Public-Key Encryption and RSA Encryption

Yu Zhang

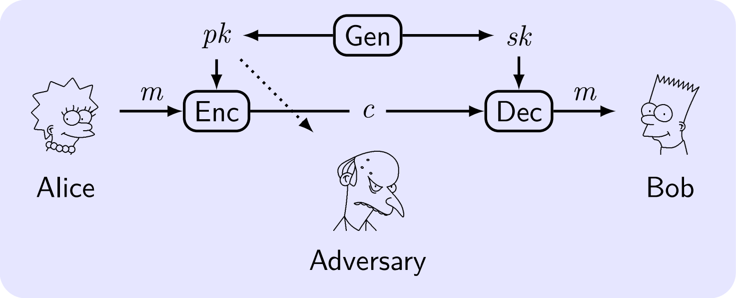
HIT/CST/NIS

Cryptography, Spring, 2014

Outline

Content

Definitions



- **Key-generation** algorithm: $(pk, sk) \leftarrow$ Gen, key length $\geq n$.
- **Plaintext space** $\mathcal{M}$ is associated with pk.
- **Encryption** algorithm: $c \leftarrow \mathsf{Enc}_{pk}(m)$.
- **Decryption** algorithm: $m := \mathsf{Dec}_{sk}(c)$, or outputs $\perp$.
- **Requirement**: $\Pr[\mathsf{Dec}_{sk}(\mathsf{Enc}_{pk}(m)) = m] \geq 1 - \mathsf{negl}(n)$.

Security against Eavesdroppers = CPA

The eavesdropping indistinguishability experiment $\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$:

1. $(pk, sk) \leftarrow \mathsf{Gen}(1^n)$.
2. $\mathcal{A}$ **is given input** $\mathbf{pk}$ **and so oracle access to** $\mathsf{Enc}_{\mathbf{pk}}(\cdot)$, outputs m_0, m_1 of the same length.
3. $b \leftarrow \{0,1\}$. $c \leftarrow \mathsf{Enc}_{pk}(m_b)$ (challenge) is given to $\mathcal{A}$.
4. $\mathcal{A}$ **continues to have access to** $\mathsf{Enc}_{\mathbf{pk}}(\cdot)$ and outputs b'.
5. If $b' = b$, $\mathcal{A}$ succeeded $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi} = 1$, otherwise 0.

Definition 1

Π is **CPA-secure** if $\forall$ PPT $\mathcal{A}$, $\exists$ negl such that

$$\Pr\left[\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(n).$$

Security Properties of Public-Key Encryption

Theorem 2

No deterministic public-key encryption scheme is secure in the presence of an eavesdropper.

Proposition 3

If Π is secure in the presence of an eavesdropper, then Π also is CPA-secure.

Theorem 4

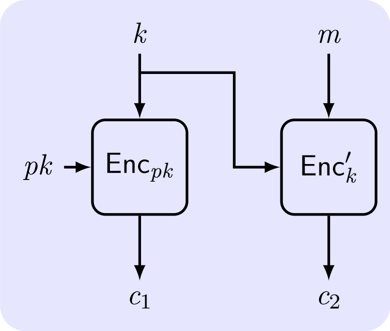
If Π is secure in the presence of an eavesdropper, then Π is secure for multiple encryptions.

Proposition 5

Perfectly-secret public-key encryption is impossible.

Construction of Hybrid Encryption

To speed up the encryption of long message, use private-key encryption Π' in tandem with public-key encryption Π.



Construction 6

$\Pi^{\mathsf{hy}} = (\mathsf{Gen}^{\mathsf{hy}}, \mathsf{Enc}^{\mathsf{hy}}, \mathsf{Dec}^{\mathsf{hy}})$:

- $\mathsf{Gen}^{\mathsf{hy}}$: $(pk, sk) \leftarrow \mathsf{Gen}(1^n)$.
- $\mathsf{Enc}^{\mathsf{hy}}$: pk *and* m.
  1. $k \leftarrow \{0,1\}^n$.
  2. $c_1 \leftarrow \mathsf{Enc}_{pk}(k)$, $c_2 \leftarrow \mathsf{Enc}'_k(m)$.
- $\mathsf{Dec}^{\mathsf{hy}}$: sk *and* $\langle c_1, c_2 \rangle$.
  1. $k := \mathsf{Dec}_{sk}(c_1)$.
  2. $m := \mathsf{Dec}'_k(c_2)$.

