Reason - A Technical Magazine
Vol. – VIII, 2007 & 2008

NEW KEY GENERATION TECHNIQUE IN RSA ALGORITHM

Satyendra Nath Mandal*, Abhijit Mondal, Nil Kamal Basak**, Souvik Naskar** and Subhadip Bhattacharya****

*Lecturer, Department of IT, Kalyani Govt. Engineering College

**Student, B.Tech. 4th Year, Department of IT, Kalyani Govt. Engineering College

ABSTRACT : The RSA cryptosystem, invented by Ron Rivest, Adi Shamir and Len Adleman was first publicized in the August 1977 issue of Scientific American. The cryptosystem is most commonly used for providing privacy and ensuring authenticity of digital data. These days RSA is deployed in many commercial systems. It is used by web servers and browsers to secure web traffic, it is used to ensure privacy and authenticity of e-mail, it is used to secure remote login sessions, and it is at the heart of electronic credit-card payment systems. In short, RSA is frequently used in applications where security of digital data is a concern. Symmetric key distribution problem is solved by it.

Goal of this work is to derive a new prime generation method with the help of current system time. That is why the method generates a unique prime every time the program is executed.

1. INTRODUCTION TO SECURITY

"The art of war teaches us to rely not on the likelihood of the enemy's not coming, but on our readiness to receive it; not on the chance of his not attacking, but rather on the fact that we have made our position un assailable."

– *The Art of War, Sun Tzu*

The requirements of information security within an organization have undergone two major changes in the last several decades. Before the widespread use of data processing equipment the security of information failed to be valuable to an organization was provided primarily by physical and administrative means.

With the introduction of the computer the lead of automated tools for protecting files and other information stored on the computer became evident, especially the case for a shared system. The second major change comes with use of the network and communication facilities for carrying data between terminals.

No one can deny the importance of security in data communications and networking. Security in networking is based on cryptography, the science and art of transforming messages to make them secure and immune to attack. Cryptography can provide several aspects of security related to the interchange of messages through networks. These aspects are confidentiality, integrity, authentication and no repudiation.

Network security is mostly achieved through the use of cryptography, a science based on abstract algebra.

2. CRYPTOGRAPHIC TECHNIQUES

Cryptography, a word with Greek origins, means "secret writing". However, we use the term to the science and art of transforming messages to make them secure and immune to attacks. There are several techniques to implement cryptography. These methods will be described in the following sections.

2.1 Plain text And Cipher text

Plaintext:

The original message, before transformed, is called plaintext. After the message is transformed, it is called cipher text. An encryption algorithm transforms the plaintext into cipher text; a decryption algorithm transforms the cipher text back into plaintext. The sender uses an encryption algorithm, and the receiver uses a decryption algorithm.

2.2 Technique of cryptography

i) Substitution techniques
ii) Transposition techniques

2.3 Substitution Techniques

The substitution techniques are –

2.3.1 Caesar Cipher :

It is a special case of substitution techniques where each alphabet in a message is replaced by an alphabet three places down the line.
Example= plain text 'ATUL' will become cipher text 'DWXQ'.

2.3.2 Modified Version of Caesar Cipher :

In this scheme the cipher text alphabets corresponding to the original plain text alphabet is not necessarily be three places down the order, but instead, can be any places down the order.

Example= the plain text alphabet 'A' can be replaced by any alphabet i.e., from 'b' to 'z'.

So this scheme has 25 possibilities of replacement.

This method is susceptible to "BRUT FORCE ATTACK".

2.3.3 Mono-alphabetic cipher :

Rather than using a uniform scheme for all the alphabets in a given plain text message, we decide to use random substitution.

This means 'A' can be replaced by any other alphabet ('B' through 'Z'),

Each 'B' can also be replaced by any other random alphabet ('A' or 'C' through 'Z').
This is also susceptible to attack.

2.3.4 Homophonic Substitution Cipher :

It is similar to monoalphabatic cipher. Like a plain substitution cipher technique we replace one alphabet with another in this scheme. However the difference the two techniques is that the replacement alphabet set in the case of simple substitution technique is fixed, where as in the case of this cipher one plain text alphabet can map to more than one cipher text alphabet. For instance, 'A' can be replaced by 'D', 'H', 'P'. 'B' can be replaced by 'E', 'I', 'Q'.

