IPCC - Improved Perfect Code Cryptosystems*

Jieun Ryu[1], Yongbhin Kim[2], Seungtai Yoon[3], Ju-Sung Kang[4], and Yongjin Yeom[5]

[1] Kookmin University, Seoul, Republic of Korea
`ofryuji@kookmin.ac.kr`
[2] Kookmin University, Seoul, Republic of Korea
`coji67@kookmin.ac.kr`
[3] Cold Spring Harbor, NY 11724, United States
`yoon@cshl.edu`
[4] Kookmin University, Seoul, Republic of Korea
`jskang@kookmin.ac.kr`
[5] Kookmin University, Seoul, Republic of Korea
`salt@kookmin.ac.kr`

Abstract. IPCC is a graph-based public key cryptosystem based on Perfect Code Cryptosystem(PCC) proposed by Koblitz in 1993. The security of IPCC relies on the difficulty of finding the Perfect Dominating Set(PDS) in a given 3-regular graph. The existing PCC has not received much attention so far because of its low efficiency. In fact, encryption speed of PCC is slower than standard PKC such as RSA and the size of ciphertext is extremely large. However, we can improve the efficiency of PCC by combining several graphs during the encryption process. With multiple graphs, IPCC not only increases the level of but also reduce encryption times drastically. The unique properties of IPCC, the small public key and large ciphertext, make it suitable for one-way function or white box encodings that allow for large amouonts of memory. In this proposal, IPCC key generation and encryption/decryption algorithms are explained. In addition, design rationale, security, and performance analysis of IPCC are briefly discussed.

Keywords: Perfect dominating set · Perfect code cryptosytem · combinatorics · Public key cryptosystem.

1 Introduction

We propose a public key cryptosystem called IPCC(Improved Perfect Code Cryptosystem) that is secure against attacks using quantum computers. The security of IPCC is based on the hardness of finding the Perfect Dominating Set(PDS) in a 3-regular graph [2]. The problem of determining whether a graph has PDS is an NP-complete problem [4] and finding the PDS in the graph containing this is also believed to be an NP-complete problem but not proved yet. A

---

* This work is submitted to 'Korean Post-Quantum Cryptography Competition' (www.kpqc.or.kr).

cryptosystem based on this problem has first been described in 1993 by Koblitz and Fellows under the name Perfect Code Cryptosystems [1]. The public key of Perfect Code Cryptosystems is a 3-regular graph that hides the PDS, and the private key is a function called PDF that maps elements of the PDS to 1 and the remaining vertices to 0. This cryptography system could be used as post-quantum cryptography, since finding PDFs seems to be an NP-complete problem [5].

In this proposal, we introduce the design rationale of the improved cryptographic system first. Then, the security of the algorithm with proposing parameter set is analyzed against dedicated attacks. Also, implementation techniques and experimental results are discussed.

1.1 Design rationale

Perfect Code Cryptosystem has a problem in that the level of security is greatly decreased by a plaintext recovery attack among known attack techniques for cryptosystems. Adjusting the security parameters to solve this problem makes the efficiency of the cryptographic system worse. In the existing PCC, it is very difficult to achieve an appropriate balance between security and efficiency.

The design of IPCC is based on the additional difficulty of polynomial factorization to alleviate this. The key recovery attack depends on the size n of the graph, and the plaintext recovery attack depends on the maximum order k of the ciphertext polynomials in Perfect Code Cryptosystem.

Since the encryption speed of a cryptographic system is highly dependent on k, we came up with a way to build a high-order polynomial with polynomials of low degree. By multiplying and adding the low-order polynomials to generate a high-order polynomial, it is possible to encrypt message at high speed and increase the security against plaintext recovery attacks. Given a ciphertext, an adversary cannot attack each polynomial separately, unless the ciphertext polynomial is factored into low-order polynomials. Since polynomial factorization is a hard problem, we can enhance the security in this way. IPCC does not modify the key generation and decryption designs of the existing Perfect Code Cryptosystems, which were already efficient enough, but improves the encryption process so that PCC could be practical.

1.2 Advantages and limitations

The IPCC has the advantage of very fast key generation and decryption speed, though encryption speed is relatively slow. In addition, it has a distinctive property based on a new problem that is different from the encryption schemes that have been proposed in the post-quantum cryptography standardization process hosted by NIST. This encryption system has limitations in that the size of the memory used in the encryption process and the size of the ciphertext are extremely large. However, it is expected to be more useful than other PQCs for one-way functions or whitebox encryption in environmemts without memory limitations.

