Public Key Encryption

Chanathip Namprempre

Department of Computer Science
Reed College

Agenda: Public Key Encryption

1. Basic math
2. Syntax of PKE
3. Security Definitions
4. Constructions

Outline

Basic math

- $\mathbf{Z}_N = \{0, \ldots, N-1\}$
- $\mathbf{Z}_N^* = \{x \in \mathbf{Z}_N | x \text{ and } N \text{ are co-prime.}\}$
- $\mathbf{Z}_N^* =$ set of invertible elements in $\mathbf{Z}_N$
- Easy operations modulo N: addition, multiplication, exponentiation, inversion

Suppose G is a group with operation $\cdot$.

- Order of $G = |G|$
- Order of $x \in G$:

  $$\mathrm{ord}_G(x) = |\langle x \rangle| = \{ \text{ smallest } a > 0 \text{ such that } x^a = 1 \text{ in } G\} \qquad .$$

  So,

  $$x^{\mathrm{ord}_G(x)} = 1.$$

- For any $x \in G$,

  $$\langle x \rangle = \{x^0, x^1, \ldots, x^{\mathrm{ord}_G(x)-1}\} \text{ is a subgroup of } G.$$

- For any subgroup S of G, $|S| \Big| |G|$.
- For any element $x \in G$, $\mathrm{ord}_G(x) \Big| |G|$
- For any element $x \in G$, $x^i = x^{i \bmod |G|}$

Let p be a prime.

- $\mathbf{Z}_p^* = \{1, 2, \ldots, p-1\}$
- $\mathbf{Z}_p^*$ is cyclic.
- Fermat's theorem: If p is prime, then $\forall x \in \mathbf{Z}_p^*$, $x^{p-1} \equiv 1$.

  Fermat's theorem can be used to test whether a number p is prime. If p is chosen at random, there's a very small chance that p would pass this test yet isn't prime.

- There's a generator $g \in \mathbf{Z}_p^*$ such that $\{1, g, g^2, g^3, \ldots, g^{p-2}\} = \mathbf{Z}_p^*$.
- Not everything in $\mathbf{Z}_p^*$ is a generator.
- Lagrange Thm: $\forall x \in \mathbf{Z}_p^*, \mathrm{ord}_p(x) \,\Big|\, p-1$

Recall definition

$\mathbf{Z}_N^* =$ set of invertible elements in $\mathbf{Z}_N$

Let N be an integer.

- Euler's phi function: $\phi(N) = |Z_N^*|$
- Euler's theorem: For any integer N, $\forall x \in \mathbf{Z}_N^*, x^{\phi(N)} = 1$.
- If $N = pq$ where p and q are distinct primes, then $\phi(N) = (p-1)(q-1)$.

Let N be an integer.

- Solving linear equations in $\mathbf{Z}_N$ is easy. For $a, b \in \mathbf{Z}_N$,
$$ax + b \equiv 0$$
Solution: $x \equiv -b \cdot a^{-1}$ in $\mathbf{Z}_N$. Use Euclidean algorithm.

- Solving higher degree polynomial in $\mathbf{Z}_N$ is more complicated, e.g.,
$$x^3 - 12 \equiv 0 \pmod{15}$$

Outline

RSA

Notation

Let $N, e \geq 1$ be integers.
The RSA function associated to N, e is $RSA_{N,e} : \mathbf{Z}_N^* \to \mathbf{Z}_N^*$ defined by

$$RSA_{N,e}(x) = x^e \bmod N \text{ for all } x \in \mathbf{Z}_N^* \ .$$

Claim

Let $N \geq 2$, $e, d \in \mathbf{Z}^*_{\phi(N)}$ be integers such that

$$ed \equiv 1 \pmod{\phi(N)} \ .$$

i.e., $[d = e^{-1}$ in $\mathbf{Z}^*_{\phi(N)}]$. Then,

$RSA_{N,e}$ is a permutation over $\mathbf{Z}_N^*$; $\quad RSA_{N,d}^{-1} = RSA_{N,e}$;
$RSA_{N,d}$ is a permutation over $\mathbf{Z}_N^*$; $\quad RSA_{N,e}^{-1} = RSA_{N,d}$.

