More on Public-Key Cryptography

Yu Zhang

Harbin Institute of Technology

Cryptography, Autumn, 2020

Outline

Content

Integer Factorization/Factoring

> *"The problem of distinguishing prime numbers from composite numbers and of resolving the later into their prime factors is known to be one of the most important and useful in arithmetic." – Gauss (1805)*

The "hardest" numbers to factor seem to be those having only large prime factors.

- The best-known algorithm is the **general number field sieve** [Pollard] with time $\mathcal{O}(\exp(n^{1/3} \cdot (\log n)^{2/3}))$.
- RSA Factoring Challenge: RSA-768 (232 digits)
  - Two years on hundreds of machines (2.2GHz/2GB, 1500 years)
  - Factoring a 1024-bit integer: about 1000 times harder.

Generating Random Primes

Algorithm 1: 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
```

To show its efficiency, we need understand two issues:

- the probability that a randomly-selected n-bit integer is prime.
- how to efficiently test whether a given integer p is prime.

The Distribution of Prime

Theorem 1 (Prime number theorem)

$\exists$ a constant c such that, $\forall n > 1$, a randomly selected n-bit number is prime with probability at least c/n.

The probability that a prime is *not* chosen in $t = n^2/c$ iterations is

$$\left(1 - \frac{c}{n}\right)^t = \left(\left(1 - \frac{c}{n}\right)^{n/c}\right)^n \leq \left(e^{-1}\right)^n = e^{-n}.$$

The algorithm will fail with a negligible probability.

Testing Primality

- **Trial division**: Divide N by $a = 2, 3, \ldots, \sqrt{N}$.
- **Probabilistic algorithm for approximately computing**:
  - Atlantic City algorithm with two-sided error.
  - Monte Carlo algorithm with one-sided error.
  - Las Vegas algorithm with zero-sided error.
- **Fermat primality test**: $a^{N-1} \equiv 1 \pmod{N}$.
- a is a **witness** that N is composite if $a^{N-1} \not\equiv 1 \pmod{N}$.
- a is a **liar** if N is composite and $a^{N-1} \equiv 1 \pmod{N}$.
- **Carmichael numbers**: composite numbers without witnesses.

Theorem 2

If $\exists$ a witness, then at least half the elements of $\mathbb{Z}_N^*$ are witnesses.

The Miller-Rabin Primality Test

$N - 1 = 2^r u$, u is odd. $a \in \mathbb{Z}_N^*$ is a **strong witness** if

1. $a^u \neq \pm 1$, and
2. $a^{2^i u} \neq -1$ for $i \in \{1, \ldots, r-1\}$.

Lemma 3

*$x \in \mathbb{Z}^*$ is a **square root of 1 modulo** N if $x^2 \equiv 1 \pmod{N}$. If N is an odd prime then the only x are $[\pm 1 \bmod N]$.*

Theorem 4

N is an odd, composite number that is not a prime power. Then at least half the elements of $\mathbb{Z}_N^*$ are strong witnesses.

Theorem 5

If N is prime, then the Miller-Rabin test always outputs "prime". If N is composite, then the algorithm outputs "prime" with probability at most 2^{-t} [1].

[1] Actually, it is at most 4^{-t}.

Describing The Algorithm

Algorithm 2: The Miller-Rabin primality test

