A Fast Public-Key Cryptography Using RSA and T-invariants of Petri Nets

Qi-Wei Ge and Takako Okamoto

Faculty of Education, Yamaguchi University
1677-1 Yoshida , Yamaguchi, 753-8513, Japan

Tel: +81-83-933-5401, Fax: +81-83-933-5304
E-mail: {gqw,okamoto}@inf.edu.yamaguchi-u.ac.jp

Abstract: This paper deals with cryptography by applying RSA and Petri nets. Firstly, we introduce RSA cryptography and a Petri net based private-key cryptography. Then we propose a new public-key cryptography, Petri Net based Public-Key Cryptography dented as PNPKC, by taking the advantages of these two proposed cryptographys and give an example to show how to apply PNPKC. Finally, we do the comparison between RSA cryptography and PNPKC on security as well as computation order. As the results, the security of PNPKC is as strong as RSA cryptography and further the encryption and decryption of PNPKC are in average 210 times as fast as RSA cryptography from our experimental data.

1. Introduction

The Internet has become so widespread that anyone can obtain and provide information easily through the open electronic network. To guarantee the safety of such electronic communications, cryptography becomes ever important in order to avoid leak of secret information or dishonest alteration of information.

There are two types of cryptography: private-key and public-key cryptosystems. Private-key cryptosystem is such that it uses a common private key to encrypt and decrypt messages and can process the encryption and the decryption very fast, but is faced with a problem how to distribute the private key through the networks without leak of the secrecy. Public-key cryptosystem successfully solves this problem by preparing a pair of keys (public and private keys), and publishing the public key meanwhile keeping the private key secret [1]. Besides, by encrypting a message with private key authentication of electronic text can be realized.

As a public-key cryptosystem, RSA has been extensively used [2]. Recently, Elliptic Curve cryptography, an improved cryptography of RSA, has been noticed. The only problem of these cryptosystem is that they take too much time in the encryption and decryption. On the other hand, a private-key cryptosystem using Petri nets has been proposed by applying the property of T-invariants [3], which can also process the encryption and decryption as fast as usual private-key cryptography.

In this paper, we propose a Petri net based public-key cryptosystem, PNPKC, by taking advantages of both RSA and the private-key cryptosystem of [3], and further show that PNPKC has the same strong security as RSA but can do encryption and decryption of messages faster than RSA.

2. Preliminary

A Petri net is a bipartite graph and composed of transitions $T = \{t_1, t_2, \cdots, t_{|T|}\}$, places $P=\{p_1, p_2, \cdots, p_{|P|}\}$, directed edges $E=\{e_1, e_2, \cdots e_{|E|}\}$ and token distribution on places (called also marking) $M = (m_1, m_2, \cdots, m_{|P|})^t$ [4]. It can be expressed by Place-Transition incidence matrix N. If a firing sequence with firing count vector $J=(j_1, j_2, \cdots, j_{|T|})^t$ leads marking M_I to M_F, then $M_F=M_I+NJ$ holds. A vector J satisfying $NJ=0$ is called T-invariant. A set of T-invariants TB is called a T-base iff i) all the T-invariants included in the set are linearly independent each other, ii) any T-invariant of the Petri net PN can be expressed as a linear combination of the elements from the set [5].

The contrivances of RSA and the private-key cryptography of [3] are introduced in the following.

(1) Contrivance of RSA

- Find two large secret prime numbers, p and q, and compute n as $n=p\times q$; further choose a number e satisfying $gcd(e,(p-1)\times(q-1))=1$ and obtain d from $e\times d=1 (\mathrm{mod}\ (p-1)\times(q-1))$.
- Treat (e,n) and (d,n) as public and private keys respectively. Usually the ciphertext CT of a plaintext PT is obtained by $CT=PT^e (\mathrm{mod}\ n)$ and the plaintext is restored by $PT=CT^d (\mathrm{mod}\ n)$.