RSA : $RSA_{N,e}$ and $RSA_{N,d}$ are inverses of each other

Let $x \in \mathbf{Z}_N^*$

Then,

$$\begin{aligned} RSA_{N,d}(RSA_{N,e}(x)) &\equiv (x^e)^d \\ &\equiv x^{ed \bmod \phi(N)} \\ &\equiv x^1 \equiv x \end{aligned}$$

The second equation holds because $\phi(N)$ is the order of the group $\mathbf{Z}_N^*$.

Similarly, we can show that $RSA_{N,e}(RSA_{N,d}(y)) = y$ for all $y \in \mathbf{Z}_N^*$.

RSA (cont.)

Notice

$RSA_{N,e}(\cdot)$ and $RSA_{N,d}(\cdot)$ are efficiently computable.

Intuition for security : *one-wayness of RSA*

Given N, e, y, it's hard to compute $RSA^{-1}_{N,e}(y)$ without d.

Modulus Generator

Definition

A modulus generator with associated security parameter $k \geq 2$ is a randomized algorithm taking no inputs & returning integers N, p, q such that

1. p, q are distinct, odd primes.
2. $N = pq$
3. $2^{k-1} \leq N < 2^k$

RSA Generator : K_{rsa}

Definition

An RSA generator with associated security parameter $k \geq 2$ is a randomized algorithm taking no inputs & returning $((N, e), (N, p, q, d))$ such that N, e, p, q, d are integers and

1. p, q are distinct, odd primes.
2. $N = pq$
3. $2^{k-1} \leq N < 2^k$
4. $e, d \in \mathbf{Z}^*_{\phi(N)}$
5. $ed \equiv 1 \pmod{\phi(N)}$

Outline

Syntax of PKE

Syntax

A public key encryption scheme $\mathsf{PKE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is a triple of algorithms.

alg	input	output	notation	maybe randomized?	maybe stateful?
$\mathcal{K}$	-	key pk, sk	$(pk, sk) \xleftarrow{\$} \mathcal{K}$	yes	no
$\mathcal{E}$	$(pk, sk) \in Keys(\mathsf{PKE})$ $M \in \{0,1\}^*$	ciphertext $C \in \{0,1\}^* \cup \{\perp\}$	$C \xleftarrow{\$} \mathcal{E}_{pk}(M)$	yes	yes
$\mathcal{D}$	$(pk, sk) \in Keys(\mathsf{PKE})$ $C \in \{0,1\}^*$	plaintext $M \in \{0,1\}^* \cup \{\perp\}$	$M \leftarrow \mathcal{D}_{sk}(C)$	no	no

Correctness

For all $(pk, sk) \in Keys(\mathsf{PKE})$ and all $M \in \{0,1\}^*$,

$$\Pr\left[\, C = \perp \text{ OR } \mathcal{D}_{sk}(C) = M \;:\; C \xleftarrow{\$} \mathcal{E}_{pk}(M) \,\right] = 1\,.$$

Outline

Privacy notion for PKE: Indistinguishability against CPA

Let $\mathsf{PKE} = (\mathsf{KG}, \mathcal{E}, \mathcal{D})$ be a PKE scheme, and let A be an adversary with access to an oracle.

Subroutine $\texttt{Initialize}$ $\quad b \xleftarrow{\$} \{0,1\}$; $(pk, sk) \xleftarrow{\$} \mathsf{KG}$ $\quad$Return pk Subroutine $\texttt{Enc}(M_0, M_1)$ $\quad$If $\lvert M_0\rvert \neq \lvert M_1\rvert$ then return $\perp$ $\quad$Return $\mathsf{Enc}_{pk}(M_b)$ Subroutine $\texttt{Finalize}(d)$ $\quad$Return $(d = b)$	Experiment $\mathbf{Exp}^{\mathrm{ind\text{-}cpa}}_{\mathsf{PKE}}(A)$ $\quad pk \xleftarrow{\$} \texttt{Initialize}$ $\quad d \xleftarrow{\$} A^{\texttt{Enc}}(pk)$ $\quad$Return $\texttt{Finalize}(d)$

ind-cpa advantage of A mounting a CPA against PKE:

$$\mathbf{Adv}^{\mathrm{ind\text{-}cpa}}_{\mathsf{PKE}}(A) = 2 \cdot \Pr\left[\, \mathbf{Exp}^{\mathrm{ind\text{-}cpa}}_{\mathsf{PKE}}(A) \Rightarrow \mathsf{true} \,\right] - 1 \; .$$

Privacy notion for PKE: Indistinguishability against CCA

Let $\mathsf{PKE} = (\mathsf{KG}, \mathcal{E}, \mathcal{D})$ be a PKE scheme, and let A be an adversary with access to an oracle.