input : Integer $N > 2$ and parameter t

output: A decision as to wether N is prime or composite

if *N is a perfect power* **then return** “composite”
compute $r \geq 1$ and u odd such that $N - 1 = 2^r u$
LOOP: **for** $s = 1$ **to** t **do**
    $a \leftarrow \{2, \ldots, N-2\}$
    $x = a^u \bmod N$
    if $x = \pm 1$ **then** do next **LOOP**
    for $i = 1$ **to** r **do**
        $x = x^2 \bmod N$
        if $x = -1$ **then** do next **LOOP**
    return “composite”
return “prime”

Examples of Primality Tests

Liars in Fermat primality test

$2^{340} \equiv 1 \pmod{341}$, but $341 = 11 \cdot 31$.
$5^{560} \equiv 1 \pmod{561}$, but $561 = 3 \cdot 11 \cdot 17$.
Carmichael numbers < 10000:
561, 1105, 1729, 2465, 2821, 6601, 8911.

Examples of Miller-Rabin test

Carmichael number $1729 = 7 \cdot 13 \cdot 19$.
$1729 - 1 = 1728 = 2^6 \cdot 27$. So $r = 6, u = 27$. $a = 671$.

$$671^{27} \equiv 1084 \pmod{1729}$$
$$671^{27\cdot 2} \equiv 1065 \pmod{1729}$$
$$671^{27\cdot 2^2} \equiv 1 \pmod{1729}$$

Algorithms for Factoring

- **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 $\mathcal{O}(\exp(n^{1/3} \cdot (\log n)^{2/3}))$.

Pollard's $p-1$ Method

Idea: Fermat's little theorem: $y = x^{(p-1)\cdot k} \equiv 1 \pmod{p}$. Then $(y-1) \equiv 0 \pmod{p}$ and $p \mid (y-1)$. So $p = \gcd(y-1, N)$. To make the exponent a large multiple of $(p-1)$:

$$M = lcm(\{i|i \leq B\}) = \prod_{\text{prime } i \leq B} i^{\lfloor \log_i B \rfloor}.$$

If $p-1$ has only "small" factors, then the bound B will be small.

Algorithm 3: Pollard's $p-1$ algorithm for factoring

input : Integer N

output: A non-trivial factor of N

$x \leftarrow \mathbb{Z}_N^*$

$y := [x^M \bmod N]$

$p := \gcd(y-1, N)$

if $p \notin \{1, N\}$ **then return** p

Pollard's Rho (ρ) Method

Idea: Using the improved birthday attack[2] to find x, x' such that $x \neq x' \wedge x \equiv x' \pmod{p}$. Then $p \mid (x - x')$, $p = \gcd(x - x', N)$. $F(x) = x^2 + b$, where $b \not\equiv 0, -2 \pmod{N}$.

Algorithm 4: Pollard's rho algorithm for factoring

---

input : Integer N

output: A non-trivial factor of N

$x_0 \leftarrow \mathbb{Z}_N^*$
for $i = 1$ **to** $2^{n/2}$ **do**
    $x_i := [F(x_{i-1}) \bmod N]$
    $x_{2i} := [F(F(x_{2i-2})) \bmod N]$
    $p := \gcd(x_{2i} - x_i, N)$
    if $p \notin \{1, N\}$ **then return** p

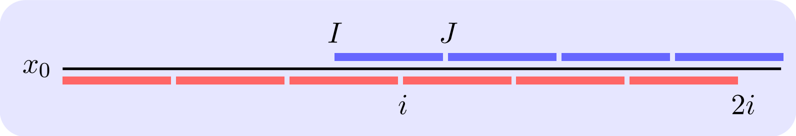
---

[2] Floyd's cycle-finding algorithm (the "tortoise and the hare" algorithm).

Proof of Pollard's ρ Method

Lemma 6

Let $x_1, \ldots$ be a sequence with $x_m \equiv F(x_{m-1}) \pmod{N}$. F satisfies that $x \equiv x' \pmod{N} \implies F(x) \equiv F(x') \pmod{N}$. If $x_I \equiv x_J \pmod{p}$ with $I < J$, then $\exists\, i < J$ such that $x_i \equiv x_{2i} \pmod{p}$.



Proof.

See the proof of improved birthday attack. □

According to the lemma of birthday problem, given a sequence of length $O(N^{1/4})$, find such pair with probability $1/4$.

