RSA Cryptanalysis and Factoring: A Survey

Nadir Murru - Università di Torino
Michele Elia - Politecnico di Torino

De Cyfris Section: Cryptanalysis

Ancona, February 4th, 2020

Outline of the presentation

I

General considerations on secret-key and public-key encryption systems.
Their cryptanalysis.

RSA cryptanalysis and the Factoring problem

II

Factoring

Fermat, Euler, Shanks, and Pollard

Continued fractions, ECC, and other tools

Cryptography

The millenarian subject of cryptography has the aim of protecting the information in its widest sense.

It has evolved in a science, a mathematical science, developing methods of always more abstract character.

The protection is achieved by transforming the information using method, technique, and algorithms borrowed from mathematics, or other physical sciences.

According to Adrian A. Albert, it is not an exaggeration to say that

abstract cryptography is identical with abstract mathematics.

Cryptanalysis

It is a branch of the Cryptography.

Cryptanalysis concerns methods of attacking cryptographic systems put in place by enemies or partners.

Cryptanalysis has two main objectives:

- To verify the validity of cryptographic schemes and protocols, acting with all publicly-available information.

- To break cryptographic schemes of adversaries (military, political, industrial, commercial, or social) acting with partial or no side information about systems, protocols, algorithms, and keys.

Goals

Both endeavors may have many goals:

- To find the cryptographic algorithm employed in a protected connection.
- To disclose the text of an encrypted message.
- To capture the secret key of a cryptographic system.

- To impersonate the authentic author of a message.
- To falsify a digital signature.

Cryptanalysis

Due to its importance in the human affairs, in particular in war time, it has received a great attention, however, no rigorous formulation, comparable to Shannon's theory of perfect secrecy, has been proposed.
The attacks to private key systems have been and still are more an art than a science.
The attacks to public-key systems have a less fuzzy formulation, although the art character is still predominant.

Anyway, an axiomatic formulation of the cryptanalysis should be based, at least, on the following two axioms.

Cryptanalysis: Axiom 1

Axiom

When the cryptanalysis of an encrypted message $\mathcal{T}$ produces a message $\mathcal{U}$, the cost for recognizing whether $\mathcal{U}$ is the right message has negligible (i.e. $O(1)$) complexity.

Cryptanalysis; Axiom 2 [(Kerkoff principle)]

Axiom (Kerkoff principle)

Let a message $M \in \mathcal{M}$ be encrypted into a message $E \in \mathcal{E}$, by a transformation f driven by a key $K \in \mathcal{K}$.
The strength of an enciphering scheme must rely totally on the key K and its secrecy. The transformation f is, or should be considered to be, publicly known.

Axiom consequences

The axioms highlight the difference between secret and public key encryption:

1. In secret-key encryption the key is kept secret, and the only available information is the encrypted message, with possibly the encryption algorithm. Axiom 1 is indispensable for establishing whether the decrypted message is correct.
2. In public-key encryption the available information is the encryption algorithm and part of the key. Axiom 1 is easily satisfied since the known mathematical relation between the secret and the public parts of the key.

Public-key schemes

Before the advent of the quantum computing, all public key encryption algorithms stood on three problems that were believed to be of difficult solution without side information, namely

1. Difficulty of factoring an integer, that is, given a number $N = pq$, product of two primes it is believed to be difficult to isolate the factors p and q.
2. Difficulty of computing a discrete logarithm, that is, given N and a primitive element α in $\mathbb{Z}_N$, it is difficult to find an integer n such that $\beta = \alpha^n \bmod N$ for some $\beta \in \mathbb{Z}_N$.
3. Searching a set of unsorted data. It seems that, in an unforeseeable future, only this method will survive, possibly together other emerging algorithms based on non-commutative algebras.

Computational complexity

In secret-key enciphering a measure of security is given by the length of the key, typically measured in bit. We have

1. Perfect security, is achieved when the size of the key, randomly generated, is equal to the size of the message (Shannon's theorem)
2. For practical purposes, the size of the key is short $(60 \div 1000)$ and is used by the encrypting machines to generate a key of the size of the message.
   For design purposes it should be assumed that the machine structure is known to attackers, (lesson taught by Enigma's story.

