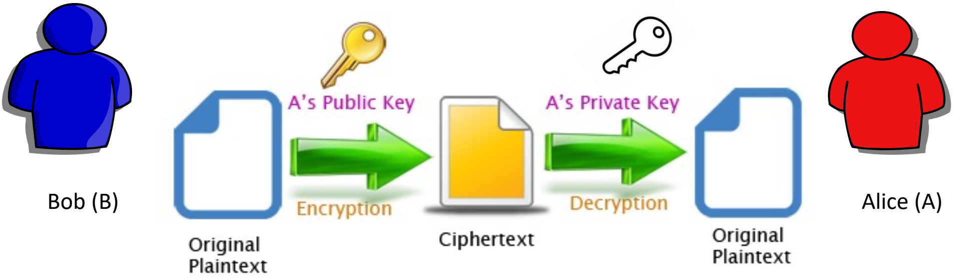
Public-Key Encryption



Bob (B)

Original Plaintext

A’s Public Key

Encryption

Ciphertext

A’s Private Key

Decryption

Original Plaintext

Alice (A)

Algorithms

A Public-Key Encryption scheme consists of three algorithms $PKE = (\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec})$

- $(pk, sk) \leftarrow \mathsf{Keygen}(1^n)$: Keygen generates the keypair (pk, sk). pk is the public-key. sk is the secret-key.

- $C \leftarrow \mathsf{Enc}(pk, m; r)$: The randomized encryption algorithm Enc takes the public-key pk and the message $m \in \mathcal{M}$ as input (alongwith a random string r), and outputs a ciphertext C.

- $m' \leftarrow \mathsf{Dec}(sk, C)$: The decryption algorithm Dec takes the secret key sk and the ciphertext C. The output is a (candidate) plaintext m.

Properties

Messages encrypted using a public-key should be retrieved when decrypted using the corresponding secret key

Correctness

For all $(pk, sk) \leftarrow \mathsf{Keygen}(1^n)$, for all message $m \in \mathcal{M}$, for all random string r, it should hold that

$$\mathsf{Dec}\left(sk, \mathsf{Enc}(pk, m; r)\right) = m$$

Properties

Messages encrypted using a public-key should be retrieved when decrypted using the corresponding secret key

Correctness

For all $(pk, sk) \leftarrow \mathsf{Keygen}(1^n)$, for all message $m \in \mathcal{M}$, for all random string r, it should hold that

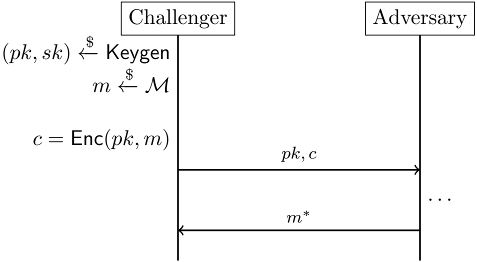
$$\mathsf{Dec}\left(sk, \mathsf{Enc}(pk, m; r)\right) = m$$

Note: Modern systems sometimes allow little correctness errors.

Security

One-way Security

The adversary can not find the message corresponding to a randomly generated ciphertext



Challenger

Adversary

$(pk, sk) \xleftarrow{\$} \mathsf{Keygen}$

$m \xleftarrow{\$} \mathcal{M}$

$c = \mathsf{Enc}(pk, m)$

pk, c

$\dots$

m^*

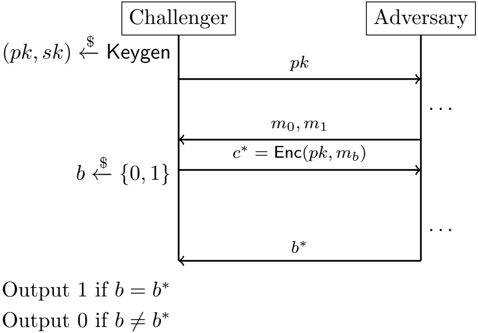
Output 1 if $m = m^*$

Output 0 if $m \neq m^*$

Security

Indistinguishability under Chosen Plaintext Attack

The adversary can not distinguish between encryptions of two messages, even when the adversary themselves choose the messages.



Challenger

Adversary

$(pk, sk) \xleftarrow{\$} \mathsf{Keygen}$

pk

$\dots$

m_0, m_1

$c^* = \mathsf{Enc}(pk, m_b)$

$b \xleftarrow{\$} \{0, 1\}$

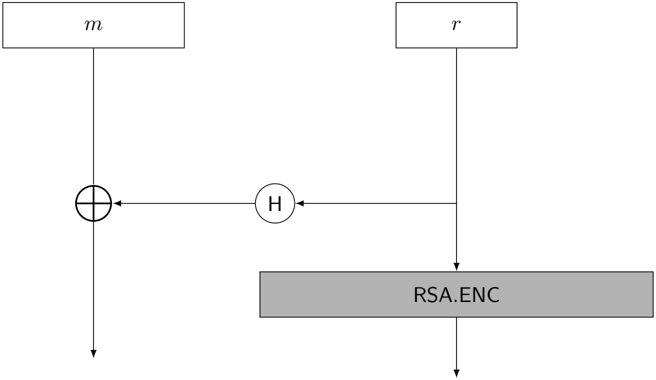
$\dots$

b^*

Output 1 if $b = b^*$

Output 0 if $b \neq b^*$

IND-CPA secure Constructions: Encryption using RSA



Intuitive Idea

One time pad with a randomly chosen key. The key is transformed using RSA function. H ensures that a random string is xored with m.