Example of Pollard's $p-1$ and ρ methods

Factorizing $N = 5917$ with Pollard's $p-1$ method

Choose $B = 5$, $M = lcm(1,2,3,4,5) = 60$.
For $x = 2$, $y \equiv x^M \equiv 2^{60} \equiv 3417 \pmod{5917}$.
$p = gcd(y-1, N) = \gcd(3416, 5917) = 61$.

Factorizing $N = 8051$ with Pollard's ρ method

$f(x) = x^2 + 1$, $x_0 = 2$.

i	x_i	x_{2i}	$\gcd(x_{2i} - x_i, N)$
1	5	26	1
2	26	7474	1
3	677	871	97

The Quadratic Sieve Algorithm

Idea: Find x, y with $x^2 \equiv y^2 \pmod N$ and $x \not\equiv \pm y \pmod N$.
$x^2 - y^2 \equiv 0 \pmod N \implies (x+y)(x-y) \equiv 0 \pmod N$.
$\gcd(x+y, N)$ and $\gcd(x-y, N)$ will give p.
Finding congruence of squares:

1. Choose a factor base $B = \{p_1, \ldots, p_k\}$ of prime numbers.
2. Use '**sieve theory**' to find $\ell = k+1$ distinct $x_1, \ldots, x_\ell$ for which $[x_i^2 \bmod N]$ decompose into the elements of B: $x_i^2 \equiv \prod_{j=1}^{k} p_j^{e_j} \pmod N$.
3. Write x_i^2 as an exponent vector $\langle e_{i,1}, \ldots, e_{i,k} \rangle \pmod 2$.
4. Find the addition of vectors $=$ the zero vector $\pmod 2$. $X = \{x_{\ell_1}, \ldots, x_{\ell_n}\}$. $\forall i$, $E_i = \sum_{j=1}^{n} e_{\ell_j,i} \equiv 0 \pmod 2$.
5. Find a pair: $x = \prod_{i=1}^{n} x_{\ell_i} \not\equiv y = \prod_{i=1}^{k} p_i^{E_i/2} \pmod N$.

Example of Quadratic Sieve Algorithm

Factorizing $N = 377753$ with quadratic sieve algorithm

$B = \{2, 13, 17, 23, 29\}$.

$$
\begin{aligned}
620^2 &\equiv 17^2 \cdot 23 \pmod N \\
621^2 &\equiv 2^4 \cdot 17 \cdot 29 \pmod N \\
645^2 &\equiv 2^7 \cdot 13 \cdot 23 \pmod N \\
655^2 &\equiv 2^3 \cdot 13 \cdot 17 \cdot 29 \pmod N
\end{aligned}
$$

$$
\begin{gathered}
[620 \cdot 621 \cdot 645 \cdot 655 \bmod N]^2 \equiv [2^7 \cdot 13 \cdot 17^2 \cdot 23 \cdot 29 \bmod N]^2 \\
\implies 127194^2 \equiv 45335^2 \pmod N,
\end{gathered}
$$

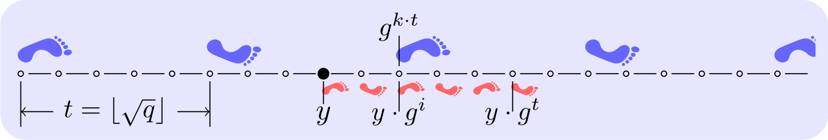
Computing $\gcd(127194 - 45335, 377753) = 751$.

Content

Overview of Discrete Logarithm Algorithms

- Given a generator $g \in \mathbb{G}$ and $y \in \langle g \rangle$, find x such that $g^x = y$.
- **Brute force**: $\mathcal{O}(q)$, $q = \mathsf{ord}(g)$ is the order of $\langle g \rangle$.
- **Baby-step/giant-step** method [Shanks]: $\mathcal{O}(\sqrt{q} \cdot \mathsf{polylog}(q))$.
- **Pohlig-Hellman** algorithm: when q has small factors.
- **Index calculus** method: $\mathcal{O}(\exp(\sqrt{n \cdot \log n}))$.
- The best-known algorithm is the **general number field sieve** with time $\mathcal{O}(\exp(n^{1/3} \cdot (\log n)^{2/3}))$.
- Elliptic curve groups vs. $\mathbb{Z}_p^*$: more efficient for the honest parties, but that are equally hard for an adversary to break. (Both 1024-bit $\mathbb{Z}_p^*$ and 132-bit elliptic curve need 2^{66} steps.)

The Baby-Step/Giant-Step Algorithm



Algorithm 5: The baby-step/giant-step algorithm

input : $g \in \mathbb{G}$ and $y \in \langle g \rangle$; $q = \mathrm{ord}(g)$ $(t := \lfloor\sqrt{q}\rfloor)$

output: $\log_g y$

