International Journal of Scientific and Innovative Mathematical Research(IJSIMR)
Volume 9, Issue 1, 2021, PP 1-10
ISSN No. (Online) 2347-3142
DOI: https://doi.org/10.20431/2347-3142.0901001
www.arcjournals.org

Attack and Implementation of Public Key Cryptosystems by the Algorithm of Factorization

Francis MAYALA LEMBA*, Boniface ENGOMBE WEDI

Faculty of Science /Department of Mathematics and Computer Science / National Pedagogical University (UPN) Democratic Republic of Congo (DRC)

***Corresponding Author:** **Francis MAYALA LEMBA,** Faculty of Science /Department of Mathematics and Computer Science / National Pedagogical University (UPN) Democratic Republic of Congo (DRC)*

Abstract: *Coined by Whitfield Diffie and Martin Hellman, the concept of public key cryptography is in the origin of a new era of cryptology. After the description of sure schemes heuristically, the formalization of security notions and models enhanced the birth of proved security cryptography. The security of public key cryptography is based on difficult mathematical problems, in particular the theory of numbers such as the factorization of integers for RSA or the discrete logarithm for El Gamal.*

The public key cryptography is one of the most used cryptographical systems all over the world. It is found in on-line exchanges as well as within the owner application in order to provide a high level of confidentiality. Yet, the algorithm of cyphering must obey some fundamental rules to make sure these levels of protection. If one of these rules is not obeyed, the breaking of a key in a public key system or of a cyphered message would become possible under some conditions.

The aim of this article is to study the calculatorily difficult problems of factorization based on the public key cryptography and the algorithms which help to solve them.

Keywords: *Cryptanalysis, factorization, cryptography, cryptosystem, RSA, public key, private key.*

1. INTRODUCTION

In 1970's, the concept of public key cryptography emerges for the first time, it has helped to free from the delicate problem of the distribution of keys. Diffie and Hellman's [1] founder article which is based upon Merkle's ideas, has introduced the notion of heart of one-way function which is in the heart of this new cryptographical paradigm. Thanks to the concomitant development of the power of the calculation of computers, cryptologists were interested in mathematics to find such functions. Two main "candidates" were considered in that time: modular exponential function of which the opposite is called discrete logarithm [4], which is in the origin of Diffie-Hellman's exchange of key, El Gamal's cryptosystem and the schemes of signature which derive from it [1, 2], factorization, and the multiplication of two prime numbers on which is based the famous cryptosystem RSA [3].

The decomposition into prime factors of big integers (numbers composed of many tens figures) is nowadays again very hard to achieve. Although the power of computers which increases incessantly since more than a half-century, it is not always possible to factorize some numbers which have many hundred figures and of which the prime factors are few and of which the smallest of them has, it also, a biggest number of figures.

If one chooses two prime numbers sufficiently big (composed of many hundred figures each), it is very easy to multiply them among them with our actual strong computers, that will take only a little time less than a second. Moreover, once the result is obtained, it is nearly impossible to do its decomposition into prime factors, this could take a certain moment [3].

The paragraph below also rises many other matters on the security of a cryptosystem:

- Is it possible to factorize n which is an integer of big size into product of prime numbers?
- By factorizing n can we attack a public key cryptosystem?

The questions above are not easy and rose many researches [14, 15]. Whatever the model of attack used to break a public key cryptographical system; the best attack known against the algorithm of public key crypto graphical system remains the factorization of n.

If we can factorize n in a reasoning moment, hence we can calculate $\emptyset(n) = (p-1)(q-1)$. We find therefore d by inverting e modulo $\emptyset(n)$ by Euclid's extended algorithm. So, we can decipher all the message ciphered with (n, e). Hence, we have broken the system.

we have implemented an algorithm that helps to factorize an integer n produced from prime numbers to enhance the attack of a public key cryptosystem.

In the fallowing lines, we are going to present the public key cryptanalysis, that is, the theory of attacks of public key cryptography. A first approach in cryptanalysis consists in studying even in denying the calculatory hypothesis related to any public key cryptosystem.

