Public-Key Encryption

Viet Tung Hoang

Some slides are based on material from Prof. Stefano Tessaro, University of Washington
Agenda

1. High-level PKE
2. Building PKE
3. Padding-oracle attack on PKCS1
4. CCA Security and OAEP
5. App: Authenticated Key Exchange
Motivation

**Problem:** Alice and Bob must be online simultaneously for key exchange
Public-Key Encryption (PKE): Syntax

Key Gen

\[ \mathcal{K} \xrightarrow{\$} \{pk, sk\} \]

Encrypt

\[ M \xrightarrow{\$} E \xrightarrow{pk} C \]

Decrypt

\[ C \xrightarrow{sk} D \xrightarrow{M \text{ or } \perp} \]
Alice generates a pair of secret key and public key. She keeps $sk$ to herself, and stores $pk$ in a public, trusted database.
PKE Usage

First retrieve Alice’s public key

Then email the encrypted message to Alice under her public key

Alice can later decrypt using her secret key

$E_{pk}(M)$
Exercise: Sharing Encrypted Files

Encrypt a file so that when we place the ciphertext in a shared folder, only selected people can decrypt, assuming everybody has a public key.

Alice  Bob  Carol  Dave
PKE: CPA Security

- Similar to the Left-or-Right security of Symmetric encryption
- **Difference**: The adversary is given the public key

**Left**

\[
\text{procedure } \text{Enc}(m_0, m_1) \\
\text{Return } \mathcal{E}_{pk}(m_0)
\]

**Right**

\[
\text{procedure } \text{Enc}(m_0, m_1) \\
\text{Return } \mathcal{E}_{pk}(m_1)
\]
Performance Issue

Standard PKE schemes can only encrypt short messages (say \( \leq 2048 \) bits)
How should we encrypt long ones?

A (not so good) solution:

- Break the message into small chunks
- Encrypt each chunk individually

\[
\begin{array}{cccc}
M_1 & M_2 & M_3 & M_4 \\
\end{array}
\]

\[
\begin{array}{cccc}
E_{pk}(M_1) & E_{pk}(M_2) & E_{pk}(M_3) & E_{pk}(M_4) \\
\end{array}
\]

Problem: PKE is very expensive, so this solution is several thousands times slower than AES-CTR
Hybrid Encryption

- Generate a random key $K$
- Encrypt the key $K$ by PKE, and use CTR under key $K$ to encrypt the message

Can replace CTR by your favorite symmetric encryption
Agenda

1. High-level PKE

2. Building PKE

3. Padding-oracle attack on PKCS1

4. CCA Security and OAEP

5. App: Authenticated Key Exchange
Number Theory Basics

For \( n \in \{1, 2, 3, \ldots \} \), define

\[
\mathbb{Z}_n^* = \{ t \in \mathbb{Z}_n \mid \gcd(t, n) = 1 \}
\]

\[
\varphi(n) = |\mathbb{Z}_n^*|
\]

Theorem:

- For any \( s \in \mathbb{Z}_n^* \), \( s^{\varphi(n)} \equiv 1 \pmod{n} \)
- \( \varphi \) is multiplicative: if \( \gcd(a, b) = 1 \) then \( \varphi(ab) = \varphi(a)\varphi(b) \)

Examples: For distinct primes \( p \) and \( q \):

\[
\varphi(p) = p - 1
\]

\[
\varphi(pq) = (p - 1)(q - 1)
\]
The RSA Function

Given $e, d \in \mathbb{Z}_{\varphi(n)}^*$ such that $ed \equiv 1 \pmod{\varphi(n)}$

Define a permutation $f$ and its inverse $f^{-1}$ as follows:

$$f(x) = x^e \mod n$$

$$f^{-1}(y) = y^d \mod n$$

**Exercise**: Try $n = 55$ and $e = 3$
A Bad PKE: Plain RSA

Key generation:
- Pick two large primes $p, q$ and compute $n = pq$
- Pick $e, d \in \mathbb{Z}_n^*$ such that $ed \equiv 1 \pmod{\varphi(n)}$
- Return $pk \leftarrow (n, e), sk \leftarrow (n, d)$

Encryption:
- To encrypt message $x$ under $pk = (n, e)$, return $c \leftarrow x^e \mod n$

Decrypt:
- To decrypt a ciphertext $c$ under $sk = (n, d)$, return $x \leftarrow c^d \mod n$

Often $e = 3$ for efficiency
Cracking Plain RSA: First Attempt

Public $e$, $N=pq$  \[ ed \equiv 1 \pmod{(p-1)(q-1)} \]

Secret $d$

A plausible attack:
- Recover $(p-1)(q-1)$
- Compute $d$ such that $ed \equiv 1 \pmod{(p-1)(q-1)}$ in $O(\log(N))$ time using (extended) Euclidean algorithm

Question: Given $N=pq$ and $(p-1)(q-1)$, recover $p$ and $q$
Cracking Plain RSA: Second Attempt

For $e = 3$, a very common choice

For small messages $x < n^{1/3}$:

\[ c = x^3 \mod n \quad \rightarrow \quad x = c^{1/3} \]

**Exercise:** Recover message $x$ when one encrypts $x, x + 1, x + 2$
Why Is Plain RSA Bad?

It doesn’t meet the CPA notion

Reason: Plain RSA is deterministic

In 2016, QQ Browser was found to use Plain RSA to encrypt user data.

