A new class of system oriented PKC, K(I)SOPKC.

Masao KASAHARA *†

Abstract

In this paper, we present a new type of PKC, system-oriented PKC,referred to as K(I)SOPKC that can be well adapted to a secure and a high speed communication between various systems and organizations of the present day network society. K(I)SOPKC is constructed on the basis of K(XIV)SE(1)PKC, a modified version of K(XII)SE(1)PKC [1], K(XIII)SE(1)PKC [2] and $\mathrm{K_p}$(XIII)SE(1)PKC [2].

keyword

Code based PKC, Multivariate PKC, System oriented PKC, K(I)SOPKC.

1 Introduction

The author proposed a several classes of linear multivariate PKC's that are constructed by many sets of linear equations [3] $\sim$ [5] based on error-correcting codes. It should be noted that McEliece PKC [6], a class of code based PKC(CB·PKC), can be regarded as a class of the linear multivariate PKC. Excellent analyses and survey are given, for example, in Refs. [7] and [8].

Recently, we presented a new class of public key cryptosystems, by modifying K(XII)SE(1)PKC [1], referred to as K(XIII)SE(1)PKC [2] and $\mathrm{K_p}$(XIII)SE(1)PKC [2].

In 2011, Tsujii and Gotaishi presented an interesting PKC for selected communication between organizations based on complementary STS-MKC [9].

In this paper, we present a new type of PKC, system oriented PKC, referred to as K(I)SOPKC that can be well adapted to a secure and a high speed communication between various systems and organizations of the present day network society. K(I)SOPKC is constructed on the basis of K(XIV)SE(1)PKC, a modified version of K(XII)SE(1)PKC [1], K(XIII)SE(1)PKC [2].

Throughout this paper, when the variable v_i takes on a value $\tilde{v}_i$, we shall denote the corresponding vector $\boldsymbol{v} = (v_1, v_2, \cdots, v_n)$ as

$$\tilde{\boldsymbol{v}} = (\tilde{v}_1, \tilde{v}_2, \cdots, \tilde{v}_n). \tag{1}$$

The vector $\boldsymbol{v} = (v_1, v_2, \cdots, v_n)$ will be represented by the polynomial as

$$v(x) = v_1 + v_2 x + \cdots + v_n x^{n-1}. \tag{2}$$

The $\tilde{u}$, $\tilde{u}(x)$ et al. will be defined in a similar manner.

*Research Institute for Science and Engineering, Waseda University.
†Research and Development Initiative, Chuo University. kasahara@ogu.ac.jp

2 K(XIV)SE(1)PKC

2.1 Theoretical background of present paper

In 1970's, the various works were made of the jointly optimization problems for realizing a high speed and a reliable digital transmission system. The author was also much involved in the study of the jointly optimization problems for source and channel coding, based on syndrome coding. Let us apply the idea of syndrome coding to constructing K(I)SOPKC as shown in Fig.1.

As illustrated in Fig.1, the idea of K(XIV)SE(1)PKC is closely related to the idea of syndrome coding proposed for a jointly optimization problem for source and channel coding.

In Feb.1986, the author presented a survey paper on cryptgraphy [10]. In Ref. [10], the author suggested the using of McEliece PKC on noisy channel and presented a very simple scheme of joint coding for encryption and error control coding, based on McEliece PKC.

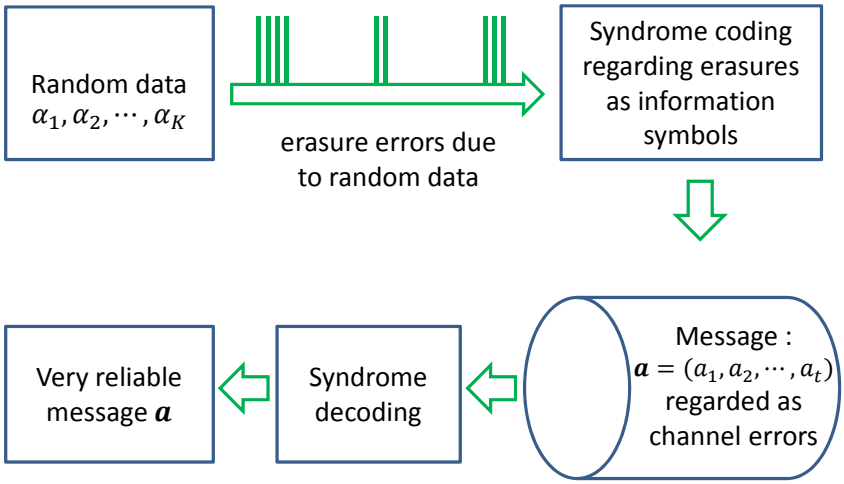

Fig. 1: Syndrome coding

2.2 Construction of K(XIV)SE(1)PKC

Let us define several symbols.

$G(x)$	:	generator polynomial of Reed Solomon code over $\mathbb{F}_{2^m}$.
g	:	degree of $G(x)$.
D	:	minimum distance, $g+1$.
K	:	length of information symbols, $2^m - 1 - g$.
K'	:	shortened length of information symbols, $K' < K$.
$\varphi(*)$	:	transformation function.

