RSA Problem and Encryption

Yu Zhang

Harbin Institute of Technology

Cryptography, Autumn, 2020

Outline

Content

RSA Overview

- **RSA**: Ron Rivest, Adi Shamir and Leonard Adleman, in 1977
- **RSA problem**: Given $N = pq$ (two distinct big prime numbers) and $y \in \mathbb{Z}_N^*$, compute y^{-e}, e^{th}-root of y modulo N
- Open problem:RSA problem is easier than factoring N?
- **Standards**: PKCS#1 (RFC3447/8017), ANSI X9.31, IEEE 1363
- **Key sizes**: 1,024 to 4,096 bit
- **Best public cryptanalysis**: a 768 bit key has been broken
- **RSA Challenge**: break RSA-2048 to win $200,000 USD

Key lengths with comparable security :

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

"Textbook RSA"

Construction 1

- 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.

Q: How to generate N, e, d? What's $\mathbb{Z}_N^*$? How to compute $m^e \bmod N$? Is it TDP? Why is it hard?

Textbook

"*A Computational Introduction to Number Theory and Algebra*" (Version 2) by Victor Shoup

Primes and Modular Arithmetic

- The set of **integers** $\mathbb{Z}$, $a, b, c \in \mathbb{Z}$.
- $p > 1$ is **prime** if it has no factors; otherwise, **composite**.
- **Greatest common divisor** $\gcd(a, b)$ is the largest integer c such that $c \mid a$ and $c \mid b$. $\gcd(0, b) = b$, $\gcd(0, 0)$ undefined.
- Remainder $r = [a \bmod N] = a - b\lfloor a/b \rfloor$ and $r < N$. N is called **modulus**.
- $\mathbb{Z}_N = \{0, 1, \ldots, N-1\} = \{a \bmod N | a \in \mathbb{Z}\}$.
- a is **invertible modulo** $N \iff \gcd(a, N) = 1$. If $ab \equiv 1 \pmod N$, then $b = a^{-1}$ is **multiple inverse** of a **modulo** N.

Examples of Modular Arithmetic

Euclidean algorithm: $\gcd(a, b) = \gcd(b, [a \bmod b])$.

Find $\gcd(12, 27)$

Extended Euclidean algorithm: Given a, N, find X, Y with $Xa + YN = \gcd(a, N)$[1].

Find the inverse of $11 \pmod{17}$

Reduce and then add/multiply

Compute $193028 \cdot 190301 \bmod 100$

Cancellation law: If $\gcd(a, N) = 1$ and $ab \equiv ac \pmod{N}$, then $b \equiv c \pmod{N}$.

$a = 3, c = 10, b = 2, N = 24$

[1] Bézout's lemma

$\mathbb{Z}_N^*$ Group

$$\mathbb{Z}_N^* \overset{\text{def}}{=} \{a \in \{1, \ldots, N-1\} \,|\, \gcd(a, N) = 1\}$$

A **group** is a set $\mathbb{G}$ with a binary operation $\circ$:

- (**Closure**:) $\forall g, h \in \mathbb{G}$, $g \circ h \in \mathbb{G}$.
- (**Existence of an Identity**:) $\exists$ **identity** $e \in \mathbb{G}$ such that $\forall g \in \mathbb{G}, e \circ g = g = g \circ e$.
- (**Existence of Inverses**:) $\forall g \in G$, $\exists\, h \in \mathbb{G}$ such that $g \circ h = e = h \circ g$. h is an **inverse** of g.
- (**Associativity**:) $\forall g_1, g_2, g_3 \in \mathbb{G}$, $(g_1 \circ g_2) \circ g_3 = g_1 \circ (g_2 \circ g_3)$.

$\mathbb{G}$ with $\circ$ is **abelian** if

- (**Commutativity**:) $\forall g, h \in \mathbb{G}, g \circ h = h \circ g$.

Existence of inverses implies **cancellation law**.

When $\mathbb{G}$ is a **finite group** and $|\mathbb{G}|$ is the **order** of group.

Is $\mathbb{Z}_N^*$ a group under '$\cdot$'? How about $\mathbb{Z}_N$ under '$\cdot$'?

$\mathbb{Z}_{15}^* = ?$ $\mathbb{Z}_{13}^* = ?$

Group Exponentiation

$$g^m \overset{\text{def}}{=} \underbrace{g \circ g \circ \cdots \circ g}_{m \text{ times}}.$$

Theorem 2

Euler's theorem: $\mathbb{G}$ *is a finite group. Then* $\forall g \in \mathbb{G}, g^{|\mathbb{G}|} = 1$.

alculate all exponentiation of $3 \in \mathbb{Z}_7^*$

Corollary 3

Fermat's little theorem: $\forall g \in \mathbb{G}$ *and* i, $g^i \equiv g^{[i \bmod |\mathbb{G}|]}$.