Can we use RSA directly for one-way security

- No. RSA function with small message and small e is insecure:
  - Suppose $p, q > 10^6$. Thus $N = pq > 10^{12}$
  - Suppose $e = 7$ and message is $m = 5$.
  - We compute $C = m^e \mod N = 5^7 \mod N = 78125 \mod N = 78125$

Can we use RSA directly for one-way security

- No. RSA function with small message and small e is insecure:
  - Suppose $p, q > 10^6$. Thus $N = pq > 10^{12}$
  - Suppose $e = 7$ and message is $m = 5$.
  - We compute $C = m^e \mod N = 5^7 \mod N = 78125 \mod N = 78125$
- Ciphertext is small, so $\mod N$ has no effect. $C = m^e$
  - $\log_{10} C = e \log_{10} m \implies m \approx 10^{\frac{\log C}{e}}$

Can we use RSA directly for one-way security

- No. RSA function with small message and small e is insecure:
  - Suppose $p, q > 10^6$. Thus $N = pq > 10^{12}$
  - Suppose $e = 7$ and message is $m = 5$.
  - We compute $C = m^e \mod N = 5^7 \mod N = 78125 \mod N = 78125$
- Ciphertext is small, so $\mod N$ has no effect. $C = m^e$
  - $\log_{10} C = e \log_{10} m \implies m \approx 10^{\frac{\log C}{e}}$
  - In our example $\log 78125 = 4.892$. Thus $\frac{\log C}{e} == 0.698$.

Can we use RSA directly for one-way security

- No. RSA function with small message and small e is insecure:
  - Suppose $p, q > 10^6$. Thus $N = pq > 10^{12}$
  - Suppose $e = 7$ and message is $m = 5$.
  - We compute $C = m^e \mod N = 5^7 \mod N = 78125 \mod N = 78125$
- Ciphertext is small, so $\mod N$ has no effect. $C = m^e$
  - $\log_{10} C = e \log_{10} m \implies m \approx 10^{\frac{\log C}{e}}$
  - In our example $\log 78125 = 4.892$. Thus $\frac{\log C}{e} == 0.698$.
  - We retrieve $m \approx 10^{0.698} = 4.98$. Rounding off, we get $m = 5$.

Padding:PKCS $\#1v1.5$

Randomized Padding

$y = 0\text{x}00||0\text{x}02||r||0\text{x}00||m$
$c = y^e \mod N$

Post Decryption Processing in SSL (pre 1998)

$y = c^d \mod N$
Check first two bytes of y. If $y \neq 0\text{x}00||0\text{x}02|| \ldots$ output **Bad Format**
Else output m

Padding:PKCS $\#1v1.5$

Randomized Padding

$y = 0\text{x}00||0\text{x}02||r||0\text{x}00||m$
$c = y^e \mod N$

Post Decryption Processing in SSL (pre 1998)

$y = c^d \mod N$
Check first two bytes of y. If $y \neq 0\text{x}00||0\text{x}02|| \ldots$ output **Bad Format**
Else output m

Observation

$2.2^{8(k-2)} \leq y \mod N < 3.2^{8(k-2)}$ where N is a k-byte number.

Chosen Ciphertext Attacks: Bleichenbacher Attack

- We are given a ciphertext $c = y^e \mod N$.
- Compute $c' = c.s^e \mod N = (y.s)^e \mod N$ for a *suitable* s.
- Check if the server accepts c'.
- If yes, then we know first two bytes of $y.s$ is $0\text{x}00||0\text{x}02$

Chosen Ciphertext Attacks: Bleichenbacher Attack

- We are given a ciphertext $c = y^e \mod N$.
- Compute $c' = c.s^e \mod N = (y.s)^e \mod N$ for a *suitable* s.
- Check if the server accepts c'.
- If yes, then we know first two bytes of $y.s$ is $0\text{x}00||0\text{x}02$
- Recall N is a k-byte number
  - $2.2^{8(k-2)} \leq ys \mod N < 3.2^{8(k-2)}$ and
  - $2.2^{8(k-2)} \leq y \mod N < 3.2^{8(k-2)}$

