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 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]∼[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 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 \( \mathbf{v} = (v_1, v_2, \ldots, v_n) \) as

\[
\tilde{\mathbf{v}} = (\tilde{v}_1, \tilde{v}_2, \ldots, \tilde{v}_n).
\]

(1)

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

\[
v(x) = v_1 + v_2 x + \cdots + v_n x^{n-1}.
\]

(2)

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

---

*Research Institute for Science and Engineering, Waseda University.
†Research and Development Initiative, Chuo University. kasahara@oogu.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 cryptography [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.

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

Let us define several symbols.

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

Let the vector \( \mu_i \) over \( \mathbb{F}_{2^m} \) be defined
\[ \mu_i = (\mu_{i1}, \mu_{i2}, \cdots, \mu_{iK}) ; \quad i = 1, 2, \cdots, k ; \quad \mu_i \neq \mu_j \text{ for } (i \neq j). \quad (3) \]

Let \( \mu_i(x) \) be
\[ \mu_i(x) = e_{i(1)}x^{\circ(1)} + e_{i(2)}x^{\circ(2)} + \cdots + e_{i(k)}x^{\circ(k)} ; \quad i = 1, 2, \cdots, k, \quad (4) \]
where the exponent \( \circ \) takes on a random value such that
\[ 0 \leq \circ \leq K - 1 ; \quad \circ \neq \circ \text{ for } i \neq j, \quad (5) \]
and the coefficient $c_{ij}$, a random value over $\mathbb{F}_{2^m}$.

Let $\mu_i(x)$ be transformed into

$$\mu_i(x)x^g \equiv r_i(x) \mod G(x),$$

$$= r_{i1} + r_{i2}x + \cdots + r_{ig}x^{g-1}; i = 1, 2, \ldots, k. \quad (6)$$

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

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

Let the code words of $\{v_i\}$ be

$$v_1 = (\mu_{11}, \mu_{12}, \cdots, \mu_{1K}, r_{11}, r_{12}, \cdots, r_{1g}),$$

$$v_2 = (\mu_{21}, \mu_{22}, \cdots, \mu_{2K}, r_{21}, r_{22}, \cdots, r_{2g}),$$

$$\vdots$$

$$v_k = (\mu_{k1}, \mu_{k2}, \cdots, \mu_{kK}, r_{k1}, r_{k2}, \cdots, r_{kg}). \quad (8)$$

Let $A_r$ be

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

The matrix $A_r$ is transformed into

$$A_rP_1 = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1g} \\ u_{21} & u_{22} & \cdots & u_{2g} \\ \vdots & \vdots & \ddots & \vdots \\ u_{k1} & u_{k2} & \cdots & u_{kg} \end{bmatrix}, \quad (10)$$

where $P_1$ is a random column permutation matrix.

Let $u_i$ be defined

$$u_i = (u_{i1}, u_{i2}, \cdots, u_{ig}) \ ; \ i = 1, 2, \cdots, k. \quad (11)$$

We assume that the elements of set $\{u_i\}$ are ordered as $u_1, u_2, \cdots, u_k$. The set $\{u_i\}$ will be publicized. We shall refer to subscript $ij$ as location $j$.

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

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

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

$$1 \leq [1] < [2] < \cdots < [t - 1] < [t] \leq g - 1. \quad (13)$$

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

$$\varphi(a_T) : a_T \mapsto \alpha = (\alpha_1, \alpha_2, \cdots, \alpha_k). \quad (14)$$

Let the word $w$ be defined

$$w = \alpha_1u_1 + \alpha_2u_2 + \cdots + \alpha_ku_k. \quad (15)$$
The ciphertext $C$ is then

$$C = w + a_T. \quad (16)$$

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 $\alpha$ is

$$E(x) = \alpha_1\mu_1(x) + \alpha_2\mu_2(x) + \cdots + \alpha_k\mu_k(x)$$

$$= \sum_{i=1}^{k} \alpha_i e_i(1)x^{i(1)} + \sum_{i=1}^{k} \alpha_i e_i(2)x^{i(2)} + \cdots + \sum_{i=1}^{k} \alpha_i e_i(\Omega)x^{i(\Omega)}. \quad (17)$$

**Proof:** Straightforward. \qed

The following relation:

$$2t + \eta + 1 = D, \quad (18)$$

is required to hold so that the erasure error value and $\alpha$ may be correctly decoded.