Calculate $3^{78} \in \mathbb{Z}_7^*$

Arithmetic algorithms

- **Addition/subtraction**: linear time $O(n)$.
- **Mulplication**: naively $O(n^2)$. Karatsuba (1960): $O(n^{\log_2 3})$
  Basic idea: $(2^b x_1 + x_0) \times (2^b y_1 + y_0)$ with 3 mults.
  Best (asymptotic) algorithm: about $O(n \log n)$.
- **Division with remainder**: $O(n^2)$.
- **Exponentiation**: $O(n^3)$.

Algorithm 1: Exponentiating by Squaring

input : $g \in G$; exponent $x = [x_n x_{n-1} \dots x_2 x_1 x_0]_2$

output: g^x

$y \leftarrow g; z \leftarrow 1$

for $i = 0$ **to** n **do**

  if $x_i == 1$ **then** $z \leftarrow z \times y$

  $y \leftarrow y^2$

return z

Euler's Phi Function

Euler's phi function: $\phi(N) \stackrel{\mathsf{def}}{=} |\mathbb{Z}_N^*|$.

Theorem 4

$N = \prod_i p_i^{e_i}$ [2], $\{p_i\}$ *are distinct primes,* $\phi(N) = \prod_i p_i^{e_i - 1}(p_i - 1)$.

$N = pq$ **where** p, q **are distinct primes.** $\phi(N) =?$
$\phi(12) =? \quad \phi(30) =?$

Corollary 5 (Euler's theorem & Fermat's little theorem)

$a \in \mathbb{Z}_N^*$. $a^{\phi(N)} \equiv 1 \pmod{N}$.
If p *is prime and* $a \in \{1, \ldots, p-1\}$, *then* $a^{p-1} \equiv 1 \pmod{p}$.

$3^{43} \bmod 49 =?$

[2] Fundamental theorem of arithmetic

Permutation by Group Exponentiation Function

Exponentiation function $f_e : \mathbb{Z}_N^* \to \mathbb{Z}_N^*$ by $f_e(x) = [x^e \bmod N]$.
e**'th root of** y: $x^e \equiv y$, $x \equiv y^{1/e}$.

Corollary 6

If $\gcd(e, \phi(N)) = 1$, *then* f_e *is a permutation.*

Proof.

Let $d = [e^{-1} \bmod \phi(N)]$, then f_d is the inverse of f_e.
$y \equiv x^e; \quad f_d(y) \equiv y^d \equiv x^{ed} \equiv x$. □

In $\mathbb{Z}_{10}^*$, $e = 3,\ d =?,\ f_e(3) =?,\ f_d(f_e(3)) =?,\ 9^{\frac{1}{3}} =?$

What if we cannot get $\phi(N)$ for some 'special' N?
What if we cannot factorize these 'special' N?

Factoring Is Hard

- **Factoring** $N = pq$. p, q are of the same length n.
- **Trial division**: $\mathcal{O}(\sqrt{N} \cdot \text{polylog}(N))$.
- **Pollard's** $p - 1$ method: effective when $p - 1$ has "small" prime factors.
- **Pollard's rho** method: $\mathcal{O}(N^{1/4} \cdot \text{polylog}(N))$.
- **Quadratic sieve** algorithm [Carl Pomerance]: sub-exponential time $\mathcal{O}(\exp(\sqrt{n \cdot \log n}))$.
- The best-known algorithm is the **general number field sieve** [Pollard] with time $\mathcal{O}(\exp(n^{1/3} \cdot (\log n)^{2/3}))$.

The RSA Problem Is Hard

Idea: factoring is hard
$\Longrightarrow$ for $N = pq$, finding p, q is hard
$\Longrightarrow$ computing $\phi(N) = (p-1)(q-1)$ is hard
$\Longrightarrow$ computing $e^{-1} \bmod \phi(N)$ is hard
There is a gap.
$\Longrightarrow$ RSA problem is hard:
Given $y \in \mathbb{Z}_N^*$, compute y^{-e} modulo N.

Open problem

RSA problem is easier than factoring?

Generating Random Primes

Algorithm 2: Generating a random prime

input : Length n; parameter t

output: A random n-bit prime

```
for i = 1 to t do
    p' ← {0,1}^(n-1)
    p := 1||p'
    if p is prime then return p
return fail
```

- $\exists$ a constant c such that, $\forall n > 1$, a randomly selected n-bit number is prime with probability at least c/n.
- If N is prime, then the Miller-Rabin primality test always outputs “prime”. If N is composite, then the algorithm outputs “prime” with probability at most 2^{-t}.

Generating RSA Problem

Let $\mathsf{GenModulus}(1^n)$ be a polynomial-time algorithm that, on input 1^n, outputs (N, p, q) where $N = pq$, and p, q are n-bit primes except with probability negligible in n.

Algorithm 3: GenRSA

input : Security parameter 1^n

output: N, e, d

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

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

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

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

return N, e, d

Show an example of RSA problem

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 7

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

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

Constructing Trap-Door Permutations

Construction 8

Define a family of permutations with GenRSA*:*

- Gen*: on input 1^n, run* GenRSA(1^n) *to obtain* (N, e, d) *and output* $I = \langle N, e \rangle, \mathsf{td} = d$*, Set* $\mathcal{D}_I = \mathcal{D}_{\mathsf{td}} = \mathbb{Z}_N^*$*.*
- Samp*: on input* I*, choose a random element* x *of* $\mathbb{Z}_N^*$*.*
- $f_I(x) = [x^e \bmod N]$*.*
- *deterministic **inverting algorithm*** $\mathsf{Inv}_{\mathsf{td}}(y) = [y^d \bmod N]$*.*

Reduce the RSA problem to the inverting problem.