2 Preliminaries

This section introduces the basic concept, definitions and notation for describing out cryptosystem.

2.1 Definition

Definition 2.1 (k-regular graph). For some positive integer k, A graph $G = (V, E)$ such that $\deg(v) = k$ for $\forall v \in V$.

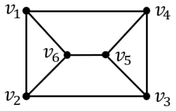
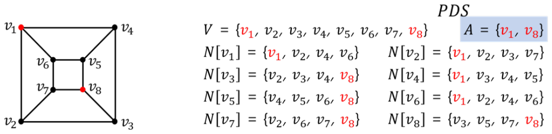


Fig. 1. example of 3-regular graph

Definition 2.2 (Perfect Dominating Set (PDS) [6]). A set of vertices $A \subseteq V$ is called a *perfect dominating set*(PDS) of G if for every vertex $v \in V$, $N[v]$ contains exactly one element of A.



$V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$

$A = \{v_1, v_8\}$

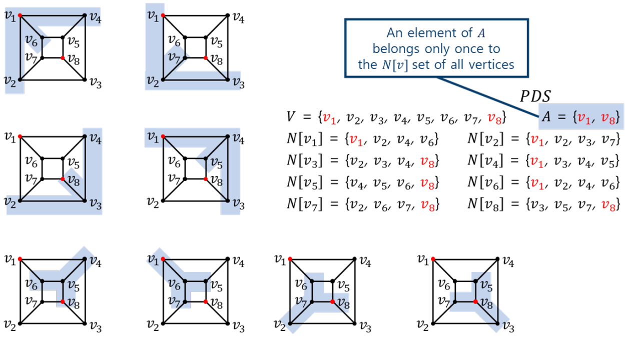
$N[v_1] = \{v_1, v_2, v_4, v_6\}$ $N[v_2] = \{v_1, v_2, v_3, v_7\}$

$N[v_3] = \{v_2, v_3, v_4, v_8\}$ $N[v_4] = \{v_1, v_3, v_4, v_5\}$

$N[v_5] = \{v_4, v_5, v_6, v_8\}$ $N[v_6] = \{v_1, v_2, v_4, v_6\}$

$N[v_7] = \{v_2, v_6, v_7, v_8\}$ $N[v_8] = \{v_3, v_5, v_7, v_8\}$

Fig. 2. example of Perfect Dominating Set


v_1
v_4
v_6
v_5
v_7
v_8
v_2
v_3
An element of A belongs only once to the $N[v]$ set of all vertices
PDS

$V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$

$A = \{v_1, v_8\}$

$N[v_1] = \{v_1, v_2, v_4, v_6\}$ $N[v_2] = \{v_1, v_2, v_3, v_7\}$

$N[v_3] = \{v_2, v_3, v_4, v_8\}$ $N[v_4] = \{v_1, v_3, v_4, v_5\}$

$N[v_5] = \{v_4, v_5, v_6, v_8\}$ $N[v_6] = \{v_1, v_2, v_4, v_6\}$

$N[v_7] = \{v_2, v_6, v_7, v_8\}$ $N[v_8] = \{v_3, v_5, v_7, v_8\}$

Fig. 3. explain for example of Perfect Dominating Set

Definition 2.3 (Perfect Dominating Function (PDF) [3]). Let $f : V \to \{0, 1\}$ be a function which assigns to each vertex of a graph G an element of the set $\{0, 1\}$. f is a perfect dominating function(PDF) if for every vertex $v \in V$,