2.3.5 Polygram Substitution Cipher :

Polygram substitution cipher technique replaces one block of a plain text with a block of cipher text- it does not work on a character by character basis. For instance 'HELLO' could be replaced by 'YUQQW'. But 'HELL' could be replaced by totally different cipher text block 'TEUI'.

2.3.6 Polyalphabatic Substitution Cipher :

This cipher uses multiple one character key. Each of the keys encrypts one plain text character.

The first key encrypts the first plain text character; and the second key encrypts the second plain text character, and so on. After all the keys are used they are recycled. Thus if we have 30 one letter keys, every 30th character in the plain text would be replaced with the same key. This number is called as the period of the cipher.

2.4 Transposition Technique

Transposition techniques differ from substitution techniques in the way that they do not simply replace one alphabet with another: they also perform some permutation over plain text alphabets.

In this category there are the following types of techniques:

- Rail Fence Technique
- Simple Columnar Transposition Technique
- Vernam Cipher (One-Time Pad)
- Book Cipher / Running Key Cipher

2.5 Encryption and Decryption

The process of encoding plain text messages into cipher text messages is called encryption.

The reverse process of transforming cipher text messages back to plain text is called decryption.

2.6 Symmetric Key Cryptography

If the same key is used for encryption and decryption, then the mechanism is called Symmetric Key Cryptography.

Cipher text = encrypt (plaintext, key)
Plaintext = decrypt (cipher text, key)

Symmetric key cryptography is useful if one wants to encrypt files on his computer, and he intends to decrypt them himself. (But if one intends to send them to someone else to be decrypted, then he has a "key distribution problem".) Regarding security, the authors assume the encryption *algorithms* that were chosen to use are publicly known; only the key is secret to the participants.

2.7 Asymmetric Key Cryptography

If two different keys are used in a cryptographic mechanism, wherein one key is used for encryption, and another, different keys are used for decryption; the mechanism is called Asymmetric Key Cryptography.

3. SYMMETRIC KEY CRYPTOGRAPHY

3.1 Algorithm Types and Modes

Algorithm Types :

Types are shown below as diagrammatically:

- Stream Cipher –
  In Stream Ciphers, the plain text is encrypted one bit at a time.
- Block Cipher –
  In Block Ciphers, rather than encrypting one bit at a time, a block of bits is encrypted at one go.

  Algorithm Modes

3.2 Data Encryption Standard (DES)

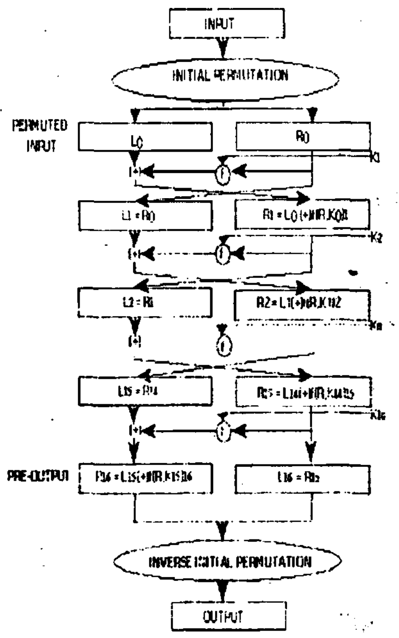
The Data Encryption Standard is a cipher (a method for encrypting information) selected as an official Federal Information Processing Standard (FIPS) for the United States in 1976, and which has subsequently enjoyed widespread use internationally. The algorithm was initially controversial, with classified design elements, a relatively short key length, and suspicions about a National Security Agency (NSA) backdoor. DES consequently came under intense academic scrutiny, and motivated the modern understanding of block ciphers and their cryptanalysis.

How DES works?

DES is a block cipher. It encrypts data in blocks of size 64 bits each. That is 64 bits of plain text goes as the input to DES , which produces 64 bits of cipher text .The same algorithm and key are used for encryption and decryption, with minor difference.

Steps of DES : –

- In the first step, the 64-bit plain text block is handed over to an Initial Permutation (IP) function.
- The initial Permutation is performed on plain text.



- Next, the Initial Permutation (IP) produces two halves of the permutation block; say Left and Right Plain Text (RPT).
- Now, each of LPT and RPT goes through 16 rounds of encryption process, each with its own key.
- In the end, LPT and RPT are rejoined, and a Final Permutation (FP) is performed on the combined block.
- The result of this process produces 64-bits cipher text.

