Admin:

Outline: RSA security; digital signatures

- RSA security; factoring
- RSA-OAEP for ACCA security
- Digital Signatures - definitions & security
- Signing with RSA
- El Gamal signatures
- Digital signature standard
RSA is homomorphic (like El Gamal):

\[ E(m_1) \cdot E(m_2) = E(m_1 m_2) = (m_1 m_2)^e \pmod{n} \]

RSA is not even semantically secure (since it is deterministic).

Adversary can easily tell \( E(m_0) \) from \( E(m_1) \) since these values are fixed.

Need to repair this...

**Notes on factoring:**

- knowing \( e \) doesn't help in factoring \( n \) (\( e \) is random)
- computing \( d \) is as hard as factoring \( n \)
  knowing \( e, d \Rightarrow \) know multiple of \( \phi(n) \Rightarrow \) know \( p, q \)

**Best factoring algorithms (Number field sieve):**

- \( \exp \left\{ \text{const} \cdot (\ln(n))^{1/3} \cdot (\ln \ln n)^{2/3} \right\} \)
  \( \approx \exp \left\{ k^{1/3} \right\} \) for \( k \)-bit \( n \)

**Now:** 512-bit \( n \)'s can be factored, 1024-bit seem a bit out of reach...

Using \( n \) in range 1024...4096 seems fine... (for now...)
How to make RSA IND-CCA2 secure?

"OAEP" = Optimal asymmetric encryption padding \[ [8R97] \]

Given \( m \), \( |m| = t \) bits

Pick \( r \) at random, \( |r| = k_0 \)

\[
\begin{align*}
G &: 50,13 \rightarrow 50,13 \uparrow t+k_1 \\
N &: 50,13 \rightarrow 50,13 \uparrow k_0 \\
G, H &: \text{"random oracles" like UFE (!) of Desai}
\end{align*}
\]

On decryption: invert RSA
invent OAEP
reject if \( O^k \) not present

**Thm:** RSA with OAEP secure against ACCA, assuming

RO model & that RSA hard to invert on random inputs.

OAEP: used in practice

**theory:** (we don't have random oracles... )
Digital Signatures

- Invented by Diffie/Hellman in 1976 (New Directions)
- First implementation: RSA (1977) [key motivator for me for PK... !]
- Initial idea: switch PK/sk - enc with secret key = sig
  - if PK decrypts it - then sig ok

- Current way of describing digital signatures
  - (Note: law is confused (include hashes, MACs, etc...) - ignore it

  - KeyGen(1^λ) → (PK, SK)
    - verification key
    - signing key
    - λ = "security parameter"
      - all lengths are polynomial
      - security may be negligible for λ.

  - Ignore for now - "PKI" issue:
    - Knowing that you have "right" PK

  - Sign(SK, M) → σ_{SK}(M)
    - M ∈ {0, 1}^* (may be randomized)

  - Verify(PK, M, σ) = True/False

Correctness: \((∀M)\) Verify(PK, M, Sign(SK, M)) = True
Security: (Weak) existential unforgeability under adaptive chosen message attack:

Game:

Challenger

\((PK, sk) \leftarrow \text{Keygen}(1^\lambda)\)

\[ \begin{array}{c}
PK \\
\downarrow \\
\sigma(M_1) \\
\downarrow \\
m_2 \\
\downarrow \\
\sigma(m_2) \\
\downarrow \\
\vdots \\
\downarrow \\
m_k \\
\downarrow \\
\sigma(m_k) \\
\downarrow \\
M, \sigma_\star \\
\end{array} \]

\begin{align*}
\text{Adv wins if } & \text{Verify}(PK, M, \sigma_\star) = \text{True} \\
& \text{and } M \notin \{M_1, \ldots, M_k\}
\end{align*}

Scheme is secure (i.e., weakly existentially unforgeable against adaptive chosen message attack)

\[ \text{Prob[Adv wins]} \text{ is negligible (i.e., } \leq \frac{1}{\lambda^c} \text{ for all sufficiently large } \lambda) \]

Scheme is strongly secure if adversary can't even produce new sig for previous message previously signed

\[ \text{i.e., Adv wins if } \text{Verify}(PK, M, \sigma_\star) = \text{True} \]

\[ \& (M, \sigma_\star) \notin \{ (M_1, \sigma_1), (M_2, \sigma_2), \ldots, (M_k, \sigma_k) \} \]
Sign with RSA

