

**Computer Methods for Integer Factorization and
Discrete Logarithm:
*A Cryptographic Perspective***

LITERATURE REVIEW

Student : Shi Bai

Supervisor: Prof. Richard Brent

OUTLINE

- I. BACKGROUND
 - Preliminaries and Concepts
 - History, Importance and Motivation
- II. INTEGER FACTORIZATION
 - "Birthday" paradox
 - Floyd cycle-finding algorithm
 - Pollard rho method for factoring
 - Example
 - Brent method for factoring
- III. DISCRETE LOGARITHM
 - Rho method for DL problems
- IV. CONCLUSION AND PERSPECTIVE

1.1 PRELIMINARIES AND CONCEPTS

Cryptography is about the communication in the presence of adversaries (eavesdropper) [Rivest 1990]. It assumes adversaries have limited computational ability - only can solve Not-hard problems.

Hard, infeasible or **intractable**: no known deterministic algorithm which could solve the problem in polynomial time (to input size). Exponential-time - not practical.

Trapdoor function: easy to compute in one direction $f(x)$, while believed to be difficult to find its inverse $f^{-1}(x)$. However with special information - "trapdoor", it is "easy" to compute its inverse.

Public-key cryptography: public key and private key. - Bob sent to Alice

- Alice generates public key p ; private key t .
- Bob encrypts message m by $f(m, p)$. Note that $f^{-1}(m, p)$ is Hard.
- Alice decrypts m by $m = f^{-1}(m, p, t)$ because m is as "trapdoor".

1.1 PRELIMINARIES AND CONCEPTS

- Any positive integer N has a unique prime power decomposition such that [Brent 1999],

$$N = p_1^{a_1} \cdot p_2^{a_2} \cdots p_k^{a_k} \quad (240 = 2^4 \cdot 3^1 \cdot 5^1)$$

- Discrete logarithms: analogues of ordinary logarithms within scope of modulo prime p .

$$3^x \equiv 13 \pmod{17} \quad x = 4 \text{ because } 3^4 \% 17 = 13$$

- Both above problems are believed to be difficult, while their inverses (multiplication and discrete exponentiation) are "easy". Could be utilized to construct "trapdoor" functions.
- Modular arithmetic: $114 \% 5 = 4$ because $114 = 5 \cdot 22 + 4$.
- Congruence: $114 \equiv 19 \pmod{5}$ because $116 \% 5 = 19 \% 5 = 4$. It implies a kind of equivalence.

1.2 HISTORY, IMPORTANCE AND MOTIVATION

- **Motivation** - Integer factorization and discrete log problems are of great interest because of their relevance to public key cryptography.
- **Importance** - ATM cards, computer passwords, E-commerce, military communication.
- **History**
 - Substitution alphabets , 2000 B.C. by Egyptian soldiers ("hello" ~ "ifmmp" by shift 1.)
 - By WorldWar II, poly-alphabetic cipher.
 - Symmetric-key, modern embodiments of poly-alphabetic. (DES, AES)
 - Public-key cryptography, based on computational complexity of "hard" problems 1976.

2.1 "BIRTHDAY" PARADOX

If a group of 23 or more people are chosen randomly, the probability that at least two of them have birthday in same day exceeds 50%. By probability theory, we have, [Handout figure 2]

$$P = 1 - \bar{P} = 1 - \frac{365}{365} \cdot \frac{365 - 1}{365} \cdot \frac{365 - 2}{365} \cdots \frac{365 - 23 + 1}{365} > 50\%$$

Suppose that we have p as the total number of integers and if we choose k integers randomly out of p , the probability that at least two of them are congruent mod p is,

$$P(k) = 1 - \bar{P}(k) = 1 - \frac{n}{n} \cdot \frac{n - 1}{n} \cdots \frac{n - k + 1}{n} = \frac{n(n - 1) \cdots (n - k + 1)}{n^k} \quad (1)$$

If the equation is at 50%, then $k \approx 1.177\sqrt{p}$ ($O(\sqrt{p})$). Thus two randomly chosen integers will be congruent mod p very likely after $1.177\sqrt{p}$ numbers have been checked.