(2) Contrivance of the cryptosystem of [3]

- Choose a T-invariant J of a Petri net PN with Place-Transition incidence matrix N and divide J into two, J_1 and J_2, as $J=J_1+J_2$. Mainly, N and $J=J_1+J_2$ are used as private key.

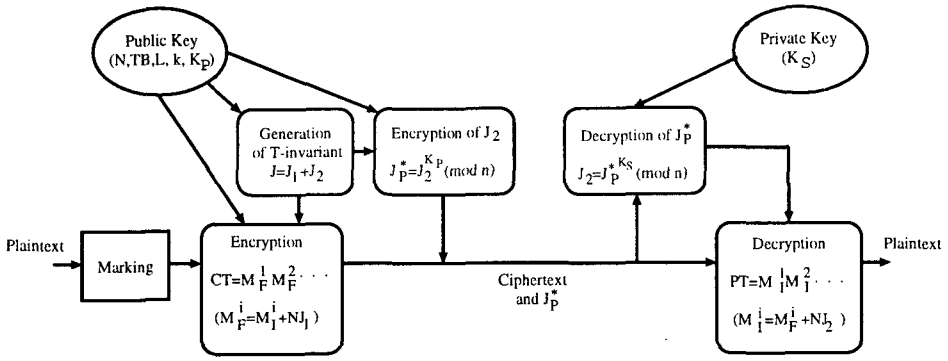

Fig. 1 A Petri net based public-key cryptosystem PNPKC.

- A plaintext PT is expressed by a marking M_I and further encrypted to the ciphertext CT expressed also by a marking as $M_F=M_I+NJ_1$. The decryption is done by computing M_F+NJ_2 that is equal to M_I due to $NJ_1 + NJ_2=NJ=0$.

3. A New Cryptosystem: PNPKC

Based on RSA and the private-key cryptosystem using Petri net, we propose a new public-key cryptosystem, PNPKC, as shown in Fig.1. In the following discussions, we assume the encryption is done by public key. Basically, the encryption and the decryption of PNPKC is as same as the proposed cryptosystem of [3]. In order to safely transmit a half of T-invariant (J_2) to be used in decryption, we use RSA to encrypt it. The T-invariant is generated by different nonnegative random integer each time in order to defend the T-invariant from attack. Concretely PNPKC works as follows:

- Public key is (N, TB, L, k, K_p) and private key is K_s, where (i) N is the incidence matrix; (ii) TB is a T-base consisting of b linear independent T-invariants $\{I_1, I_2, \cdots, I_b\}$; (iii) L is the length of a block (to be described later), expressed by number of bits; (iv) k is the capability of places to contain tokens; (v) K_p and K_s are the pair of keys generated by RSA.

- A T-invariant is generated from TB [5] as $J=c_1I_1+c_2I_2+\cdots+c_bI_b$, where c_i is chosen by nonnegative random integer. It is divided into J_1 and J_2, $J=J_1+J_2$.

- Encryption is done as: (i) express a plaintext with binary and divide it into blocks $PT=B_1B_2\cdots$, each of which contains L bits; (ii) divide each block B_i into $|P|$ parts, each with $L/|P|$ bits (whose value is no more larger than k) and map them to a marking M_I^i; (iii) for each M_I^i, compute $M_F^i=M_I^i+NJ_1$ and then the ciphertext CT is $CT=M_F^1M_F^2\cdots$; (iv) meanwhile apply RSA to encrypt J_2 with the key K_p to obtain J_p^*.

- Receiving the ciphertext CT and J_p^*, the decryption is done as: (i) apply RSA to decrypt J_p^* with the key K_s to obtain J_2; (ii) for each received M_F^i of CT, compute $M_F^i+NJ_2$ to get M_I^i and thus each block B_i is restored.

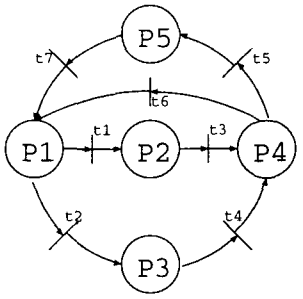

Fig. 2 A Petri net N used in the example.