China’s Top Web Browsers Leave User Data Vulnerable, Group Says

Report from Citizen Lab accuses Tencent of weak encryption practices with its QQ Browser

By Juro Osawa and Eva Dou

March 28, 2016 5:00 p.m. ET
What Plain RSA Gives: Trapdoor permutation

A triple of algorithms (Gen, Samp, Inv)

\[(f, d) \leftarrow \$ \text{Gen}, \text{ with } f : \text{Dom} \rightarrow \text{Range}\]

For \(x \leftarrow \$ \text{Samp},\) it’s easy to compute \(y = f(x),\) but hard to invert \(f^{-1}(y)\) without knowing the trapdoor \(d\)
Building PKE from Trapdoor Permutation

Plain RSA $\rightarrow$ Hashed RSA

Given a trapdoor permutation \((\text{Gen}, \text{Samp}, \text{Inv})\) and a hash function \(H\)

**Key generation:** Run \((f, d) \leftarrow \text{Gen}\) and return \(pk \leftarrow f, sk \leftarrow d\)

**Encryption:** To encrypt message \(M\) under \(pk = f\)

**Question:** How to decrypt?
1. High-level PKE

2. Building PKE

3. Padding-oracle attack on PKCS1

4. CCA Security and OAEP

5. App: Authenticated Key Exchange
PKCS #1 Encryption

encrypt byte strings only

Give shorter ciphertexts than Hashed RSA

Uses encrypt-with-redundancy paradigm:
Decryption will reject if the format is incorrect

2 bytes
0002

non-zero, random bytes
$$

1 byte
00

1024 bits padded message

Plain RSA Enc
Padding-Oracle Attack

**Context:** Alice is establishing a TLS session with a server

Adversary uses server as a decryption oracle by observing server’s accepting/rejecting of its fake ciphertexts
Padding-Oracle Attack

Recall $C = X^e \mod n$, with $pk = (e, n)$

Padded message

Pick some $r$

$C' \leftarrow C r^e \mod n$

$(X r)^e \mod n$

Accept only if $X r$ has valid PKCS encoding

By using several $r$, can fully recover $X$, and also $M$
Illustrative Toy Problem

1 bit

0 \quad M

X

Plain RSA Enc

\[ C' \leftarrow C r^e \mod n \]

Accept only if
\[ (X r \mod n) < n/2 \]

\[ C' = (X r)^c \mod n \text{ since } C = X^c \mod n \]
Key Idea: Binary Search

Initial search range of $X$: $\{0, \ldots, n - 1\}$

At each step, try to half the range of $X$ by carefully choosing $r$

- Pick $r = 1$ if $X < n$
- Pick $r = 2$ if $(Xr \mod n) < n/2$

- If $X < n/2$:
  - Pick $r = 1$ if $X < n/4$
  - Pick $r = 2$ if $n/4 < X < n/2$

- If $n/2 < X < n$:
  - Pick $r = 1$ if $n/2 < X < 3n/4$
  - Pick $r = 2$ if $3n/4 < X < n$
A Quick Fix and Its Problem

**Want:** Change only server side, for backward compatibility

The change in TLS 1.0:
- If format or length of the decrypted message is incorrect, decryption returns a random 48-byte strings

Hiding decryption failure

**Problem:** Might be broken if implementation is not done properly to ensure that the timing is constant in both decryption success and failure.
Agenda

1. High-level PKE
2. Building PKE
3. Padding-oracle attack on PKCS1
4. CCA Security and OAEP
5. App: Authenticated Key Exchange
Resisting Padding-Oracle Attacks: CCA Security

**Left**

procedure $\text{Enc}(m_0, m_1)$
Return $\mathcal{E}_{pk}(m_0)$

**Right**

procedure $\text{Enc}(m_0, m_1)$
Return $\mathcal{E}_{pk}(m_1)$

$A$ is **prohibited** from feeding ctx from Enc to Dec.
Achieving CCA Security: OAEP

**Use:** 1024-bit Plain RSA and **two** hash functions $H$ and $G$

Modeled as independent random oracles

How to get two hash functions from SHA-256: **Domain separation**

$$H(x)$$

$$G(x)$$
OAEP Design: Feistel Networks

One round Feistel

Inverse of Feistel

Question: How to invert?
OAEP Design: Feistel Networks

**Design paradigm:** Two-round (unbalanced) Feistel

Feistel (in **decryption**)

![Feistel decryption diagram]

Inverse Feistel (in **encryption**)

![Inverse Feistel encryption diagram]
OAEP Encryption

128 bits

\[ r \]

896 bits

\[ 0^{128} \| M \| 10^* \]

Use *encrypt-with-redundancy.*
OAEP Decryption

If $X[1:128] = 0^{128}$ then
Decode $X$ to get $M$
Else return ⊥
1. High-level PKE
2. Building PKE
3. Padding-oracle attack on PKCS1
4. CCA Security and OAEP
5. App: Authenticated Key Exchange
Motivation

Previously: Diffie-Hellman doesn’t work for active adversaries
Asymmetric Authenticated Key Exchange

“Bank”, \( K \)

\( \ast \), \( K \)

Bank doesn’t know who it’s establishing the key with.
That’s why you still need to log in using your password.
Question: Does this work?
Simple Asymmetric AKE

\[ K \leftarrow \{0, 1\}^k \]

\[ R \leftarrow \{0, 1\}^r \]

\[ C \leftarrow \mathcal{E}(pk_B, R\|K) \]

Decrypt and check for matching

**Question:** Find a CPA-secure scheme that breaks this