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

- Gen(1^k) \rightarrow (PK, SK, M, C)
  - Ciphertext space
  - Message space
  \mid M \mid = \mid C \mid

- Enc(PK, \cdot) maps M to C \hspace{1cm} 1 to 1
  - Efficiently computable

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

Enc & Dec are inverse functions (given PK & SK)

- For signatures, we rename Enc as Verify, Dec as Sign

Sign
  \sigma = Sign(SK, m)
  \sigma = \text{signature on } m \text{ by } PK

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

Correctness: Verify(PK, \sigma) = m
  \text{if } \sigma = Sign(SK, m)
  \text{(Security defined in a bit...)}

Enc & Dec are trapdoor permutations
  (SK is trapdoor)
SK is "signing key"

PK is "signature verification key"

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

Verify($PK, m, \sigma$) ∈ {true, false}?

(Note: have pulled $m$ inside as arg to Verify)

Security:

Signature scheme $(Gen, Sign, Verify)$ is secure against adaptive chosen message attack

if $(\forall \text{PPT } A)$ (Adversary)

$$\Pr \left[ A^{\text{Sign}(SK, \sigma)}(PK) = (m^*, \sigma^*) \text{ such that } (PK, SK) \leftarrow \text{Gen}(1^\lambda) \text{ Verify}(PK, m^*, \sigma^*) = \text{true} \& m^* \text{ was not ever given to Sign} \right] = \text{negl}(\lambda)$$

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:

- Gen$(1^x) \rightarrow (PK, SK, M, C)$
  
  \[ PK = (n, e) \]
  
  \[ SK = (n, d), \quad d = e^{-1}(\text{mod } \phi(n)) \]

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

- 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 $m$.

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

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

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

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

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 untrusted, 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 $p = q \cdot r + 1$
  and $q$ is prime
- $G = \{ h^* \pmod{p}, h \in \mathbb{Z}_p^* \}$
- $|G| = q$
- To find $g$ generator of $G$, choose $h \in \mathbb{Z}_p^*$, $h \not\equiv 1 \pmod{p}$; let $g = h^* \pmod{p}$.
- 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** $P$

- $(\text{knows } x)$

- $k \leftarrow \mathbb{Z}_q$

- $\text{commit}$

- $r = g^k \pmod{\text{p}}$

**Verifier** $V$

- $(\text{knows } g^x)$

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

- $\text{challenge}$

- $g = k - xe$

- $\text{response}$

- $s = r / pk^e$

Accepts iff $g^s = r / (g^x)^e$

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

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

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

\[ g^{s_1} pk^{e_1} = g^{s_2} pk^{e_2} = r \]

\[ (s_1 - s_2) / (e_1 - e_2) = pk \]

\[ (s_1 - s_2) \text{ is } SK \times ! \]

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?

Commit \( r \)

challenge \( e \) (e is “public coin”)

response \( s \)

Accept based on PK, r, e, s
Answer: \textbf{Fiat-Shamir heuristic}

Let \( e = H(m, r) \) \hspace{1cm} \text{ROM}

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

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 ACMA)

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

Proof ideas:

Seeing signs of other messages is just like
Seeing attacks on ID protocol - just seeing
\( H(m, r) \) instead of 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" \( SK \times x \).
Digital Signature Standard (DSA)

Like Schnorr signature scheme, except:

- $r$ is computed as $g^x \pmod{p} \pmod{q}$ (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:**
- \( q = 160 \) bit prime
- \( p = 1024 \) bit prime s.t. \( q \mid p-1 \)
- \( g = h^{(p-1)/q} \) generates group of order \( q \)

**Gen:**
- \( SK = x \in \mathbb{Z}_q^* \)
- \( PK = g^x \) \hspace{1cm} (PK=y below)

**Sign:**
- \( k \in \mathbb{Z}_q^* \) \hspace{1cm} (must be random & new!)
- \( r = (g^k \mod p) \mod q \) \hspace{1cm} restart if \( r = 0 \)
- \( s = \left( \frac{H(m) + xr}{k} \right) \mod q \) \hspace{1cm} restart if \( s = 0 \)
- \( \sigma = (r, s) \)

**Verify (PK, m, \sigma):**

Check that \( 0 \leq r < q \) & \( 0 < s < q \)

- \( w = s^{-1} \mod q \)
- \( u_1 = H(m) \cdot w \mod q \)
- \( u_2 = r \cdot w \mod q \)
- \( v = (g^{u_1} \cdot u_2 \mod p) \mod q \)

Accept iff \( v = r \)