On Encryption of a Petri Net based Multi-Stage-Encryption Public-Key Cryptography

Qi-Wei Ge†, Chie Shigenaga†, Mitsuru Nakata† and Ren Wu††

† Faculty of Education, Yamaguchi University, Japan
†† Part-time Lecturer of Yamaguchi University, Japan
1677-1 Yoshida, Yamaguchi 753-8513, Japan
† Tel: +81-83-933-5401, Fax: +81-83-933-5304
E-mail: † {gqw, shigenaga, nakata}@inf.edu.yamaguchi-u.ac.jp

Abstract: A new conception of public-key cryptography MEPKC, Petri net based Multi-stage-Encryption Public-Key Cryptography, has been proposed in order to guarantee stronger network communication security. Different from an ordinary public-key cryptography that opens only a single public key to the public, MEPKC opens a key-generator that can generate multiple encryption keys and uses these keys to encrypt a plain text to a cipher text stage by stage. In this paper, we propose the methods how to carry out the encryption operations. First, we describe how to design a hash function H that is used to conceal the encryption keys from attack. Then, given with a key-generator (a Petri net supposed to possess a large number of elementary T-invariants), we discuss how to randomly generate a series of encryption keys, the elementary T-invariants. Finally, we show how to use these encryption keys to encrypt a plain text to a cipher text by applying a private-key cryptography, say DES.

1. Introduction

To guarantee safety electronic communications, public-key cryptography becomes ever important in order to avoid leak of secret information or dishonest alteration of the information [1]. As public-key cryptography, there are RSA [2], ElGamal [3], etc., among which RSA is most extensively used in various network communications. Recently, elliptic curve cryptography that is developed and improved from RSA has attracted a great deal of attention [4] and also PGP cryptography [5] has been widespreadly used in email text encryption. These cryptosystems possess security as strong as subexponential or exponential computation time.

To make the security furthermore stronger, a new public-key cryptography, Petri net based Multi-stage-Encryption Public-Key Cryptography (MEPKC for short hereafter), has been proposed [6]. Different from an ordinary public-key cryptography that opens only a single public key to the public, MEPKC opens to the public a key-generator that can generate multiple encryption keys and uses these keys to encrypt a plain text to a cipher text stage by stage. So that the security of a ciphertext will be increased exponentially with the number of the encryption stages [6].

In this paper, we mainly discuss how to generate encryption keys from a given key-generator and how to do the encryption of a plaintext by using MEPKC. Firstly, we describe how to design a hash function H that is used to conceal the encryption keys. Then, given with a key-generator (a Petri net supposed to possess a large number of elementary T-invariants), we discuss how to randomly generate a series of encryption keys, the elementary T-invariants. Finally, we show how to use these encryption keys to encrypt a plain text to a cipher text by applying a private-key cryptography, say DES [1].

2. Preliminary

The necessary definitions are given in the following.

Definition 1. A Petri net [7] is a bipartite graph and expressed by $PN=(T,P,E,\alpha,\beta)$, where $E=E^+\cup E^-$ and

T: a set of transitions $\{t_1,t_2,\cdots,t_{|T|}\}$
P: a set of place $\{p_1,p_2,\cdots,p_{|P|}\}$
E^+: a set of edges from transitions to places $e=(t,p)$
E^-: a set of edges from places to transitions $e=(p,t)$
α: $\alpha(e)$ is the weight of edge $e=(p,t)$
β: $\beta(e)$ is the weight of edge $e=(t,p)$ □

Definition 2.

(1) When there exist neither edge (p_i,t_j) nor edge (t_j,p_i) for any p_i and t_j of a Petri net PN, PN is called pure Petri net. The $P-T$ incidence matrix of a pure Petri net is expressed by $N=N^+-N^-=[N^+_{pt}]-[N^-_{pt}]$, where,

$$N^+_{pt}=\begin{cases}\beta_e & \text{if } e=(t,p)\\ 0 & \text{otherwise,}\end{cases}\quad N^-_{pt}=\begin{cases}\alpha_e & \text{if } e=(p,t)\\ 0 & \text{otherwise.}\end{cases}$$

(2) Token distribution to places is called marking and expressed by $M=(m_1,m_2,\cdots,m_{|P|})^t$, where, m_i is the number of tokens at p_i. If capacity of all the places is no more than k then PN is k-bounded.