Hybrid encryption is a public-key encryption without any secret key in advance.

Security of Hybrid Encryption

Theorem 7

If Π is a CPA-secure public-key encryption scheme and Π' is a private-key encryption scheme that has indistinguishable encryptions in the presence of an eavesdropper, then Π^{hy} is a CPA-secure public-key encryption scheme.

$$
\begin{tikzcd}
\langle pk, \mathsf{Enc}_{pk}(k), \mathsf{Enc}'_k(m_0)\rangle \arrow[r, leftrightarrow, dotted, "\text{(by transitivity)}"] \arrow[d, leftrightarrow, "\text{(by security of } \Pi\text{)}"] & \langle pk, \mathsf{Enc}_{pk}(k), \mathsf{Enc}'_k(m_1)\rangle \arrow[d, leftrightarrow, "\text{(by security of } \Pi\text{)}"'] \\
\langle pk, \mathsf{Enc}_{pk}(0^n), \mathsf{Enc}'_k(m_0)\rangle \arrow[r, leftrightarrow, "\text{(by security of } \Pi'\text{)}"'] & \langle pk, \mathsf{Enc}_{pk}(0^n), \mathsf{Enc}'_k(m_1)\rangle
\end{tikzcd}
$$

Content

Scenarios of CCA in Public-Key Setting

1. An adversary $\mathcal{A}$ observes the ciphertext c sent by $\mathcal{S}$ to $\mathcal{R}$.
2. $\mathcal{A}$ send c' to $\mathcal{R}$ in the name of $\mathcal{S}$ or its own.
3. $\mathcal{A}$ infer m from the decryption of c' to m'.

Scenarios

- **login to on-line bank with the password**: trial-and-error, learn info from the feedback of bank.
- **reply an e-mail with the quotation of decrypted text**.
- **malleability of ciphertexts**: e.g. doubling others' bids at an auction.

Definition of Security Against CCA/CCA2

The CCA/CCA2 indistinguishability experiment $\mathsf{PubK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(n)$:

1. $(pk, sk) \leftarrow \mathsf{Gen}(1^n)$.
2. $\mathcal{A}$ **is given input** pk **and oracle access to** $\mathsf{Dec}_{sk}(\cdot)$, outputs m_0, m_1 of the same length.
3. $b \leftarrow \{0,1\}$. $c \leftarrow \mathsf{Enc}_{pk}(m_b)$ is given to $\mathcal{A}$.
4. $\mathcal{A}$ **have access to** $\mathsf{Dec}_{sk}(\cdot)$ **except for** c **in CCA2**[1] and outputs b'.
5. If $b' = b$, $\mathcal{A}$ succeeded $\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi} = 1$, otherwise 0.

Definition 8

Π has **CCA/CCA2-secure** if $\forall$ PPT $\mathcal{A}$, $\exists$ negl such that

$$\Pr\left[\mathsf{PubK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(n) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(n).$$

[1] CCA is also called Lunchtime attacks; CCA2 is also called Adaptive CCA.

State of the Art on CCA2-secure Encryption

- **Zero-Knowledge Proof**: complex, and impractical. (e.g., Dolev-Dwork-Naor)
- **Random Oracle** model: efficient, but not realistic (to consider CRHF as RO). (e.g., RSA-OAEP and Fujisaki-Okamoto)
- **DDH(Decisional Diffie-Hellman assumption) and UOWHF(Universal One-Way Hashs Function)**: x2 expansion in size, but security proved w/o RO or ZKP (e.g., Cramer-Shoup system).

CCA2-secure implies Plaintext-aware: an adversary cannot produce a valid ciphertext without "knowing" the plaintext.

Open problem

Constructing a CCA2-secure scheme based on RSA problem as efficient as "Textbook RSA".

Private Key Encryption vs. Public Key Encryption

	Private Key	**Public Key**
Secret Key	both parties	receiver
Weakest Attack	Eav	CPA
Probabilistic	CPA/CCA	always
Assumption against CPA	OWF	TDP
Assumption against CCA	OWF	TDP+RO
Efficiency	fast	slow

Content

The RSA Problem

Recall group exponentiation on $\mathbb{Z}_N^*$

Define function $f_e : \mathbb{Z}_N^* \to \mathbb{Z}_N^*$ by $f_e(x) = [x^e \bmod N]$.
If $\gcd(e, \phi(N)) = 1$, then f_e is a permutation.
If $d = [e^{-1} \bmod \phi(N)]$, then f_d is the inverse of f_e.
e**'th root of** c: $g^e = c$, $g = c^{1/e} = c^d$.