$$\sum_{u \in N[v]} f(u) = 1.$$

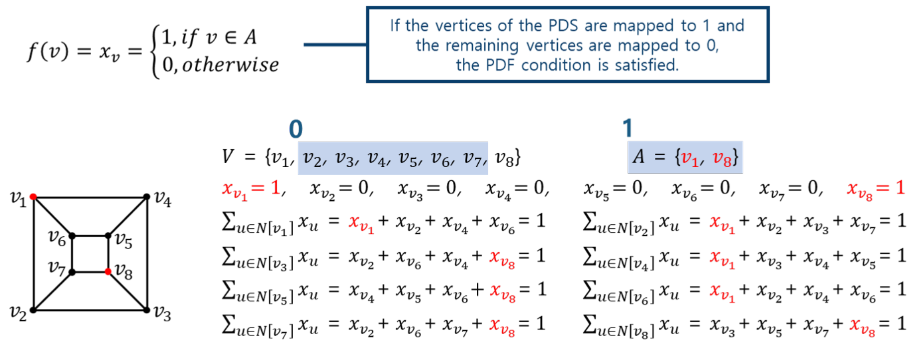

$f(v) = x_v = \begin{cases} 1, if\ v \in A \\ 0, otherwise \end{cases}$
If the vertices of the PDS are mapped to 1 and the remaining vertices are mapped to 0, the PDF condition is satisfied.
0
1
$V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$
$A = \{v_1, v_8\}$
$x_{v_1} = 1, \quad x_{v_2} = 0, \quad x_{v_3} = 0, \quad x_{v_4} = 0, \quad x_{v_5} = 0, \quad x_{v_6} = 0, \quad x_{v_7} = 0, \quad x_{v_8} = 1$
$\sum_{u \in N[v_1]} x_u = x_{v_1} + x_{v_2} + x_{v_4} + x_{v_6} = 1$
$\sum_{u \in N[v_2]} x_u = x_{v_1} + x_{v_2} + x_{v_3} + x_{v_7} = 1$
$\sum_{u \in N[v_3]} x_u = x_{v_2} + x_{v_6} + x_{v_4} + x_{v_8} = 1$
$\sum_{u \in N[v_4]} x_u = x_{v_1} + x_{v_3} + x_{v_4} + x_{v_5} = 1$
$\sum_{u \in N[v_5]} x_u = x_{v_4} + x_{v_5} + x_{v_6} + x_{v_8} = 1$
$\sum_{u \in N[v_6]} x_u = x_{v_1} + x_{v_2} + x_{v_4} + x_{v_6} = 1$
$\sum_{u \in N[v_7]} x_u = x_{v_2} + x_{v_6} + x_{v_7} + x_{v_8} = 1$
$\sum_{u \in N[v_8]} x_u = x_{v_3} + x_{v_5} + x_{v_7} + x_{v_8} = 1$
v_1
v_4
v_6
v_5
v_7
v_8
v_2
v_3

Fig. 4. example of Perfect Dominating Function

2.2 Algorithm of PDF cryptosystems

This subsection describes the algorithm of the existing PDF cryptosystem [5].

```
Algorithm 1 Key generation in PDF-cryptography
Input: The security parameters $(n_0, k)$ and modulus $p$
Output: A pair of keys $(pk, sk)$
Step 1: Divide a set $V$ into 4 subsets which are called $A, B, C$,and $D$ such
    that $|A| = |B| = |C| = |D| = n_0$
Step 2: Randomly create six one-to-one correspondences between the sets and
    connect related vertices each. Then the resulting graph $G = (V, E)$ with a
    PDS is generated
Step 3: WLOG set $A$ is used as $PDS$, and assign a value to each vertex $v \in V$
    from $PDF$ mapping $f : V \to \{0, 1\}$ by
        $f(v) = \begin{cases} 1, & \text{if } v \in A, \\ 0, & \text{otherwise.} \end{cases}$
Step 4: $pk \leftarrow$ graph $G = (V, E)$ and $sk \leftarrow PDF f$
Step 5: return $pk, sk$
```

Algorithm 2 Encryption in PDF-cryptography

```
Input: public key $pk(G=(V,E))$, message $m \in Z_p$, degree of polynomial $k$, and prime $p$
Output: ciphertext $ct$
Step 1: Choose an arbitrary subset $I$ of $\mathcal{P}(V)$.
        $I = \{S_1, S_2, ..., S_t\}$ for som positive integer $t$
        such that for all $S \in I, |S| \leq k$.
Step 2: Assign an integer $c_S$ to each set in $I$ such that
        $$\sum_{S \in I} c_S = m \pmod{p}.$$
    We denote this integer set $C$ by
        $C = \{c_{S_1}, c_{S_2}, ..., c_{S_t}\}$ such that $\sum_{S \in I} c_S = m \pmod{p}$.
Step 3: Form the following polynomial over $H = \mathbb{Z}/p\mathbb{Z}$
        $h(x_1, x_2, ..., x_n) = \sum_{S \in I} c_S \prod_{u \in S} \sum_{v \in N[u]} x_v \ \in H[\{x_v\}]$.
Step 4: Expand a polynomial $h$, replace all higher powers of a variable by its
    first power, and delete any monomial in which two variables occur that
    correspond to vertices whose distance from one to another in the graph $G$
    is $\leq 2$. Consequently, $ct$ is obtained as a simplified form of $h$.
Step 5: return $ct$
```