Set of keys are:

- **Public key:** \{u\}.
- **Secret key:** \{\mu\}, Ar, \{r\}P.

## 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.

Alice : president of a large company

- $D_1, \cdots, D_i$ (Director), $\cdots, D_d$
- $H_1, \cdots, H_i$ (Head), $\cdots, H_h$
- $S_1, \cdots, S_i$ (Supervisor), $\cdots, S_s$
- $M_1, \cdots, M_i$ (Member of Staff), $\cdots, M_m$

Fig. 2: Personnel organization of Alice’s company

In Fig. 3, let us show an example of system-oriented secure communication.
Referring to Figs.2 and 3, let us define several symbols:

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

\( OS_B \) : order sheet submitted from Member \( M_B \).

\( a_0 \) : 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.

\( a_i \) : 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 \( a_i \) is assumed to be non-zero.

\( a_{i,j}(x) \) : randomly permuted version of \( a_i, a_{i1}x^{[i1]} + a_{i2}x^{[i2]} + \cdots + a_{it}x^{[it]} \).

\([ij] \) : exponent randomly selected by Bob.

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

\( c_{ij} \) : component of \( C_i \) at location \( j \).

When Alice uploads the public \( \{u_i\}_A \), she performs the following:

(I) Alice, the president of Co.A, publishes the public key \( \{u_i\}_A \), where \( u_i \) is given by Eq. (11).

(II) Alice sends the following decryption key, \( k_i \) to Member \( M_i \); \( i = 1, 2, \cdots, I \).

\[ k_i = \{(i1), (i2), \cdots, (i\eta)\}; i = 1, 2, \cdots, I \lesssim 10^4; \]
\[ 1 \leq (i1) < (i2) < \cdots < (i\eta) = K. \quad (19) \]

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

\[ x^i \cdot x^g \equiv \lambda_i(x) \mod G_M(x) \]
\[ = \lambda_{i1} + \lambda_{i2}x + \cdots + \lambda_{ig}x^{g-1}; i = 0, 1, \cdots, K - 1. \quad (20) \]
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 \( \{u_i\}_\Lambda \).

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 \( \{u_i\}_\Lambda \), Bob encrypts an order sheet, \( OS_B \), of the size 10 \( \sim \) 80KB to a series of ciphertexts, \( C_0, C_1, \cdots, C_N \), where \( C_0 \) is the encrypted version of the key words of \( OS_B \) and \( C_1, C_2, \cdots, C_N \), those of the text of \( OS_B \). Bob sends them to Alice.

O3: Alice decrypts only \( C_0 \) 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 \( C_1 \), Alice transforms \( C_i \mapsto C_i^{(J)} = C_i + v_i^{(J)} ; i = 1, 2, \cdots, N \). Alice provides all \( v_i^{(J)}'s \) before completing the receiving of whole \( C_1 \) (See Fig.4).

O6: Alice sends \( C_i^{(J)} \) to \( M_i ; i = 1, 2, \cdots, N \).

O7: Given \( C_i^{(J)}, M_J \) decodes message \( a_i ; i = 1, 2, \cdots, N \), 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 \( a_0 \) that represents key words.

Step 1: Given \( a_0 = (a_{01}, a_{02}, \cdots, a_{0r}) \), Bob randomly selects the locations of the components of ciphertext \( C_0 \), \{01\}, \{02\}, \cdots, \{0r\}.

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

Step 3: Bob generates \( \varphi(a_0) = (\alpha_0, \alpha_1, \cdots, \alpha_N) \), where \( \alpha_i = (\alpha_{i1}, \alpha_{i2}, \cdots, \alpha_{ik}) \), and we assume that the Hamming weight of \( \alpha_i \) is \( \eta \).

Step 4: Bob calculates the word:
\[ w_0 = a_{01}u_1 + a_{02}u_2 + \cdots + a_{0r}u_r. \]

Step 5: Bob calculates the ciphertext:
\[ C_0 = w_0 + a_{0T}. \]

Given \( C_0 \), Alice performs the followings:

Decryption process for decoding \( a_0 \) and \( \alpha_0 \).

Step 1: Receiving \( C_0 \), Alice decodes \( w_0 \) and \( a_0 \), with erasure and error decoding [11].

Step 2: From \( a_0 \) 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
\[ \varphi(a_0) = \alpha = (\alpha_0, \alpha_1, \cdots, \alpha_N), \] where \( \alpha_i = (\alpha_{i1}, \alpha_{i2}, \cdots, \alpha_{ik}) \).

