Number Theory and Cryptographic Hardness Assumptions (数论与密码学困难度假设) - 2

Sheng Zhong   Yuan Zhang

Computer Science and Technology Department
Nanjing University

Outline

Example 8.25

Denote by $\times_{15}, \times_3, \times_5$ multiplications modulo 15, 3, and 5 resp. Consider two groups: $Z_{15}^* = \{1, 2, 4, 7, 8, 11, 13, 14\}$ with group operation $\times_{15}$ and $Z_5^* \times Z_3^* = \{(x, y)\}_{(x \in Z_5^*, y \in Z_3^*)}$ with group operation $\times_{5,3} \stackrel{def}{=} (\times_5, \times_3)$.

1. There is a one-to-one mapping from Z_{15}^* to $Z_5^* \times Z_3^*$, e.g.
$$1 \leftrightarrow (1,1), 2 \leftrightarrow (2,2), 4 \leftrightarrow (4,1), 7 \leftrightarrow (2,1)$$
$$8 \leftrightarrow (3,2), 11 \leftrightarrow (1,2), 13 \leftrightarrow (3,1), 14 \leftrightarrow (4,2)$$

2. Denote the above mapping by f, f is homomorphic: for any $a, b \in Z_{15}^*$, we have
$$f(a \times_{15} b) = f(a) \times_{5,3} f(b).$$
E.g. $f(2 \times_{15} 11) = f(7) = (2, 1)$
$f(2) \times_{5,3} f(11) = (2, 2) \times_{5,3} (1, 2) = (2, 1)$.

An example of two isomorphic groups

Example 8.25

Denote by $\times_{15}, \times_3, \times_5$ multiplications modulo 15, 3, and 5 resp. Consider two groups: $Z_{15}^* = \{1, 2, 4, 7, 8, 11, 13, 14\}$ with group operation $\times_{15}$ and $Z_5^* \times Z_3^* = \{(x, y)\}_{(x \in Z_5^*, y \in Z_3^*)}$ with group operation $\times_{5,3} \overset{def}{=} (\times_5, \times_3)$.

1. There is a one-to-one mapping from Z_{15}^* to $Z_5^* \times Z_3^*$.
2. Denote the above mapping by f, f is homomorphic: for any $a, b \in Z_{15}^*$, we have

$$f(a \times_{15} b) = f(a) \times_{5,3} f(b).$$

Like Z_{15}^* and $Z_5^* \times Z_3^*$, when any two groups G_1 and G_2 have the above two properties, we say they are isomorphic, the mapping f is an isomorphism from G_1 to G_2, and write $G_1 \simeq G_2$.

The Chinese Remainder Theorem

THEOREM 8.24 (Chinese remainder theorem)

Let $N = pq$ where p, q are relatively prime. Then

$$Z_N \simeq Z_p \times Z_q \text{ and } Z_N^* \simeq Z_p^* \times Z_q^*.$$

Moreover, let f be the function mapping elements $x \in \{0, \ldots, N-1\}$ to pairs (x_p, x_q) with $x_p \in \{0, \ldots, p-1\}$ and $x_q \in \{0, \ldots, q-1\}$ defined by

$$f(x) \stackrel{def}{=} ([x \mod p], [x \mod q]).$$

Then f is an isomorphism from Z_N to $Z_p \times Z_q$, and the restriction of f to Z_N^* is an isomorphism from Z_N^* to $Z_p^* \times Z_q^*$.

- First appeared in the 5th-century book Sunzi's Mathematical Classic (孙子算经) by the Chinese mathematician Sunzi.
- Often used to solve congruence equations.

The CRT Theorem with multiple factors

An simple extension of Thm 8.24 is as follows.