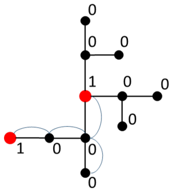


Fig. 5. example of reduction process in Step 4

Algorithm 3 Decryption in PDF-cryptography

```
Input: Private key $sk(PDF f)$, ciphertext $ct \in H[\{x_v\}]$, and prime $p$
Output: message $m$
Step 1: Evaluate the polynomial $h = ct$ using the private key $sk$
        $m \leftarrow h(f(x_1), f(x_2), ..., f(x_n))$.
Step 2: return $m$
```

3 Specification

This section introduces the notation, definition, and algorithm used to describe IPCC.

3.1 Notation

This subsection introduces the definitions of symbols.

Table 1: definitions of Symbols

symbol	definition
$\overset{\$}{\leftarrow}$	$a \overset{\$}{\leftarrow} A$: Select one of the elements of set A randomly (i.e., $a \in A$)
$\overset{\$}{\underset{i}{\leftarrow}}$	$A \overset{\$}{\underset{i}{\leftarrow}} B$: Select set A with size i among subsets of set B randomly (i.e., $A \subseteq B$)
$\overset{\$}{\leftrightarrow}$	$A \overset{\$}{\leftrightarrow} B$: Generate a random one-to-one correspondence between two sets A and B where A, B have equal size
$P^*(S)$	The power set (all subsets) of the set S excluding the empty set i.e., $P^*(S) = \{A : A \subseteq S\} \setminus \phi$
$N[v]$	For a vertex $v \in V$ of a graph $G = (V, E)$, a set having adjacent vertices of v and itself
$G = (V, E)$	A graph G consisting of a set of vertices V and a set of edges $E(\subseteq V \times V)$
n	Security parameter. The number of vertices in the graph, i.e., the size of the vertex set. All 3-regular graphs satisfy $n = 4 \times n_0$ for some integer n_0
k	Security parameter. Maximum order of ciphertext. i.e., degree of the ciphertext polynomial
p	Security parameter. Determine the range of message m to be encrypted, $m \in \{0, 1, 2, ..., p-1\}$.
x_v	Variable assigned to vertex $v(\in V)$
$PDF(G)$	A private key function that maps the PDS of the 3-regular graph G to 1 and the remaining vertices to 0
$Gen(n_1, n_2, ...)$	A key generation function that generates a public key graph $G_i = (V_i, E_i)$ with $\lvert V_i \rvert = n_i$ and a private key PDF corresponding to $G_1, G_2, ...$ for $i = 1, 2, ...$ $(PDF, G_1, G_2, ...) \leftarrow Gen(n_1, n_2, ...)$ and $PDS = PDS1 \vert\vert PDS2 \vert\vert ...$
$\mathcal{G}$	family of graphs that generated by function Gen

m	The message to encrypt. Based on the implementation of this proposal, a message is a fixed-length block of 32-bit.
$SubEnc(m, k, G)$	A function that encrypts a message m into a polynomial f_G^k of maximum degree k using the public key graph $G = (V, E)$ $$f_G^k \leftarrow SubEnc(m, k, G)$$
$F(f_{G_1}^{k_1}, f_{G_2}^{k_2}, ...)$	A function that combines polynomials $f_{G_1}^{k_1}, f_{G_2}^{k_2}, ...$ generated by the function $SubEnc$ to form polynomial F_G^k with maximum order k $$F_G^k \leftarrow \mathcal{F}(f_{G_1}^{k_1}, f_{G_2}^{k_2}, ...)$$
$\mathcal{F}^k$	family of multivariate mixing functions. A set of multivariate polynomials with a maximum degree k generated using an invatiant polynomial as a variable (indeterminate)
$Enc(m, k, G_1, G_2, ...)$	A function that encrypts message m into ciphertext polynomial ct with maximum degree k using public key graphs $G_1, G_2, ...$ $$ct \leftarrow Enc(m, k, G_1, G_2, ...) = F(f_{G_1}^{k_1}, f_{G_2}^{k_2}, ...)$$
$Dec(ct, PDS)$	A function that decrypts the ciphertext ct into message m using the private key PDS $$m \leftarrow Dec(ct, PDS)$$

