Admin:

- Pset #3 due 4/5 (Mon)
- Take-home quiz out 4/5 due 4/21 (Wed)
- Projects!
- Guest lecture by Jim Bidzos on 4/12 (Mon)

Today:

- Digital Signatures!
  - Diffie-Hellman concept of PK signatures
  - ACMA defn of security ("Adaptive chosen message attack")
  - Textbook RSA signatures
  - Hash & Sign (Full Domain Hash)
  - Schnorr ID scheme
  - Fiat-Shamir paradigm
  - Schnorr signatures
  - Digital Signature Algorithm (DSA) by NIST
Diffie & Hellman ("New Directions in Cryptography")
1976

- \( \text{Gen}(1^k) \to (\text{PK}, \text{SK}, M, C) \)
- \( |M| = |C| \)
- \( \text{Enc}(\text{PK}, \cdot) \) maps \( M \) to \( C \) 1 to 1 efficiently computable
- \( \text{Dec}(\text{SK}, \cdot) \) maps \( C \) to \( M \) 1 to 1

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

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

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

\[ \sigma \xmapsto{\text{Sign}} m \xmapsto{\text{Verify}} \sigma \]

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

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

Correctness: \( \text{Verify}(\text{PK}, \sigma) = m \) if \( \sigma = \text{Sign}(\text{SK}, m) \)

(\text{Security defined in a bit...})

\( \text{Enc} \) & \( \text{Dec} \) are trapdoor permutations
(\text{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 Verify)

Security:

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

\[
\text{if } (\forall \text{ PPT } A) \quad \text{(Adversary)}
\]

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

\[
(PK, sk) \leftarrow \text{Gen}(1^\lambda) \quad \text{Verify}(PK, m^*, \sigma^*) = \text{true}
\]

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

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

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:

- \( \text{Gen} (1^k) \rightarrow (PK, SK, M, c) \)
  
  \( PK = (n, e) \)
  
  \( SK = (n, d) \)
  
  \( d = e^{-1} (\text{mod} \ \phi(n)) \)

- \( \text{Sign} (SK, m) \rightarrow \sigma = m^d (\text{mod} \ n) \)

- \( \text{Verify} (PK, m, \sigma) = \text{true} \) iff \( \sigma^e = m \ (\text{mod} \ n) \)

This is just basic RSA encryption "turned around"!

This is not secure against ACMA:

- if \( \sigma \) is signature for \( m \)
  
  then \( \sigma^2 \) is signature for \( m^2 \)

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

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

$$(\text{hard to compute } x^d \text{, given } n, e, x \text{ mod } n)$$

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

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

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 $r = \sigma^e \pmod{n}$, program $H(m) = r$, output $\sigma$ as signature for $m$. 
(If Adv asks for $H(m)$, where $m$ previously unspent, choose random $s \in \mathbb{Z}_n^*$, set $r = s^e (\text{mod } n)$ 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 $G$

- $g$ is a generator of $G$

- E.g. work $\text{mod } p$, where $p = q \cdot r + 1$ and $q$ is prime

- $G = \left\{ h^r \mod p, h \in \mathbb{Z}_p^* \right\}$

- $1 \leq 1 = q$

- To find $g$ generator of $G$, choose $h \in \mathbb{Z}_p^*$, $h \neq 1 \mod p$; let $g = h^r (\text{mod } p)$.

- Typically $p$ has 1024 bits (to defeat DL attacks)

- $q$ has 160 bits (to defeat birthday attacks)
User has keypair $(g^x, x)$ for $x \in \mathbb{Z}_q$.

**Prover $P$**

(knows $x$)

$k \leftarrow \mathbb{Z}_q$

**Verifier $V$**

(knows $g^x$)

$r = g^k \mod p$

$e \leftarrow \mathbb{Z}_q$ ("random challenge")

$e$ = e

$g = k^{-x} e$

Accepts iff

$g^e = r/pk^e$

Note: $g^e = g^{k^{-x}e} = r/(g^x)^e = r/pk^e$

**Claim:** Prover "must know" SK $x$ if he can answer most challenges $\frac{e_1}{s_1}$

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

$g^{s_1} pk^{e_1} = g^{s_2} pk^{e_2} = r$

$(s_1-s_2)/(e_2-e_1)$

$g = \text{PK}$

$\Rightarrow (s_1-s_2)/(e_2-e_1)$ is $\text{SK} \times$ !

Prover "knows" $x$!
Claim: Verifier gains no information about x.

"Honest" (who picks e at random from $\mathbb{Z}_p$)

Proof idea:

Verifier can generate transcript on his own! (from pk)

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

How?

Verifier chooses e at random from $\mathbb{Z}_p$

s at random from $\mathbb{Z}_q$

computes $r = g^s \text{PK}^e$

(called Honest Verifier Zero Knowledge)

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

commit $\rightarrow r$

challenge $\leftarrow e$ (e is "public coin")

response $\rightarrow s$

Accept based on \(\text{PK}, r, e, s\)
Answer: **Fiat-Shamir heuristic**

Let $e = H(m, r)$

$\text{Sign} (sk, m) = (r, e, s)$

where $e = H(m, r)$

$\text{Verify} (pk, m, (r, e, s))$

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 ACGA)

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

Proof idea:

Seeing signs of other messages is just like seeing attacks on ID protocol — just seeing $H(m, r)$ instead of verifier's $e$. 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" $SK x$. 

H is hash function (CR)
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:**
- \( SK = \{ x \in \mathbb{Z}_q^* \} \)
- \( PK = g^x \) \( (PK = y \text{ below}) \)

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

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