(3) A transition sequence σ is called firing sequence from M_I to M_F, if the firing simulation of σ on M_I can be carried out all the way to the last element of σ, which leads to the marking M_F. The marking transition is expressed by $M_I[\sigma>M_F$ and the firing numbers of all the transitions are expressed by a firing count vector $J=(j_1,j_2,\cdots,j_{|T|})^t$. The relationship among N, J, M_I and M_F can be expressed by $M_F=M_I+NJ$. □

All the Petri nets used in this paper are supposed to be pure ones.

Definition 3.

(1) A non-negative vector J satisfying $NJ=0$ is called T-invariant and the set of transitions $T_J=\{t_i\in T|j_i\neq 0\}$ is called support of J.

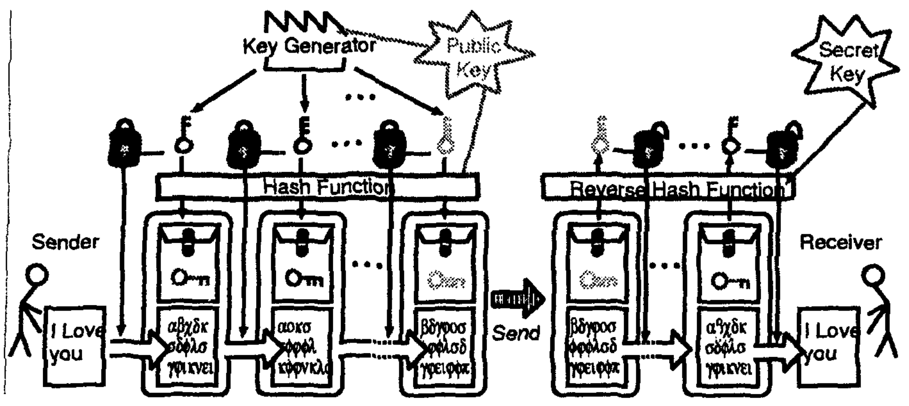

Figure 1. The proposed public-key cryptography: MEPKC

(2) For a T-invariant J with support T_J, if there exists no such T-invariant J' whose support $T_{J'}$ satisfies $T_{J'} \subset T_J$, then T_J is called minimum support. Further for a T-invariant J with minimum support T_J, if all the values $\{j_i | t_i \in T_J\}$ *have no common divisor then* J is called elementary T-invariant. □

The proposed cryptography MEPKC is illustrated in Fig.1. The public key is $<N, H>$, where N is a key generator that is P-T incidence matrix of a Petri net PN and H is a hash function that maps an elementary T-invariant J_i^e to a value $V_i = H(J_i^e)$ in order to conceal the elementary T-invariant. The private key is $<\{J_i^e\}, \overline{H}>$, where $\{J_i^e\}$ is the set of all the elementary T-invariants of PN and $\overline{H}$ is a reverse hash function $\overline{H}: V_i \rightarrow J_i^e$. The encryption and decryption are as following.

Encryption: Given with a plaintext $\mathcal{P}$, a sender at first generates a series of elementary T-invariants $\{J_{k_1}^e, J_{k_2}^e, ..., J_{k_l}^e\}$ randomly from key generator N and then uses these elementary T-invariants to do encryption operations stage by stage. The first encryption stage is as *follows: (i) a plaintext $\mathcal{P}$ is encrypted to a ciphertext* C_1' by a private-key cryptography (e.g. DES), in which $J_{k_1}^e$ is used as the encryption key; (ii) $J_{k_1}^e$ is mapped to V_{k_1} by *the hash function H; (iii) $C_1 = (C_1', V_{k_1})$ is* treated as the ciphertext of the first encryption stage. And i-th encryption stage is similar as the first stage and the only difference is that the ciphertext C_{i-1} is treated as a plaintext of i-th stage. The sender sends the final ciphertext $C_l = (C_l', V_{k_l})$ to the receiver.

Decryption: Receiving the ciphertext $C_l = (C_l', V_{k_l})$, the receiver takes out V_{k_l} from C_l and uses V_{k_l} to deduce $J_{k_l}^e$ *by the reverse hash function $\overline{H}$. Further, using $J_{k_l}^e$ to do* decryption (of the private-key cryptography) for C_l', the sender gets the $(l-1)$-th ciphertext $C_{l-1} = (C_{l-1}', V_{k_{l-1}})$. Repeating this decryption operation successively, the original plaintext $\mathcal{P}$ is finally obtained.