Computational complexity

The public-key crypto-systems are based on an asymmetry in complexity between encryption and decryption rules.
The principle is to refer at a one-way function f such that

1. f is easy to compute, i.e. $O(\ln N)$
2. $f^{-1}$is difficult to compute, i.e. $O(N)$

The most popular public-key crypto-systems, when were discovered had a believed difference in computational complexity of $O(\ln N)$ versus $O(\sqrt{N})$. Defining the faction $L(\gamma, c) = Q(e^{c(\ln N)^{\gamma}(\ln \ln N)^{1-\gamma}})$, at the present time the upper complexity is sub-exponential, i.e.
$L(\frac{1}{3}, c) = Q(e^{c\sqrt[3]{\ln N}\sqrt[3]{(\ln \ln N)^2}})$ which is the probabilistic complexity of factoring using the number field sieve.

Gauss

In his *Disquisitiones Arithmeticae*, Gauss pointed out the importance of the factoring problem

Problema, numeros primos a compositis dignoscendi, hosque in factores suos primos resolvendi, ad gravissima ac utilissima totius arithmeticae pertinere, et geometrarum tum veterum tum recentiorum industriam ac sagacitatem occupavisse, tam notum est, ut de hac re copiose loqui superfluum foret. Nihilominus fateri oportet, omnes methodos hucusque prolatas vel ad casus valde speciales restrictas esse, vel tam operas et prolixas, ut iam pro numeris talibus, qui tabularum a viris meritis constructarum limites non excedent, i.e. pro quibus methodi artificiales supervacuae sunt, calculatoris etiam exercitati patientiam fatigent, ad maiores autem plerumque vix applicari possint.

C. F. Gauss [Art. 329]

Schemes based on the factoring problem

Two centuries later, factoring became of fundamental importance in public-key cryptography:

1. RSA: the Rivest-Shamir-Adleman scheme is defined as follows: given a product of two primes $N = pq$, a message M, encoded into a number of $\mathbb{Z}_N$, is encrypted as follows

$$\mathbf{C = M^E \bmod N}$$

where the publicly known E is relatively prime with the Euler function $\varphi(N) = (p-1)(q-1)$; the decryption is performed using a secret exponent D computed as $D \cdot E = 1 \bmod \varphi(N)$: $\mathbf{M = C^D \bmod N}$

2. Rabin: given $N = pq$, a message M encoded into a number of $\mathbb{Z}_N$ is encrypted as follows

$$C = M^2 \bmod N$$

Electronic signature

Electronic Signatures are obtained by exchanging the role of the public key and the secret key
Let M be the message to be signed, and S be the signature.
Using RSA the signer, using the secret D and his public N computes

$$S = M^D \bmod N$$

The verifier, using the public key E and N, checks

$$S^E \bmod N \stackrel{?}{=} M$$

Using Rabin the signer and verifier respectively compute:

$$S = \sqrt{M} \bmod N \quad \text{and} \quad S^2 \bmod N \stackrel{?}{=} M$$

RSA versus Rabin

To break Rabin is fully equivalent fo factor the modulo N

If the modulo N is factored then RSA is broken.
It is still an open problem to show that, if RSA is broken then N can be factored.

From a purely theoretical point of view Rabin is more secure if the same number of bits is used for the public key

- RSA: the public key consists of two numbers $[\mathbf{N}, \mathbf{E}]$, i.e. $\log_2 N + \log_2 E \approx 2 \log_2 N$ bits
- Rabin: the public key consists of a single number $[\mathbf{N}]$, i.e. $\log_2 N$ bits

Then with Rabin, it is possible to use a modulo N consisting of the double number of digits; however, practical considerations have favored the predominance of RSA.

Factoring methods

1. Brute force approach: it serves as a reference for the computational complexity:
   if a composite N has k prime factors, the complexity for a direct search of the most small factor is $O(\sqrt[k]{N})$
2. Methods based on ideas of Fermat, that is: the search for two distinct integers x_1 and x_2 such that $x_1^2 = x_2^2 \bmod N$
3. Methods based on the continued fraction expansion of $\sqrt{N}$

Conclusions

- At the present state, the complexity of factoring and the complexity of computing the discrete logarithm are comparable.
- At the present state, factoring is still challenging the mathematicians.
- The advent of quantum computer leaves the fascination of the factoring problem.

The case of public-key shows indisputably that cryptanalysis is an art.