Subroutine $\texttt{Initialize}$

$b \xleftarrow{\$} \{0,1\}$; $(pk, sk) \xleftarrow{\$} \mathsf{KG}$

$S \leftarrow \emptyset$; Return pk

Subroutine $\texttt{Enc}(M_0, M_1)$

If $|M_0| \neq |M_1|$ then return $\perp$

$C \xleftarrow{\$} \mathsf{Enc}(pk, M_b)$

$S \leftarrow S \cup \{C\}$; Return C

Subroutine $\texttt{Dec}(C)$

If $C \in S$ then return $\perp$

Return $\mathsf{Dec}(sk, C)$

Subroutine $\texttt{Finalize}(d)$

Return $(d = b)$

Experiment $\mathbf{Exp}_{\mathsf{PKE}}^{\mathrm{ind\text{-}cca}}(A)$

$\texttt{Initialize}$

$d \xleftarrow{\$} A^{\texttt{Enc},\texttt{Dec}}$

Return $\texttt{Finalize}(d)$

ind-cca advantage:

$$\mathbf{Adv}_{\mathsf{PKE}}^{\mathrm{ind\text{-}cca}}(A) = 2 \cdot \Pr\left[\mathbf{Exp}_{\mathsf{PKE}}^{\mathrm{ind\text{-}cca}}(A) \Rightarrow \mathsf{true}\right] - 1 .$$

Outline

ElGamal PKE modulo prime

Let p be a prime and g a generator of $\mathbf{Z}_p^*$. ElGamal PKE is $(\mathrm{KG}, \mathcal{E}, \mathcal{D})$ as follows.

Alg KG	**Alg** $\mathcal{E}_X(M)$	**Alg** $\mathcal{D}_x(Y, W)$
$x \xleftarrow{\$} \mathbf{Z}_{p-1}$	$y \xleftarrow{\$} \mathbf{Z}_{p-1}$; $Y \leftarrow g^y$	$K \leftarrow Y^x$
$X \leftarrow g^x$	$K \leftarrow X^y$	$M \leftarrow W \cdot K^{-1}$
Return (X, x)	$W \leftarrow K \cdot M$	Return M
	Return (Y, W)	

In $\mathbf{Z}_p^*$, ElGamal PKE is NOT IND-CPA secure.

Hint: Use $M_0 = g$ and $M_1 = 1$. What are the Jacobi symbols of these messages?

ElGamal PKE modulo prime

Let p be a prime and g a generator of $\mathbf{Z}_p^*$. ElGamal PKE is $(\mathrm{KG}, \mathcal{E}, \mathcal{D})$ as follows.

Alg KG	**Alg** $\mathcal{E}_X(M)$	**Alg** $\mathcal{D}_x(Y, W)$
$x \xleftarrow{\$} \mathbf{Z}_{p-1}$	$y \xleftarrow{\$} \mathbf{Z}_{p-1}$; $Y \leftarrow g^y$	$K \leftarrow Y^x$
$X \leftarrow g^x$	$K \leftarrow X^y$	$M \leftarrow W \cdot K^{-1}$
Return (X, x)	$W \leftarrow K \cdot M$	Return M
	Return (Y, W)	

In $\mathbf{Z}_p^*$, ElGamal PKE is NOT IND-CPA secure.

Hint: Use $M_0 = g$ and $M_1 = 1$. What are the Jacobi symbols of these messages?

Basic math intermission: Quadratic Residue

Let p be an odd prime, and let g be a generator for $\mathbf{Z}_p^*$.

In $\mathbf{Z}_p^*$, the map $x \to x^2$ is a 2-to-1 function.

QR

All these statements are equivalent.

- $x \in \mathbf{Z}_p$ is a quadratic residue (QR)
- x has a square root in $\mathbf{Z}_p$
- Legendre symbol of x over $p = 1$

Legendre/Jacobi symbol of x over p

$J_p(x) = x^{(p-1)/2} \pmod{p}$

Try with $\mathbf{Z}_{11}^*$ and $g = 2$. For each $x \in \mathbf{Z}_{11}^*$, compute x^2 and $J_p(x)$.

i	1	2	3	4	5	6	7	8	9	10
2^i	2	**4**	8	**5**	10	**9**	7	**3**	6	**1**

Quadratic Residue (cont.)