2. PUBLIC KEY CYPHERING

2.1. INTRODUCTION

The concept of asymmetric on public key cryptosystem appeared for the first time in Whitfield and Martin Mellman's works [5].

In such a system, the cyphering function can be broadly expanded. Deciphering function is obviously kept secret by the receiver. Generally, it is he who chooses his deciphering function and expands to all the persons susceptible to communicate with him the associated cyphering function. Thus his only concern is to insure the authenticity of key (if Alice wants to send a message to Bob, she must to be sure that the key that she is going to use is indebtedly Bob's one). In addition, when n persons want to communicate together, it is only necessary to have n couples (secret key, public key) in an asymmetric system (one by person) instead of $n(n-1)/2$ keys in a traditional system (one by peer of persons).

Such a system cannot be unconditionally sure, since an attacker can in principle decrypt a message c by cyphering all the possible clear messages until he gets c.

So that such a system works, the receiver must know an information (the secret key) which must be very hard to be found by someone who only knows the public key. The construction of such a system can be based upon the existence of mathematical functions called "one-way functions" because they are easily calculatable but their opposite is practically impossible to calculate as more costly.

The receiver then chooses his secret key and uses a one-way function to calculate the public key that he expands. The construction of such a system can also be based upon the existence of trapping functions of which the opposite (and the function itself perhaps) is hard to calculate, except for someone who knows a secret information.

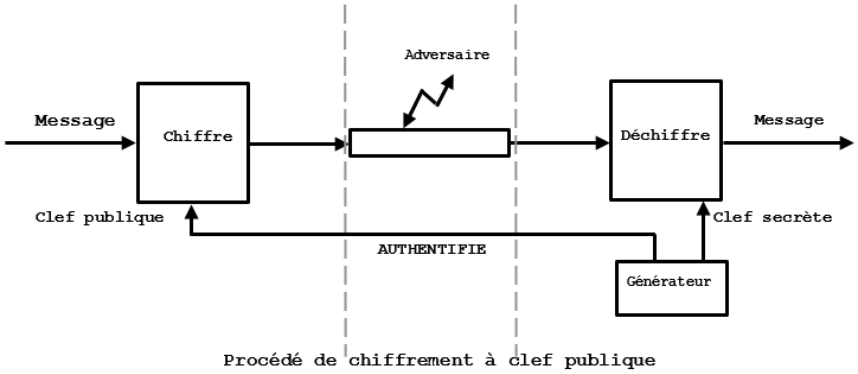
2.2. PARAMETERS OF A PUBLIC KEY CYPHERING

This is what Whithield Diffie and Martin Hellman conceived at Stanford University. Their researches were published in 1976 in a founder article of modern cryptography. Although it seems that investigations agencies have known such systems before these results, the first published instance of such a system goes back to 1978. It is owed to Ronald Rivest, Adi Shanir and Leonard Adleman of MIT: it is the RSA system [6].

A public key system of cyphering is composed of three algorithms: an algorithm which generates couples consisted of a secret key and a public key, an algorithm which helps to cypher a message by using a K_{pub} public key and an algorithm which helps to decipher a message by using a K_{sec} secret key. The functionality makes sure that if we generate one of K_{pub} and K_{sec} keys and if we decipher with K_{sec} a message cyphered with K_{pub}, we get the original message as shown clearly in the table below.

Intuitively, security makes sure the confidentiality of the clear message even when public key is known to the receiver. The public key cyphering is frequently used to start secure communications, for instance for on line trades [7, 8].

As both keys are distinct, the public key cyphering is also called asymmetric cyphering [9].



In public key cryptography where a user U has a couple (K_{pub}, K_{sec}) of keys related among them: the key K_{pub} is public, whereas the key K_{sec} is kept secret by U. We also defined asymmetric cyphering (also called public key cyphering): anyone who can cipher a message send to U, in using the public key K_{pub} of U. But only U is able to decipher cyphered messages in using his secret key K_{sec}.

