Adams

Quiz (in-class) 4/4 (One page of notes)

Projects

Today:

Digital Signatures!

- Diffie-Hellman notion of PK signatures
- ACMA (Adaptive Chosen Message Attack) security defn.
- "Textbook" RSA signatures
- Hash & Sign (Full Domain Hash)
- Schnorr ID scheme
- Fiat-Shamir paradigm
- Schnorr signatures
- NIST DSA (Digital Signature Algorithm)
Diffie & Hellman ("New Directions in Cryptography")

- $\text{Gen}(1^k) \rightarrow (PK, SK, M, C)$
  - Ciphertext space
  - Message space
  $|M| = |C|$

- $\text{Enc}(PK, \cdot)$ maps $M$ to $C$ \hspace{2cm} 1 to 1 \hspace{2cm} \text{efficiently computable}$

- $\text{Dec}(SK, \cdot)$ maps $C$ to $M$ \hspace{2cm} 1 to 1

\[ m \xrightarrow{\text{Enc}} C \xrightarrow{\text{Dec}} m \]

$\text{Enc} \& \text{Dec}$ are inverse functions (given $PK$ \& $SK$)

- For signatures, we rename $\text{Enc}$ as $\text{Verify}$, $\text{Dec}$ as $\text{Sign}$

\[ \sigma = \text{Sign}(SK, m) \quad \sigma = \text{Signature on } m \text{ by } SK \]

Verify by checking if $\text{Verify}(PK, \sigma) = m$?

Correctness: $\text{Verify}(PK, \sigma) = m$

\[ \text{if } \sigma = \text{Sign}(SK, m) \]

(Security defined in a bit...) \hspace{2cm} SK is trapdoor

Enc \& Dec are trapdoor permutations

(SK is trapdoor)
SK is "signing key"

PK is "signature verification key"

\[ \sigma = \text{Sign}(SK, m) \]

\[ \text{Verify}(PK, m, \sigma) \in \{\text{true, false}\} \]

(Note: have pulled \( m \) inside as arg to \text{Verify})

**Security:**

Signature scheme \((\text{Gen}, \text{Sign}, \text{Verify})\) is secure against adaptive chosen message attack

if \((A, \text{PPT} A)\) (Adversary)

\[
\Pr \left[ A^{\text{Sign}(SK,m)}(PK) = (m^*, \sigma^*) \text{ such that } \right.

\[ \text{Verify}(PK, m^*, \sigma^*) = \text{true} \]

\[ \text{and } m^* \text{ was not ever given to } \text{Sign} \]

\[
\leq \text{negl}(\lambda)
\]

i.e. Adversary cannot forge a new message/signature pair, even after having seen signatures for polynomially many messages of his choice.
Textbook RSA signatures:

1. Gen \( (1^n) \rightarrow (PK, SK, M, C) \)
   
   \( PK = (n, e) \)
   
   \( SK = (n, d) \)
   
   \( d = e^{-1} \pmod{\phi(n)} \)

2. Sign \( (SK, m) \rightarrow \sigma = m^d \pmod{n} \)

3. Verify \( (PK, m, \sigma) = \) true if \( \sigma^e = m \pmod{n} \)

This is just basic RSA encryption "turned around"!

This is not secure against ACMA:

- If \( \sigma^e \) is signature for \( m \)
  
  then \( \sigma^\text{-2} \) is signature for \( m^2 \)

- Worse: \( \sigma \) is signature for \( m = \sigma^e \pmod{n} \)

How to fix?
Hash & Sign (aka Full Domain Hash)

Assume \( H \) is a hash function mapping messages (of arbitrary length) to \( \mathbb{Z}_n \) where \( H \) is modeled as "random oracle".

Idea: Sign \( H(m) \) rather than signing \( m \).

Note: This provides efficiency gains for long messages, as \( H \) is fast.

Claim: This scheme is now secure against ACMA in ROM, under RSA Assumption (hard to compute \( x^d \), given \( n, e, x \mod n \)).

Note: \( \text{Sign}(sk,m) = H(m)^d \mod n \)

\( \text{Verify}(pk,m,\sigma) = \text{true \ if \ } \sigma^e = H(m) \mod n. \)

Proof of claim (sketch):

- Without signing oracle, hard to compute any valid signature, since this requires breaking RSA assumption.

- With signing oracle: Adv can compute transcript of requests to Sign himself, so he learns nothing from Sign. Idea: "program" \( H \). Given \( m \), choose \( \sigma \in \mathbb{Z}_n^* \) compute \( r = \sigma^e \mod n \), program \( H(m) = r \), output \( \sigma \) as signature for \( m \).
If Adv asks for $H(m)$, where m previously unseen, choose random $s \in \mathbb{Z}_n^*$, set $r = s^e \pmod{n}$ return $H(m) = r$.

Schnorr Signature Scheme
- Based on Schnorr Identification Scheme
- Fiat-Shamir paradigm
- Basis for NIST Digital Signature Standard.