Fermat's idea

The majority of factoring methods is based on an idea of **Fermat**, that is, given a composite number N, find **two squares** x^2 and y^2 such that their **difference is divisible by** N, i.e.,

$$x^2 \equiv y^2 \pmod{N}.$$

The factors are obtained from the identity by computing

$$\gcd\{(x-y), N\} \quad \text{and} \quad \gcd\{(x+y), N\}$$

If N has more than two factors, the mechanism should be iterated.
Various methods differ in the way the two squares are obtained.

Fermat's factoring algorithm

The simplest idea is to **search by trial** x and y such that $x^2 - y^2 = N$.

1. Set $k = \lfloor \sqrt{N} \rfloor + 1$, $h = k^2 - N$, $d = 1$
2. If $\lfloor \sqrt{h} \rfloor = \sqrt{h}$ go to (4), else set $h = h + 2k + d$ and $d = d + 2$
3. If $\lfloor \sqrt{h} \rfloor < N/2$ go to (2), else no factor found and terminate algorithm
4. Set $x = \sqrt{N + h}$ and $y = \sqrt{h}$, $x - y$ and $x + y$ are factor of N

If $N = pq$, with $|p - q| < C \cdot N^{1/4}$, then the Fermat's algorithm is efficient.

Quadratic Sieve

Invented by **Pomerance** in 1981, the idea is to find congruences of the form $x_i^2 \equiv a_i \pmod{N}$ **where** $\prod a_i$ **is a square**. Pomerance gave the following example.
Consider $N = 1649$, with Fermat's method, we will consider

- 41^2 and $41^2 - 1649 = 32$ (which is not a square)
- 42^2 and $42^2 - 1649 = 115$ (which is not a square)
- 43^2 and $43^2 - 1649 = 200$ (which is not a square)
- ...
- 57^2 and $57^2 - 1649 = 1600 = 40^2$, from which we get the factors 17 and 97 of 1649

Quadratic Sieve

With the idea of Pomerance, 41^2, 42^2 and 43^2 are enough to factor 1649, indeed

$$41^2 \equiv 32 \pmod{1649},\ 42^2 \equiv 115 \pmod{1649},\ 43^2 \equiv 200 \pmod{1649}$$

and $32 = 2^5$, $200 = 2^3 \cdot 5^2$. Thus we are sure that $32 \cdot 200$ is a square and in particular we have

$$(41 \cdot 43)^2 \equiv 80^2 \pmod{1649}$$

where $41 \cdot 43 \equiv 114 \pmod{1649}$, from which $\gcd(114 - 80, 1649) = 17$.

General Number Field Sieve

The GNFS is one of the most powerful factorization methods and the main idea is **still based on finding two squares** congruent modulo N.

- Choose an integer polynomial $f(x)$ irreducible over $\mathbb{Q}$ and an integer m such that $f(m) \equiv 0 \pmod{N}$.
- The main task of the GNFS method is to find pairs of integers a_i and b_i such that $\prod_i (a_i + b_i m)$ is a square x^2 in $\mathbb{Z}$ and $\prod_i (a_i + b_i \alpha)$ is a square β^2 in the ring $\mathbb{Z}[\alpha]$, where $\alpha \notin \mathbb{Q}$ is a root of $f(x)$.
- Exploit the ring morphism $\varphi : \mathbb{Z}[\alpha] \to \mathbb{Z}_N$, defined substituting α with $m \pmod{N}$; in this way, we have that $\varphi(\beta^2) = y^2 \equiv x^2 \pmod{N}$

Continued fractions

Continued fractions provide a representation for any real number α by means of a sequence of integers:

$$\alpha = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cdots}}},$$

where $a_0, a_1, ...$ (*partial quotients*) are obtained by

$$\begin{cases} a_k = [\alpha_k] \\ \alpha_{k+1} = \dfrac{1}{\alpha_k - a_k} \quad \text{if } \alpha_k \text{ is not integer} \end{cases} \qquad k = 0, 1, 2...$$

where α_i are called *complete quotients*. Shortly, we write $\alpha = [a_0, a_1, a_2, a_3, ...]$.

Some properties of continued fractions

- A continued fraction is **finite** if and only if α is a **rational number**.
- A continued fraction is **infinite** if and only if α is an **irrational number**.
- A continued fraction is **periodic** if and only if α is a **quadratic irrationality**.
- $\sqrt{N} = [a_0, \overline{a_1, \ldots, a_{L-1}, 2a_0}]$.