In the following lines we are going to present public key cryptanalysis, that is, the theory of attacks of public key cryptography. A first approach in cryptanalysis consists in studying even in denying the calculatory hypothesis related to any public key cryptosystem.

3. Cryptographic Attack

3.1. Introduction

We are going to speak here about a technique of cryptanalysis. The general hypothesis to make is that the opponent, Oscar, knows the cryptographically system used. This is called Kerckoff's principle [9]. Of course, if Oscar does not know the process used, his task will be more difficult, but we do not wish to base the security of the system on the protection (uncertain, no doubt) of the description of cryptographic functions. The goal is therefore to study cryptographic systems according to Kerckoff's system.

An obvious attack of the public key cyphering consists to try to factorize n. If this can be done, it is easy to then calculate $\emptyset(n) = (p-1)(q-1)$ and to calculate the exponent of the deciphering a from b as Bob does it (it was presumed that breaking public key system is polynomially[1] equivalent to factorizing n, but this remains an open problem).

3.2. Elementary Attacks

The great advantage of these cryptosystems is that the cyphered text can be only deciphered if the attacker is able to factorize the public key n efficiently. These cryptosystems are probably sure enough methods of cyphering.

3.3. Limits of Factorization

Crypting and decriptating method for factorization is long in terms of calculations – Attack by factorization. This attack consists in finding the private key corresponding to the public key attacked, in factorizing the n into two prime numbers p and q and finding d in applying suitable functions. Nevertheless, this technique is too fastidious when n is very big. There exists an algorithm of factorization baptized Carl Promenance's QS (Quadratic Cribble) which is used if the number to factorize is not very big [10, 11].

4. Attack by Method of Factorization

4.1. Proposition

Either n a strictly positive integer, product of prime numbers superior to 2.

Either k the smallest perfect square superior or equal to n such as $k-n$ either also a perfect square. So, there exists a number $p = \left(\sqrt{k} - \sqrt{k-n}\right)$ which divides n.

[1] Two problems are polynomially called equivalent if any oracle that solves one of them can be used in a polynomial algorithm that solves the other.

4.2. PROOF

k perfect square, iff there exists $a \in \mathbb{N}$ tq $k = a^2$ (i)

$k - n$ perfect square if there exists $b \in \mathbb{N}$ tq $k - n = b^2$ (ii)

We must show that $p = \sqrt{a^2} - \sqrt{b^2} = a - b$ divides n.

In fact,

(i) $-$ (ii) $\Longrightarrow n = a^2 - b^2 = (a - b)(a + b) = p(a + b)$ then, p divides n. ■

4.3. COROLLARY

According to the theorem above, if $p = \left(\sqrt{k} - \sqrt{k - n}\right)$, then we have

$n = p.p' = \left(\sqrt{k} - \sqrt{k - n}\right)\left(\sqrt{k} + \sqrt{k - n}\right)$ (which $p' = \sqrt{k} + \sqrt{k - n}$);

if p and p' are note prime, we apply the theorem above and so forth until to get all the prime factors of n.

4.4. PROOF *(Example of Fermat's number factorization by factorization method)*

We take the same number which has been given by Fermat 2 027 651 281, it is asked if it is prime or composed, and what numbers it is composed of, if it is.

We note $n = 2\ 027\ 651\ 281$ et $k = b^2 + n \Longrightarrow k = 1040400 + 2027651281$

$k = 2028691681$

$n = p.p' = \left(\sqrt{k} - \sqrt{k - n}\right)\left(\sqrt{k} + \sqrt{k - n}\right)$

$$n = p.p' = \left(\sqrt{2028691681} - \sqrt{2028691681 - 2\ 027\ 651\ 281}\right)\left(\sqrt{2028691681} + \sqrt{2028691681 - 2\ 027\ 651\ 281}\right)$$

$n = (45041 - 1020)(45041 + 1020) \Longrightarrow n = 44021$ x 46061

Hence $p = 44021$ and $q = 46061$ ■

4.5. RSA AND ALGORITHM OF FACTORIZATION

In this attack, we will need only the n to find the secret key.