```
for i = 0 to ⌊q/t⌋ do compute g_i := g^(i·t)   /* giant steps */
sort the pairs (i, g_i) by g_i
for i = 0 to t do
    compute y_i := y · g^i                      /* baby steps */
    if y_i = g_k for some k then return [kt − i mod q]
```

The time complexity is $\mathcal{O}(\sqrt{q} \cdot \mathrm{polylog}(q))$.

Example of Baby-Step/Giant-Step Algorithm

In $\mathbb{Z}_{29}^*$, $q = 28$, $g = 2$, $y = 17$.

$t = 5$, compute the giant steps:

$$2^0 = 1,\ 2^5 = 3,\ 2^{10} = 9,\ 2^{15} = 27,\ 2^{20} = 23,\ 2^{25} = 11.$$

compute the baby steps:

$$17 \cdot 2^0 = 17,\ 17 \cdot 2^1 = 5,\ 17 \cdot 2^2 = 10,$$

$$17 \cdot 2^3 = 20,\ 17 \cdot 2^4 = 11,\ 17 \cdot 2^5 = 22.$$

$2^{25} = 11 = 17 \cdot 2^4$. So $\log_2 17 = 25 - 4 = 21$.

The Pohlig-Hellman Algorithm

Idea: when q is known and has small factors, reduces the discrete logarithm instance to multiple instances in groups of smaller order.

According to CRT: If $q = \prod_{i=1}^{k} q_i$ and $\forall i \neq j, \gcd(q_i, q_j) = 1$, then

$$\mathbb{Z}_q \simeq \mathbb{Z}_{q_1} \times \cdots \times \mathbb{Z}_{q_k} \text{ and } \mathbb{Z}_q^* \simeq \mathbb{Z}_{q_1}^* \times \cdots \times \mathbb{Z}_{q_k}^*$$

$$(g_i)^x \stackrel{\text{def}}{=} \left(g^{q/q_i}\right)^x = (g^x)^{q/q_i} = y^{q/q_i} \text{ for } i = 1, \ldots, k.$$

We have k instances in k smaller groups, $\mathrm{ord}(g_i) = q_i$. [3]
Use any other algorithm to solve $\log_{g_i}(y^{q/q_i})$.
Answers are $\{x_i\}_{i=1}^{k}$ for which $g_i^{x_i} \equiv y^{q/q_i} \equiv g_i^x$.
$\forall i,\ x \equiv x_i \pmod{q_i}$. $x \bmod q$ is uniquely determined (CRT).
The time complexity is $\mathcal{O}(\max_i\{\sqrt{q_i}\} \cdot \mathrm{polylog}(q))$.

[3] If $p \mid q$, then $\mathrm{ord}(g^p) = q/p$.

Example of Pohlig-Hellman Algorithm

In $\mathbb{Z}_{31}^*$, $q = 30 = 5 \cdot 3 \cdot 2$, $g = 3$, $y = 26 = g^x$.

$$
\begin{aligned}
(g^{30/5})^x = y^{30/5} &\implies (3^6)^x = 26^6 \implies 16^x \equiv 1 \\
(g^{30/3})^x = y^{30/3} &\implies (3^{10})^x = 26^{10} \implies 25^x \equiv 5 \\
(g^{30/2})^x = y^{30/2} &\implies (3^{15})^x = 26^{15} \implies 30^x \equiv 30
\end{aligned}
$$

$$x \equiv 0 \pmod 5,\ x \equiv 2 \pmod 3, x \equiv 1 \pmod 2,$$

so $x \equiv 5 \pmod{30}$.

The Index Calculus Method

Idea: find a relatively small factor base and build a system of ℓ linear equations related to g; find a linear equation related to y; solve $\ell + 1$ linear equations to give $\log_g y$.

1. for $\mathbb{Z}_p^*$, choose a base $B = \{p_1, \ldots, p_k\}$ of prime numbers.
2. find $\ell \geq k$ distinct $x_1, \ldots, x_\ell$ for which $[g^{x_i} \bmod p]$ decompose into the elements of B: $g^{x_i} \equiv \prod_{j=1}^{k} p_j^{e_j} \pmod{p}$.
3. ℓ equations: $x_i = \sum_{j=1}^{k} e_{i,j} \cdot \log_g(p_j) \pmod{p-1}$.
4. find x^* for which $[g^{x^*} \cdot y \bmod p]$ can be factored.