2.2 FLOYD CYCLE-FINDING ALGORITHM

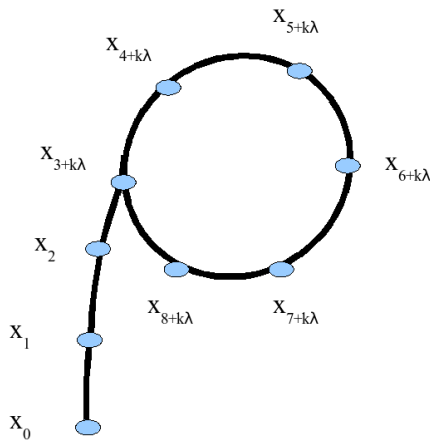
A pseudorandom generator is needed to produce random numbers for above use.

$$x_{i+1} \equiv f(x_i) \pmod{p}$$

The minimal such α and λ are the length of aperiodic and periodic part of the sequence x_i respectively

$$x_{n+\lambda} \equiv x_n \pmod{p} \quad (n > \alpha)$$

● $x \equiv f(x_0) \pmod{p} \sim y \equiv f(f(x_0)) \pmod{p}$



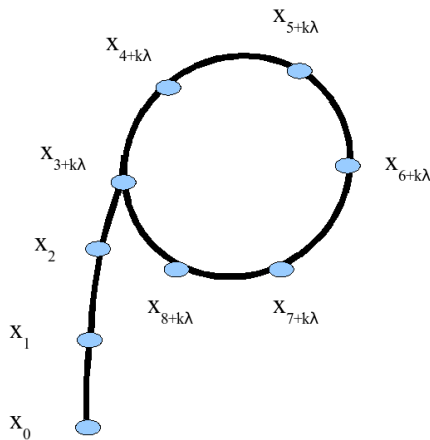
2.2 FLOYD CYCLE-FINDING ALGORITHM

A pseudorandom generator is needed to produce random numbers for above use.

$$x_{i+1} \equiv f(x_i) \pmod{p}$$

The minimal such α and λ are the length of aperiodic and periodic part of the sequence x_i respectively

$$x_{n+\lambda} \equiv x_n \pmod{p} \quad (n > \alpha)$$



- $x \equiv f(x_0) \pmod{p} \sim y \equiv f(f(x_0)) \pmod{p}$
- $x_1 \sim x_2 \pmod{p}$

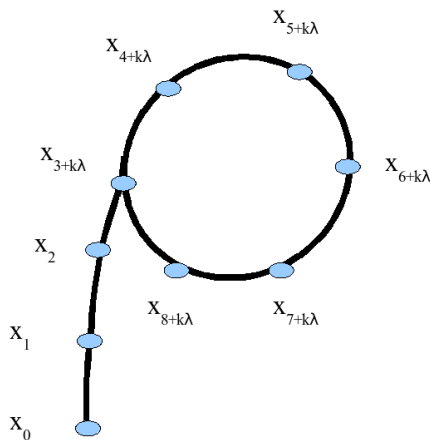
2.2 FLOYD CYCLE-FINDING ALGORITHM

A pseudorandom generator is needed to produce random numbers for above use.

$$x_{i+1} \equiv f(x_i) \pmod{p}$$

The minimal such α and λ are the length of aperiodic and periodic part of the sequence x_i respectively

$$x_{n+\lambda} \equiv x_n \pmod{p} \quad (n > \alpha)$$



- $x \equiv f(x_0) \pmod{p} \sim y \equiv f(f(x_0)) \pmod{p}$
- $x_1 \sim x_2 \pmod{p}$
- $x_2 \sim x_4 \pmod{p}$

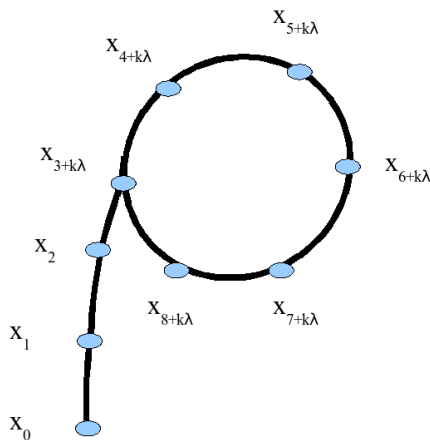
2.2 FLOYD CYCLE-FINDING ALGORITHM

A pseudorandom generator is needed to produce random numbers for above use.