Table: *RSA and Attack by the Factorization Algorithm*

RSA	*Algorithm of factorization*
$\boldsymbol{n = 85}$, $\boldsymbol{p = 5}$ and $\boldsymbol{q = 17}$ $\boldsymbol{\emptyset(n) = (p - 1)(q - 1) = 16 \times 4 = 64}$ $\boldsymbol{e \Longrightarrow pgcd\left(e, \emptyset(n)\right) = 1 \Longrightarrow e = 19}$ $\boldsymbol{d \times e \equiv 1\ (mod\ \emptyset(n)) \Longrightarrow d = 27}$ So, the public key is $(\boldsymbol{n, e})\Longrightarrow$ (85, 19) and the private key is $(\boldsymbol{n, d}) \Longrightarrow$ (85, 27)	Going from the public key (n, e)$\Longrightarrow$ (15, 7) $n = 85$ *et* $k = b^2 + n \Longrightarrow k = 36 + 85$ $k = 121$ $n = p.p' = \left(\sqrt{k} - \sqrt{k - n}\right)\left(\sqrt{k} + \sqrt{k - n}\right)$ $n = p.p' = \left(\sqrt{121} - \sqrt{121 - 85}\right)\left(\sqrt{121} + \sqrt{121 - 85}\right)$ $n = (11 - 6)(11 + 6) \Longrightarrow n = 5 \times 17$ Hence $p = 5$ *and* $q = 17$ $\emptyset(n) = (p - 1)(q - 1) = 16 \times 4 = 64$ $d \times e \equiv 1\ (mod\ \emptyset(n)) \Longrightarrow d = 27$ So, the private key is $(n, d) \Longrightarrow$ (85, 27)

5. IMPLÉMENTATION OF ALGORITHM

5.1. PRESENTATION OF PYTHON TECHNOLOGIES [12]

Computer revolution is born in a machine. Our programmation languages tend therefore to resemble this machine.

But computers are not as machines as tools at disposal of the mind and a new means of expression. So, these tools start to resemble less machines and more the parts of our brain or other forms of expression such as scripture, paint, sculpture or realization of films. The language of Python programmation is part of this movement which uses computer as means of expression.

Python is an interpreted, multiparadigm and multiplatform language of programmation. It enhances the structured, functional and oriented imperative programmation of the object. It is equipped with a strong dynamic typing, an automatic management of ramasse-miettes memory and an exceptional management system; so, it is similar to Perl, Ruby, Scheme, Smallkalk and Tcl. The Python language is placed under a free license close to BSD license and works on most computer platforms, smartphones with central computers, Windows to Unix with notably GNU/Linux passing through macOs; or again Android, iOS and can be also translated into Java or .Net. It is designed to optimize programmers' productivity in providing high level tools and an easy syntax to use [13].

It is also praised by some pedagogists who find there a language in which syntax, clearly separated from low level mechanisms, facilitates an easy initiation to basic concepts of programmation.