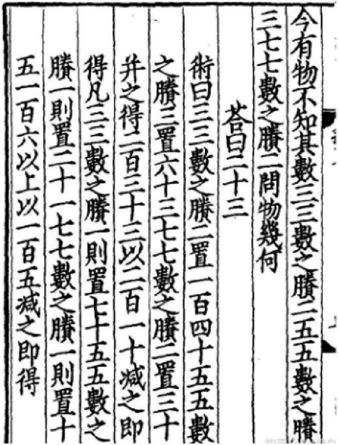
CRT (multiple factors)

Let $N = p_1 p_2 \ldots p_l$ where all $p_1, p_2, \ldots, p_l$ are pairwise relatively prime (i.e. p_i, p_j are relatively prime for all $i \neq j$). Then it holds

$$Z_N \simeq Z_{p_1} \times \ldots \times Z_{p_l} \text{ and } Z_N^* \simeq Z_{p_1}^* \times \ldots \times Z_{p_l}^*.$$

And a corresponding isomorphism can be obtained by a natural extension of the one used in Thm 8.24.

The CRT Theorem in Sunzi's Mathematical Classic



今有物不知其數三三數之賸二五五數之賸
三七七數之賸二問物幾何
荅曰二十三
術曰三三數之賸二置一百四十五五數
之賸三置六十三七七數之賸二置三十
并之得二百三十三以二百一十減之即
得凡三三數之賸一則置七十五五數之
賸一則置二十一七七數之賸一則置十
五一百六以上以一百五減之即得

图 1: CRT theorem in Sunzi's Mathematical Classic

Using Chinese Remainder Theorem - 1

Example 8.26.
Q: Compute $14 \cdot 13$ modulo 15.
A:

$$\begin{aligned}
14 \cdot 13 &= 13 + 13 + \ldots + 13 \\
&\leftrightarrow ([13 \mod 5], [13 \mod 3]) +_{5,3} ([13 \mod 5], [13 \mod 3]) \\
&\quad +_{5,3} \ldots +_{5,3} ([13 \mod 5], [13 \mod 3]) \\
&= (3, 1) +_{5,3} (3, 1) \ldots +_{5,3} (3, 1) \\
&= ([14 \cdot 3 \mod 5], [14 \cdot 1 \mod 3]) \\
&= ([[14 \mod 5] \cdot 3 \mod 5], [[14 \mod 3] \cdot 1 \mod 3]) \\
&= ([4 \cdot 3 \mod 5], [2 \cdot 1 \mod 3]) \\
&= (2, 2).
\end{aligned}$$

It is easy to see $2 \leftrightarrow (2, 2)$, thus $14 \cdot 13$ modulo 15 is 2.

Using Chinese Remainder Theorem - 2

Example 8.27.
Q: Compute 11^{54} modulo 15.
A: Since $11 \leftrightarrow (1, -1)$, we know

$$\begin{aligned} 11^{54} &\leftrightarrow ([1^{54} \mod 5], [(-1)^{54} \mod 3]) \\ &= (1, 1) \end{aligned}$$

It is easy to see $1 \leftrightarrow (1, 1)$, thus 11^{54} modulo 15 is 1.

Using Chinese Remainder Theorem - 3

Q: Let $N = p \times q$, where p, q are relatively prime. It is easy to compute $f(x) = ([x \mod p], [x \mod q])$. But how to compute f^{-1}, the inverse of f?

Example 8.30. Take $p = 5, q = 7$, and $N = 35$, what's the number in Z_{35} that corresponds to representation (4,3) in $Z_5 \times Z_7$?

Using Chinese Remainder Theorem - 3

Q: Let $N = p \times q$, where p, q are relatively prime. It is easy to compute $f(x) = ([x \mod p], [x \mod q])$. But how to compute f^{-1}, the inverse of f?
A: Given an arbitrary (x_p, x_q) in $Z_p \times Z_q$, we know

$$\begin{aligned}(x_p, x_q) &= x_p \cdot (1, 0) + x_q \cdot (0, 1) \\ &\leftrightarrow x_p \cdot f^{-1}((1, 0)) + x_q \cdot f^{-1}((0, 1)).\end{aligned}$$

So our problem reduces to compute $1_p \leftrightarrow (1, 0)$ and $1_q \leftrightarrow (0, 1)$.