Example: $\mathbf{Z}_{11}^*$ with $g = 2$.

i	1	2	3	4	5	6	7	8	9	10
2^i	2	**4**	8	**5**	10	**9**	7	**3**	6	**1**

Facts

$\forall x, y \in \mathbf{Z}_p^*, a, b \in \mathbf{Z}_{p-1}$,

$$J_p(x) \in \{1, -1\}$$
$$x \text{ is a QR iff } J_p(x) = 1$$
$$J_p(xy) = J_p(x) \cdot J_p(y)$$
$$J_p(x^{-1}) = J_p(x)$$
$$J_p(g^{ab}) = 1 \text{ iff } J_p(g^a) = 1 \text{ or } J_p(g^b) = 1$$

Quadratic Residue (cont.)

Try it with $p = 13$

a	1	2	3	4	5	6	7	8	9	10	11	12
$a^2 \bmod 13$												
$J_{13}(a)$												
a^{-1}												
$J_{13}(a^{-1})$												

ElGamal PKE modulo prime: NOT IND-CPA

If we ask the encryption oracle $\texttt{Enc}(g, 1)$ and call what we get back (Y, W_0) if it's a left oracle and (Y, W_1) if it's a right oracle, then we have

$$\begin{aligned} X &= g^x \\ Y &= g^y \\ W_0 &= g^{xy} \cdot g \\ W_1 &= g^{xy} \cdot 1 \end{aligned}$$

$J_p(X)$	$J_p(Y)$	$J_p(g^{xy})$	$J_p(W_0)$	$J_p(W_1)$
1	1	1	-1	1
-1	1			
1	-1			
-1	-1			

Algorithm $A(X)$:

$(Y, W) \xleftarrow{\$} \texttt{Enc}(g, 1)$; $J^* \leftarrow J_p(W)$

switch $(J_p(X), J_p(Y))$:

case $(1, 1)$: case $(-1, 1)$: case $(1, -1)$:

If $J^* = -1$ then return ? else return ?

case $(-1, -1)$:

If $J^* = -1$ then return ? else return ?

ElGamal PKE is ok in certain other groups

However, ElGamal PKE is secure in any group where DDH is hard.
e.g., prime-order subgroups of $\mathbf{Z}_p^*$, elliptic curve groups of prime order

DHIES PKE

Let $G = \langle g \rangle$ be a group of order m, $H : \{0,1\}^* \to \{0,1\}^k$ be a hash function, and $\mathrm{AE} = (\mathrm{KG}_{ae}, \mathcal{E}_{ae}, \mathcal{D}_{ae})$ be an AE scheme with k-bit keys. DHIES PKE is $(\mathrm{KG}, \mathcal{E}, \mathcal{D})$ as follows.

Alg KG	**Alg** $\mathcal{E}_X(M)$	**Alg** $\mathcal{D}_x(Y, C)$
$x \xleftarrow{\$} \mathbf{Z}_m$	$y \xleftarrow{\$} \mathbf{Z}_m$; $Y \leftarrow g^y$	$Z \leftarrow Y^x$
$X \leftarrow g^x$	$Z \leftarrow X^y$	$K \leftarrow H(Y \Vert Z)$
Return (X, x)	$K \leftarrow H(Y \Vert Z)$	$M \leftarrow \mathcal{D}_{ae}(K, C)$
	$C \xleftarrow{\$} \mathcal{E}_{ae}(K, M)$	Return M
	Return (Y, C)	

Textbook RSA

Textbook RSA is insecure!

Alg KG	Alg $\mathcal{E}_{pk}(M)$	Alg $\mathcal{D}_{sk}(C)$
$(N, p, q, e, d) \xleftarrow{\$} K_{rsa}$	$C \leftarrow M^e \bmod N$	$M \leftarrow C^d \bmod N$
$pk \leftarrow (N, e)$	Return C	Return M
$sk \leftarrow (N, d)$		
Return (pk, sk)		

Adversary gets $C = M^e \pmod N$. Suppose M is 64 bits long.
If $M = M_1 \cdot M_2$ where $M_1, M_2 < 2^{34}$ (This happens with prob. approx 20%), then

$$C/M_1^e = M_2^e \pmod N$$

Meet in the middle attack:

1. Build table $C/1^e, C/2^e, \ldots C/2^{34e}$
2. For $M_2 = 0, \ldots, 2^{34}$, test if M_2^e is in table.
3. Output matching $M = M_1 \cdot M_2$

Time: much less than 2^{64}

Source: Dan Boneh's coursera slides

Textbook RSA

Textbook RSA is insecure!