5.2. IMPLEMENTATION OF ALGORITHM

```
import math
#-----------------------------------------------------------
#La fonction permet de générer le "P" à partir de "N"
def genereP(n) :
    #Calcule de racine carrée de de "N" et l'affectation de "P"
   p = int(math.sqrt(n))
   #La Réaffectation de "P" en respectant la condition ci-dessous :
   if p * p < n :
      p = p + 1
   #La Réaffectation de "P" en respectant la condition ci-dessous :
   while int(math.sqrt(p * p - n)) - math.sqrt(p * p - n) != 0 :
      p = p + 1
   #Génération de "P"
   return math.sqrt(p * p) - math.sqrt((p * p)-n)
#-----------------------------------------------------------
#-----------------------------------------------------------
#Calcule de PGCD de deux entier "A et B"
def pgcd(a,b) :
   while b != 0 :
      r = a % b
      a , b = b , r
   return a
#-----------------------------------------------------------
```

```
#La fonction permet de générer le "E" à partir de "FideN(qdn), P et Q"
def getE(qdn,p,q) :
    i = 2
    #simulation de Do while
    flag =True
    while flag  :
        i = i + 1
        if i > p and i > q :
            if pgcd(i,qdn) == 1 :
                flag = False
    return i
#-----------------------------------------------------------
#La fonction permet de générer le "D" à partir de "E, FideN(qdn), P et Q"
def getD (e,qdn,p,q):
    j = 0
    flag2 = True
    while flag2 :
        j = j + 1
        if j > p and j > q :
            if ((j* e) % qdn) == 1 :
                if e != j :
                    flag2 = False
    return j
#-----------------------------------------------------------
#Récupération de valeur saisie par l'utilisateur en chaine de caractères
st= input("Saisir un nombre entier : ")
#Convertion de la valeur saisie en entier avec la fonction int()
n = int(st)
#LA variable "NN" stock ne "N" temporairement
nn = n
#Création d'un tableau de 10000 cellule
t = [0] * 10000
#Initialition de "I" et "Q"
i = 0
q = 0
#Recuperation de "N" s'il est paire et initialisation de la première cellule de notre tableau par 2
while n % 2 == 0 :
    n = n / 2
```

```
    t[i] = 2
    i = i + 1
#Variable utilitaire
#-------------------------------------------------
#Initialisation de nouvelle valuer de n modifier ci-haut
A = n
c = 1
#Recupération de "I"
j = i
#Simulation de Do while
flag = True
while flag :
    #Récuperation de "P" apartir de "N"
    p = genereP(n)
    #Ajout des valeurs dans notre tableau si "P" égale à 1
    if p == 1 :
        t[j] = n
        #Récupération de "C, N, J"
        c = c * n
        n = A / c
        j = j + 1
    #Ajout des valeurs dans notre tableau si la génération de "P" par lui-même est égale à 1
    elif genereP(p) == 1 :
        t[j] = p
         #Récupération de "C, N, J"
        c = c * p
        n = A / c
        j = j + 1
    #sinon notre réaféctation de "N" par "A / P"
    else :
        n = A / p
    #si "N" est égale a 1 on sort dans la boucle while
    if n == 1 :
        flag = False
#La trie de notre tableau "T" en alignant les valeurs de plus petit au plus grand
t.sort()
#Affichage du resultat
print("Decomposition en facteurs premiers de ", nn , " est : ")
```

```
#Initialisation de "I"
i = 0
#Pour chaque items(valuer) de "T" qui est notre tableau on exucute le code ci-dessous
for valeur in t :
    #si la valeur du cellule(Item) est différent de 0
    if valeur != 0 :
        i = i + 1
        if i == 1:
            p = int(valeur)
        elif i == 2:
            q = int(valeur)
#----------------------------------------------------------
if nn % 2 != 0 :
    #si "N" est premier
    if p != 0 and q != 0:
        qdn = (p-1)*(q-1)
        e = getE(qdn,p,q)
        d = getD(e,qdn,p,q)
        print("p =", p)
        print("q =", q)
        print("qdn =", qdn)
        print("e =",e)
        print("d =",d)
    #sinon
    else :
        print(nn," est premier !")
else :
        print(nn," est paire")
#appuyer sur le clavier pour arrêter
input()
```

6. CONCLUSION

This paper's goal is to study the calculatorily difficult problem of the factorization based on public key cryptography to facilitate the attack.

The decomposition into prime factors of big integers (numbers composed of many tens figures) is again nowadays very difficult to achieve. Despite the power of computers which increased since more than a half-century, it is not always possible to factorize some numbers which have many hundreds figures and of which prime factors are few numerous and of which the smallest of them has, it also, a biggest number of figures.

