Outline

- Bleichenbacher attack
- CCA Secure Encryption
- RSA OAEP
Public Key Encryption
Textbook RSA System

\[ \text{Gen}(\cdot): \text{ choose random primes } p, q \approx 1024 \text{ bits. Set } N = pq. \]

choose integers \( e, d \) s.t. \( e \cdot d \equiv 1 \pmod{\phi(N)} \)

output \( pk = (N, e) \), \( sk = (N, d) \)

\[ \text{RSA-Enc } (pk, x) = x^e \quad \text{(in } Z_N) \]

\[ \text{RSA-Dec } (pk, y) = y^d \quad \text{(in } Z_N) \]

\[ y^d = x^{ed} = x^{k\phi(N)+1} = (x^{\phi(N)})^k \cdot x = (1)^k \cdot x = x \]

Trapdoor Permutation:
Trapdoor Permutation

Three algorithms: \((G, F, F^{-1})\)

- **G**: outputs \(pk, sk\). \(pk\) defines a function \(F(pk, \cdot) : X \rightarrow X\)

- **F\((pk, x)\)**: evaluates the function at \(x\)

- **\(F^{-1}(sk, y)\)**: inverts the function at \(y\) using \(sk\)

**Secure trapdoor permutation:**

The function \(F(pk, \cdot)\) is one-way without the trapdoor \(sk\)

\[
\Pr \left[ A(pk, y) \rightarrow x \right] = \text{negl}
\]

Given \(x\) s.t. \(F(pk, x) = y\).
The RSA assumption

RSA assumption: RSA is one-way permutation

For all efficient algs. A:

\[ \Pr \left[ A(N,e,y) = y^{1/e} \right] < \text{negligible} \]

where \( p, q \leftarrow^R \text{n-bit primes}, \ N \leftarrow pq, \ y \leftarrow^R Z_N^* \)
Textbook RSA System

\[\text{Gen(\(.)\): } pk = (N, e), \ sk = (N, d) \text{ such that } e \cdot d = 1 \pmod{\varphi(N)}\]

\[\text{RSA-Enc } (pk, x) = x^e \quad \text{(in } Z_N)\]

\[\text{RSA-Dec } (pk, y) = y^d \quad \text{(in } Z_N)\]

Is not semantically secure

How would you make it semantically secure?

\[\text{Enc } (pk, x; r) = (r^e, H(r) \oplus m)\]

extractor
Speeding up RSA

To speed up RSA use a small $e$: $c = m^e \pmod{N}$

- Recommended value: $e \geq 65537 = \sqrt{2^{16} + 1}$ (gcd$(e, \varphi(N)) = 1$)
  
  Encryption: 17 multiplications

Asymmetry of RSA: fast enc. / slow dec.

- ElGamal (next module): approx. same time for both.
RSA in practice: PKCS1 v1.5

PKCS1 mode 2: (encryption)

- Resulting value is RSA encrypted
- Widely deployed, e.g. in HTTPS
- Suffered from a CCA attack
CCA Attack on PKCS1 v1.5

PKCS1 used in HTTPS:

Chosen-ciphertext attack: to decrypt a given ciphertext $C$ do:

1. Choose $r \in \mathbb{Z}_N$. Compute $c' \leftarrow r^e \cdot C = \left( r \cdot \text{PKCS1}(m) \right)^e$
2. Send $c'$ to web server and use response

\[ e \cdot d = 1 \mod \Phi(N) \]
\[ (x^e)^d \equiv x \mod N \] (Bleichenbacher 1998)

\[ c = \text{ciphertext} \]
\[ \text{(PKCS1(m))} \]

Is this PKCS1?

\[ \text{02} \]
\[ \text{FF} \]
\[ \text{msg} \]

\[ \Rightarrow \text{attacker can test if 16 MSBs of plaintext = '02'} \]
Suppose $N$ is $N = 2^n$ (an invalid RSA modulus). Then:

- Sending $c$ reveals $\text{msb}(x)$
- Sending $2^e \cdot c = (2x)^e$ in $\mathbb{Z}_N$ reveals $\text{msb}(2x \mod N) = \text{msb}_2(x)$
- Sending $4^e \cdot c = (4x)^e$ in $\mathbb{Z}_N$ reveals $\text{msb}(4x \mod N) = \text{msb}_3(x)$
- ... and so on to reveal all of $x$
Chosen Ciphertext Security for Public Key Encryption

