TOPIC: 6.857

DATE: 3/31/21

FILE UNDER:

PAGE: L 12.1

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

TOPIC:

DATE:

FILE UNDER:

PAGE: L12.2

Diffie & Hellman ("New Directions in Cryptography") 1976

- $\text{Gen}(1^\lambda) \to (PK, SK, M, C)$

  C ← ciphertext space

  M ← message space

  $$|M| = |C|$$

- $\text{Enc}(PK, \cdot)$ maps M to C — 1 to 1

  efficiently computable

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

$$\begin{tikzcd} m \arrow[r, bend left, "\text{Enc}"] & c \arrow[l, bend left, "\text{Dec}"] \end{tikzcd}$$

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

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

$$\begin{tikzcd} \sigma \arrow[r, bend left, "\cancel{\text{Enc}}\ \text{Verify}"] & m \arrow[l, bend left, "\text{Sign}"] \end{tikzcd}$$

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

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

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

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

(Security defined in a bit...)

Enc & Dec are trapdoor permutations

(SK is trapdoor)

SK is "signing key"

PK is "signature verification key"

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

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

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

Security:

ACMA

Signature scheme (Gen, Sign, Verify) is secure against adaptive chosen message attack if ($\forall$ PPT A) (Adversary)

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

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

TOPIC:

FILE UNDER:

DATE:

Textbook RSA signatures:

- $\text{Gen}(1^\lambda) \rightarrow (PK, SK, M, C)$

  $PK = (n, e)$
  $SK = (n, d) \quad d = e^{-1} \pmod{\varphi(n)}$

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

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

This is just basic RSA encryption "turned around"!

This is not secure against ACMA:

- if σ is signature for m

  then σ^2 is signature for m^2

- Worse: σ is signature for $m = \sigma^e \pmod n$

How to fix?

TOPIC:

DATE:

FILE UNDER:

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 $n, e, x \bmod n$)

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

$\text{Verify}(PK, m, \sigma) = \text{true}$ iff $\sigma^e \equiv H(m) \bmod n$.

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 $\sigma \leftarrow \mathbb{Z}_n^*$ compute $r = \sigma^e \pmod n$, program $H(m) = r$, output σ as signature for m.

TOPIC:

DATE:

FILE UNDER:

(If Adv asks for $H(m)$, where m previously unseen, choose random $\sigma \leftarrow Z_n^*$, set $r = \sigma^e \pmod n$ 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 <u>prime order</u>

  g is a generator of G
- E.g. work mod p, where
  $$p = q \cdot r + 1$$
  and q is prime
- $G = \{ h^r \bmod p,\ h \in Z_p^* \}$

  $|G| = q$
- To find g generator of G, choose $h \in Z_p^*$,

  $h^r \neq 1 \bmod p$; let $g = h^r \pmod p$.
- Typically p has 1024 bits (to defeat DL attacks)

  q has 160 bits (to defeat birthday attacks)

User has ~~key~~ keypair (g^x, x) for $x \in \mathbb{Z}_q$

$$(\underbrace{g^x}_{PK}, \underbrace{x}_{SK})$$

Prover P		Verifier V
(knows x)		(knows g^x)
$k \leftarrow \mathbb{Z}_q$		
commit $\rightarrow$	$\xrightarrow{r = g^k \pmod{p}}$	
challenge $\leftarrow$	$\xleftarrow{e}$	$e \leftarrow \mathbb{Z}_q$ ("random challenge")
response $\rightarrow$	$\xrightarrow{s = k - xe}$	
		<u>Accepts</u> iff $g^s = r / PK^e$

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

<u>Claim:</u> Prover "must know" SK x if he can answer most challenges $\xleftarrow{e_i}\ \xrightarrow{s_i}$

<u>Pf idea:</u> Suppose Prover can answer e_1 & s_2

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

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

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

Prover "knows" x !

TOPIC:

DATE:

FILE UNDER:

Claim: Verifier gains no information about x.

"Honest" (who picks e at random from 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 Z_q

s at random from Z_q

computes $r = g^s PK^e$ ▨

(called Honest Verifier Zero Knowledge)

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

Commit $\xrightarrow{\quad r \quad}$

challenge $\xleftarrow{\quad e \quad}$ (e is "public coin")

response $\xrightarrow{\quad s \quad}$

Accept based on PK, r, e, s

TOPIC:

FILE UNDER:

DATE:

H is hash function (CR)

Answer: Fiat-Shamir heuristic

Let $e = H(m, r)$ 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 idea:

Seeing sigs 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.

∎

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)).

TOPIC:

FILE UNDER:

DATE:

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: $SK = x \in Z_q^*$
$PK = g^x$ (PK = y below)

Sign: $k \leftarrow Z_q^*$ (must be random & new!)

$r = (g^k \bmod p) \pmod q$ restart if $r=0$

$s = \left(\frac{H(m)+xr}{k}\right) \bmod q$ restart if $s=0$

$\sigma = (r,s)$

Verify (PK, m, σ)

Check that $0<r<q$ & $0<s<q$

$w = s^{-1} \pmod q$

$u_1 = H(m)\cdot w \bmod q$

$u_2 = r \cdot w \bmod q$

$v = (g^{u_1} y^{u_2} \bmod p) \pmod q$

Accept iff $v = r$