Let the vector $\boldsymbol{\mu}_i$ over $\mathbb{F}_{2^m}$ be defined

$$\boldsymbol{\mu}_i = (\mu_{i1}, \mu_{i2}, \cdots, \mu_{iK}) \ ; \ i = 1, 2, \cdots, k \ ; \ \boldsymbol{\mu}_i \neq \boldsymbol{\mu}_j \text{ for } (i \neq j). \tag{3}$$

Let $\mu_i(x)$ be

$$\mu_i(x) = e_{i\textcircled{1}} x^{\textcircled{1}} + e_{i\textcircled{2}} x^{\textcircled{2}} + \cdots + e_{i\textcircled{\eta}} x^{\textcircled{\eta}} \ ; \ i = 1, 2, \cdots, k, \tag{4}$$

where the exponent $\textcircled{i}$ takes on a random value such that

$$0 \leq \textcircled{i} \leq K - 1 \ ; \ \textcircled{i} \neq \textcircled{j} \quad \text{for} \quad i \neq j, \tag{5}$$

and the coeffcient $e_{i\textcircled{j}}$, a random value over $\mathbb{F}_{2^m}$.

Let $\mu_i(x)$ be transformed into

$$\begin{aligned}\mu_i(x)x^g &\equiv r_i(x) \mod G(x),\\ &= r_{i1} + r_{i2}x + \cdots + r_{ig}x^{g-1}\ ; i = 1, 2, \cdots, k.\end{aligned} \tag{6}$$

The code word $v_i(x)$ generated by $G(x)$ is

$$v_i(x) = \mu_i(x)x^g + r_i(x) \equiv 0 \mod G(x). \tag{7}$$

Let the code words of $\{\boldsymbol{v}_i\}$ be

$$\begin{aligned}\boldsymbol{v}_1 &= (\mu_{11}, \mu_{12}, \cdots, \mu_{1K}, r_{11}, r_{12}, \cdots, r_{1g}),\\ \boldsymbol{v}_2 &= (\mu_{21}, \mu_{22}, \cdots, \mu_{2K}, r_{21}, r_{22}, \cdots, r_{2g}),\\ &\vdots\\ \boldsymbol{v}_k &= (\mu_{k1}, \mu_{k2}, \cdots, \mu_{kK}, r_{k1}, r_{k2}, \cdots, r_{kg}).\end{aligned} \tag{8}$$

Let A_r be

$$A_r = \begin{bmatrix} r_{11}, & r_{12}, & \cdots, & r_{1g} \\ r_{21}, & r_{22}, & \cdots, & r_{2g} \\ \vdots & \vdots & & \vdots \\ r_{k1}, & r_{k2}, & \cdots, & r_{kg} \end{bmatrix}. \tag{9}$$

The matrix A_r is transformed into

$$A_r \cdot P_I = \begin{bmatrix} u_{11}, & u_{12}, & \cdots, & u_{1g} \\ u_{21}, & u_{22}, & \cdots, & u_{2g} \\ \vdots & \vdots & & \vdots \\ u_{k1}, & u_{k2}, & \cdots, & u_{kg} \end{bmatrix}, \tag{10}$$

where P_I is a random column permutation matrix.

Let $\boldsymbol{u}_i$ be defined

$$\boldsymbol{u}_i = (u_{i1}, u_{i2}, \cdots, u_{ig}) \ \ ; \ \ i = 1, 2, \cdots, k. \tag{11}$$

We assume that the elements of set $\{\boldsymbol{u}_i\}$ are ordered as $\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_k$. The set $\{\boldsymbol{u}_i\}$ will be publicized. We shall refer to subscript ij as location j.

When Bob encrypts a message $\boldsymbol{a} = (a_1, a_2, \cdots, a_t)$, Bob transforms it into :

$$a_T(x) = a_1x^{[1]} + a_2x^{[2]} + \cdots + a_tx^{[t]}\ ;\ 0 \le [i] \le g - 1, \tag{12}$$

where the exponents $[1], [2], \cdots, [t]$ are randomly chosen by Bob under the condition that

$$1 \le [1] < [2] < \cdots < [t-1] < [t] \le g - 1. \tag{13}$$

We assume that every time Bob encrypts the message $\boldsymbol{a}$, he is required to transform $\boldsymbol{a}$ into $\boldsymbol{a}_T$ over all again. Bob then generates a random sequence $\boldsymbol{\alpha}$ over $\mathbb{F}_{2^m}$, using a transformation function $\varphi(*)$:

$$\varphi(\boldsymbol{a}_T) : \boldsymbol{a}_T \mapsto \boldsymbol{\alpha} = (\alpha_1, \alpha_2, \cdots, \alpha_k). \tag{14}$$

Let the word $\boldsymbol{w}$ be defined

$$\boldsymbol{w} = \alpha_1\boldsymbol{u}_1 + \alpha_2\boldsymbol{u}_2 + \cdots + \alpha_k\boldsymbol{u}_k. \tag{15}$$