Content

Recall "Textbook RSA"

Construction 9

- 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.

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 $m = \_\_$?
- 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 Chinese Remainder Theory[3], $\hat{c} \equiv m^3 \pmod{N^*}$. Since $m^3 < N^*$, $m = \hat{c}^{1/3}$.

[3] $N = pq$ where $\gcd(p, q) = 1$. $\mathbb{Z}_N \simeq \mathbb{Z}_p \times \mathbb{Z}_q$ and $\mathbb{Z}_N^* \simeq \mathbb{Z}_p^* \times \mathbb{Z}_q^*$.

A Quadratic Improvement in Recovering m

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

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

Algorithm 4: 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

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 $Xe_1 + Ye_2 = 1$[4].

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

An example of common modulus attack

$N = 15, e_1 = 3, e_2 = 5, c_1 = 8, c_2 = 2, m = ?$

[4] Bézout's lemma

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 $m = ?$

Doubling the bid at an auction

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

$$(c')^d \equiv ?$$

Content

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 $2n^3$.

Padded RSA

Idea: add randomness to improve security.

Construction 10

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 $\hat{m}$.*

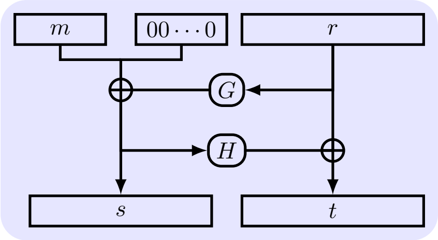
ℓ should neither be too large (r is too short in theory) nor be too small (m is too short in practice).

Theorem 11

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

PKCS #1 v2.1 (RSAES-OAEP)

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



Q: How to decipher?

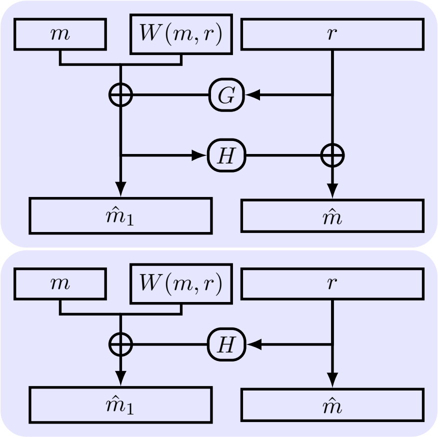
RSA-OAEP is CCA-secure in Random Oracle model. [5] [RFC 3447]

CPA: To learn r, attacker has to learn s from $(s\|t)^e$

CCA: Effective decryption query is disabled by checking "00...0" in the plaintext before the response

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

OAEP Improvements



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

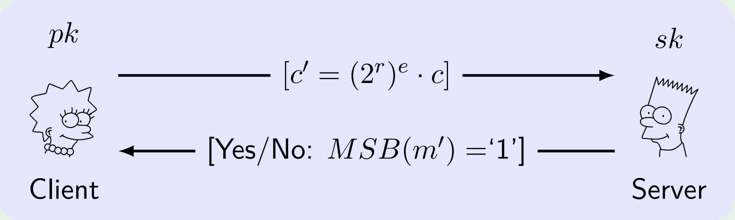
SAEP+: RSA (e=3) is a trap-door permutation, RSA-SAEP+ is CCA-secure.

W, G, H are Random Oracles.

Implementation Attacks on RSA

Simplified CCA on PKCS1 v1.5 in HTTPS [Bleichenbacher]

Server tells if the MSB of plaintext (Version Number) = '1' for a given ciphertext. Attacker sends $c' = (2^r)^e \cdot c$. If receiving Yes, then $(r+1)$-th $MSB(m) =$?



Defense: treating incorrectly formatted message blocks in a manner indistinguishable from correctly formatted blocks. See [RFC 5246]

Implementation Attacks on RSA (Cont.)

Timing attack: [Kocher et al. 1997] The time it takes to compute c^d can expose d. (require a high-resolution clock)
Power attack: [Kocher et al. 1999] The power consumption of a smartcard while it is computing c^d can expose d.
Defense: **Blinding** by choosing a random r and deciphering $r^e \cdot c$.

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).
Q: N_1, N_2 from different devices, $\gcd(N_1, N_2) = ?$
Experiment result: factor 0.4% of public HTTPS keys.

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

Using Chinese Remainder Theory 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 $m' \equiv c^d \pmod p$, $m' \not\equiv c^d \pmod q$.
So $(m')^e \equiv c \pmod p$, $(m')^e \not\equiv c \pmod q$.

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

Defense: check output. (but 10% slowdown)

Summary

- RSA, "textbook RSA", padded RSA, PKCS
- small e, common modulus attacks, CCA, implementation/faults attack