Data Encryption Standard steps

◎ Round face of DES :

Step - 1 : Key Transformation

It is noted that the initial 64-bit key is transformed into a 56-bit key by discarding every initial key. Thus, for each round, a 56-bit key is available from this 56-bit key, a different 48-bit is **sub-key** generated during each round using a process called as key transformation. For this, the 56-bit key is divided into two halves, each of 28 bits.

Step - 2 : Expansion Permutation

Recall that after Initial Permutation, we had 32-bit plain text area, called Left Plain Text and Right Plain Text. During expansion permutation, the RPT is expanded 32 bits to 48 bits. Besides increasing the bit size from 32 to 48, the bits are permuted as well, hence the name expansion permutation.

Step - 3 : S-box Substitution

S-box substitution is a process that accepted the 48-bit input from the XOR operation involving the compressed key and expanded RPT, and produces a 32 bit output using the substitution technique.

Step - 4 : P-box Permutation

The output of S-box consists of 32 bits. These 32 bits are permuted using a P-box. This straightforward permutation mechanism involves simple permutation (i.e. replacement of each bit with another bit, as specified in the P-box table, without any expansion or compression).This is called as **P-box Permutation.**

Step - 5 : XOR and Swap

Note that the authors have been performing all these operation only on the 32-bit right half portion of the 64-bit original plain text. The left half portion was untouched so far. At this juncture the left half portion of the initial 64-bit plain text block is XORed with the output produced by P-box permutation. The result of this XOR operation becomes the new right half. The old right half becomes the left half, in a process of Swapping.

Variations of DES

- Double DES-Double DES is quite simple to understand. Essentially, it does twice what DES normally does only once. Double DES used only two keys, say k1 and k2.It first performs DES on the original plain text using k1 to get the encrypted text. It again performs DES on the encrypted text, but this time with the other key, i.e. k2.
- Triple DES

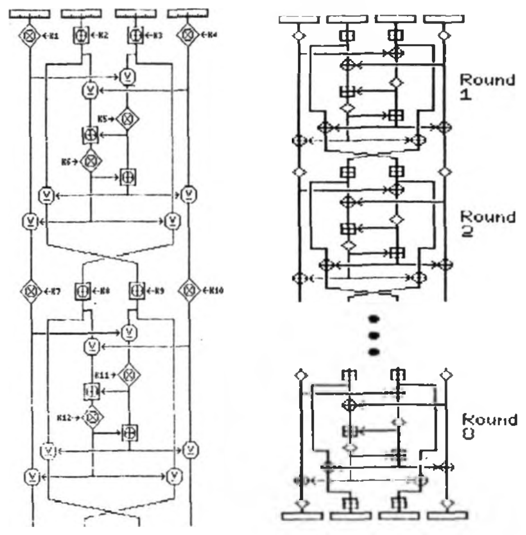
  Although the meet-in-the-middle attack on Double DES is not quite practical yet, in cryptography, it is always better to take the minimum possible chances. Consequently; Double DES seemed inadequate, paving way for Triple DES. As one can imagine, Triple DES is DES-three times. It comes in two flavors: one that uses three keys, and the other that uses two keys.

3.3 Internation Data Encryption Standard

In cryptography, the International Data Encryption Algorithm (IDEA) is a block cipher designed by Xuejia Lai and James Massey of ETH Zurich and was first described in 1991. The algorithm was intended as a replacement for the Data Encryption Standard. IDEA is a minor revision of an earlier cipher, PES (Proposed Encryption Standard); IDEA was originally called IPES (Improved PES).

The cipher was designed under a research contract with the Hasler Foundation, which became part of Ascom-Tech AG. The cipher is patented in a number of countries but is freely available for non-commercial use. The name "IDEA" is also a trademark. The patents will expire in 2010–2011. Today, IDEA is licensed worldwide by MediaCrypt.

IDEA was used in Pretty Good Privacy (PGP) v2.0, and was incorporated after the original cipher used in v1.0, BassOmatic, was found to be insecure.[1] IDEA is an optional algorithm in the Open PGP standard.