Chosen Ciphertext Attacks: Bleichenbacher Attack

- We are given a ciphertext $c = y^e \mod N$.
- Compute $c' = c.s^e \mod N = (y.s)^e \mod N$ for a *suitable* s.
- Check if the server accepts c'.
- If yes, then we know first two bytes of $y.s$ is $0\text{x}00||0\text{x}02$
- Recall N is a k-byte number
  - $2.2^{8(k-2)} \leq ys \mod N < 3.2^{8(k-2)}$ and
  - $2.2^{8(k-2)} \leq y \mod N < 3.2^{8(k-2)}$
  - $y \mod N < 3.2^{8(k-2)}/s$

Chosen Ciphertext Attacks: Bleichenbacher Attack

- We are given a ciphertext $c = y^e \mod N$.
- Compute $c' = c.s^e \mod N = (y.s)^e \mod N$ for a *suitable* s.
- Check if the server accepts c'.
- If yes, then we know first two bytes of $y.s$ is $0x00||0x02$
- Recall N is a k-byte number
  - $2.2^{8(k-2)} \leq ys \mod N < 3.2^{8(k-2)}$ and
  - $2.2^{8(k-2)} \leq y \mod N < 3.2^{8(k-2)}$
  - $y \mod N < 3.2^{8(k-2)}/s$
- Repeat the procedure with $s' > s$.
- Choose s via binary search.

Mitigating Bleichenbacher Attack in TLS

Internet Engineering Task Force RFC5246

“In any case, a TLS server MUST NOT generate an alert if processing an RSA-encrypted premaster secret message fails, or the version number is not as expected. Instead, it MUST continue the handshake with a randomly generated premaster secret. It may be useful to log the real cause of failure for troubleshooting purposes; however, care must be taken to avoid leaking the information to an attacker”

Mitigating Bleichenbacher Attack in TLS

Internet Engineering Task Force RFC5246

“In any case, a TLS server MUST NOT generate an alert if processing an RSA-encrypted premaster secret message fails, or the version number is not as expected. Instead, it MUST continue the handshake with a randomly generated premaster secret. It may be useful to log the real cause of failure for troubleshooting purposes; however, care must be taken to avoid leaking the information to an attacker”

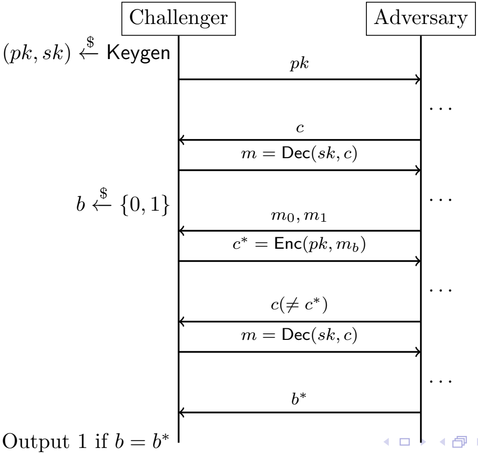
Other Solution

Use a scheme secure against Chosen Ciphertext Attack

Security

Indistinguishability under Chosen Ciphertext Attack

The adversary can not distinguish between encryptions of two messages, even when the adversary themselves choose the messages and the adversary could get decryptions of some *chosen* ciphertexts.



Challenger

Adversary

$(pk, sk) \xleftarrow{\$}$ Keygen

pk

...

c

$m = \mathsf{Dec}(sk, c)$

$b \xleftarrow{\$} \{0, 1\}$

...

m_0, m_1

$c^* = \mathsf{Enc}(pk, m_b)$

...

$c (\neq c^*)$

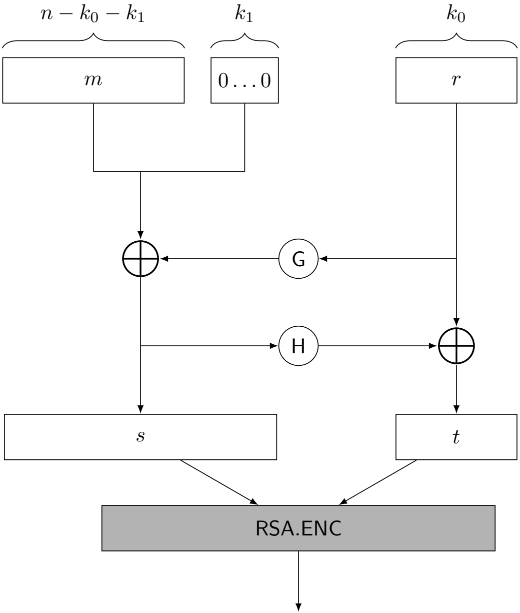
$m = \mathsf{Dec}(sk, c)$

...

b^*

Output 1 if $b = b^*$

IND-CCA secure Constructions: OAEP


$n - k_0 - k_1$
k_1
k_0
m
$0 \ldots 0$
r
G
H
s
t
RSA.ENC