$$x_{i+1} \equiv f(x_i) \pmod{p}$$

The minimal such α and λ are the length of aperiodic and periodic part of the sequence x_i respectively

$$x_{n+\lambda} \equiv x_n \pmod{p} \quad (n > \alpha)$$



- $x \equiv f(x_0) \pmod{p} \sim y \equiv f(f(x_0)) \pmod{p}$
- $x_1 \sim x_2 \pmod{p}$
- $x_2 \sim x_4 \pmod{p}$
- $x_3 \sim x_6 \pmod{p}$

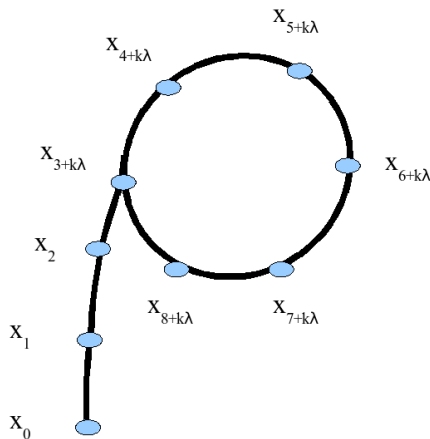
2.2 FLOYD CYCLE-FINDING ALGORITHM

A pseudorandom generator is needed to produce random numbers for above use.

$$x_{i+1} \equiv f(x_i) \pmod{p}$$

The minimal such α and λ are the length of aperiodic and periodic part of the sequence x_i respectively

$$x_{n+\lambda} \equiv x_n \pmod{p} \quad (n > \alpha)$$



- $x \equiv f(x_0) \pmod{p} \sim y \equiv f(f(x_0)) \pmod{p}$
- $x_1 \sim x_2 \pmod{p}$
- $x_2 \sim x_4 \pmod{p}$
- $x_3 \sim x_6 \pmod{p}$
- $\dots \sim \dots$ until $x_i \equiv x_{2i} \pmod{p}$

2.3 POLLARD RHO METHOD FOR FACTORING

- 1. Now we could built a (pseudo)-random sequence to make two numbers in the sequence are very likely congruent to a number.
- 2. We could find such two numbers by Floyd cycle-finding algorithm $x_i \equiv x_{2i} \pmod{p}$.

They are the two key concepts in rho method which was first proposed by Pollard [Pollard 1975]. For example, for a prime integer p , we construct a (pseudo)-random sequence in which two numbers are congruent to such p . Therefore we have $a \equiv b \pmod{p}$. Such a and b could be ultimately found by $x_i \equiv x_{2i} \pmod{p}$. Then we know $|a - b| = |x_i - x_{2i}| \equiv 0 \pmod{p}$ now, which means $|a - b|$ is a multiple of p . So if we apply $\gcd(|a - b|, p)$, we could get p . However p is not known before because it's our target, we could use $\gcd(|a - b|, N)$, where N is our candidate number for factoring.

2.4 EXAMPLE

- 1. Pseudo-random sequence: $x_{i+1} = x_i^2 - 1$, $x_0 = 3$, Factor $N = 91643$;
- 2. We could find such two numbers by Floyd cycle-finding algorithm $x_i \equiv x_{2i} \pmod{N}$.

Iteration	$x = f(x)$	$y = f(f(y))$	Comparison	$\gcd(x - y , N)$
0	$x_0 \equiv 3$	$x_0 \equiv 8$	-	-
1	$x_1 \equiv 8$	$x_2 \equiv 63$	$ x_2 - x_1 $	$\gcd(55, N) = 1$
2	$x_2 \equiv 63$	$x_4 \equiv 74070$	$ x_4 - x_2 $	$\gcd(74007, N) = 1$
3	$x_3 \equiv 3968$	$x_6 \equiv 35193$	$ x_6 - x_3 $	$\gcd(31225, N) = 1$
4	$x_4 \equiv 74070$	$x_8 \equiv 45368$	$ x_8 - x_4 $	$\gcd(28702, N) = 113$

- Inside: $x_8 \equiv x_4 \pmod{113}$, so $|x_8 - x_4|$ is a multiple of 113, therefore $\gcd(28702 = 113 \cdot x, N = 113 \cdot y) = 113$.
rho can fail but rarely.