The ciphertext $\boldsymbol{C}$ is then

$$\boldsymbol{C} = \boldsymbol{w} + \boldsymbol{a}_T. \tag{16}$$

We see that $\alpha_1\mu_1(x) + \alpha_2\mu_2(x) + \cdots + \alpha_k\mu_k(x)$ results in erasure errors, referred to as $E(x)$.
Theorem 1: Erasure error $E(x)$ due to $\boldsymbol{\alpha}$ is

$$\begin{aligned} E(x) &= \alpha_1\mu_1(x) + \alpha_2\mu_2(x) + \cdots + \alpha_k\mu_k(x) \\ &= \sum_{i=1}^{k} \alpha_i e_{i①} x^{①} + \sum_{i=1}^{k} \alpha_i e_{i②} x^{②} + \cdots + \sum_{i=1}^{k} \alpha_i e_{iⓗ} x^{ⓗ}. \end{aligned} \tag{17}$$

Proof : Straightforward. □

The following relation :

$$2t + \eta + 1 = D, \tag{18}$$

is required to hold so that the erasure error value and $\boldsymbol{a}$ may be correctly decoded.

Set of keys are :

Public key	:	$\{\boldsymbol{u}_i\}$.
Secret key	:	$\{\boldsymbol{\mu}_i\}, A_r, \{\boldsymbol{r}_i\}P_I$.

3 System oriented PKC, K(I)SOPKC

3.1 Preliminaries

K(I)SOPKC is constructed based on K(XIV)SE(1)PKC.

In Fig.2, we show the personnel organization of Alice's company.

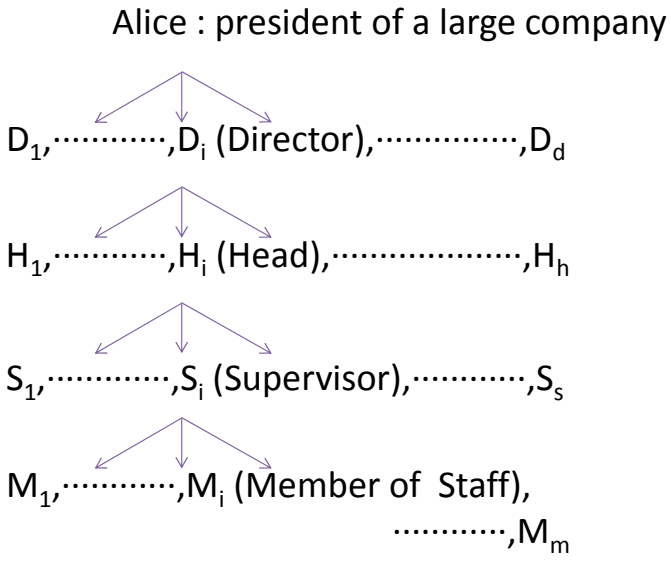

Fig. 2: Personnel organization of Alice's company

In Fig.3, let us show an example of system-oriented secure communication.

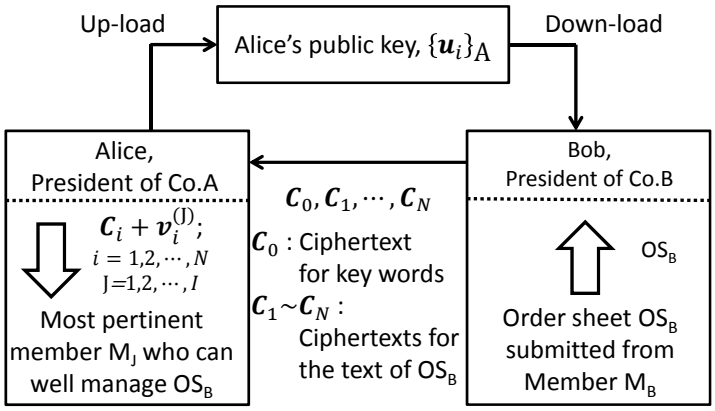

Fig. 3: An example of system-oriented secure communication

Referring to Figs.2 and 3, let us define several symbols :

$G_M(x)$: decryption key over $\mathbb{F}_{2^m}$ sent from Alice to all members of Alice's company, where the degreee of $G_M(x)$ is g.

OS_B : order sheet submitted from Member M_B.

$\boldsymbol{a}_0$: message that stands for the key words of OS_B, $(a_{01}, a_{02}, \cdots, a_{0t})$, where any component is assumed to be non-zero.

$\boldsymbol{a}_i$: message that stands for the text of OS_B,$(a_{i1}, a_{i2}, \cdots, a_{it}); i = 1, 2, \cdots, N$, where any component of $\boldsymbol{a}_i$ is assumed to be non-zero.

$a_{iT}(x)$: randomly permuted version of $\boldsymbol{a}_i$, $a_{i1}x^{[i1]} + a_{i2}x^{[i2]} + \cdots + a_{it}x^{[it]}$.

$[ij]$: exponent randomly selected by Bob.

$\boldsymbol{C}_i$: ciphertext, $(c_{i1}, c_{i2}, \cdots, c_{ig}); i = 1, 2, \cdots, N$.

c_{ij} : component of $\boldsymbol{C}_i$ at location j.