The security of MEPKC relies on such a property of Petri nets that it is extremely difficult to compute all the elementary T-invariants for a given Petri net PN, but is *comparatively easy to compute a part of them.* And if we repeat enough many encryption stages, the resultant ciphertext can get to such strong security as we expect.

3. On encryption of MEPKC

In the *followings, we focus our discussions to the* following problems on encryption of MEPKC: (1) How to design a hash function H; (2) How to randomly generates a series of encryption keys, the elementary T-invariants $\{J_{k_1}^e, J_{k_2}^e, ..., J_{k_l}^e\}$ from a given Petri net's P-T incidence matrix N; and (3) How to use these encryption keys to encrypt a plaintext by using a *private-key cryptography.*

3.1 Designing hash function H

We first discuss on how to design a so strong hash function H that it is impossible to deduce an elementary T-invariant J^e from $V = H(J^e)$. We randomly generate two vectors, R_N and R_0, both with $|T|$-*dimension. Then for a given elementary T-invariant* $J^e = (j_1^e, j_2^e, \cdots, j_{|T|}^e)^t$, we make a complement vector of its support, $\overline{S}_{J^e} = (\overline{s}_1, \overline{s}_2, \cdots, \overline{s}_{|T|})^t$ where

$$\overline{s}_i = \begin{cases} 1 & \text{if } j_i^e = 0 \\ 0 & \text{otherwise.} \end{cases}$$

Using R_N and R_0, we give our hash function as

$$V = H(J^e) = (R_N)^t J^e + (R_0)^t \overline{S}_{J^e}.$$

Theorem 1. The problem to deduce J^e from $V = H(J^e)$ is NP-complete. □

The above theorem is true, since it is a NP-complete problem even to find out J^e from $V' = (R_N)^t J^e$, which is subset sum problem [8]. Obviously, it is more difficult to deduce J^e from V than to solve a subset sum problem. Therefore our hash function is strong enough to conceal the encryption key J^e.

3.2 Random generation of encryption keys

For a given Petri net PN with its P-T incidence matrix N, we can apply Linear Programming and Gauss-Jordan reduction process to randomly generate a series of elementary T-invariants.

Theoretically, a rational non-negative solution J of $NJ = 0$ can be obtained by solving the following formulation without doing division operations.

Minimize: $W = \mathbf{1}^t Z$

Subject to:

Table 1. An initial simplex tableau

base	j_1	j_2	j_3	j_4	j_5	j_6	j_7	j_8	j_9	z_1	z_2	z_3	z_4	z_5	z_6	const.
z_1	-2	-5							1	1						0
z_2	2	5	-1	-2							1					0
z_3			1	2	-3	-4						1				0
z_4					3	4	-1	-3					1			0
z_5							1	3	-1					1		0
z_6	1	1	1	1	1	1	1	1	1						1	1
$-W$	-1	-1	-1	-1	-1	-1	-1	-1	-1							-1

$$\begin{pmatrix} n_{11} & \cdots & n_{1|T|} & 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ n_{|P|1} & \cdots & n_{|P||T|} & 0 & \cdots & 1 & 0 \\ 1 & \cdots & 1 & 0 & \cdots & 0 & 1 \end{pmatrix} \begin{pmatrix} j_1 \\ \vdots \\ j_n \\ z_1 \\ \vdots \\ z_{|P|+1} \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ c \end{pmatrix}$$

$$j_1,\cdots,j_n; z_1,\cdots,z_{|P|+1} \geq 0$$

where, n_{xy} is the element of N, z_i is the artificial variable and c is a positive integer.

Theorem 2.[9] If a rational solution J of the above formulation is obtained by simplex method with W=0, then (i) J can be converted into a T-invariant by multiplying an appropriate integer; and (ii) the support of such a converted T-invariant is minimum. □

Practically, we can efficiently compute an elementary T-invariant by doing the followings.

- Simply using simplex method (allowing division operations) to obtain a non-negative solution with W=0. Thus a support T_J is obtained.
- Deleting all the artificial variables and treat the variables related to the transitions in T_J as the basic variables to obtain a non-negative integer solution by Gauss-Jordan reduction process.