3.2 Specification of IPCC

Here, we describe the details of IPCC algorithm. Key generation, encryption, and decryption algorithms are described in each subsection; encryption algorithms are divided into sub-encryption algorithms and a newly devised encryption algorithm that generates a ciphertext by using the sub-encryption algorithm.

The modified algorithm used in the implementation to speed up is described in Section 4.3.

Specification of key generation function

```
Algorithm 4 Key generation function Gen
Input: the number of vertices $n_1$, $n_2$, ... for each graph
```

Output: public key pk, private key sk

Step 1: For $i = 0, 1, 2, \ldots,$ allocate space for the vertex set V_i and the edge set E_i of the graph G_i

$$V_1 \leftarrow \{1, 2, \ldots, n_1\},\ E_1 \leftarrow \phi$$
$$V_2 \leftarrow \{1, 2, \ldots, n_2\},\ E_2 \leftarrow \phi$$
$$\cdots,$$

Step 2: Divide each vertex set V_i into 4 subsets which are called D_{i1}, D_{i2}, D_{i3} and D_{i4} such that $|D_{i1}| = |D_{i2}| = |D_{i3}| = |D_{i4}| = n_i/4$

$$D_{i1} \underset{n_i/4}{\overset{\$}{\leftarrow}} V_i$$
$$D_{i2} \underset{n_i/4}{\overset{\$}{\leftarrow}} V_i - D_{i1}$$
$$D_{i3} \underset{n_i/4}{\overset{\$}{\leftarrow}} V_i - D_{i1} - D_{i2}$$
$$D_{i4} \leftarrow V_i - D_{i1} - D_{i2} - D_{i3}$$

Step 3: Give a random one-to-one correspondence between subsets D_{i1}, D_{i2}, D_{i3} and D_{i4}, and connect vertices according to this relationship to create edges

$$E \leftarrow \{D_{i1} \overset{\$}{\leftrightarrow} D_{i2}\} \cup E,\ E \leftarrow \{D_{i1} \overset{\$}{\leftrightarrow} D_{i3}\} \cup E,$$
$$E \leftarrow \{D_{i1} \overset{\$}{\leftrightarrow} D_{i4}\} \cup E,\ E \leftarrow \{D_{i2} \overset{\$}{\leftrightarrow} D_{i3}\} \cup E,$$
$$E \leftarrow \{D_{i2} \overset{\$}{\leftrightarrow} D_{i4}\} \cup E,\ E \leftarrow \{D_{i3} \overset{\$}{\leftrightarrow} D_{i4}\} \cup E,$$

Step 4: Create a PDS set by randomly choosing one of the subsets D_{i1}, D_{i2}, D_{i3} or D_{i4}

```
for $i = 1, 2, \ldots$ do
    $j \overset{\$}{\leftarrow} \{1, 2, 3, 4\}$
    $PDS_i \leftarrow D_{ij}$
    $PDS \leftarrow PDS_i \cup PDS$
end for
i.e., $PDS = PDS_1 \parallel PDS_2 \parallel \cdots$
```

Step 5: Define each pair (V_i, E_i) as graph G_i

$$G_1 \leftarrow (V_1, E_1),\ G_2 \leftarrow (V_2, E_2), \cdots$$

Step 6: Generate private key function PDF using PDS

$$\forall v \in V, PDF(v) = x_v = \begin{cases} 1, & \text{if } v \in PDS, \\ 0, & \text{otherwise.} \end{cases}$$

Step 7: Use graph $G_1 = (V_1, E_1), G_2 = (V_2, E_2), \cdots$ as public key and PDF as private key

$$pk \leftarrow \mathcal{G} = \{G_1, G_2, \ldots\}$$

```
        $sk \leftarrow PDF$
Step 8: return $pk, sk$
```