In the following, we give an example to show how to use PNPKC to encrypt and decrypt plaintext and ciphertext respectively.

Example.

A. Preparation

- Construction of a Petri net
  A Petri net must be prepared to be used as a part of public-key. Here we use a Petri net as shown in Fig.2, whose incidence matrix N is shown below. Note that the weights of the edges are 1.

$$N = \begin{pmatrix} -1 & -1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 \end{pmatrix}$$

- Generation of T-base
  T-base is also used as a part of public-key and is generated as follows:

  $I_1 = (1\ 0\ 1\ 0\ 1\ 0\ 1)^t$
  $I_2 = (1\ 0\ 1\ 0\ 0\ 1\ 0)^t$
  $I_3 = (0\ 1\ 0\ 1\ 1\ 0\ 1)^t$

- A T-invariant is generated from TB as
  $J = c_1I_1 + c_2I_2 + c_3I_3$.

The c_1, c_2 and c_3 are nonnegative random integer and are here generated as 3, 5, 7 respectively. Thus

$J = 3I_1 + 5I_2 + 7I_3$
$= (8\ 7\ 8\ 7\ 10\ 5\ 10)^t$.

J is further divided into J_1 and J_2 as $J=J_1+J_2$:
$J_1=(2\ 3\ 6\ 4\ 5\ 2\ 9)^t$
$J_2=(6\ 4\ 2\ 3\ 5\ 3\ 1)^t$

Note that J_1 and J_2 must not be T-invariant.

B. Generation of a pair of keys by RSA

- Find two large secret prime numbers, p and q, as follows:
  p=1208131
  q=1208117
- Compute n as $n=p\times q$:
  $n=p\times q$
  $=1208131\times 1208117$
  $=1459563599327$
- Further choose a number e satisfying $gcd(e,(p-1)\times(q-1))=1$, where
  $(p-1)\times(q-1)=1208130\times 1208116$
  $=1459561183080$.
  Thus e=1208113 is obtained.
- Obtain d from $e\times d=1(\mathrm{mod}\ (p-1)\times(q-1))$:
  $e\times d=1(\mathrm{mod}\ (p-1)\times(q-1))$
  $d=e^{-1}(\mathrm{mod}\ (p-1)\times(q-1))$
  $=1208113^{-1}(\mathrm{mod}\ 1208130\times 1208116)$.
  Therefore, d=1402325882184.

By the above operations, the public and private keys of RSA are obtained:

$K_p=(e,n)=(1208113, 1459563599327)$
$K_s=(d,n)=(1402325882184, 1459563599327)$.

Therefore the public and private keys of PNPKC are as follows:

Public key: (N, TB, L, k, K_p)
Private key: K_s

where N, TB, K_p, K_s are as have been obtained and L, k are determined as 40, 256 respectively.

C. Encryption

Here we are to encrypt a plain text of "Encrypt and Decrypt".

(i) Express a plaintext in binary and divide it into blocks $PT=B_1B_2\cdots$, each of which contains L bits.

B_1: $M_I^1 = (\mathrm{E\ n\ c\ r\ y})^t$
B_2: $M_I^2 = (\mathrm{p\ t\ \_\ a\ n})^t$
B_3: $M_I^3 = (\mathrm{d\ \_\ D\ e\ c})^t$
B_4: $M_I^4 = (\mathrm{r\ y\ p\ t\ \_})^t$

where _ is space. Note that one character consists of 8 bits.

(ii) Divide each block B_i into $|P|$ parts, each with $L/|P|$ bits (whose value is no more larger than k) and map them to a marking M_I^i.

B_1: $M_I^1 = (48\ 56\ 64\ 72\ 80)^t$
B_2: $M_I^2 = (88\ 96\ 136\ 104\ 56)^t$
B_3: $M_I^3 = (112\ 136\ 120\ 128\ 64)^t$
B_4: $M_I^4 = (72\ 80\ 88\ 96\ 136)^t$

(iii) For each M_I^i, compute $M_F^i=M_I^i+NJ_1$ and then the ciphertext CT is $CT=M_F^1M_F^2M_F^3M_F^4$.