Generally during the simplex procedure, plural variables may be the candidates of basic variables. Therefore if we randomly select the candidates as basic variables, then an unspecified elementary T-invariant can be obtained. To randomly obtain a series of unspecified elementary T-invariants, we need only to change the basic variables from the previously obtained elementary T-invariants by doing Gauss-Jordan reduction process. To avoid generating duplicated elementary T-invariants, the searching method proposed in Ref. [10] is useful.

3.3 Encryption of plain texts by using DES

Since a Petri net PN used as a key-generator is assumed to possess a large amount of elementary T-invariants, PN must have enough many transitions (say more than several hundreds transitions). Thus an elementary T-invariant may have many elements and further most of them are of value 0. As an encryption key used to encrypt a plaintext by a private-key cryptography (say 3DES that limits the length of encryption key to 1024 bytes), an elementary T-invariant J_i^e may be too long and we need to compress it into a shorter one K_i. The compression method, here we are to propose, is as follows:

- Basically concatenate the values of all the elements into an integer.

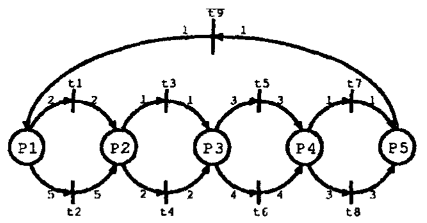

Figure 2. A Petri net used as key generator

- During the concatenation, if there are k same value D continuously appears (e.g. 123 123 123 123), then we compress them into kD (4123). Especially, if the value D is 0 then we simply write k (4).

According to the above proposed methods, we can realize the encryption of MEPKC.

4. An example

Here we are to give an example to show how encryption of MEPKC works. We carry out two stage encryption to encrypt a plain text $\mathcal{P}$= "I love you." to a cipher text. The Petri net shown in Fig.2 is used as key generator.

(1) Determination of hash function H

We first generate two random vectors with $|T|$-dimension, R_N and R_0, as follows:

$$R_N = (\ 3\ \ 7\ \ 1\ \ 9\ \ 12\ \ 6\ \ 8\ \ 1\ \ 20\ )^t$$
$$R_0 = (\ 4\ \ 1\ \ 9\ \ 17\ \ 10\ \ 8\ \ 3\ \ 2\ \ 5\ )^t$$

Thus the hash function is determined as:

$$V=H(J^e)=(R_N)^t J^e+(R_0)^t \overline{S}_{J^e}$$

(2) Generation of encryption keys

To generate encryption keys, we give the following LP formulation.

Minimize = $\mathbf{1}^t Z$

Subject to:

$$\begin{pmatrix} -2 & -5 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & & & 0 & & \\ 2 & 5 & -1 & -2 & 0 & 0 & 0 & 0 & 0 & & 1 & & & & \\ 0 & 0 & 1 & 2 & -3 & -4 & 0 & 0 & 0 & & & 1 & & & \\ 0 & 0 & 0 & 0 & 3 & 4 & -1 & -3 & 0 & & & & 1 & & \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 3 & -1 & & & & & 1 & \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & & & & 0 & & 1 \end{pmatrix} \begin{pmatrix} j_1 \\ \vdots \\ j_9 \\ z_1 \\ \vdots \\ z_6 \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}$$

$$j_1,\cdots,j_9; z_1,\cdots,z_6 \geq 0$$

where, $z_1,\cdots,z_6$ are the artificial variables. Table 1 shows the initial simplex tableau. From this table, we can find that at the beginning all the variables, $j_1,\cdots,j_9$, are possible to be selected as basic variables. Here we first select j_2 as one basic variable. Then gradually, j_4, j_6, j_8, j_9 are selected through simplex operation.