Challenger → pk → Attacker

\[ \text{Dec}_{sk}(\tilde{c}) \]

\[ \text{Dec}_{sk}(\hat{c}) \]

\[ \text{Dec}_{sk}(\hat{c}) \]

\[ m_0, m_1 \]

\[ c = \text{Enc}_{pk}(m_0) \]

\[ c' = \text{Enc}_{pk}(m_1) \]

\[ \text{Pr}(b' = 1 | b = 0) - \text{Pr}(b' = 1 | b = 1) \]

\[ b' \? \]

\[ \{ c, \hat{c}, \hat{c}, \ldots \} \]
PKCS1 v2.0: OAEP

New preprocessing function: OAEP [BR94]

Thm [FOPS’01]: RSA is a trap-door permutation \( \Rightarrow \)
RSA-OAEP is CCA secure when \( H, G \) are random oracles

in practice: use SHA-256 for H and G
What is a random oracle?

- H: Truly random function

- "Observable"

  No attacker can output $Z = H(y)$ without first querying oracle on $y$. $H(y)$ is uniform random unless attacker queries R.O. on $y$.

- "Programmable"

  Stronger than observability.

  "Proof of security / challenger" can set the output of RO on any given input $x$. 

  STRONGER: For any attacker $H(y)$ is uniform random unless attacker queries R.O. on $y$. 

  $S_1$: Uniform random string

  $S_2$: Uniform random string

  $S_3$: Uniform random string
Analyze OAEP

Fujisaki-Okamoto

Pointcheval-Stemmer

plaintext to encrypt with RSA

\[ m, 01 00..0 \]

rand.

\[ + \]

\[ H \]

\[ G \]

\[ + \]

\[ \text{plaintext to encrypt} \quad \text{with RSA} \]

\[ e \]

\[ c \]

\[ Dec(c) = pt_1 \quad pt_2 \]

\[ r_1 = G(pt_1) \oplus pt_2 \]

\[ m'_1 = H(r_1) \oplus pt_1 \]

pt_1 = m \oplus H(r)

pt_2 = G(pt_1) \oplus r

Goal: Argue plaintext-awareness.

Claim: If attackers "knows" (r, pt), they also "know" m.
Challenger MUST HAVE recorded queries \((pt, ri)\) s.t. \(ct\) can be recomputed given \((pt, ri)\).

Analyze OAEP

Plaintext awareness \(\Rightarrow\)

Any attacker that queries Dec. oracle on \(\widehat{ct}\), must "know" underlying \((m, ri)\).

\(\Rightarrow\) For any attacker that makes CCA Dec. oracle query on \(\widehat{ct}\),

Challenger sitting inside RO, must have "recorded" a query that would allow challenger to compute \((m, ri)\) without using \(sk\) at all.
Authenticated Key Exchange
Authenticated Key Exchange

- Recall key exchange

- Adversary eavesdrops on the network
Authenticated Key Exchange

- Adversary has complete control of the network
  - Modify/inject/delete packets

Man-in-the-middle attack.
Trusted Third Parties

- Adversary has complete control of the network
  - Modify/inject/delete packets
Authenticated Key Exchange

(next lecture)
Important Component: Signature Scheme

\[ \sigma = \text{Signature}(\text{sign} k, m) \]
Digital Signatures

\[ \text{Challenger} \quad (vk, sign_k) \]

\[ \text{Attacke}r \]

\[ \sigma = \text{Sign}(sign_k, m_1) \]

\[ \vdots \]

\[ \sigma = \text{Sign}(sign_k, m_i) \]

\[ \vdots \]

\[ \text{Wins if} \quad \text{Verify}(\hat{m}, \hat{\sigma}, vk) = 1 \]

\[ \text{"Public-key" MAC} \quad \text{Verification is "public" instead of private.} \]
Digital Signatures

• Public-key version of “MACs”
One-Time Signatures

• Definition of security
Lamport One-Time Signatures

- \( vk = \) 
  
- \( sk = \)
Lamport One-Time Signatures

- One-time security:
Lamport One-Time Signatures

- One-time security:
Lamport One-Time Signatures

• What about two-time security?
Summary

- RSA review
- Bleichenbacher attack
- PKCS 1 v2.0
- Authenticated Key Exchange and Signatures