When Alice uploads the public $\{\boldsymbol{u}_i\}_A$, she performs the following :

(I)Alice, the president of Co.A, publishes the public key $\{\boldsymbol{u}_i\}_A$, where $\boldsymbol{u}_i$ is given by Eq.(11).
(II)Alice sends the following decryption key, k_i to Member M_i ; $i = 1, 2, \cdots, I$.

$$\begin{aligned} k_i &= \{(i1), (i2), \cdots, (i\eta)\}; i = 1, 2, \cdots, I \lesssim 10^4; \\ & 1 \leq (i1) < (i2) < \cdots < (i\eta) = K. \end{aligned} \tag{19}$$

The component (ij)'s ; $j = 1, 2, \cdots, \eta$, are randomly selected by Alice and secretly sent to Member M_i. The (ij)'s are Member M_i's secret key for decoding erasure value.

(III)Alice calculates

$$\begin{aligned} x^i \cdot x^g &\equiv \lambda_i(x) \bmod G_M(x) \\ &= \lambda_{i1} + \lambda_{i2}x + \cdots + \lambda_{ig}x^{g-1}; i = 0, 1, \cdots, K-1. \end{aligned} \tag{20}$$

3.2 Outline of K(I)SOPKC

For an easy understanding of the present paper, let us describe an outline of K(I)SOPKC :

O1 : Alice publishes her public key $\{\boldsymbol{u}_i\}_A$.
At the same time :
(i)she secretly sends a set of decryption key k_i to Member M_i, of Co.A ; $i = 1, 2, \cdots, I$
(ii)she sends another decryption key, $G_M(x)$, to all the members of Co.A.

O2 : Referring to $\{\boldsymbol{u}_i\}_A$, Bob encrypts an order sheet, OS_B, of the size $10 \sim 80$KB to a series of ciphertexts, $\boldsymbol{C}_0, \boldsymbol{C}_1, \cdots, \boldsymbol{C}_N$, where $\boldsymbol{C}_0$ is the encrypted version of the key words of OS_B and $\boldsymbol{C}_1$, $\boldsymbol{C}_2, \cdots, \boldsymbol{C}_N$, those of the text of OS_B. Bob sends them to Alice.

O3 : Alice decrypts only $\boldsymbol{C}_0$ and decodes the key words of OS_B.

O4 : From the key words, Alice decides on the most suitable member, M_J ; $1 \leq J \leq I$ who manages the order sheets.

O5 : Given $\boldsymbol{C}_i$, Alice transforms $\boldsymbol{C}_i \mapsto \boldsymbol{C}_i^{(J)} = \boldsymbol{C}_i + \boldsymbol{v}_i^{(J)}$; $i = 1, 2, \cdots, N$.
Alice provides all $\boldsymbol{v}_i^{(J)}$'s before completing the receiving of whole $\boldsymbol{C}_1$(See Fig.4).

O6 : Alice sends $\boldsymbol{C}_i^{(J)}$ to $M_J; i = 1, 2, \cdots, N$.

O7 : Given $\boldsymbol{C}_i^{(J)}$, M_J decodes message $\boldsymbol{a}_i; i = 1, 2, \cdots, N$, using erasures and errors decoding [11] for Reed-Solomon code generated with $G_M(x)$.

3.3 Encryption and decryption process of K(I)SOPKC

Encryption process for $\boldsymbol{a}_0$ that represents key words.

Step 1 : Given $\boldsymbol{a}_0 = (a_{01}, a_{02}, \cdots, a_{0t})$, Bob randomly selects the locations of the components of ciphertext $\boldsymbol{C}_0, [01], [02], \cdots, [0t]$.

Step 2 : Bob constructs $a_{0T}(x) = a_{01}x^{[01]} + a_{02}x^{[02]} + \cdots + a_{0t}x^{[0t]}$.

Step 3 : Bob generates $\varphi(\boldsymbol{a}_0) = (\boldsymbol{\alpha}_0, \boldsymbol{\alpha}_1, \cdots, \boldsymbol{\alpha}_N)$, where $\boldsymbol{\alpha}_i$ is $\boldsymbol{\alpha}_i = (\alpha_{i1}, \alpha_{i2}, \cdots, \alpha_{ik})$, and we assume that the Hamming weight of $\boldsymbol{\alpha}_i$ is η.

Step 4 : Bob calculates the word :
$\boldsymbol{w}_0 = \alpha_{01}\boldsymbol{u}_1 + \alpha_{02}\boldsymbol{u}_2 + \cdots + \alpha_{0k}\boldsymbol{u}_k$.

Step 5 : Bob calculates the ciphertext :
$\boldsymbol{C}_0 = \boldsymbol{w}_0 + \boldsymbol{a}_{0T}$.

Given $\boldsymbol{C}_0$, Alice performs the followings :

Decryption process for decoding $\boldsymbol{a}_0$ and $\boldsymbol{\alpha}_0$.

Step 1 : Recieving $\boldsymbol{C}_0$, Alice decodes $\boldsymbol{w}_0$ and $\boldsymbol{a}_0$, with erasure and error decoding [11].

Step 2 : From $\boldsymbol{a}_0$ that represents the key words of OS_B, Alice selects the most suitable person who will well manage OS_B. At the same time, Alice generates
$\varphi(\boldsymbol{a}_0) = \boldsymbol{\alpha} = (\boldsymbol{\alpha}_0, \boldsymbol{\alpha}_1, \cdots, \boldsymbol{\alpha}_N)$, where $\alpha_i = (\alpha_{i1}, \alpha_{i2}, \cdots, \alpha_{ik})$.