Schnorr Identification Scheme
- Prove knowledge of $x$, for $PK = g^x$
- Group $G$ has prime order
  - $g$ is a generator of $G$
  - E.g. work $\mod p$, where $p = g \cdot r + 1$ and $q_p$ is prime
  - $G = \{ h^r \mod p, h \in \mathbb{Z}_p^* \}$
  - $|G| = q_p$
  - To find $g$ generator of $G$, choose $h \in \mathbb{Z}_p^*$,
    - $h \neq 1 \mod p$; let $g = h^r \pmod{p}$.
  - Typically $p$ has 1024 bits (to defeat DL attacks)
  - $q_p$ has 160 bits (to defeat birthday attacks)
User has key pair $(g^x, x)$ for $x \in \mathbb{Z}_q$

**Prover** $P$
(knows $x$)

- $k \leftarrow \mathbb{Z}_q$
- $r = g^k (\text{mod } p)$

**Verifier** $V$
(knows $g^x$)

- $e \leftarrow \mathbb{Z}_q$ ("random challenge")
- $s = k - xe$
- Accept if
  \[ g^s = r / \text{PK}^e \]

Note: $g^s = g^{k-xe} = r / (g^x)^e = r / \text{PK}^e$

**Claim:** Prover "must know" $SK \times x$ if he can answer most challenges

**Pf idea:** Suppose Prover can answer $e_1$, $e_2$

- $g^{s_1}\text{PK}^{e_1} = g^{s_2}\text{PK}^{e_2} = r$
- $(s_1-s_2) / (e_2-e_1)$
- $g = \text{PK}$
- $(s_1-s_2) \times (e_2-e_1)$

Prover "knows" $x$!
Claim: Verifier gains no information about x. ("Honest" who picks e at random from \(\mathbb{Z}_q\))

Proof idea:
Verifier can generate transcript on his own! (from PK)

\[
\text{transcript} = (PK, r, e, s)
\]

How?
Verifier chooses e at random from \(\mathbb{Z}_q\).

s at random from \(\mathbb{Z}_q\).

computes \(r = g^s \cdot PK^e\) 

(called Honest Verifier Zero Knowledge)

How to convert a three-round public coin ID protocol to a digital signature scheme?

\[
\begin{align*}
\text{commit} & \quad \rightarrow \quad r \\
\text{challenge} & \quad \leftarrow \quad e \\
\text{response} & \quad \rightarrow \quad s
\end{align*}
\]

(e is "public coin")

Accept based on PK, r, e, s
Answer: Fiat-Shamir heuristic

\[ H \text{ is hash function} \]
\[ \text{Let } e = H(m, r) \text{ from ROM} \]
\[ \text{Sign}(sk, m) = (r, e, s) \text{ where } e = H(m, r) \]
\[ \text{Verify}(pk, m, (r, e, s)) \]
\[ \text{Accepts if verifier of ID scheme accepts} \]

Claim: We can use Fiat-Shamir to convert Schnorr ID scheme to a (secure) Schnorr signature scheme. (secure against ACRs)

\[ \sigma = (r, e, s) = (g^k, H(m, r), k - x \cdot H(m, r)) \]

Proof ideas:

Seeing signs of other messages is just like seeing attacks on ID protocol - just seeing \( H(m, r) \) instead of verifying \( e \). Zero-knowledge property of ID protocol gives attacker no benefit.

If Adversary can forge, he must be able to supply good response to many possible \( e \)'s (possible \( H(m, r) \) values). This implies he "knows" \( SK \ x. \)
Digital Signature Standard (DSA)

Like Schnorr signature scheme, except:
- \( r \) is computed as \( g^k \pmod{p} \pmod{q} \)
  (for shorter signatures)
- \( e = H(m) \) rather than \( e = H(m, r) \)
  (This version not known to be secure in ROM. (Insecure in Schnorr but not known to be insecure in DSA)).
**DSA details**

**Setup:**
- \( g = 160 \text{ bit prime} \)
- \( p = 1024 \text{ bit prime s.t. } g \mid p-1 \)
- \( g = h^{(p-1)/q} \) generates group of order \( q \)

**Gen:**
- \( SK = x \in \mathbb{Z}_q \)
- \( PK = g^x \in \mathbb{Z}_q^* \)  
  \( (PK = y \text{ below}) \)

**Sign:**
- \( k \leftarrow \mathbb{Z}_q^* \) \text{  (must be random & new!)}
- \( r = \left( g^k \mod p \right) \mod q \) \text{  rest artificially fixed if } f = 0
- \( s = \left( H(m) + xk \right) \mod q \) \text{  rest artificially fixed if } s = 0
- \( \sigma = (r, s) \)

**Verify \((PK, m, \sigma)\):**
- Check that \( 0 < r < q \) & \( 0 < s < q \)
- \( w = s^{-1} \mod q \)
- \( u_1 = H(m) \cdot w \mod q \)
- \( u_2 = r \cdot w \mod q \)
- \( v = \left( g^{u_1} y^{u_2} \mod p \right) \mod q \)
- Accept if \( v = r \)