Step 3: Alice calculates the followings:
(i) \( \alpha_{i1}u_1 + \alpha_{i2}u_2 + \cdots + \alpha_{ik}u_k = \beta_i ; i = 1, 2, \cdots, N \).
(ii) \( \gamma_{i1}\lambda_{1J}(x) + \gamma_{i2}\lambda_{2J}(x) + \cdots + \gamma_{i\eta}\lambda_{\eta J}(x) = t_i^{(J)}(x) \),
where \( \gamma_{ij} \) is a randomly chosen element from \( F_{2^m} \).
(iii) \( v_i^{(J)} = t_i^{(J)} + \beta_i ; i = 1, 2, \cdots, N \).

We shall see later that, all \( v_i^{(J)}'s \) can be provided before completing the reception of whole \( C_1 \) (See Fig.4).
Encryption process for \(a_1, a_2, \ldots, a_N\) by Bob:

Step 1: Given \(a_i\), Bob calculates
\[
w_i = \alpha_{i1}u_1 + \alpha_{i2}u_2 + \cdots + \alpha_{ik}u_k; \quad i = 1, 2, \ldots, N,
\]
where \(\alpha_i = (\alpha_{i1}, \alpha_{i2}, \ldots, \alpha_{ik})\) is provided in Step 3 of Encryption process for \(a_0\).

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

Step 3: Bob calculates
\[
C_i = w_i + a_{iT}; \quad i = 1, 2, \ldots, N.
\]

Forwarding process for \(C_1, C_2, \ldots, C_N\), by Alice:

Step 1: Given \(C_i\), Alice simply adds \(v_i^{(J)}\) on \(C_i\), yielding \(C_i^{(J)}\).

Step 2: Alice sends \(C_i^{(J)}\) to \(M_J\) through a public channel.

Decryption process of \(a_1, a_2, \ldots, a_N\) by \(M_J\):

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

Step 2: Disregarding erasure value, \(M_J\) transforms
\[a_{it} \mapsto a_i.\]

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

![Time chart of ciphertext sequences](image)

**Remark 1:** Any form of re-encryption on \(C_i\) based on the conventional common key cryptosystem cannot be completed before receiving the whole \(C_i\). 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 \(v_i^{(J)}\) on \(C_i\), only after \(\tau < T\) sec, as shown in Fig.4.

### 3.4 Calculation of \(v_i^{(J)}\)

At the time when Alice publicizes her public key \(\{u_i\}\), she secretly sends decryption key \(k_i = \{(i1), (i2), \ldots, (it)\}\) to Member \(M_i\). She also calculates \(v_i^{(J)}\) as shown below.
In order to calculate $v_1^{(J)}$, Alice performs the followings:

(i) Decoding of the key words $a_0 = (a_{01}, a_{02}, \cdots, a_{0t})$ using Euclidean decoding algorithm [11].
(ii) Determination of the most suitable member, by referring to the key words.
(iii) Transformation of $a_0$ into $a_0^r$.
(iv) Generation of $\alpha = (\alpha_0, \alpha_1, \cdots, \alpha_N)$ through the transformation $\varphi(a_0)$.
(v) Calculation of $t_1^{(J)}$ such that
$$t_1^{(J)} = \gamma_{i1}\lambda_{(J1)} + \gamma_{i2}\lambda_{(J2)} + \cdots + \gamma_{i\eta}\lambda_{(J\eta)}; i = 1, 2, \cdots, N,$$
where $\gamma_{ij}$'s are randomly chosen element of $F_{2^m}$.

The calculation of $t_1^{(J)}$ requires 1 step of multiplication in a parallel processing based on table look up method over $F_{2^m}$ and $\lceil \log_2 \eta \rceil$ steps of modulo 2 addition, in a parallel processing.

(vi) Calculation of $\beta_i$ such that
$$\beta_i = \alpha_{i1}u_1 + \alpha_{i2}u_2 + \cdots + \alpha_{ik}u_k; i = 1, 2, \cdots, N.$$

The calculation of $\beta_i$ requires 1 step of multiplication, in a parallel procceing based on the table look up method over $F_{2^m}$ and $\lceil \log_2 k \rceil$ steps of modulo 2 addition, in a parallel processing.

(vii) Calculation of $v_1^{(J)}$ such that
$$v_1^{(J)} = t_1^{(J)} + \beta_1^{(J)}; i = 1, 2, \cdots, N.$$

The calculation of $v_1^{(J)}$ given above requires 1 step of modulo 2 addition, in a parallel processing.