Step 3 : Alice calculates the followings :
(i) $\alpha_{i1}\boldsymbol{u}_1 + \alpha_{i2}\boldsymbol{u}_2 + \cdots + \alpha_{ik}\boldsymbol{u}_k = \boldsymbol{\beta}_i$; $i = 1, 2, \cdots, N$.
(ii) $\gamma_{i1}\lambda_{(J1)}(x) + \gamma_{i2}\lambda_{(J2)}(x) + \cdots + \gamma_{i\eta}\lambda_{(J\eta)}(x) = t_i^{(J)}(x)$,
where γ_{ij} is a randomly chosen element from $\mathbb{F}_{2^m}$,
(iii) $\boldsymbol{v}_i^{(J)} = \boldsymbol{t}_i^{(J)} + \boldsymbol{\beta}_i$; $i = 1, 2, \cdots, N$.

We shall see later that, all $\boldsymbol{v}_i^{(J)}$'s can be provided before completing the reception of whole $\boldsymbol{C}_1$(See Fig.4).

Encryption process for $\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_N$ by Bob :

Step 1 : Given $\boldsymbol{a}_i$, Bob calculates
$\boldsymbol{w}_i = \alpha_{i1}\boldsymbol{u}_1 + \alpha_{i2}\boldsymbol{u}_2 + \cdots + \alpha_{ik}\boldsymbol{u}_k$; $i = 1, 2, \cdots, N$,
where $\boldsymbol{\alpha}_i = (\alpha_{i1}, \alpha_{i2}, \cdots, \alpha_{ik})$ is provided in Step 3 of Encryption process for $\boldsymbol{a}_0$.

Step 2 : Bob transforms $\boldsymbol{a}_i = (a_{i1}, a_{i2}, \cdots, a_{it})$ into
$a_{iT}(x) = a_{i1}x^{[i1]} + a_{i2}x^{[i2]} + \cdots + a_{it}x^{[it]}$; $i = 1, 2, \cdots, N$.

Step 3 : Bob calculates
$\boldsymbol{C}_i = \boldsymbol{w}_i + \boldsymbol{a}_{iT}; i = 1, 2, \cdots, N$.

Forwarding process for $\boldsymbol{C}_1, \boldsymbol{C}_2, \cdots, \boldsymbol{C}_N$, by Alice :

Step 1 : Given $\boldsymbol{C}_i$, Alice simply adds $\boldsymbol{v}_i^{(J)}$ on $\boldsymbol{C}_i$, yielding $\boldsymbol{C}_i^{(J)}$.

Step 2 : Alice sends $\boldsymbol{C}_i^{(J)}$ to M_J through a public channel.

Decryption process of $\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_N$ by M_J :

Step 1 : Receiving $\boldsymbol{C}_i^{(J)}$ from Alice, M_J decodes erasure value and $\boldsymbol{a}_{iT}$, based on erasure and error decoding of Reed-Solomon code [11].

Step 2 : Disregarding erasure value, M_J transforms
$\boldsymbol{a}_{it} \mapsto \boldsymbol{a}_i$.

The time chart of ciphertext sequences are given in Fig.4.

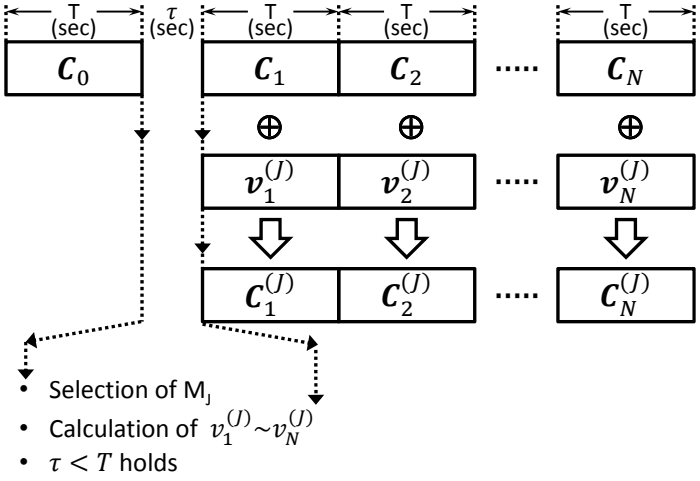

Fig. 4: Time chart of ciphertext sequences

Remark 1 : Any form of re-encryption on $\boldsymbol{C}_i$ based on the conventional common key cryptsystem cannnot be completed before receiving the whole $\boldsymbol{C}_i$. As a result the delay due to the re-encryption process based on the common key cryptosystem requires the delay of larger than T(sec). On the other hand in K(I)SOPKC, it is possible to add $\boldsymbol{v}_i^{(J)}$ on $\boldsymbol{C}_i$, only after $\tau(< T)$ sec, as shown in Fig.4.

3.4 Calculation of $\boldsymbol{v}_i^{(J)}$

At the time when Alice publicizes her public key $\{\boldsymbol{u}_i\}$, she secretly sends decryption key $k_i = \{(i1), (i2), \cdots, (i\eta)\}$ to Member M_i. She also calculates $\boldsymbol{v}_i^{(J)}$ as shown below.