Alg KG	**Alg** $\mathcal{E}_{pk}(M)$	**Alg** $\mathcal{D}_{sk}(C)$
$(N, p, q, e, d) \xleftarrow{\$} K_{rsa}$	$C \leftarrow M^e \bmod N$	$M \leftarrow C^d \bmod N$
$pk \leftarrow (N, e)$	Return C	Return M
$sk \leftarrow (N, d)$		
Return (pk, sk)		

Adversary gets $C = M^e \pmod N$. Suppose M is 64 bits long.
If $M = M_1 \cdot M_2$ where $M_1, M_2 < 2^{34}$ (This happens with prob. approx 20%), then

$$C/M_1^e = M_2^e \pmod N$$

Meet in the middle attack:

1. Build table $C/1^e, C/2^e, \ldots C/2^{34e}$
2. For $M_2 = 0, \ldots, 2^{34}$, test if M_2^e is in table.
3. Output matching $M = M_1 \cdot M_2$

Time: much less than 2^{64}

Source: Dan Boneh's coursera slides

Textbook RSA

Textbook RSA is insecure!

Alg KG	**Alg** $\mathcal{E}_{pk}(M)$	**Alg** $\mathcal{D}_{sk}(C)$
$(N, p, q, e, d) \xleftarrow{\$} K_{rsa}$	$C \leftarrow M^e \bmod N$	$M \leftarrow C^d \bmod N$
$pk \leftarrow (N, e)$	Return C	Return M
$sk \leftarrow (N, d)$		
Return (pk, sk)		

Adversary gets $C = M^e \pmod N$. Suppose M is 64 bits long.
If $M = M_1 \cdot M_2$ where $M_1, M_2 < 2^{34}$ (This happens with prob. approx 20%), then

$$C/M_1^e = M_2^e \pmod N$$

Meet in the middle attack:

1. Build table $C/1^e, C/2^e, \ldots C/2^{34e}$
2. For $M_2 = 0, \ldots, 2^{34}$, test if M_2^e is in table.
3. Output matching $M = M_1 \cdot M_2$

Time: much less than 2^{64}

Source: Dan Boneh's coursera slides

Let $H : \{0,1\}^* \to \{0,1\}^k$ be a hash function, and $\mathrm{AE} = (\mathrm{KG}_{ae}, \mathcal{E}_{ae}, \mathcal{D}_{ae})$ be an AE scheme with k-bit keys. SRSA PKE is $(\mathrm{KG}, \mathcal{E}, \mathcal{D})$ as follows.

Alg KG	**Alg** $\mathcal{E}_{pk}(M)$	**Alg** $\mathcal{D}_{sk}(C_1, C_2)$
$(N, p, q, e, d) \xleftarrow{\$} K_{rsa}$	$x \xleftarrow{\$} \mathbf{Z}_N^*$	$x \leftarrow C_1^d \bmod N$
$pk \leftarrow (N, e)$	$K \leftarrow H(x)$	$K \leftarrow H(x)$
$sk \leftarrow (N, d)$	$C_1 \leftarrow x^e \bmod N$	$M \leftarrow \mathcal{D}_{ae}(K, C_2)$
Return (pk, sk)	$C_2 \leftarrow \mathcal{E}_{ae}(K, M)$	Return M
	Return (C_1, C_2)	

Hybrid encryption

Conceptually, SRSA follows a common paradigm:

PKE = KEM + DEM

1. Use a trapdoor function (TDF) and a hash to encapsulate an ephemeral symmetric key
2. Use AE to encapsulate the payload

Alg KG
$(N, p, q, e, d) \xleftarrow{\$} K_{rsa}$
$pk \leftarrow (N, e)$
$sk \leftarrow (N, d)$
Return (pk, sk)

Alg $\mathcal{E}_{pk}(M)$
$x \xleftarrow{\$} \mathbf{Z}_N^*$
$K \leftarrow H(x)$
$C_1 \leftarrow x^e \bmod N$
$C_2 \leftarrow \mathcal{E}_{ae}(K, M)$
Return (C_1, C_2)

Alg $\mathcal{D}_{sk}(C_1, C_2)$
$x \leftarrow C_1^d \bmod N$
$K \leftarrow H(x)$
$M \leftarrow \mathcal{D}_{ae}(K, C_2)$
Return M

Note: KEM schemes are actually defined slightly differently from what's shown here. We use the above presentation for simplicity.