We see that the calculation of $v_1^{(J)}$, 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 $N \leq 100$, it would be quite possible to calculate all $v_1^{(J)}$'s before completing the receiving of a whole $C_1$ (See Fig.4).

3.5 Security considerations

Remark 2 : The using of Reed-Solomon codes is very attractive, particularly 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(1)SOPKC over a small field such as $F_{2^{10}} \sim F_{2^{16}}$, 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 $\{\mu_i\}, \{r_i\}, A_r, P_J$ can be disclosed according to the following process:

Step 1 : Letting the estimated value of $\mu_i$ be $\hat{\mu}_i$, we calculate
$$\hat{\mu}_i(x) x^d \equiv \hat{r}_1 + \hat{r}_2 x + \cdots + \hat{r}_k x_k \mod G(x).$$

Step 2 : When $\hat{r}_i = u_j$ holds for all pairs of $(i, j); i, j = 1, 2, \cdots, g$, we conclude that $\hat{\mu}_i = \mu_i$.

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

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

$$P_c[\hat{\mu}_i] = \left( \frac{K'}{\eta} \right)^{-1} (2^m - 1)^{-\eta}.$$

(21)
In order to be secure against Attack 1, we recommend that $P_c[\hat{\mu}_i]$ be
\[ P_c[\hat{\mu}_i] \leq 2^{-80} = 8.27 \times 10^{-25}. \] (22)

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

Because $(ij)$’s satisfy
\[ 1 \leq (ij) \leq K', \] (23)
the probability that all the elements of $k_i$ can be successfully estimated, $P_c[\hat{k}_i]$, is
\[ P_c[\hat{k}_i] = \left( \frac{K'}{\eta} \right)^{-1}. \] (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 disclosing $\eta$ efs’s among the $g$ components of a ciphertext.**

The probability that the $\eta$ efs’s is correctly estimated, $P_c[\hat{\text{efs}}]$, is
\[ P_c[\hat{\text{efs}}] = \left( \frac{g-t}{\eta} \right) \left( \frac{g}{\eta} \right). \] (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, η = 96, t = 208.**
$P_c[\hat{\text{efs}}] = 1.18 \times 10^{-25} < 2^{-80}$

**Example III: m = 16, g = 1024, η = 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 recommend 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 $a_T$ added on $C$.**

**Step 1 :** Charles, an attacker, successfully estimates $\eta$ efs’s at the locations $L_1, L_2, \cdots, L_\eta$ of the ciphertext $C$.

**Step 2 :** Charles randomly selects $\eta$ $u_i$’s that constructs n-dimensional vector space $V_\eta$.

**Step 3 :** From $u_j \in V_\eta$, let $\rho_j$ be defined
\[ \rho_j = (\rho_{jL_1}, \rho_{jL_2}, \cdots, \rho_{jL_\eta}); j = 1, 2, \cdots, \eta, \]
where $\rho_{jL_i}$ is the symbol at the error-free location $L_i$ of the ciphertext $C$.

**Step 4 :** Charles constructs the following equation:
\[ w = x_1\rho_1 + x_2\rho_2 + \cdots + x_\eta\rho_\eta \]
\[ = (w_1, w_2, \cdots, w_\eta), \]
where $x_1, x_2, \cdots, x_\eta$ are unknown variables.
Step 5: From Eq. (27), Charles constructs the following linear simultaneous equations:

\[ x_1 \rho_1 L_1 + x_2 \rho_2 L_1 + \cdots + x_\eta \rho_\eta L_1 = w_1, \]
\[ x_1 \rho_1 L_2 + x_2 \rho_2 L_2 + \cdots + x_\eta \rho_\eta L_2 = w_2, \]
\[ \vdots \]
\[ x_1 \rho_1 L_\eta + x_2 \rho_2 L_\eta + \cdots + x_\eta \rho_\eta L_\eta = w_\eta. \]

Step 6: Charles obtains an unique solution:

\[ x_1 = e^{x_1}, x_2 = e^{x_2}, \cdots, x_\eta = e^{x_\eta}, \]

and then discloses \( a_{\text{IT}} \) as

\[ a_{\text{IT}} = C - (\tilde{x}_1 \rho_1 + \tilde{x}_2 \rho_2 + \cdots + \tilde{x}_\eta \rho_\eta). \]