If we choose two prime numbers sufficiently big (composed of hundreds figures each), it is very easy to multiply them among them with our present computers, this will only take a little time (less than a

second). On the contrary, once the result is got, it is nearly impossible to achieve its decomposition into prime factors, this could take a certain time.

We have presented a factorization attack through this algorithm (a factorization algorithm) and an implementation with Python's programmation language. This algorithm represents a methodology or an attacker's algorithm of factorization before a public key cryptosystem.

It begins to look for information about the public key and then elaborate an algorithm to find p and q secret prime numbers but necessary to deduce the private key.

This algorithm provides a easy solution to one of more difficult problems in all cryptography, that is, the problem of factorization of n into product of two prime numbers.

REFERENCES

[1] Diffie W. and Hellman M. E., New directions in cryptography. IEEE Trans. Information Theory, ITa-22(6) :644{654, 1976.

[2] El Gamal T., A public key cryptosystem and a signature scheme based on discrete logarithms. In Advances in cryptology - CRYPTO 1984, volume 196 of Lecture Notes in Comput. Sci., pages 10{18. Springer, Berlin, 1985.

[3] R. L. Rivest, A. Shamir et L. Adleman. « A method for obtaining digital signatures and public-key cryptosystems ». In : Communications of the ACM 21.2 (1978), p. 120–126 (cité p. 1).

[4] Boneh D. and Venkatesan R., Rounding in lattices and its cryptographic applications. In Proc. of the 8th Symposium on Discrete Algorithms, pages 675–681. ACM, 1997.

[5] W. DiFFiE, M.E. Hellman : New directions in cryptography. IEEE Transactions on Information Theory, 22 A976), 644-654.

[6] Jeffrey Hoffstein, Jill Pipher, Joseph H. Silverman, An Introduction to Mathematical Cryptography, 2008, springer, New York (USA).

[7] Rivest R. L., Cryptography. In Handbook of Theoretical Computer Science volume A, chapter 13. Ed. Elsevier, 1990.

[8] Joux A., La Réduction des Réseaux en Cryptographie. PhD thesis, Ecole Polytechnique, 1993.

[9] Guillaume J., Dumas JL., Roch É., Tannier S., Varrette, correction, théorie des codes Compression, cryptage, Dunod, Paris, 2007 ISBN 9-78-210-050692-7

[10] Ainigmatias Cruptos,"Attaque de RSA par fractions continues", Acrypta Hostalot DL.,"Attaque par factorisation contre RSA",2007.

[11] Stinson D. R., Some baby-step giant-step algorithms for the low Hamming weight discrete logarithm problem. Math. Comp., 71(237) :379–391 (electronic), 2002.

[12] Python et Django, « Principes et bonnes pratiques pour les sites web dynamiques », Edition Eyrolles, Paris, 2018.

[13] Tarek Ziadé, Programmation Python « Conception et optimisation », Edition Eyrolles, Paris, 2009.

[14] Pierre Colmez, Éléments d'analyse et d'algèbre et de théorie des nombres, Palaiseau : Presses internationales polytechnique, 2009.

[15] Jean -Paul Pier, Development of mathematics 1950-2000, Edition Birkhäuser Verlag, Boston, 2000.

AUTHOR'S BIOGRAPHY

MAYALA LEMBA Francis, he teaches in the "Université Pédagogique Nationale", Democratic Republic of the Congo. His research (Cryptography).

ENGOMBE WEDI Boniface is a Professor in the Department of Mathematics and Computer Science, in the Faculty of Science, at the "Université Pédagogique Nationale", Democratic Republic of the Congo.

Citation: *Francis MAYALA LEMBA & Boniface ENGOMBE WEDI., Attack and Implementation of Public Key Cryptosystems by the Algorithm of Factorization, International Journal of Scientific and Innovative Mathematical Research (IJSIMR), vol. 9, no. 1, pp. 1-10, 2021. Available : DOI: https://doi.org/ 10.20431/2347-3142.0901001*