In order to calculates $\boldsymbol{v}_i^{(J)}$, Alice performs the followings :

(i) Decoding of the key words $\boldsymbol{a}_0 = (a_{01}, a_{02}, \cdots, a_{0t})$ using Euclidean decoding algorithm [11].
(ii) Determination of the most suitable member, by referring to the key words.
(iii) Transformation of $\boldsymbol{a}_0$ into $\boldsymbol{a}_{0T}$.
(iv) Generation of $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_0, \boldsymbol{\alpha}_1, \cdots, \boldsymbol{\alpha}_N)$ through the transformation $\varphi(\boldsymbol{a}_0)$.
(v) Calculation of $\boldsymbol{t}_i^{(J)}$ such that
$\boldsymbol{t}_i^{(J)} = \gamma_{i1}\boldsymbol{\lambda}_{(J1)} + \gamma_{i2}\boldsymbol{\lambda}_{(J2)} + \cdots + \gamma_{i\eta}\boldsymbol{\lambda}_{(J\eta)}; i = 1, 2, \cdots, N,$
where γ_{ij}'s are randomly chosen element of $\mathbb{F}_{2^m}$.
The calculation of $\boldsymbol{t}_i^{(J)}$ requires 1 step of multiplication in a parallel processing based on table look up method over $\mathbb{F}_{2^m}$ and $\lceil \log_2 \eta \rceil$ steps of modulo 2 addition, in a parallel proccesing.
(vi) Calculation of $\boldsymbol{\beta}_i$ such that
$\boldsymbol{\beta}_i = \alpha_{i1}\boldsymbol{u}_1 + \alpha_{i2}\boldsymbol{u}_2 + \cdots + \alpha_{ik}\boldsymbol{u}_k; i = 1, 2, \cdots, N.$
The calculation of $\boldsymbol{\beta}_i$ requires 1 step of multiplication, in a parallel procceing based on the table look up mathod over $\mathbb{F}_{2^m}$ and $\lceil \log_2 k \rceil$ steps of modulo 2 addition, in a parallel procceing.
(vii) Calculation of $\boldsymbol{v}_i^{(J)}$ such that
$\boldsymbol{v}_i^{(J)} = \boldsymbol{t}_i^{(J)} + \boldsymbol{\beta}_i^{(J)}; i = 1, 2, \cdots, N.$
The calculation of $\boldsymbol{v}_i^{(J)}$ given above requires 1 step of modulo 2 addtion, in a parallel proccesing.

We see that the calculation of $\boldsymbol{v}_i^{(J)}$, in all, can be performed within less than 100 steps (in a parallel processing) of modulo 2 addition, for $\eta = 256, k = 128$ and $m = 10 \sim 16$. For $N \lesssim 100$, it would be quite possible to caculate all $v_i^{(J)}$'s before completing the receiving of a whole $\boldsymbol{C}_1$(See Fig.4).

3.5 Security considerations

Remark 2 : The using of Reed-Solomon codes is very attractive, particulaly for small m. Because they are extensively used for the various storage systems such as CD, DVD, etc. However, when the degree of the generator polynomial is less than 80, the total number of different Reed-Solomon codes is not at all sufficiently large, compared with the Goppa codes. As the result, the using of Reed-Solomon code has been supposed to be dangerous as the generator polynomial can be estimated without much difficulty compared with the Goppa code, unless m takes on a large value. As we construct K(I)SOPKC over a small field such as $\mathbb{F}_{2^{10}} \sim \mathbb{F}_{2^{16}}$, we assume that $G(x)$ is correctly given, from a conservative point of view.

In the followings, we assume that K' is chosen as $K' = 500$.

Attack 1: Attack on secret keys of K(XIV)SE(1)PKC.

The secret keys $\{\boldsymbol{\mu}_i\}$, $\{\boldsymbol{r}_i\}$, A_r, P_I can be disclosed according to the following process :
Step 1 : Letting the estimated value of $\boldsymbol{\mu}_i$ be $\hat{\boldsymbol{\mu}}_i$, we calculate
$\hat{\boldsymbol{\mu}}_i(x)x^g \equiv \hat{r}_1 + \hat{r}_2 x + \cdots + \hat{r}_k x_k \bmod G(x)$.
Step 2 : When $\hat{r}_i = u_j$ holds for all pairs of $(i, j); i, j = 1, 2, \cdots, g$, we conclude that $\hat{\boldsymbol{\mu}}_i = \boldsymbol{\mu}_i$.

It is easy to see that when once $\boldsymbol{\mu}_i$ is estimated correctly, all other secret keys are disclosed.

Let the probability that $\boldsymbol{\mu}_i$ is estimated correctly be denoted $P_c[\hat{\boldsymbol{\mu}}_i]$. The $P_c[\hat{\boldsymbol{\mu}}_i]$ is

$$P_c[\hat{\boldsymbol{\mu}}_i] = \begin{pmatrix} K' \\ \eta \end{pmatrix}^{-1} (2^m - 1)^{-\eta}. \tag{21}$$

In order to be secure against Attack 1, we recommend that $P_c[\hat{\boldsymbol{\mu}}_i]$ be

$$P_c[\hat{\boldsymbol{\mu}}_i] \leq 2^{-80} = 8.27 \times 10^{-25}. \tag{22}$$

Attack 2: Attack on M_i's secret key $k_i = \{(i1), (i2), \cdots, (i\eta)\}$.