Specification of encryption function

```
**Algorithm 5** Sub-encryption function SubEnc
**Input:** message $m$, maximum degree $k$, graph $G = (V, E)$
**Output:** polynomial $f_G^k$
Step 1: Randomly choose one of the subsets of $P^*(V_i)$ to satisfy the following condition
        $I \overset{\$}{\leftarrow} P^*(V_i)$ such that $\forall S \in I, |S| \le k$
Step 2: Assign an integer $c_i$ to each set in $I$ as follows. At this time, the sum of the $c_i$ should be the message $m$
        for $i = 1$ to $|I| - 1$ do
            $c_i \overset{\$}{\leftarrow} Z_p$
        end for
        $c_{|I|} = m - \sum_{i=1}^{|I|-1} c_i (\mod p)$
Step 3: Form the following polynomial
        $f(X_{v1}, X_{v2}, ...) = \sum_{i=1}^{|I|-1} c_i \prod_{u \in I_i} \sum_{u \in N_{[u]}} x_u$
Step 4: Expand polynomial $f$, then replace all higher powers of a variable by its first power and delete a term consisting of a product of variables corresponding to a vertices with a distance equal to or less than 2
Step 5: $f$ is an invariant polynomial in which each variable in the polynomial corresponds to the vertex of $G$ and the highest degree is $k$, denoted as $f_G^k$
Step 6: return $f_G^k$
```

```
**Algorithm 6** Encryption function $Enc$
**Input:** message $m$, maximum degree $k$, public key graph set $\mathcal{G} = \{G_1, G_2, \cdots\}$
**Output:** ciphertext $ct$
Step 1: Select $F$ to combine polynomials such that the maximum order is $k$
        $F \overset{\$}{\leftarrow} \mathcal{F}^k$
Step 2: For $F$, distribute message $m$ to satisfy the following condition
        $m = F(m_1, m_2, ...)$
Step 3: For each $m_i$, create a polynomial $f_{G_i}^{k_i}$ with maximum degree $k_i$ using $G_i$
        i.e., $f_{G_i}^{k_i} \leftarrow SubEnc(m_i, k_i, G_i)$
```

```
Step 4: By combining the polynomial $f_{G_i}^{k_i}$ according to the form of $F$,
    generate ciphertext $F_G^k(x_{v_1}, x_{v_2}, ...)$ with maximum degree $k$ and the
    vertices in the ciphertext belong to $G_1$, $G_2$ or others
        $ct \leftarrow F_G^k(x_{v_1}, x_{v_2}, ...) = F(f_{G_1}^{k_1}, f_{G_2}^{k_2}, ...)$
Step 5: return $ct$
```

Specification of decryption function

```
Algorithm 7 Decryption function $Dec$
Input: ciphertext $ct$, private function $PDF$
Output: message $m$
Step 1: Put $PDF(v_i)$ into variable $x_{v_i}$ of the polynomial ciphertext
        $ct = F_G^k(x_{v_1, x_{v_2}, ...})$
        $m \leftarrow ct(PDF(v_1), PDF(v_2), \cdots)$
Step 2: return $m$
```

example of IPCC

We show an example of IPCC where $n_1 = 200, n_2 = 200, k = 2, p = 11$. The sets of vertices are $V_1 = \{v_1, v_2, ..., v_8\}$ and $V_2 = \{v_9, v_{10}, ..., v_{20}\}$.

☐ Key Generation

First, receiver Bob generates two graphs, G_1 and G_2, in which the sizes of vertex set are $n_1$and n_2, respectively, as shown in Figure 6. The process of generating graph G_1 is shown in Figure 6, 7.

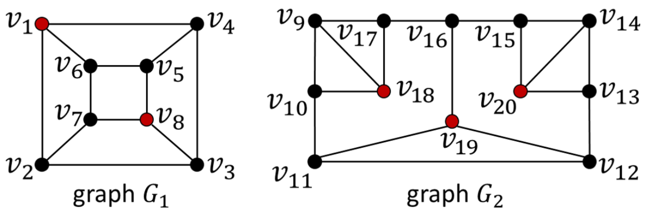


Fig. 6. generate key pair (pk, sk)