Table 2. The table obtained by dividing basic and non-basic variables and deleting artificial variables

j_2	j_4	j_6	j_8	j_9	j_1	j_3	j_5	j_7	const.
-2	-5	0	0	0	0	0	0	1	0
2	5	-1	-2	0	0	0	0	0	0
0	1	2	-3	-4	0	0	0	0	0
0	0	0	3	4	-1	-3	0	0	0
0	0	0	0	0	0	1	3	-1	0
1	1	1	1	1	1	1	1	1	1

Table 3. The table after Gauss-Jordan reduction process

j_2	j_4	j_6	j_8	j_9	j_1	j_3	j_5	j_7	const.
137	0	0	0	0	62	6	3	8	12
0	274	0	0	0	36	167	15	40	60
0	0	274	0	0	18	15	213	20	30
0	0	0	137	0	12	10	5	59	20
0	0	0	0	137	36	30	15	40	60

Thus a support $T_J=\{j_2, j_4, j_6, j_8, j_9\}$ is obtained. We divide the basic and non-basic variables and further delete all the artificial variables to obtain a table as shown in Table 2. For Table 2, we do Gauss-Jordan reduction process to get a new table as shown in Table 3. Thus an elementary T-invariant $J^e_{k_1}$ is obtained.

$$J^e_{k_1} = (\ 0\quad 12\quad 0\quad 30\quad 0\quad 15\quad 0\quad 20\quad 60\ )^t$$

For Table 3, we can randomly select the variables (from j_1, j_3, j_5, j_7) as the possible basic variables to exchange the basic variables. Here we first select j_5 and then the second elementary T-invariant is finally obtained.

$$J^e_{k_2} = (\ 0\quad 6\quad 0\quad 15\quad 10\quad 0\quad 0\quad 10\quad 30\ )^t$$

(3) Encryption of the plain text $\mathcal{P}$

The private-key cryptography used here is 3DES. First we use hash function H to conceal these two keys and compress them as follows:

$V_{k_1}=H(J^e_{k_1})=1670$, $J^e_{k_1} \rightarrow K_1$: 01203001502060;

$V_{k_2}=H(J^e_{k_2})=931$, $J^e_{k_2} \rightarrow K_2$: 060151021030.

- Encryption of 1st stage:
  Using the compressed key K_1 as the encryption key to encrypt $\mathcal{P}$ by 3DES, we get the following cipher text C'_1:

```
OPBLKH)[#^1/G::WWAVR;>P
```

  Combing C'_1 and V_{k_1} to get the resultant cipher text C_1 of the 1st stage encryption:

```
OPBLKH)[#^1/G::WWAVR;>P
1670
```

- Encryption of 2nd stage:
  Treating C_1 as a plain text and using K_2 to encrypt C_1, we get the following cipher text C'_2:

```
@K$JOMVOO F^HN\@@ &ON7.
%][507"^[MN4EYY'/38<
```

  Combing C'_2 and V_{k_2}, we get the final cipher text:

```
@K$JOMVOO F^HN\@@ &ON7.
%][507"^[MN4EYY'/38<
931
```

5. Concluding remarks

We have proposed the methods how to do encryption of a proposed public-key cryptography MEPKC. A hash function has been proposed, which is *NP*-complete and thus is strong enough to conceal encryption keys from attack. A method to randomly generate the encryption keys has been given by applying Linear Programming and Gauss-Jordan reduction process. Further we have proposed the way to encrypt a plain text to a cipher text by using a private-key cryptography 3DES. Finally, we have given an example to concretely show how our proposed methods are adopted.

As the future works related to realization of the conception of MEPKC, we need to (i) make a prototype encryption system to evaluate our proposed encryption methods; and (ii) propose the methods how to design the private key of MEPKC and to do the decryption operations.

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] T.ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms", IEEETrans. in Information Theory, vol.IT-31,no.4, pp.469-472, 1985.

[4] T.Okamoto and S.Uchiyama, "Recent topics of public-key cryptography: 1. On the security of elliptic curve cryptosystems", IPSJ Magazine, vol.39, no.12, pp.1252-1257, 1998.

[5] Simson Garfinkel, PGP: Pretty Good Privacy, O'Reilly & Associates, 1994.

[6] Q.W.Ge, C.Shigenaga, and R.Wu, "A Petri Net based New Conception of Public-Key Cryptography", Proceeding of ICFS2002, pp.S5-37 - S5-42, 2002.

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

[8] M.R.Garey and D.S.Johson, Computers and Intractability (A Guide to the Theory of NP-Completeness), W.H.Freeman and Company, New York, 1991.

[9] 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, pp.51-57, 1987.

[10] T.Araki, T.Tanida, T.Watanabe and K.Onaga, "A linear programming-based algorithm for successively enumerating elementary invariants of Petri nets", Technical Report of IEICE, pp.125-132 (CAS89-89), 1989.