Because (ij)'s satisfy

$$1 \leq (ij) \leq K', \tag{23}$$

the probability that all the elements of k_i can be successfully estimated, $P_c[\hat{k}_i]$, is

$$P_c[\hat{k}_i] = \left(\begin{array}{c} K' \\ \eta \end{array} \right)^{-1}. \tag{24}$$

We conclude that K(XIV)SE(1)PKC can be made sufficiently secure provided that $P_c[\hat{k}_i] < 2^{-80}$.

Let us consider the following attack, where efs implies an error free symbol among g components of ciphertext.

Attack 3 : An exhaustive attack for discclosing η efs's among the g components of a ciphertext.

The probability that the η efs's is correctly estimated, $P_c[\hat{\text{efs}}]$, is

$$P_c[\hat{\text{efs}}] = \frac{\left(\begin{array}{c} g-t \\ \eta \end{array} \right)}{\left(\begin{array}{c} g \\ \eta \end{array} \right)}. \tag{25}$$

Example I : $m = 8, g = 128, \eta = 64, t = 32$.
$P_c[\hat{\text{efs}}] = 1.24 \times 10^{-12}$, not a sufficiently small value.
Example II: $m = 16, g = 512, \eta = 96, t = 208$.
$P_c[\hat{\text{efs}}] = 1.18 \times 10^{-25} < 2^{-80}$
Example III: $m = 16, g = 1024, \eta = 128, t = 464$.
$P_c[\hat{\text{efs}}] = 9.49 \times 10^{-36}$.

As we see in Example 1, for $m = 8$, K(XIV)SE(1)PKC is not secure against Attack 3. Throughout this paper we reccomend that $P_c[\hat{\text{efs}}]$ be less than 10^{-30}. In the following attack, for easy understanding, ciphertext C_i will be simply denoted as C.

Attack 4 : Attack on $\boldsymbol{a}_T$ added on $\boldsymbol{C}$.

Step 1 : Charles, an attacker, successfully estimates η efs's at the locations $L_1, L_2, \cdots, L_\eta$ of the ciphertext $\boldsymbol{C}$

Step 2 : Charles randomly selects η $\boldsymbol{u}_i$'s that constructs n-dimensional vector space V_η.

Step 3 : From $\boldsymbol{u}_j \in V_\eta$, let $\boldsymbol{\rho}_j$ be defined
$\boldsymbol{\rho}_j = (\rho_{jL_1}, \rho_{jL_2}, \cdots, \rho_{jL_\eta}); j = 1, 2, \cdots, \eta,$
where ρ_{jL_l} is the symbol at the error-free location L_l of the ciphertext $\boldsymbol{C}$.

Step 4 : Charles constructs the following equation :
$\boldsymbol{w} = x_1\boldsymbol{\rho}_1 + x_2\boldsymbol{\rho}_2 + \cdots + x_\eta\boldsymbol{\rho}_\eta$
$= (w_1, w_2, \cdots, w_\eta),$
where $x_1, x_2, \cdots, x_\eta$ are unknown variables.

Step 5 : From Eq.(27), Charles constructs the following linear simultaneous equations :

$$x_1\rho_{1L_1} + x_2\rho_{2L_1} + \cdots + x_\eta\rho_{\eta L_1} = w_1,$$
$$x_1\rho_{1L_2} + x_2\rho_{2L_2} + \cdots + x_\eta\rho_{\eta L_2} = w_2,$$
$$\vdots$$
$$x_1\rho_{1L_\eta} + x_2\rho_{2L_\eta} + \cdots + x_\eta\rho_{\eta L_\eta} = w_\eta.$$

Step 6 : Charles obtains an unique solution : $x_1 = \widetilde{x}_1, x_2 = \widetilde{x}_2, \cdots, x_\eta = \widetilde{x}_\eta$, and then discloses $\boldsymbol{a}_{iT}$ as $\widetilde{\boldsymbol{a}}_{iT} = \boldsymbol{C} - (\widetilde{x}_1\boldsymbol{\rho}_1 + \widetilde{x}_2\boldsymbol{\rho}_2 + \cdots + \widetilde{x}_\eta\boldsymbol{\rho}_\eta)$.

We conclude that K(I)SOPKC is not secure against Attack 4, when once η efs's are correctly estimated.

In Table. 1, we show several examples of K(I)SOPKC. The size of public key, S_{PK}, the length of ciphertext, $|\boldsymbol{C}_i|$ and the coding rate, ρ, are

$$\begin{aligned} S_{PK} &= mgk(\text{bits}). \\ |\boldsymbol{C}_i| &= gm(\text{bits}). \\ \rho &= \frac{t}{g}. \end{aligned} \tag{26}$$

Table. 1: Examples of K(I)SOPKC ($K' = 500, k = 128$)

m	g	η	t	$P_c[\hat{\text{efs}}]$	ρ	S_{PK}(KB)
10	512	128	192	$3.46e-32$	0.375	81.9
12	600	120	240	$1.45e-31$	0.40	115
14	800	160	320	$7.42e-42$	0.40	179
16	1000	200	400	$3.79e-52$	0.40	258

We see that the coding rate is less than 0.5. In the following sub-section we present a slightly modified version of K(I)SOPKC, referred to as K_p(XIII)SE(1)PKC.