Using Chinese Remainder Theorem - 3

Q: How to compute 1_p and 1_q?
A: Since p, q are relatively prime, we know $gcd(p, q) = 1$, therefore we can find integers X, Y such that

$$Xp + Yq = 1.$$

Based on the above equation, it is easy to verify that $Xp = 0 \mod p$ and $Xp = 1 \mod q$. Thus, $1_q = [Xp \mod N]$.
Similarly, we know $1_p = [Yq \mod N]$.
To compute Xp and Yq, we can use the extended Euclidean algorithm.

Using Chinese Remainder Theorem - 3

Example 8.30.
Q: Take $p = 5, q = 7$, and $N = 35$, what's the number in Z_{35} that corresponds to representation (4,3) in $Z_5 \times Z_7$?
Assume we have know $3 \cdot 5 - 2 \cdot 7 = 1$ by running an extended Euclidean algorithm.
A: We know

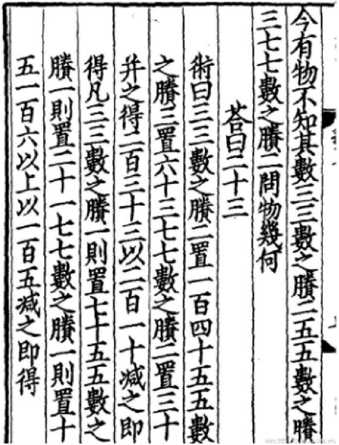
$$1_p = [-14 \mod 35],$$

and

$$1_q = [15 \mod 35]$$

.
Thus, $(4,3) \leftrightarrow [4 \times (-14) + 3 \times 15 \mod 35] = [-11 \mod 35] = 24.$

Using Chinese Remainder Theorem - 4



今有物不知其數三三數之賸二五五數之賸
三七七數之賸二問物幾何
荅曰二十三
術曰三三數之賸二置一百四十五五數
之賸三置六十三七七數之賸二置三十
幷之得二百三十三以二百一十減之即
得凡三三數之賸一則置七十五五數之
賸一則置二十一七七數之賸一則置十
五一百六以上以一百五減之即得

图 2: CRT theorem in Sunzi's Mathematical Classic

The Factoring Experiment/Problem

The Factoring Assumption assumes that the following factoring problem is hard.

- The problem is instantiated by running a polynomial-time algorithm **GenModulus**: given input 1^n, it outputs (N, p, q) where:
  1. $N = pq$.
  2. p and q are n-bit primes except with probability negligible in n.

The Factoring experiment/problem

The Factoring Assumption assumes that the following factoring problem is hard.

- The problem is instantiated by running a polynomial-time algorithm **GenModulus**.
- The problem is specified by letting an adversary $\mathcal{A}$ join in the following experiment:

The factoring experiment $Factor_{\mathcal{A},GenModulus}(n)$

1. Run **GenModulus(1^n)** to obtain (N, p, q).
2. $\mathcal{A}$ is given N, and asked to output integers $p', q' > 1$.
3. The output of the experiment is defined to be 1 if $p' \cdot q' = N$, and 0 otherwise.

The Factoring Assumption

We now formally define the factoring assumption:

DEFINITION 8.45

Factoring is hard relative to GenModulus if for all PPT algorithms $\mathcal{A}$ there exists a negligible function *negl* such that

$$Pr[Factor_{\mathcal{A},GenModulus}(n) = 1] \leq negl(n).$$

The factoring assumption is the assumption that there exists a **GenModulus** relative to which factoring is hard.

The RSA experiment/problem

The RSA problem is a problem that is closely related to the factoring problem.

The RSA experiment RSA-inv$_{\mathcal{A},GenRSA}(n)$

1. Run **GenRSA**(1^n) to obtain (N, e, d), where N is a product of two n-bit primes, $gcd(e, \phi(N)) = 1$ and $ed = 1 \mod \phi(N)$.
2. Choose a uniform $y \in \mathbb{Z}_N^*$.
3. $\mathcal{A}$ is given N, e, y, and outputs $x \in \mathbb{Z}_N^*$.
4. The output of the experiment is defined to be 1 if $x^e = y \mod N$, and 0 otherwise.