One-wayness of RSA against known-exponent attacks

Definition

Let K_{rsa} be an RSA generator with security parameter k.
Let A be an algorithm.

Experiment $\mathbf{Exp}^{\mathrm{ow\text{-}kea}}_{K_{rsa}}(A)$

$((N,e),(N,p,q,d)) \stackrel{\$}{\leftarrow} K_{rsa}$

$x \stackrel{\$}{\leftarrow} \mathbf{Z}_N^*;\ y \leftarrow x^e \bmod N$

$x' \stackrel{\$}{\leftarrow} A(N,e,y)$

If $x = x'$ then 1 else 0

$$\mathbf{Adv}^{\mathrm{ow\text{-}kea}}_{K_{rsa}}(A) = \Pr\left[\ \mathbf{Exp}^{\mathrm{ow\text{-}kea}}_{K_{rsa}}(A) = 1\ \right] .$$

One-wayness of RSA against chosen-exponent attacks

Definition

Let K_{mod} be an modulus generator with security parameter k.
Let A be an algorithm.

Experiment $\mathbf{Exp}^{\mathrm{ow\text{-}cea}}_{K_{mod}}(A)$

$(N, p, q) \xleftarrow{\$} K_{mod}$

$y \xleftarrow{\$} \mathbf{Z}_N^*$

$(x, e) \xleftarrow{\$} A(N, y)$

If $x^e \equiv y \pmod{N}$ and $e > 1$ then return 1 else return 0

$$\mathbf{Adv}^{\mathrm{ow\text{-}cea}}_{K_{mod}}(A) = \Pr\left[\,\mathbf{Exp}^{\mathrm{ow\text{-}cea}}_{K_{mod}}(A) = 1\,\right] .$$

PKCS1 encryption

PKCS1 padding (02 is the mode number):

$$\underbrace{\boxed{02}\boxed{\quad\text{random pad}\quad}\boxed{FF}\boxed{\qquad\qquad\text{msg}\qquad\qquad}}_{\text{RSA-encrypted}}$$

- The entire thing is the value that gets RSA-encrypted.
- "02" is written as a 16-bit binary string.
- "random pad" doesn't contain FF.
- Widely deployed

<u>Bleichenbacher attack</u>: An attacker tests to see if 16 MSBs of plaintext is 02.

Bleichenbacher attack (simplified)

Bleichenbacher attack uses the server as a padding oracle.

- Success: the first two bytes are 02.
- Failure: the first two bytes are not 02.

Simplified attack:

- Suppose $N = 2^n$
- Suppose instead of revealing whether the first 2 bytes are 02, the server reveals whether the MSB is 1.
- Suppose adversary snoops a ciphertext C
- Adversary sends C and gets MSB
- Adversary sends $2^e C$ and gets 2nd most MSB
  [$2^e C = (2M)^e$ so we shift M to the left 1 position]
- adversary sends $4^e C$ and gets 3rd most MSB
  [$4^e C = (4M)^e$ so we shift M to the left 2 positions]
- ...

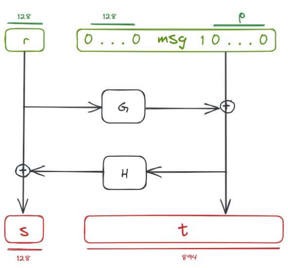
Preventing Bleichenbacher attack: HTTPS

Bleichenbacher attack uses the server as a padding oracle.

So for HTTPS (RFC 5246):

1. Generate a random string R of 46 bytes
2. Decrypt the ciphertext to get M
3. If PKCS1 padding check fails for M, then the decryption is R.

OAEP: Optimal Asymmetric Encryption Padding [BR94]



Theorem

If RSA is a trapdoor permutation, then RSA-OAEP is CCA secure in the random oracle model.

- OAEP+ replaces $10 \cdots 0$ with $W(m, r)$ where W is a hash function. This works for any trapdoor permutation, not just RSA.
- SAEP+ replaces $10 \cdots 0$ with $W(m, r)$ where W is a hash function and removes G. This works for RSA.