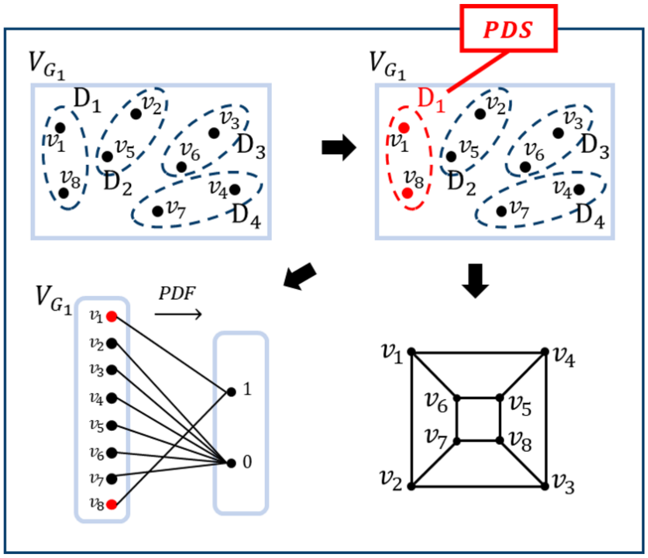


Fig. 7. public key graph *pk*

Then generates PDF to use as a secret key by combining each PDS of graphs G_1 and G_2 as following.

$$PDF(v) = x_v = \begin{cases} 1, & \text{if } v \in PDS = \{v_1, v_8, v_{18}, v_{19}, v_{20}\}, \\ 0, & \text{otherwise.} \end{cases}$$

Bob saves the PDF and sends the public key graphs G_1 and G_2 as shown in Figure 8, to the sender.

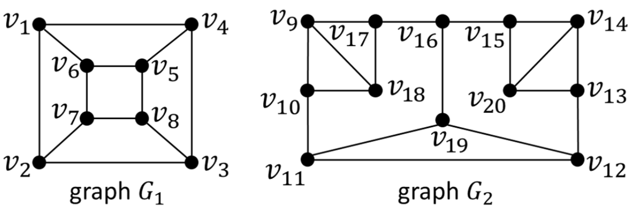


Fig. 8. public key graph *pk*

☐ Encryption

After receiving the public key, the sender Alice prepares a message m to be encrypted. When a message is prepared in the pre-shared message space $\mathbb{Z}_p$, F from $\mathcal{F}$ is randomly selected for encryption.

$$\begin{gathered} m = 1(\mod 11) \\ F = f^1_{G_1} f^1_{G_2} + f^2_{G_1} + f^2_{G_2} \leftarrow \mathcal{F} \end{gathered}$$

Next, she distributes the message according to the form of F.

$$m = m_1 m_2 + m_3 + m_4 = 5 \cdot 6 + 7 + 8 = 1 \mod 11$$

Then, with each m_i as input, subEnc is performed according to f_i.

- $f^1_{G_1} = SubEnc(m_1 = 5, k = 1, G_1)$
  $|I| = 1$, $I = \{v_2\} \overset{\$}{\underset{1}{\leftarrow}} P^*(G_1)$, $c_{\{v_2\}} = 5 \mod 11$
  $f^1_{G_1} = c_{\{v_2\}} \sum_{v \in N[v_2]} x_v = 5(x_{v_1} + x_{v_2} + x_{v_3} + x_{v_7})$

- $f^1_{G_2} = SubEnc(m_2 = 6, k = 1, G_2)$
  $|I| = 1$, $I = \{v_{11}\} \overset{\$}{\underset{1}{\leftarrow}} P^*(G_2)$, $c_{\{v_{11}\}} = 6 \mod 11$
  $f^1_{G_2} = c_{\{v_{11}\}} \sum_{v \in N[v_{11}]} x_v = 6(x_{v_{10}} + x_{v_{11}} + x_{v_12} + x_{v_{19}})$