5. new equation: $x^* + \log_g y = \sum_{j=1}^{k} e_j^* \cdot \log_g(p_j) \pmod{p-1}$.
6. Use linear algebra to solve equations and give $\log_g y$.

The time complexity is identical to that of the quadratic sieve.

Example of Index Calculus Method

$p = 101$, $g = 3$ **and** $y = 87$. $B = \{2, 5, 13\}$.

$3^{10} \equiv 65 \pmod{101}$ and $65 = 5 \cdot 13$. Similarly, $3^{12} \equiv 80 = 2^4 \cdot 5 \pmod{101}$ and $3^{14} \equiv 13 \pmod{101}$. The linear equations:

$$x_1 = 10 \equiv \log_3 5 + \log_3 13 \pmod{100}$$
$$x_2 = 12 \equiv 4 \cdot \log_3 2 + \log_3 5 \pmod{100}$$
$$x_3 = 14 \equiv \log_3 13 \pmod{100}.$$

We also have $x^* = 5$, $3^5 \cdot 87 \equiv 32 \equiv 2^5 \pmod{101}$, or

$$5 + \log_3 87 \equiv 5 \cdot \log_3 2 \pmod{100}.$$

Adding the 2nd and 3rd equations and subtracting the 1st, we derive $4 \cdot \log_3 2 \equiv 16 \pmod{100}$. So $\log_3 2$ is 4, 29, 54, or 79. Trying all shows that $\log_3 2 = 29$. The last equation gives $\log_3 87 = 40$.

Content

Additional Public-key Schemes

- **Goldwasser-Micali** based on deciding quadratic residuosity problem. (first scheme proven to be CPA-secure)
- **Rabin**: based on the computing square root problem. (security equivalent to the hardness of factoring)
- **Paillier**: based on the decisional composite residuosity problem. (efficient and homomorphic)
- **Elliptic curve**: forms a cyclic group with DH problem (efficient).

Quadratic Residues Modulo a Prime

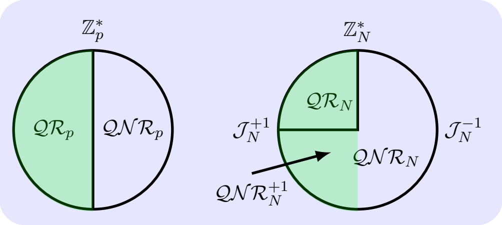
- $y \in \mathbb{G}$ is a **quadratic residue (qr)** if $\exists x \in \mathbb{G}$ with $x^2 = y$. Otherwise, y is a **quadratic non-residue (qnr)**.
- In an abelian group, the set of qr forms a subgroup.
- In $\mathbb{Z}_p^*$, $p > 2$ is prime, every qr has two square roots.
- The set of qr/qnr is $\mathcal{QR}_p/\mathcal{QNR}_p$, $|\mathcal{QR}_p| = |\mathcal{QNR}_p| = \frac{p-1}{2}$.
- $\mathcal{J}_p(x)$ is **Jacobi symbol** of x modulo p:

$$\mathcal{J}_p(x) \stackrel{\text{def}}{=} \begin{cases} +1 & \text{if } x \text{ is a qr} \\ -1 & \text{if } x \text{ is not a qr} \end{cases}$$

- $\mathcal{J}_p(x) = x^{\frac{p-1}{2}} \bmod p$.
- $\mathcal{J}_p(xy) = \mathcal{J}_p(x) \cdot \mathcal{J}_p(y)$.

Quadratic Residues Modulo a Composite

- $N = pq$, p, q distinct primes, in Chinese Remainder Theorem: $x \in \mathbb{Z}_N^*$ with $x \leftrightarrow (x_p, x_q) = ([x \bmod p], [x \bmod q])$.
- x is a qr mod $N \iff x_p/x_q$ are qr mod p/q.
- x is a qr mod $N \iff \mathcal{J}_p(x) = \mathcal{J}_q(x) = +1$.
- Qr x has 4 roots: $(\pm x_p, \pm x_q)$, so $\frac{|\mathcal{QR}_N|}{|Z_N^*|} = \frac{|\mathcal{QR}_p||\mathcal{QR}_q|}{|Z_N^*|} = \frac{1}{4}$.
- $\mathcal{J}_N(x) \stackrel{\text{def}}{=} \mathcal{J}_p(x) \cdot \mathcal{J}_q(x)$. $\mathcal{J}_N(xy) = \mathcal{J}_N(x) \cdot \mathcal{J}_N(y)$.
