Ibn Al-Haitham Journal for Pure and Applied science https://doi.org/ 10.30526/2017.IHSCICONF.1805

Counting Functions to Generate the Primes in the RSA Algorithm and Diffie-Hellman Key Exchange

Faez Ali AL-Maamori
Faez352013@gmail.com
Mazin Saied Rashid
mazinalsadawy1@gmail.com
Dept. Of Mathematics / College of Education for Pure Science/ University of Babylon

Abstract

The Rivest–Shamir–Adleman (RSA) and the Diffie-Hellman (DH) key exchange are famous methods for encryption. These methods depended on selecting the primes p and q in order to be secure enough. This paper shows that the named methods used the primes which are found by some arithmetical function .In the other sense, no need to think about getting primes p and q and how they are secure enough, since the arithmetical function enable to build the primes in such complicated way to be secure. Moreover, this article gives new construction of the RSA algorithm and DH key exchange using the primes p,qfrom areal number x.

Keywords: (RSA algorithm and Diffie-Hellman exchange, Beurling Primes, Analytic number theory).

1. Introduction

Since the seventieth of the last century, several authors introduced a cryptographic algorithm in order to replace the less secure algorithms "Diffie and Hellman", who described the idea of such an algorithm, but never truly developed it. The RSA algorithm and DH-key exchange cryptosystems (as well as digital signatures) are regarded as the important security schemes (see for instance (A. Salomaa.[8]),(J. Coron .[5]) and (S.D. Galbraith.[6]).

The majority is to know the security algorithms that focused on generating the primes which is called the code of the encryption. The main aim of this work is to secure enough the mentioned algorithms and the way for building the prime from a challenging way that no one can predict it. Here, the principle objective is to overlook the utilization of the messenger with a specific end goal to convey keys to the recipient over another protected channel before transmitting the initially planned data or message. The more vital thing here is that the keys ought to be worked in such a way that the unscrambling key may not be effortlessly found from the RSA calculation and DH-key. In this article, the work has been done concentrated on how could we make the algorithm secure strongly such that one can say it breaks down it. The main body of this work shows that,there is an arithmetical function which generates a prime number p from a large enough real number x and uses that prime in the code of the algorithm. In this work, we generate the prime p by using the arithmetical function $F^*(x)$.

2. Materials and Methods

The generating function $F^*(x)$. Let $\mathcal{P} = \{2,3,5,7,\dots\}$be a set of primes, while the set of natural number is $\mathcal{N} = \{1,2,3,4,5,6,7,\dots..\}$The counting functions of prime and integers are defined as follows:

Definition

Let $\mathcal{P} = \{2,3,5,7,\dots..\}$. The counting function for any positive real x is defined to be:

$$H(x) = \sum_{p^k \le x} \log p, \text{ where } k \in N \text{ and } p \in \mathcal{P}.$$

We can write $H(x)$ as follows:

$H(x) = \sum_{k=1}^{\infty} \sum_{p \le x^{\frac{1}{k}}} \log p = \sum_{k=1}^{\infty} h\left(x^{\frac{1}{k}}\right)$,(see[1]for more details).

$Where\ \ h(x) = \sum_{p \le x} \log p$. Now using the MobiusInversion Formula, we get

$$F(x) = \sum_{n=1}^{\infty} \mu(n)\, H\left(x^{\frac{1}{n}}\right),$$

where $\mu(n)$ is $mäbius$ function. The counting function H(x) is represented as the integer part of and positive x minus 1,this tells us that the F(x) would be the sum of product $\mu(x)$ and the integer part of and positive x minus 1. Where the sum is running from 1 to infinity.

We note that the sum is vanished when the index is greater than $\frac{log\ x}{\log 2}$.In the other words, F(x) = g(x) +zero, where the g(x) means the sum of product $\mu(x)$ and the integer part of positive x minus 1 for the index is less than or equal to $\frac{log\ x}{\log 2}$.

The function F(x) is increasing step function. So, choosing a (large enough) real number x gives two options on the F(x) which are given as follows: Either the output of F(x) is a prime number or a natural number.

The reader must be familiar with proof of F(x) being increased and step function. For more details, one can see [2].

