll p$ will shorten the signature, desired for Smart Card applications.

- ElGamal signature scheme uses:

$$\mathbb{Z}_p^* = \{ \alpha^0, \alpha^1, \alpha^2, \ldots, \alpha^{p-2} \} \quad \mathbb{Z}_{p-1} = \{0, 1, 2, \ldots, p-2\}.$$ 

A signature is $(r, s) \in \mathbb{Z}_p^* \times \mathbb{Z}_{p-1} \approx \mathbb{Z}_{p-1} \times \mathbb{Z}_{p-1}$.

- DSA uses:

$\langle g \rangle = \{ g^0, g^1, \ldots, g^{q-1} \} \subset \mathbb{Z}_p^*$, where $g = \alpha^b, bq = p-1$.

A signature is $(\hat{r}, s) \in Z_q \times Z_q$.
1. Key generation

- choose two primes $p$ and $q$ such that $q \mid (p - 1)$.
  $(q \ll p$, e.g., $\|q\| = 160, \|p\| = 1024.)$
- let $g \in \mathbb{Z}_p^*$ be an element of order $q$.
- randomly choose $0 \neq x \in \mathbb{Z}_q$ and compute $h = g^x \mod p$;
- system parameters: $(p, q, g)$
- $sk = (x)$ and $pk = (h)$.

(Remark: The DLP will be working in $\langle g \rangle$.)
2. **Signing**: to sign a message $m$,

- randomly choose $k \in Z_q^*$; compute $\hat{r} = \left( g^k \mod p \right) \mod q$.
- compute $s = (H(m) + \hat{r}x)k^{-1} \mod q$.
// choose a different $k$ if $\hat{r} = 0$ or $s = 0$ //</br>
- $(m, \hat{r}, s)$ is the signed message.

3. **Verification**: accept $(m, \hat{r}, s)$ iff $\hat{r}, s \in Z_q^*$ and

$$\hat{r} = \left( (g^{H(m) \hat{r}})^t \mod p \right) \mod q$$
where $t = s^{-1} \mod q$. 
DSA Correctness: if the message is signed correctly, the signature will be verified/accepted.

\[ s = (H(m) + \hat{r}x)k^{-1} \mod q \]

\[ \Rightarrow k \equiv (H(m) + \hat{r}x)t \mod q \quad (t = s^{-1} \mod q) \]

\[ \Rightarrow g^k \equiv \left( g^{H(m)}g^{\hat{r}x} \right)^t \mod p \]

\[ \Rightarrow g^k \equiv \left( g^{H(m)}h^{\hat{r}} \right)^t \mod p \]

\[ \Rightarrow g^k \mod p = \left( g^{H(m)}h^{\hat{r}} \right)^t \mod p \]

\[ \Rightarrow (g^k \mod p) \mod q = \left( (g^{H(m)}h^{\hat{r}})^t \mod p \right) \mod q \]
Remarks on DSA

- SHA-1 is suggested for use as the hash function.
- \(\|q\| = 160\) bits, \(\|p\| = 512 + 64t, \ 0 \leq t \leq 8\).
- Because a hash is used, there is no restriction on \(m\).
- Why use \(\langle g \rangle\) instead of \(\mathbb{Z}_p^*\)?
- \(\langle g \rangle\) is a subgroup of \(\mathbb{Z}_p^*\) of order \(q\).
- Shorter message length: \(|(\hat{r}, s)| = 2\|q\|\) bits, rather than \(2\|p\|\).
- Why not work in \(\mathbb{Z}_q^*\)? The message length would be \(2\|q\|\), too.
- Reason: The Index Calculus method works for \(\mathbb{Z}_q^*\) but not for \(\langle g \rangle\).