The RSA assumption

DEFINITION 8.46

The RSA problem is hard relative to GenRSA if for all PPT algorithms $\mathcal{A}$ there exists a negligible function *negl* such that

$$Pr[\text{RSA-inv}_{\mathcal{A},GenRSA}(n) = 1] \leq negl(n).$$

The RSA assumption is that there exists a **GenRSA** algorithm relative to which the RSA problem is hard.

Group element's order and (finite) cyclic group

Let $\mathbb{G}$ be a finite group of order m. For arbitrary $g \in \mathbb{G}$, define the set

$$\langle g \rangle \stackrel{def}{=} \{g^0, g^1, \ldots, g^{i-1}\},$$

where i is the smallest positive integer such that $g^i = 1$.

- i always exists and $i \leq m$. (SINCE we know $g^m = 1$.)
- i is called the order ("阶") of g.
- $\langle g \rangle$ is a group. We call it the the subgroup generated by g.
- If there is an element $g \in \mathbb{G}$ whose order equal m, then $\langle g \rangle = \mathbb{G}$. In this case, we call G is a cyclic group("循环群"), and say g is a generator ("生成元") of $\mathbb{G}$.
- Given cyclic groups, we can define several computational problems on them that are conjectured to be hard.

Examples of cyclic group

Example: consider the additive group $\mathbb{Z}_5$. What's the order of its elements? Is it a cyclic group?

Example: consider the multiplicative group $\mathbb{Z}_5^*$. What's the order of its elements? Is it a cyclic group?

The discrete logarithm problem

Let $\mathbb{G}$ be a cyclic group of order q with generator g, then

$$\mathbb{G} = \{g^0, g^1, \ldots, g^{q-1}\}.$$

- For every $h \in \mathbb{G}$, there is a unique $x \in Z_q$ such that $g^x = h$.
- We call this x the discrete logarithm of h with respect to g, and write $x = \log_g h$.

The discrete-logarithm problem

The discrete logarithm problem is to compute the discrete logarithm of a uniformly chosen element in a cyclic group.
Specifically, the discrete-logarithm problem can be described using the following experiment for a group-generation algorithm $\mathcal{G}$, and parameter n.

The discrete-logarithm experiment $\mathsf{DLog}_{\mathcal{A},\mathcal{G}}$

1. Run $\mathcal{G}(1^n)$ to obtain $(\mathbb{G}, q, g)$, where $\mathbb{G}$ is a cyclic group of order q (with $||q|| = n$), and g is a generator of $\mathbb{G}$.
2. Choose a uniform $h \in \mathbb{G}$.
3. $\mathcal{A}$ is given $\mathbb{G}, q, g, h$, and outputs $x \in Z_q$.
4. The output of the experiment is defined to be 1 if $g^x = h$, and 0 otherwsie.

The discrete-logarithm assumption

DEFINITION 8.62

We say that **the discrete-logarithm problem is hard relative to** $\mathcal{G}$ if for all PPT algorithms $\mathcal{A}$ there exists a negligible function *negl* such that

$$Pr[\mathrm{DLog}_{\mathcal{A},\mathcal{G}}(n) = 1] \leq negl(n).$$

The discrete logarithm assumption is the assumption that there exists a $\mathcal{G}$ for which the discrete-logarithm problem is hard.

There are two important variants of D-H problems:

1. **The computational D-H (CDH) problem**: Given g, g^x and g^y, can you compute g^{xy}?
2. **The decisional D-H (DDH) problem**: Given g, g^x, g^y, and g^{xy}, can you differentiate g^{xy} from a uniform random group element g^z?

- First proposed by Whitfield Diffie and Martin Hellman.
- Closed related to the discrete-logarithm problem, but not known to be equivalent.
- The D-H assumptions are assumptions that there exists instances of CDH/DDH problem which are hard.