Convergents of a continued fraction

Definition

Given a continued fraction $\alpha = [a_0, a_1, a_2, ...]$, the n–the convergent is the finite continued fraction $[a_0, ..., a_n] = \frac{p_n}{q_n}$.

The convergents can be evaluated by

$$\begin{pmatrix} a_0 & 1 \\ 1 & 0 \end{pmatrix} \cdots \begin{pmatrix} a_n & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} p_n & p_{n-1} \\ q_n & q_{n-1} \end{pmatrix}$$

or

$$\begin{cases} p_0 = a_0, \quad p_1 = a_0 a_1 + 1 \\ p_n = a_n p_{n-1} + p_{n-2}, \quad \forall n \geq 2 \end{cases}, \quad \begin{cases} q_0 = 1, \quad q_1 = a_1 \\ q_n = a_n q_{n-1} + q_{n-2}, \quad \forall n \geq 2 \end{cases}$$

CFRAC and SQUFOF algorithms

The CFRAC and SQUFOF algorithms exploit the **continued fraction expansion** of $\sqrt{N}$ for **finding two squares** congruent modulo N. The first idea is to find a complete quotient $(P_n + \sqrt{N})/Q_n$ such that $Q_n = R^2$; moreover it is known that

$$p_{n-1}^2 \equiv (-1)^n Q_n \pmod{N}$$

where p_n/q_n is the n–th convergent of $\sqrt{N}$. Then N is factored evaluating $\gcd(p_{n-1} - R, N)$ e $\gcd(p_{n-1} + R, N)$.
The method was improved by Shanks exploiting the theory of **quadratic forms** and obtaining an algorithm with complexity $O(\sqrt[5]{N})$.

Pollard's $p-1$ algorithm

Input: N number to factor, B bound
Output: p prime factor of N
Algorithm:

1. Set $a = 2$
2. **for** $j = 2$ **to** B **do**

$$a = a^j \pmod{N}$$

3. $d = \gcd(a - 1, N)$

Pollard's $p-1$ algorithm

1. The bound B must be such that $q \leq B$, for all $q|p-1$, with p factor of N and q power of a prime.
2. In this way we know that $p-1$ divides $B!$.
3. At the end of the iterations of the algorithm, we have $a \equiv 2^{B!} \pmod{N}$.
4. For the Fermat's Little Theorem (since $p-1|B!$), we know that $a = 2^{B!} \equiv 1 \pmod{p}$, i.e., $p|a-1$.
5. Hence $p|d = \gcd(a-1, N)$.

Pollard's $p-1$ algorithm

- The method needs the evaluation of $B-1$ modular exponentiations and a GCD.
- It is efficient if $p-1$ has small factors.
- The RSA modulo $N=pq$ must be chosen such that $p-1$ and $q-1$ have not small factors.
- Do not use Fermat and Mersenne numbers.

The Shor's algorithm

The Shor's algorithm is a **quantum algorithm** that can be applied for solving the integer factorization problem in **polynomial time**. In fact, it is an algorithm able to efficiently **find the period** of an element in residue class rings. If we are able to find the period t of an element $a \in \mathbb{Z}_N$, then we can factor N because from $a^t \equiv 1 \pmod{N}$, assuming t even, we get

$$(a^{t/2} - 1)(a^{t/2} + 1) \equiv 0 \pmod{N}.$$

If N does not divide $a^{t/2} + 1$ (note that N surely does not divide $a^{t/2} - 1$, since t is the period of a), then we factor N evaluating $\gcd(a^{t/2} - 1, N)$ and $\gcd(a^{t/2} + 1, N)$. The Shor's algorithm provides an efficient method exploiting quantum computing techniques and specifically the **quantum Fourier transform**.

Some factored RSA–numbers

- **RSA–129** (426 bits) factored by Atkin, Lenstra et al., with Quadratic Sieve, 1994.
- **RSA–155** (512 bits) factored by Montgomery, Zimmermann et al., with Number Field Sieve, 2000.
- **RSA–174** (576 bits) factored by Franke et al., with General Number Field Sieve, 2003. (prize of 10 000 dollars)
- **RSA–180** (596 bits) factored by Danilov and Popovyan, with General Number Field Sieve, 2010. (with three Intel Core i7)
- **RSA–240** (795 bits) factored by Zimmermann et al., with Number Field Sieve, November 2019. (https://lists.gforge.inria.fr/pipermail/cado-nfs-discuss/2019-December/001139.html)

Thank you!