3.6 Improving the coding rate

In this sub-section, let us improve the coding rate. The improved version is referred to as K_p(I)SOPKC.

Let us modify K(I)SOPKC through following steps :

S1 : Let the message $\boldsymbol{a} = (a_1, a_2, \cdots, a_t)$ be partitioned into :
$\boldsymbol{a} \mapsto (\boldsymbol{a}_\pi; \boldsymbol{a}_\theta)$,
where
$\boldsymbol{a}_\pi = (a_1, a_2, \cdots, a_\pi)$,
$\boldsymbol{a}_\theta = (a_{\pi+1}, a_{\pi+2}, \cdots, a_{\pi+\theta})$,
$\pi + \theta = t$.

S2 : Let $\boldsymbol{a}_\pi(x)$be
$a_\pi(x) = a_1 + a_2 x + \cdots + a_\pi x^{\pi-1}$.

S3 : Let $a_{\pi+i}$, a component of $\boldsymbol{a}_\theta$ be randomly located at $[j]$ by Bob, where we let $[j] > \pi$.

The probability $P_c[\hat{\text{efs}}]$ is

$$P_c[\hat{\text{efs}}] = \frac{\begin{pmatrix} g-\pi-\theta \\ \eta \end{pmatrix}}{\begin{pmatrix} g-\pi \\ \eta \end{pmatrix}}. \tag{27}$$

The coding rate, ρ_p, is

$$\rho_p = \frac{\pi + \theta}{g}. \tag{28}$$

Let us show several examples in Table. 2.

Table. 2: Examples of K_p(I)SOPKC (k=128,K'=500)

m	g	η	θ	π	$P_c[\{(\hat{\boldsymbol{u}}_i)_\theta\}]$	ρ	S_{PK}(KB)
10	500	200	50	200	$3.2e-29$	0.500	80
11	1000	200	75	650	$1.9e-34$	0.725	176
12	2000	200	100	1600	$4.0e-38$	0.850	384
13	4000	200	100	3600	$4.0e-38$	0.925	832
14	8000	200	100	7600	$4.0e-38$	0.963	1792
15	16000	200	100	15600	$4.0e-38$	0.981	3840
16	32000	200	100	31600	$4.0e-38$	0.991	8192

4 Conclusion

We presented a new type of PKC, system oriented PK, referred to as K(I)SOPKC. We have shown that K(I)SOPKC can be well adapted to the various systems and organization of the present day network society.

References

[1] M. Kasahara, "A New Class of Public Key Cryptosystems Constructed Based on Reed-Solomon Codes K(XII)SE(1)PKC.– Along with a presentation of K(XII)SE(1)PKC over $\mathbb{F}_{2^8}$, the field extensively used for various storage and transmission systems –", Cryptology ePrint Archive, Report, 2013/363 (2013-06).

[2] M. Kasahara, "Presentation of a new class of public key cryptosystems K(XIII)SE(1)PKC along with K_p(XIII)SE(1)PKC that realizes the coding rate of exactly 1.0, constructed by modifying K(XII)SE(1)PKC.", Cryptology ePrint Archive, Report, 2013/605 (2013-09).

[3] M. Kasahara "Construction of New class of Linear Multivariate Public Key Cryptosystem - Along With a Note on the Number 9999990 and its Application", Technical Report of IEICE, ISEC 2009-44 (2009-09).

[4] M. Kasahara "A New Class of Public Key Cryptosystems Constructed Based on Perfect Error-Correcting Codes Realizing Coding Rate of exactly 1.0", Cryptology ePrint Archive , Report 2010/139 (2010-03).

[5] M. Kasahara: "Public Key Cryptosystems Constructed Based on Pseudo Cyclic Codes, K(IX)SE(1)PKC, Realizing Coding Rate of Exactly 1.0", Cryptology ePrint Archive, Report 2011/545, (2011-09).

[6] R. J. McEliece: "A Public-key Cryptosystem Based on Algebraic Coding Theory", DSN Progress Report, no.42-44, pp.114-116 (1978).

[7] J. C. Faugere and A. Otomoni, L. Perret, J. P. Tillich: "Algebraic Cryptanalysis of McElierce Variants with Compact Keys", Eurocrypt'10.

[8] E. M. Gabidulin: "Public-key cryptosystems based on linear codes", Report 95-30, TU Delft (1995).

[9] S. Tsujii and M. Gotaishi, "Public Key Cryptsystems for Selected Communication between Organizations based on Complementary STS-MPKC", 2011 SCIS, (2011-01).

[10] M. Kasahara: "On Cryptography", Proc. of Symposium on Information and Communication Network Security, pp.154-179 (Feb. 1986).

[11] Y. Sugiyama, M. Kasahara, S. Hirasawa and T. Namekawa: "An Erasures-and-Errors decoding Algorithm for Goppa Codes", IEEE Trans. on Inform. Theory, IT-22, 2, pp.238-241 (1976-03).