1. Hash & sign with PKCS
   Let $H(M) = $SHA256$(M) \quad \text{(normal hash)}$
   Let $H'(M) = 0x\ 00\ 01\ FF\ FF\ ...\ FF\ 00\ ||\ H(M)\ ||\ H(M)$
   $\sigma(m) = (H'(m))^d \mod n$

   Some problems with $e=3$ (bad implementations can form 0
   padding ASN.1
   take $H(M)$
   miss other stuff after $H(M)$)

   Otherwise seems OK, but no proofs. (even
   assuming collision resistance & RSA hard to invert...)

   Commonly used, none the less...

2. PSS [Bellare & Rogaway 1996]
\[
\text{Sign}(m): \begin{cases} \\
\text{w} \leftarrow h(M \| r) \quad \text{note!} \quad |w| = k_1 \\
\text{r}^* \leftarrow g_1(w) \oplus r \\
\text{y} \leftarrow 0 \| w \| r^* \| g_2(w) \\
\text{return } y^d \pmod{n} \end{cases}
\]

\[
\text{Verify}(M, x): \begin{cases} \\
y \leftarrow x^e \pmod{n} \\
\text{parse } y \text{ as } b \| w \| r^* \| \gamma \\
r \leftarrow r^* \oplus g_1(w) \\
\text{if } h(M \| r) = w \land g_2(w) = \gamma \land b = 0 \text{ return True} \\
\text{else return False} \end{cases}
\]

**Theorem:** PSS is (weakly) semantically unforgeable against chosen message attack in ROM if RSA is not invertible on random inputs.

(\text{\texttt{TEAdv}} who can produce } x^d \text{ given } x.)
El Gamal Signatures

Public system parameters: \( p \) prime, \( g \) generator

Keygen: \( \mathbf{x} \in \mathbb{Z}_p \rightarrow \mathbb{Z} \), \( \mathbf{y} = g^\mathbf{x} \)

\( \mathbf{SK} = \mathbf{x} \)

\( \mathbf{PK} = \mathbf{y} \)

\[ \text{Sign}(M): \quad m = h(M) \]

\[ k \in \mathbb{Z}_{p-1}^* \quad [\text{gcd}(k, p-1) = 1] \]

\[ r = g^k \quad [\text{hard work is independent of } M] \]

\[ ks + rx = m \]

\[ s = (m-rx) \pmod{p-1} \]

\[ \sigma(M) = (r, s) \]

\[ \text{Verify:} \quad \begin{cases} \text{check } 0 < r < p \\ y^r s = g^m \pmod{p} \quad \text{where } m = h(M) \end{cases} \]

Return True if both checks pass else return False

Correctness: \( g^{rx} g^{sk} = g^{rx+sk} \equiv g^m \pmod{p} \)

\[ \equiv \]

\[ rx + ks = m \pmod{p-1} \]

\[ s = (m-rx) \pmod{p-1} \]

(if \( \text{gcd}(k, p-1) = 1 \))
Theorem: El Gamel is existentially forgeable (without h fn
or h = id(h),)

Proof: Let \( e \in R \mathbb{Z}_{p-1} \)

\[
\begin{align*}
    r &\leftarrow g^e \mod p \\
    s &\leftarrow -r \mod (p-1)
\end{align*}
\]

\((r, s)\) is sig for message \( m = es \mod (p-1) \)

\[y^r r^s = g^m\]

\[g^{xr} (g^e)^{-r} = g^{-r} = g^e = g^m \text{ for } m = es \mod (p-1)\]

But: It is easy to fix.

Modified El Gamel (Pointcheval/Stern 1996)

**Sign** *(m)*: \( k \in R \mathbb{Z}_p^* \)

\[r = g^k \mod p \]

\[m = h(M || r) \leq 255\]

\[s = (m - rx) \mod (p-1) \]

\([s] \leq 255\]

\[\sigma(M) = (r, s)\]

**Verify:** check \( 0 < r < p \)

check \( y^r r^s = g^m \) where \( m = h(M || r) \)
Thm: (Modified) El Gamal is existentially unforgeable against adaptive chosen message attack, in ROM, assuming DLP is hard.