- $\mathcal{QNR}_N^{+1}(x) \stackrel{\text{def}}{=} \{x | x \text{ is qnr, but } \mathcal{J}_N(x) = +1\}$.



Goldwasser-Micali Scheme

- **Deciding quadratic residuosity (DQR)** of x, where x is randomly chosen from $\mathcal{J}_N^{+1}$ ($\mathcal{QR}_N$ and $\mathcal{QNR}_N^{+1}$).
- For DQR, no solution is better than factoring N.

Construction 7

- Gen*:* (N, p, q)*,* $z \leftarrow \mathcal{QNR}_N^{+1}$*.* $pk = \langle N, z \rangle$ *and* $sk = \langle p, q \rangle$*.*
- Enc*:* $m \in \{0, 1\}$*,* $x \leftarrow \mathbb{Z}_N^*$*, output* $c := [z^m \cdot x^2 \bmod N]$*.*
- Dec*: If* c *is a qr, output* 0*; otherwise* 1*.*

Goldwasser-Micali scheme is CPA-secure if DQR problem is hard.

Computing Square Roots mod a Prime

Algorithm 6: computing square root of a prime

input : Prime p; quadratic residue $a \in \mathbb{Z}_p^*$

output: A square root of a

case $p = 3 \bmod 4$: **return** $[a^{\frac{p+1}{4}} \bmod p]$

case $p = 1 \bmod 4$: let b be a qnr modulo p

compute l and m odd with $2^\ell \cdot m = \frac{p-1}{2}$

$r := 2^\ell$, $r' := 0$

for $i = \ell$ **to** 1 **do**

  $r := r/2$, $r' := r'/2$ `/* maintain` $a^r \cdot b^{r'} = 1 \bmod p$ `*/`

  if $a^r \cdot b^{r'} = -1 \bmod p$ **then** $r' := r' + 2^\ell \cdot m$

`/* now` $r = m$, r' `is even, and` $a^r \cdot b^{r'} = 1 \bmod p$ `*/`

return $\left[a^{\frac{r+1}{2}} \cdot b^{\frac{r'}{2}} \bmod p\right]$

Rabin Scheme

- **Computing square roots (CSR)** of qr mod N is **proven to be hard** if factoring N is hard.
- $N = pq$ is a **Blum integer** if $p \neq q$ and $p \equiv q \equiv 3 \bmod 4$.
- $\mathcal{QR}$ for Blum integer can form TDP.

Construction 8

- Gen: *Blum integer* $N = pq$, $pk = N$ *and* $sk = \langle p, q \rangle$.
- Enc: $m \in \{0, 1\}$, $x \leftarrow \mathcal{QR}_N$, *output* $c := \langle [x^2 \bmod N], \mathsf{lsb}(x) \oplus m \rangle$.
- Dec: *Input* $\langle c, c' \rangle$. $x = c^{1/2}$, *output* $\mathsf{lsb}(x) \oplus c'$.

Rabin scheme is CPA-secure if factoring problem is hard.

Paillier Scheme

- $\mathbb{Z}_N \times \mathbb{Z}_N^* \simeq \mathbb{Z}_{N^2}^*$ with $f(a,b) = [(1+N)^a \cdot b^N \bmod N^2]$.
- Res(N^2) is the set of Nth residue mod N^2: $\{(0,b) | b \in \mathbb{Z}_N^*\}$.
- **Decisional composite residuosity (DCR)** problem is to distinguish a random element of $\mathbb{Z}_{N^2}^*$ from one of Res(N^2).

Construction 9

- Gen*:* (N,p,q)*,* $pk = N$ *and* $sk = \langle N, \phi(N)\rangle$*.*
- Enc*:* $m \in \mathbb{Z}_N$*,* $r \leftarrow \mathbb{Z}_N^*$*, output* $c := [(1+N)^m \cdot r^N \bmod N^2]$*.*
- Dec*: output* $\left[\frac{[c^{\phi(N)} \bmod N^2]-1}{N} \cdot \phi(N)^{-1} \bmod N\right]$*.*

$c^{\phi(N)} \bmod N^2 \leftrightarrow (m,r)^{\phi(N)} = (m \cdot \phi(N), r^{\phi(N)})$.
Paillier scheme is CPA-secure if DCR problem is hard.