We conclude that K(I)SOPKC is not secure against Attack 4, when once \( \eta \) 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, \( |C_i| \) and the coding rate, \( \rho \), are

\[ S_{PK} = mgk \text{(bits)}, \]
\[ |C_i| = gm \text{(bits)}, \]
\[ \rho = \frac{t}{g}. \]  

\[ \text{(26)} \]

<table>
<thead>
<tr>
<th>m</th>
<th>g</th>
<th>\eta</th>
<th>t</th>
<th>( P_{c}[\text{efs}] )</th>
<th>\rho</th>
<th>( S_{PK} ) (KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>512</td>
<td>128</td>
<td>192</td>
<td>( 3.46e-32 )</td>
<td>0.375</td>
<td>81.9</td>
</tr>
<tr>
<td>12</td>
<td>600</td>
<td>120</td>
<td>240</td>
<td>( 1.45e-31 )</td>
<td>0.40</td>
<td>115</td>
</tr>
<tr>
<td>14</td>
<td>800</td>
<td>160</td>
<td>320</td>
<td>( 7.42e-42 )</td>
<td>0.40</td>
<td>179</td>
</tr>
<tr>
<td>16</td>
<td>1000</td>
<td>200</td>
<td>400</td>
<td>( 3.79e-52 )</td>
<td>0.40</td>
<td>258</td>
</tr>
</tbody>
</table>

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(I)SOPKC.

### 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 \( a = (a_1, a_2, \cdots, a_t) \) be partitioned into:

\[ a \rightarrow (a_x; a_\theta), \]

where

\[ a_x = (a_1, a_2, \cdots, a_\pi), \]
\[ a_\theta = (a_{\pi+1}, a_{\pi+2}, \cdots, a_{\pi+\theta}), \]
\[ \pi + \theta = t. \]

S2: Let \( a_x(x) \) be

\[ a_x(x) = a_1 + a_2 x + \cdots + a_\pi x^{\pi-1}. \]

S3: Let \( a_{\pi+1}, \) a component of \( a_\theta \) be randomly located at \([j]\) by Bob, where we let

\[ [j] > \pi. \]

The probability \( P_c[\text{efs}] \) is

\[ P_c[\text{efs}] = \left( \frac{g - \pi - \theta}{\eta} \right)^{\frac{x}{\eta}}. \]  

\[ \text{(27)} \]
The coding rate, $\rho_p$, is
\[ \rho_p = \frac{\pi + \theta}{g}. \]  
(28)

Let us show several examples in Table 2.

<table>
<thead>
<tr>
<th>$m$</th>
<th>$g$</th>
<th>$\eta$</th>
<th>$\theta$</th>
<th>$\pi$</th>
<th>$P_t({(u_i)_g})$</th>
<th>$\rho$</th>
<th>$S_{PK}(KB)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>500</td>
<td>200</td>
<td>50</td>
<td>200</td>
<td>$3.2e-29$</td>
<td>0.500</td>
<td>80</td>
</tr>
<tr>
<td>11</td>
<td>1000</td>
<td>200</td>
<td>75</td>
<td>650</td>
<td>$1.9e-34$</td>
<td>0.725</td>
<td>176</td>
</tr>
<tr>
<td>12</td>
<td>2000</td>
<td>200</td>
<td>100</td>
<td>1600</td>
<td>$4.0e-38$</td>
<td>0.850</td>
<td>384</td>
</tr>
<tr>
<td>13</td>
<td>4000</td>
<td>200</td>
<td>100</td>
<td>3600</td>
<td>$4.0e-38$</td>
<td>0.925</td>
<td>832</td>
</tr>
<tr>
<td>14</td>
<td>8000</td>
<td>200</td>
<td>100</td>
<td>7600</td>
<td>$4.0e-38$</td>
<td>0.963</td>
<td>1792</td>
</tr>
<tr>
<td>15</td>
<td>16000</td>
<td>200</td>
<td>100</td>
<td>15600</td>
<td>$4.0e-38$</td>
<td>0.981</td>
<td>3840</td>
</tr>
<tr>
<td>16</td>
<td>32000</td>
<td>200</td>
<td>100</td>
<td>31600</td>
<td>$4.0e-38$</td>
<td>0.991</td>
<td>8192</td>
</tr>
</tbody>
</table>

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 $F_{2^n}$, the field extensively used for various storage and transmission systems -”, Cryptology ePrint Archive, Report, 2013/363 (2013-06).