IDEA steps

Operation : –

IDEA operates on 64-bit blocks using a 128-bit key, and consists of a series of eight identical transformations (a *round*, see the illustration) and an output transformation (the *half-round*). The processes for encryption and decryption are similar. IDEA derives much of its security by interleaving operations from different groups — modular addition and multiplication, and bitwise exclusive OR (XOR) — which are algebraically "incompatible" in some sense. In more detail, these operators, which all deal with 16-bit quantities, are –

- Bitwise eXclusive OR (denoted with a blue).
- Addition modulo 2^{16} (denoted with a green).
- Multiplication modulo $2^{16}+1$, where the all-zero word (0x0000) is interpreted as 2^{16} (denoted by a red).

Key schedule

The first 6 keys-parts (K_1 through K_6) for the first round are taken directly as the first 6 consecutive blocks of 16 bits (MSB being the first). This means that only 96 of the 128 bits are used in each round. After each round, including before the concluding half-round, the 128 bit key undergoes a 25 bit rotation to the left, i.e. the LSB becomes the 25th LSB. Then the 6 key-parts are again taken from the first 6 consecutive blocks of 16 bits.

3.4 Disadvantages of Symmetric Key Cryptography

- The key distribution problem is inherently link with the symmetric key operation.
- Based on linear approximations linear cryptanalysis attack can be take place.
- Similarly differentials cryptanalysis attack can be take place.

4. ASYMMETRIC KEY CRYPTOGRAPHY

4.1 Overview of RSA key Cryptography

In 1977 Ron Rivest, Adi Shamir and Len Adleman [RSA 78] introduced an important public key crypto-system based on computing modular exponentials. The security of RSA cryptography ultimately lies on our inability to effectively factor large integers. As a point in case, [LLMP 92] used hundreds of computers world-wide for a number months in order to explicitly factor the 9-th Fermat number ***Fg*** = 2^2^9 + 1, a 513 bit integer. Admittedly this factorization uses special properties of ***FS*** and fully general techniques for factoring numbers over 512b are even slower. Nevertheless, RSA implementations with key lengths of 512b must be prepared to renew their keys regularly and cannot be used for reliably transmitting any data which must remain secret for more than a few weeks. Longer keys (7683 or1 Kb) appear safe within the current state-of-the-art on integer factoring.

4.2 The RSA Algorithm

```
RSA_Key_Generation
{
    Select two large primes p and q such that p‘eq
    n ← p*q
    Ø(n) ← (p-1)*(q-1)
    Select e such that 1<e< Ø(n) and e is co prime to Ø(n)
    d ← e^ (-1)*mod Ø(n)   //d is inverse of modulo Ø(n)
    Public_key ← (e, n) //To be announced publicly
    Private_key ← d    //To be kept secret
    Return Public_key and Private_key
```

◎ Algorithm for RSA encryption

```
RSA_Encryption (P, e, n) //P is the plain text and p<n in Zn
{
        C ← Fast_Exponentiation(P, e, n)
                        //Calculation of (P^e mod n)
        Return C
}
```

◎ Algorithm for RSA decryption

```
RSA_Decryption(C, d, n)   //C is the cipher text in Zn
{
        P ← Fast Exponentiation(C, d, n)
                        //Calculation of (C^d mod n)
        Return P
}
```

In RSA, p and q must be at least 512 bits; n must be at least 1024bits.

◎ Proof of RSA

It can be proved that encryption and decryption are inverses of each other using Euler's theorem

If n=p*q, a<n, and k is an integer, then a^(k* Ø(n)+1)equivalent to a(mod n)

Assume that the plain text retrieved b Bob is P1 and prove that it is equal to P.

P1=C^d mod n=(P^e mod n)^d mod n=P^(e* d) mod n
e*d=k *Ø(n)+1
P1=P^(e*d) mod'!P1=P^(k *Ø(n)+1) mod n
P1=P^(k *Ø(n)+1) mod n=P mod n

◎ Trivial example

p=7, q=11   calculate n=7*11=77
So, Ø(n)=(7-1)*(11-1)=60
Choose e=13, then d=37.

Note that e*d mod 60=1 (they are inverse of each of other)

Now imagine that Alice wants to send the plain text 5 to Bob. She uses the public exponent 13 to encrypt 5.

Plain text: 5   C=5^13 mod 77=26   Cipher text=26
Bob receives the cipher text 26 and uses the private key 37 to decipher the cipher text:
Cipher text: 26   P=26^37 mod 77=5   Plaintext=5
The plaintext 5 sent by Alice is received as plaintext 5 by Bob.