2.5 BRENT METHOD

A new cycle-finding algorithm along with a faster refinement of the rho algorithm is proposed in [Brent 1980].

elements	$x_j - x_{2^k}$
x_1, x_2	$x_2 - x_1$
x_3, x_4	$x_4 - x_2$
x_5, x_6, x_7	$x_7 - x_4$
x_8	$x_8 - x_4$
$x_9, x_{10}, x_{11}, x_{12}, x_{13}$	$x_{13} - x_8$
x_{14}	$x_{14} - x_8$
x_{15}	$x_{15} - x_8$
x_{16}	$x_{16} - x_8$
x_{17}, \dots, x_{25}	$x_{25} - x_{16}$
x_{26}	$x_{26} - x_{16}$
x_{27}	$x_{27} - x_{16}$
\dots	\dots

RHO METHOD FOR DISCRETE LOGARITHM PROBLEM

Pollard's rho algorithm [Pollard 1978] for the discrete logarithm problem is a randomized algorithm with the same idea as his rho algorithm for solving the integer factorization problem. We will not repeat such ideas and we will only introduce the differences. The main difference lies in the pseudorandom generator.

$$x_{i+1} = f(x_i) = \begin{cases} \beta \times x_i & : \chi_i \in G_1 \\ x_i^2 & : x_i \in G_2 \\ \alpha \times x_i & : x_i \in G_3 \end{cases}$$

The above sequence of group elements in turn defines other two sequences of integers a_i and b_i who satisfy $\chi_i = \alpha^{a_i} \beta^{b_i}$,

$$a_{i+1} = \begin{cases} a_i & : x_i \in G_1 \\ 2a_i \bmod n & : \chi_i \in G_2 \\ a_i + 1 \bmod n & : x_i \in G_3 \end{cases}$$

$$b_{i+1} = \begin{cases} b_i + 1 \bmod n & : x_i \in G_1 \\ 2b_i \bmod n & : x_i \in G_2 \\ b_i & : x_i \in G_3 \end{cases}$$

CONCLUSION AND PERSPECTIVE

- Little has been proved about rho algorithms because the walk performed is not truly random. Thus pure theoretical analysis based upon probability theory is not ensured to be accurate. HOW GOOD THE RANDOM ASSUMPTION IT IS?
- Assume it is truly random, the expected running time is $O(\sqrt{n})$ from probability theory. For integer factorization, let w be the sum of the period and the length of the non-periodic part of the walk. Then w^2 should have a certain exponential distribution and *we can investigate this empirically by taking walks mod n for various primes p .*
- For integer factorization, Guy conjectured that in the worst case is $w = O(\sqrt{p \log p})$ [Guy 1975]. *Test this conjecture by computing $\frac{w}{\sqrt{\log p}}$ for all primes p up to some limit B . As a byproduct we can obtain a rigorous bound on the running time of the Pollard rho factoring algorithm on numbers whose least prime factor is at most B .*

CONCLUSION AND PERSPECTIVE - CONTINUE

- Because the running time depends on the randomness number by pseudo-random generator. A consideration of different generating functions for sequence may perform differently. *For integer factorization, we could compare the performance by different generators.*
- For discrete logarithm, it was investigated empirically that the Pollard rho method is slower than the conjectured expected running time in [Teske 1998, 2001] and it was proved recently to be $O(\sqrt{n}(\log n)^3)$ in [Miller and Venkatesan 2006]. *Some new iteration functions have been proposed, which we could test and also could be used to help us in the empirical investigation.* The investigation will be optional depending on the progress.

THANK YOU!
ANY QUESTIONS? $\Theta\Theta$
 ω