Idea: factoring is hard

$\implies$ for $N = pq$, finding p, q is hard

$\implies$ computing $\phi(N) = (p-1)(q-1)$ is hard

$\implies$ computations modulo $\phi(N)$ is not available

There is a gap.

$\implies$ **RSA problem** [Rivest, Shamir, and Adleman] is hard:
Given $y \in \mathbb{Z}_N^*$, compute y^{-e}, e^{th}-root of y modulo N.

Open problem

RSA problem is easier than factoring?

Generating RSA Problem

Algorithm 1: GenRSA

input : Security parameter 1^n

output: N, e, d

$(N, p, q) \leftarrow$ GenModulus(1^n)

$\phi(N) := (p-1)(q-1)$

find e such that $\gcd(e, \phi(N)) = 1$

compute $d := [e^{-1} \bmod \phi(N)]$

return N, e, d

The RSA Assumption

The RSA experiment $\mathsf{RSAinv}_{\mathcal{A},\mathsf{GenRSA}}(n)$:

1. Run $\mathsf{GenRSA}(1^n)$ to obtain (N, e, d).
2. Choose $y \leftarrow \mathbb{Z}_N^*$.
3. $\mathcal{A}$ is given N, e, y, and outputs $x \in \mathbb{Z}_N^*$.
4. $\mathsf{RSAinv}_{\mathcal{A},\mathsf{GenRSA}}(n) = 1$ if $x^e \equiv y \pmod{N}$, and 0 otherwise.

Definition 9

RSA problem is hard relative to GenRSA if $\forall$ PPT algorithms $\mathcal{A}$, $\exists$ negl such that

$$\Pr[\mathsf{RSAinv}_{\mathcal{A},\mathsf{GenRSA}}(n) = 1] \leq \mathsf{negl}(n).$$

Content

“Textbook RSA”

Construction 10

- Gen: *on input* 1^n *run* GenRSA(1^n) *to obtain* N, e, d.
  $pk = \langle N, e \rangle$ *and* $sk = \langle N, d \rangle$.
- Enc: *on input* pk *and* $m \in \mathbb{Z}_N^*$, $c := [m^e \bmod N]$.
- Dec: *on input* sk *and* $m \in \mathbb{Z}_N^*$, $m := [c^d \bmod N]$.

Insecurity

Since the “textbook RSA” is deterministic, it is insecure with respect to any of the definitions of security we have proposed.

RSA Implementation Issues

- **Encoding binary strings as elements of $\mathbb{Z}_N^*$**: $\ell = \|N\|$. Any binary string m of length $\ell - 1$ can be viewed as an element of Z_N. Although m may not be in Z_N^*, RSA still works.
- **Choice of** e: Either $e = 3$ or a small d are bad choices. Recommended value: $e = 65537 = 2^{16} + 1$
- **Using the Chinese remainder theorem**: to speed up the decryption.

$$[c^d \bmod N] \leftrightarrow ([c^d \bmod p], [c^d \bmod q]).$$

Assume that exponentiation modulo a v-bit integer takes v^3 operations. RSA decryption takes $(2n)^3 = 8n^3$, whereas using CRT takes $2n^3$.

Example of "Textbook RSA"

$N = 253$, $p = 11$, $q = 23$, $e = 3$, $d = 147$, $\phi(N) = 220$.

$m = 0111001 = 57$.
Encryption: $250 := [57^3 \bmod 253]$.
Decryption: $57 := [250^{147} \bmod 253]$.

Using CTR,

$$[250^{[147 \bmod 10]} \bmod 11] = [8^7 \bmod 11] = 2$$

$$[250^{[147 \bmod 22]} \bmod 23] = [20^{15} \bmod 23] = 11$$

$57 \leftrightarrow (2, 11)$.

Attacks on "Textbook RSA" with a small e

Small e and small m make modular arithmetic useless.

- If $e = 3$ and $m < N^{1/3}$, then $c = m^3$ and $m = c^{1/3}$.
- In the hybrid encryption, 1024-bit RSA with 128-bit DES.