B_1': $M_F^1 = M_I^1 + NJ_1 = (54\ 52\ 63\ 75\ 76)^t$
B_2': $M_F^2 = M_I^2 + NJ_1 = (94\ 92\ 135\ 109\ 52)^t$
B_3': $M_F^3 = M_I^3 + NJ_1 = (118\ 132\ 119\ 133\ 60)^t$
B_4': $M_F^4 = M_I^4 + NJ_1 = (78\ 76\ 87\ 101\ 132)^t$

(iv) Meanwhile apply RSA to encrypt J_2 by the key K_p to obtain J_p^*. Then the ciphertext CT as well as J_p^* can be safely sent to the receiver possessing the public-key (N, TB, L, k, K_p) and the private key K_s.

D. Decryption

(i) Receiving the ciphertext CT and J_p^*, the receiver applies RSA to decrypt J_p^* with the private key K_s to obtain J_2.

(ii) Divide each block B_i into $|P|$ parts, each with $L/|P|$ bits (whose value is no more larger than k) and map them to a marking M_F^i.

B_1: $M_F^1 = (54\ 52\ 63\ 75\ 76)^t$
B_2: $M_F^2 = (94\ 92\ 135\ 109\ 52)^t$
B_3: $M_F^3 = (118\ 132\ 119\ 133\ 60)^t$
B_4: $M_F^4 = (78\ 76\ 87\ 101\ 132)^t$

(iii) For each M_F^i, compute $M_I^i=M_F^i+NJ_2$ and then the plaintext PT is $PT = M_I^1M_I^2M_I^3M_I^4$.

B_1': $M_I^1 = M_F^1 + NJ_2 = (48\ 56\ 64\ 72\ 80)^t$
B_2': $M_I^2 = M_F^2 + NJ_2 = (88\ 96\ 136\ 104\ 56)^t$
B_3': $M_I^3 = M_F^3 + NJ_2 = (112\ 136\ 120\ 128\ 64)^t$
B_4': $M_I^4 = M_F^4 + NJ_2 = (72\ 80\ 88\ 96\ 136)^t$

Finally, restoring the plaintext from its binary expression PT, the original text "Encrypt and Decrypt" can be obtained.

4. Comparison of RSA and PNPKC

Here we compare the two cryptosystems, RSA and PNPKC, on their security and computation time.

(1) Security

The security of RSA depends on the difficulty of factoring the large integer n $(= p\times q)$ [2] and it is well know that the computational order is $O(\exp(c(\log n)(\log\log n))^{1/2})$ (c is constant), which is the time required in applying factorization algorithm. If p and q leak out, then the private key (d, n) can be easily obtained from $e\times d=1(\mathrm{mod}\ (p-1)\times(q-1))$. Therefore n is required to be as larger as two hundred figures and the security of RSA is guaranteed by this consideration.

Table 1 Computation Times of PNPKC and RSA

Size of Texts	Computation Times (ms/KB)			
	PNPKC		RSA	
	encryption	decryption	encryption	decryption
1KB - 9KB	2.00275	1.97777	61.03496	59.69664
10KB - 99KB	0.34876	0.34380	59.29288	58.17512
100KB - 999KB	0.22158	0.22159	58.65152	57.56400
1KB - 999KB	0.27305	0.27039	58.80878	57.71171

In the case of PNPKC, obviously J_p^* has the same security as RSA. Since J_2 is a half of a T-invariant, it can not be cryptanalyzed from N and TB unless all the possible combinations of the $|T|$ elements' values are tested. Therefore, the computation order is $O(f_m^{|T|})$, where f_m is the maximum integer of the elements. RSA is said to be safe enough if 1024 bits is used and therefore if we set f_m and $|T|$ larger than 2^{10} and 16 respectively then PNPKC can possess the same security as RSA. If necessary, it is possible to increase a security of J_2 by increase the maximum integer of the elements f_m and $|T|$.