Moreover, if we define D_F= F(x)-F(x-1), then D_F must be either 1 or 0. If one can calculatethe jump of F(x) he can see that,

$$F(d) - F(d-1) = \begin{Bmatrix} 1 \; if \; d \neq n^k, n, k > 1 \; and \; d \geq 2 \\ 0 \qquad\qquad\qquad otherwise \end{Bmatrix}.$$

More details, one can see [1]. Therefore, getting a prime p from a real x is guaranteed by taking the power, which is the characteristic function $m_{F(x)}$,of the output of F(x)to the characteristic function $m_{F(x)}$. This means that $F^*(x) = (\mathrm{F(x)})^{\boldsymbol{m}_{F(x)}}$, where the characteristic function is defined by

$$\boldsymbol{m}_{\mathrm{F(x)}} = \begin{Bmatrix} 1 \; if \; F(x) \neq k \cdot l \;\; for \; any \; k,l > 1 \; in \; N, \\ 0 \; if \; F(x) = k \cdot l \;\; for \; some \; k,l > 1 \; in \; N. \end{Bmatrix}$$

Therefore, the step function $F^*(x)$ is either zero or prime number. The mathematical technique shows that picking a prime number from a large real number Is not easy without knowing the function F(x).

3. Example and its applicable algorithm

Here, we give a very simple example in order to show how the method works:

Choice x = 41.5.

By definition 2.1, we get

F (41.5) = M(1) ([$41.5^{1/1}$]-1) + M(2) ([$41.5^{1/2}$]-1) + M(3) ([$41.5^{1/3}$] -1) + M(4) ([$41.5^{1/4}$] -1) + M(5) ([$41.5^{1/5}$] -1) (The function F(x) is vanished for n > $\frac{\log x}{\log 2}$).

= 40 – 5 – 2 + 0 – 1 = 32 not prime.

By corollary 2.7, we get

$\hat{F}(41.5) = (32)^{\gamma_{32}}$,where γ_{32}=0

$= (32)^0$

= 1.

Program (1): Shows generating the prime p from a real number x:

```
clear
clc
x=24.5;
n=1; t=1; sum=0;
while (t)
n
if n>(log2(x)/log2(2))
break
end
if n==1
mobn=1;
else
if isprime(n)
mobn=-1;
else
n1=n; base=2; k=1;
while n1>1
while mod(n1,base)~=0
base=base+1;
end
a(k)=base;
n1=n1/base;
base=2;
k=k+1;
end
t=1;
for i=1:k-1
if a(i)~=a(1)
t=2;
end
end
if t==1
mobn=0;
else
if mod(k,2)==0
mobn=1;
else
mobn=-1
end
end
end
end
mobn
xn=fix(x^(1/n));
sum=sum+(mobn*(xn-1));
n=n+1;
```

end

4. Discussion of the results and future works:

The simple example given above was just to reflect the hard work of building the prime p from a large real x. One could use this approach in various manners and subject especially in security branches.

References

[1] T. M. Apostol, Mathematical Analysis Second Edition, Addison-Wesley, 1974.

[2] Luma. N.M. Tawfiq, Numerical solution of singular perturbation problems by using collocation neural networks, MJ Journal on Numerical Analysis, 1 (1) (2017) 1-7.

[3] Q. Baodong, "Leakage-resilient lossy trapdoor functions and public-key encryption." Proceedings of the first ACM workshop on Asia public-key cryptography. ACM, 2013.

[4] Q. Baodong, and S. Liu, "Leakage-flexible CCA-secure public-key encryption: simple construction and free of pairing." International Workshop on Public Key Cryptography. Springer Berlin Heidelberg, 2014.

[5] J. Coron, "Fully homomorphic encryption over the integers with shorter public keys." Annual Cryptology Conference. Springer Berlin Heidelberg, 2011.

[6] S.D. Galbraith. Mathematics of public key cryptography. Cambridge University Press, 2012.

[7] B. Persis Urbana, Purshotam Mandiwa, and M. Kumar, "A modified RSA cryptosystem based on 'n'prime numbers." International Journal Of Engineering And Computer Science 1 (2012): 63-66.

[8] A. Salomaa, Public-key cryptography. Springer Science & Business Media, 2013.

[9] L.N.M.Tawfiq and R.S.Naoum, (2007). Density and approximation by using feed forward artificial neural networks, Ibn Al-Haitham Journal for Pure & Applied Sciences, **20**(1): 67-81.