A general attack when small e is used:

- $e = 3$, the same message m is sent to 3 different parties.
- $c_1 = [m^3 \bmod N_1]$, $c_2 = [m^3 \bmod N_2]$, $c_3 = [m^3 \bmod N_3]$.
- N_1, N_2, N_3 are coprime, and $N^* = N_1 N_2 N_3$, $\exists$ unique $\hat{c} < N^*$: $\hat{c} \equiv c_1 \pmod{N_1}$, $\hat{c} \equiv c_2 \pmod{N_2}$, $\hat{c} \equiv c_3 \pmod{N_3}$.
- With CRT, $\hat{c} \equiv m^3 \pmod{N^*}$. Since $m^3 < N^*$, $m = \hat{c}^{1/3}$.

A Quadratic Improvement in Recovering m

If $1 \leq m < \mathcal{L} = 2^\ell$, there is an attack that recovers m in time $\sqrt{\mathcal{L}}$.

Algorithm 2: An attack on textbook RSA encryption

input : Public key $\langle N, e \rangle$; ciphertext c; parameter ℓ

output: $m < 2^\ell$ such that $m^e \equiv c \pmod{N}$

set $T := 2^{\alpha\ell}$ /* $\frac{1}{2} <$ constant $\alpha < 1$ */

for $r = 1$ **to** T **do** $x_r := [c/r^e \bmod N]$

sort the pairs $\{(r, x_r)\}_{r=1}^{T}$ by x_r

for $s = 1$ **to** T **do**

  if $[s^e \bmod N] \stackrel{?}{=} x_r$ *for some* r **then**

    return $[r \cdot s \bmod N]$

return fail

It can be shown that with good probability that $m = r \cdot s$:

$$c \equiv m^e = (r \cdot s)^e = r^e \cdot s^e \pmod{N}$$

Common Modulus Attacks

Common Modulus Attacks: the same modulus N.

Case I: for multiple users with their own secret keys.
Each user can find $\phi(N)$ with his own e, d, then find others' d.

Case II: for the same message encrypted with two public keys.
Assume $\gcd(e_1, e_2) = 1$, $c_1 \equiv m^{e_1}$ and $c_2 \equiv m^{e_2} \pmod{N}$.
$\exists X, Y$ such that $Xe_1 + Ye_2 = 1$.

$$c_1^X \cdot c_2^Y \equiv m^{Xe_1} m^{Ye_2} \equiv m^1 \pmod{N}.$$

CCA in "Textbook RSA" Encryption

Recovering the message with CCA

$\mathcal{A}$ choose a random $r \leftarrow \mathbb{Z}_N^*$ and compute $c' = [r^e \cdot c \bmod N]$, and get m' with CCA. Then $m = [m' \cdot r^{-1} \bmod N]$.

$$m' \cdot r^{-1} \equiv (c')^d r^{-1} \equiv (r^e \cdot m^e)^d r^{-1} \equiv r^{ed} m^{ed} r^{-1} \equiv rmr^{-1} \equiv m.$$

Doubling the bid at an auction

The ciphertext of an bid is $c = [m^e \bmod N]$. $c' = [2^e c \bmod N]$.

$$(c')^d \equiv (2^e m^e)^d \equiv 2^{ed} m^{ed} \equiv 2m.$$

Content

Padded RSA

Idea: add randomness to improve security.

Construction 11

Let ℓ be a function with $\ell(n) \leq 2n - 2$ for all n.

- Gen*: on input 1^n, run* GenRSA(1^n) *to obtain* (N, e, d)*. Output $pk = \langle N, e \rangle$, and $sk = \langle N, d \rangle$.*
- Enc*: on input $m \in \{0,1\}^{\ell(n)}$, choose a random string $r \leftarrow \{0,1\}^{\|N\| - \ell(n) - 1}$. Output $c := [(r \| m)^e \bmod N]$.*
- Dec*: compute $\hat{m} := [c^d \bmod N]$, and output the $\ell(n)$ low-order bits of $\hat{m}$.*

ℓ should neither be too large (r is too short in theory) nor be too small (m is too short in practice).