4.3 Advantages of RSA

◎ ClassicSys as a standard

Besides ClassicSys ciphering at high speed, two more advantages make ClassicSys prime candidate for THE standard application in cryptography:

1. ClassicSys uses only 1 secret key to meet ALL the cryptographic needs of an end-user such as:

- To authenticate himself
- To authenticate messages with a time reference
- To generate all the SessionKeys he needs for Email (as one possible application)
- To generate several keys for other applications: banking, electronic commerce, electronic voting, casino games at home,

2. ClassicSys is designed in such a way that there is no valid reason to forbid its use in any country in the world. ClassicSys gives all the required guarantees to its users and their government: secret keys must not be divulged and Security Services can always decipher suspect messages.

◎ Advantages & benefits for the End-User

◎ ClassicSys offers more than the known advantages of encryption solutions

- Very high speed of encryption (see below).
- The chip contains the SED algorithm and all the other features of ClassicSys. One system covers all cryptographic needs, for all applications.
- New applications can be added without updating the chip.
- ClassicSys works is fully automated, requests to the TA are returned directly, without human intervention.
- Private Keys are completely unknown to everybody, even the Trust Authorsity's manager! All keys are written into chips and are not accessible to humans or other machines. This guarantees the privacy of all the end-users.
- Once an end-user has received the information to generate his Application Keys, he does not need the intervention of the TA anymore. Email for example, users do not need the TA to exchange messages between themselves.
- ClassicSys acts like a public key cryptosystem: every end-user has one public ID number, which is used in a similar way to public keys. Email for example, when somebody wants to communicate with another end-user, he sends to the TA his ID number and the one from his correspondent. In return he receives information from the TA to generate their Session Key.

◎ Advantages & benefits for the Authorsity

- ClassicSys enables the TA and National Security Service (NSS) to act completely separately, under different authorsities, as required by our Democracies. Requests from the NSS to the TA are recorded encrypted by the TA (TA doesn't know the ID of Alice or Bob in a suspect message). This guarantees the confidentiality of the NSS's investigation; however, the recorded provides an audit trail for any Competent Investigating Authorsity. Optimum ClassicSys operation should have the TA and NSS under different authorsities, but every country can implement it as seen fit.
- ClassicSys enables the NSS to decrypt the content of suspect incoming and outgoing international messages, without the necessity for users to deposit their private secret keys in the corresponding countries (as with the RSA).
- Only the NSS is able to request necessary information to the TA to investigate suspect messages.
- Each country remains independent regarding the deciphering of the incoming and outgoing messages: each message contains the necessary information to be deciphered by the 2 National Security Services.
- Each Trust Authorsity has its own Private Key. Consequently they can only compute Private Keys for domestic users.

◎ Technical advantages & benefits

- ClassicSys is easy to implement in integrated circuits because :
  - ❑ it uses only XOR and branching functions
  - ❑ no reporting arithmetic bits are needed
  - ❑ Programming can be done with a polynomial structure.
  - ❑ The length of the blocks of key and data are identical and equal to 128 bits (16 bytes).
- Security of ClassicSys is enhanced compared to other systems because :
  - ❑ Deciphering is not the reverse of ciphering
  - ❑ The ciphering and deciphering keys are different
  - ❑ All the Private Keys (end-users, TAs, NSSs) are included in an IC and therefore not accessible.
- There is no known way to reconstruct, by cryptanalysis, the secret key, knowing a clear and its corresponding encrypted message.
- Differential cryptanalysis is not suitable to the SED algorithm. On average, there is only one key corresponding to a clear and its associated encrypted text and therefore, each bit of the key has equal weight in the algorithm.
- Only 1 secret key of 128 bits is enough to meet all the cryptographic needs of an end-user such as
  - ❑ to generate all the SessionKeys he needs
  - ❑ to authenticate himself
  - ❑ to authenticate messages with a time reference
  - ❑ to generate several keys for other applications (banking, electronic commerce, electronic voting, casino games at home, ...)
- Unlike the RSA algorithm, where every key requires a determined space, the SED algorithm can use every block contained in the space 2^{128}.
- The SED algorithm is very fast for the following reasons :
  - ❑ The length of the blocks (key and data) is small (128 bits against more than 512 bits) but long enough to disable every exhaustive cryptanalysis.
  - ❑ On average, it is possible to compute at 1/3 of the clock frequency(8to10 Mbytes/sec).
- The SED algorithm is completely transparent. Due to the theory of Multiplicative Groups we can confirm that there is no Trojan Horse in the SED algorithm.
- The SED algorithm permits chained mode ciphering, allowing reduction of the authentication information to one block of 128 bits, whatever the length of the data to authenticate.