(2) Computation time

As for the computation order of encryption and decryption, RSA takes $O((\log n)LEN_{PT})$ while PNPKC takes $O((|T||P|/L)LEN_{PT})$, where LEN_{PT} is the block number of the plaintext PT. If we choose n as 1024 and $|T|$, $|P|$, L as 16, 8, 64 respectively, then PNPKC takes only about one hundredth of RSA's. We have done experiments to compare real computation times between the two cryptosystems.

We downloaded 100 texts randomly from the Internet, whose text sizes are from several kilobytes (KBs) to hundreds kilobytes, and encrypt and decrypt them by using RSA and PNPKC. The average computation times are summarized in Table 1. From this table, it is clear that PNPKC takes only the computation times about 1/30, 1/170 and 1/260 of RSA's respectively for several KBs, tens KBs and hundreds KBs. In total average, PNPKC can reduce RSA's encryption and decryption times by 210 times. Obviously, these experimental results are consistent with our above theoretical analysis. Therefore PNPKC is more efficient than RSA and is definitely practically useful.

5. Conclusion

In this paper, we proposed a new public-key cryptosystem PNPKC based on RSA and the Petri net based private-key cryptosystem [3], and gave an example to show how to use PNPKC. Through theoretical analysis as well as experiments, we revealed that (i) the security of PNPKC is as strong as RSA; and (ii) the encryption and decryption of PNPKC are in average 210 times as fast as RSA. That is PNPKC is useful.

To make a practical cryptosystem using PNPKC, the following research problems need to be solved: (a) To design an algorithm of generating T-base by referring the known research results on computation of T-invariant [6]-[9]; (b) To find out structurally safe Petri Nets in order to guarantee strong security. Futurely it is expected to develop PNPKC to dissolve the dependence of use of RSA.

Acknowledgment

The authors would like to thank Miss. Yuko Matsuo for her help of doing the experiments when she was studying at Information Processing Laboratory, Faculty of Education, Yamaguchi University, This research is partly supported by Grant-in-Aid for Scientific Research (C) 11680417 of the Ministry of Education, Science, Sports and Culture of Japan.

References

[1] A. Salomaa, Public-Key Cryptography, Springer-Verlag Berlin Heidelberg (1990).

[2] R.L.Rivest, A.Shamir and L.Adleman, "A method of obtaining digital signatures and public-Key cryptosystems", Comm. of ACM, vol.21, no.2, pp.120-126 (1978).

[3] H.Shiizuka and Y.Mouri, "A model of net structured cryptography" Research reports of the Kogakuin University, vol.74, pp.155-164 (1993).

[4] J.Peterson, Petri Net Theory and the Modeling of Systems, Englewood Cliffs, NJ: Prectice-Hall (1981).

[5] Q.W.Ge, T.Tanida, N.Ono and K.Onaga, "Construction of a T-base and design of a periodic firing sequence of a live and bounded Petri net", Proc. Twenty-Fifth Annual Allerton Conference on Communication, Control and Computing, pp.305-314 (1987).

[6] J.Martinez and M.Silva, "A simple and fast algorithm to obtain all invariants of a generalized Petri net", In C.Girault and W.Reisig, editors, Fachberichte Informatik, vol.52, pp.301-310, Springer Verlag, Berlin, Germany (1982).

[7] T.Matsumoto, H.Itoh and Y.Jiang, "On a New Method for Finding All Minimal Invariants in General Petri Nets", Technical Report of IEICE, vol.98, no.87, pp.31-38 (1998).

[8] M.Wakuda, S.Taoka and T.Watanabe, "Computing Petri Net Invariants based on Siphon-Trap Supports", Technical Report of IEICE, vol.99, no.98, pp.9-16 (1998).

[9] M. Wakuda, M. Yamauchi, S. Taoka and T. Watanabe, "A fast and space-saving algorithm FMSN for computing Petri net invariants with supports containing all specified nodes", Technical Report of IEICE, vol.99, no.206, pp.37-44 (1999).