Theorem 12

If the RSA problem is hard relative to GenRSA*, then Construction with $\ell(n) = \mathcal{O}(\log n)$ is CPA-secure.*

PKCS #1 v1.5 (RSAES-PKCS1-v1_5)

Public-Key Cryptography Standard (PKCS) #1 version 1.5:

- N has k bytes, $2^{8(k-1)} \leq N < 2^{8k}$.
- Message m has $D(\leq k-11)$ bytes.
- Random pad r has $(k-D-3)$ bytes without $\{0\}^8$.
- The ciphertext:

$$[(\{0\}^8 \| \{0\}^6 10 \| r \| \{0\}^8 \| m)^e \bmod N]$$

Security: PKCS #1 v1.5 is believed to be CPA-secure, although no proof based on the RSA assumption has ever been shown.

Attack on PKCS #1 v1.5

PKCS #1 v1.5 used in HTTPS:
if the first 16 bits of message is not "02" which is standing for "PKCK #1", then the web server returns error.

CCA to infer the message m of ciphertext c:

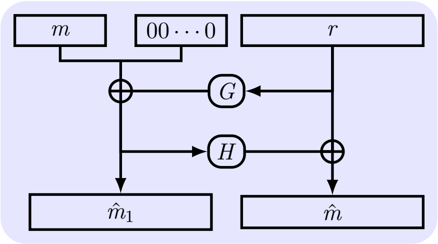
1. choose a string r, compute $c' \leftarrow r^e \cdot c = (r \cdot \text{PKCS1}(m))^e$.
2. send c' to the web server. If the server does not return error, some bits of m can be learned.
3. change r and learn other bits of m.

HTTPS Defense [RFC 5246]: if not "02", set the message as a random string.

PKCK #1 v2.1 (RSAES-OAEP)

Optimal Asymmetric Encryption Padding (OAEP): encode m of length $n/2$ as $\hat{m}$ of length $2n$. G, H are **Random Oracles**.

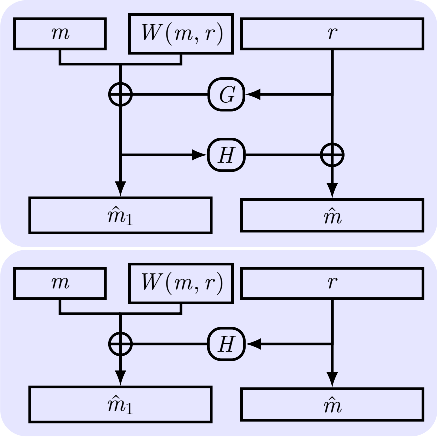
$$\hat{m}_1 := G(r) \oplus (m \| \{0\}^{n/2}), \hat{m} := \hat{m}_1 \| (r \oplus H(\hat{m}_1)).$$



RSA-OAEP is CCA-secure in Random Oracle model. [2] [RFC 3447]

[2] It may not be secure when RO is instantiated.

OAEP Improvements



OAEP+: $\forall$ trap-door permutation F, F-OAEP+ is CCA-secure.

SAEP+: RSA (e=3) is a trap-door permutation, RSA-SAEP+ is CCA-secure.

W, G, H are Random Oracles.

Remarks on RSA in Practice

Key lengths with comparable security :

Symmetric	RSA
80 bits	1024 bits
128 bits	3072 bits
256 bits	15360 bits

Implementation attacks:
Timing attack: The time it takes to compute c^d can expose d.

Power attack: The power consumption of a smartcard while it is computing c^d can expose d.

Key generation trouble (in OpenSSL RSA key generation):
Same p will be generated by multiple devices (due to poor entropy at startup), but different q (due to additional randomness).
N_1, N_2 from different devices, $\gcd(N_1, N_2) = p$.
Experiment result: factor 0.4% of public HTTPS keys.

Faults Attack on RSA

Faults attack: A computer error during $c^d \bmod N$ can expose d.

Using CRT to speed up the decryption:

$$[c^d \bmod N] \leftrightarrow ([m_p \equiv c^d \pmod{p}], [m_q \equiv c^d \pmod{q}]).$$

Suppose error occurs when computing m_q, but no error in m_p.

Then output is m' where $m' \equiv c^d \pmod{p}$, $m' \not\equiv c^d \pmod{q}$.
So $(m')^e \equiv c \pmod{p}$, $(m')^e \not\equiv c \pmod{q}$.

$$\gcd((m')^e - c, N) = p.$$

A common defense: check output. (but 10% slowdown)

Summary

- eavesdropper=CPA, CCA/CCA2 in public-key encryptions.
- hybrid argument, multiple encryptions.
- hybrid encryption, "textbook RSA", padded RSA, PKCS.
- small e, common modulus attacks, CCA, faults attack.