5. THE APPROACH

5.1 Why RSA?

The RSA algorithm is the most popular and proven asymmetric key cryptographic algorithm. The RSA algorithm is based on the mathematical fact that it is easy to find and multiply large prime numbers together, but it is extremely difficult to factor their product .The private and public keys in RSA are based on very large (made up of 100 or more digits) prime numbers. However, the real challenge in the case of RSA is the Selection and generation of the private and public keys.

5.2 Used Interface

The authors are using p and q which are 1024 bits each it is not compatible to use it in C, C++ or Visual C++ or Java, so they are shiftting to Linux platform and using the GNU Bc Compiler. The test program is **"rsakeys.bc"** A program written for Bc is well suited to this experimental work, because it can handle numbers of an arbitrary size.

The Bc utility is a programmable calculator. It allows several types of calculations and provides simple looping logic. It is also much easier to read than the 'expr' expression evaluator.

5.3 Implementation

◎ The Approach to generate the key

◎ The process to generate the first prime(p)

1. The first prime generation process is dependent on the concept of DES.
2. Here the user has to give an authentication code. If it matched with the authenticate code then only he can be able to execute the code.
3. In the execution the user will give a 32 bit number.
4. Then he gives the position he want to discard.
5. Then the position will be discarded.
6. Then the bit positions are summed and divided by 8.
7. The result represents the number of circular right shift will be done.
8. Then the number will be regenerated.

◎ The program which gives the initial number to generate it's next prime that is given below(prime_des.c)

◎ Output of the above program is given below for six test runs

Bits taken	Number generated	Time taken(sec.)
35	34361187461	6.97
36	68720925859	9.80
37	137440402531	14.65
38	274879356019	18.68
40	1099513076861	39.25
50	1125899908291709	2410.75

6. CONCLUSION

The main aim is to generate a new way to generate keys of RSA algorithm. To achieve this authors have tried to derive a new way to prime number. Because what they have found that apart from the computational complexity of the RSA, the toughest part is to find two random prime. If prime numbers are same then the keys will be same. But, authors have tried to randomize the prime generation process, so that the two prime will be equal is highly unlikely. This program is highly protected because after authentication of the user than only he can get the chance to run the program. Then the user input a number from 1 to 2^32. Then he gives eight bit position which will be discarded. From the entered bit position the number of times the circular shift will be done is determined.

So, the chance of two generated prime becoming equal is highly unlikely. With the help of this program the headache of the user to find the two prime is solved. Authors hope more work on this field will be done in future.

◎ Constraints

Due to lack of high speed computer continuous backup power supply, the processing of 1024 bit number is not possible.

If the prime numbers of 1024 bits each then it will be completely unbreakable to the user.

◎ Achievements made

1. Authors have used DES concept in generating the first prime number. DES is powerful encryption algorithm. That is why it is very much infrequent due to the extra randomness which is added by them. Hence, it is quite tough to guess.
2. In generation of second prime, the current system time has been used. Due to using the current system time the generated number will be unique each time.

7. REFERENCE

[1] Behrouz A Forouzan : "Data Communications and Networking."

[2] William Stallings : "Cryptography and Network Security."

[3] Pradip K. Sahani : "Distributed Computing System."

[4] M. Shand and J. Vuillemin : "Journal of Fast Implementations of RSA Cryptography", Digital Equipment Corp., Paris Research Laboratory (PRL), 85 AV. Victor Hugo. 92500 Rueil-Mdmaison, France.

[5] Simson L. Garfinkel : "Public key cryptography."

[6] Atul Kahate : "Computer security."

[7] Calvin C. Clawson : "Mathematical Mysteries", 1996, Plenum Press.

[8] Donald C. Benson : "The Moment of Proof", 1999, Oxford University Press.

[9] Martin Gardner : "Penrose Tiles to Trapdoor Ciphers", 1989, W.H. Freeman & Co.

[10] Paulo Ribenboim : "The Little Book of Big Primes", 1991, Springer-Verlag.

[11] Keith Devlin : "The entire Math that's Fit to Print", 1994, The Mathematical Association of America.