- $f^2_{G_1} = SubEnc(m_3 = 7, k = 2, G_1)$
  $|I| = 1$, $I = \{v_4, v_5\} \overset{\$}{\underset{1}{\leftarrow}} P^*(G_1)$, $c_{\{v_4, v_5\}} = 7 \mod 11$

$$\begin{aligned} f^2_{G_1} &= c_{\{v_4,v_5\}} \prod_{u \in I} \sum_{v \in N[u]} x_v \\ &= 7(x_{v_1} + x_{v_3} + x_{v_4} + x_{v_5})(x_{v_4} + x_{v_5} + x_{v_6} + x_{v_8}) \\ &= 7(x_{v_1}x_{v_4} + x_{v_1}x_{v_5} + x_{v_1}x_{v_6} + x_{v_1}x_{v_8} + x_{v_3}x_{v_4} + x_{v_3}x_{v_5} + x_{v_3}x_{v_6} + x_{v_3}x_{v_8} \\ &\quad + x_{v_4}x_{v_4} + x_{v_4}x_{v_5} + x_{v_4}x_{v_6} + x_{v_4}x_{v_8} + x_{v_5}x_{v_4} + x_{v_5}x_{v_5} + x_{v_5}x_{v_6} + x_{v_5}x_{v_8}) \\ &= 7(x_{v_1}x_{v_8} + x_{v_3}x_{v_6} + x_{v_4} + x_{v_5}) \end{aligned}$$

- $f^2_{G_2} = SubEnc(m_4 = 8, k = 2, G_2)$
  $|I| = 1$, $I = \{v_{16}, v_{20}\} \overset{\$}{\underset{1}{\leftarrow}} P^*(G_2)$, $c_{\{v_{16}, v_{20}\}} = 8 \mod 11$

$$\begin{aligned} f^2_{G_2} &= c_{\{v_{16},v_{20}\}} \prod_{u \in I} \sum_{v \in N[u]} x_v \\ &= 8(x_{v_{15}} + x_{v_{16}} + x_{v_{17}} + x_{v_{19}})(x_{v_{13}} + x_{v_{14}} + x_{v_{15}} + x_{v_{20}}) \\ &= 8(x_{v_{15}}x_{v_{13}} + x_{v_{15}}x_{v_{14}} + x_{v_{15}}x_{v_{15}} + x_{v_{15}}x_{v_{20}} + x_{v_{16}}x_{v_{13}} + x_{v_{16}}x_{v_{14}} + x_{v_{16}}x_{v_{15}} \\ &\quad + x_{v_{16}}x_{v_{20}} + x_{v_{17}}x_{v_{13}} + x_{v_{17}}x_{v_{14}} + x_{v_{17}}x_{v_{15}} + x_{v_{17}}x_{v_{20}} + x_{v_{19}}x_{v_{13}} \\ &\quad + x_{v_{19}}x_{v_{14}} + x_{v_{19}}x_{v_{15}} + x_{v_{19}}x_{v_{20}}) \\ &= 8(x_{v_{15}} + x_{v_{16}}x_{v_{13}} + x_{v_{17}}x_{v_{13}} + x_{v_{17}}x_{v_{14}} + x_{v_{17}}x_{v_{20}} + x_{v_{19}}x_{v_{13}} + x_{v_{19}}x_{v_{14}} + x_{v_{19}}x_{v_{20}}) \end{aligned}$$

Finally, the ciphertext ct is generated by combining the polynomial f_i according to F.

$$\begin{aligned} ct = F &= f^1_{G_1} f^1_{G_2} + f^2_{G_1} + f^2_{G_2} \\ &= 5(x_{v_1} + x_{v_2} + x_{v_3} + x_{v_7})6(x_{v_{10}} + x_{v_{11}} + x_{v_1 2} + x_{v_{19}}) \\ &\quad + 7(x_{v_1}x_{v_8} + x_{v_3}x_{v_6} + x_{v_4} + x_{v_5}) + 8(x_{v_{15}} + x_{v_{16}}x_{v_{13}} + x_{v_{17}}x_{v_{13}} \\ &\quad + x_{v_{17}}x_{v_{14}} + x_{v_{17}}x_{v_{20}} + x_{v_{19}}x_{v_{13}} + x_{v_{19}}x_{v_{14}} + x_{v_{19}}x_{v_{20}}). \end{aligned}$$

Alice sends the generated ciphertext ct to Bob.

☐ Decryption

Bob obtains the message by substituting the PDF into the ciphertext.

$$\begin{aligned} m &= ct(PDF) \\ &= 0+0+0+8+0+0+0+0+0+0+0+0+0+0+0+0+7 \\ &\quad +0+0+0+0+0+0+0+0+0+0+8 \\ &= 1 \mod 11. \end{aligned}$$

3.3 Parameter sets

The parameter set selected to satisfy 128-bit safety is as follows. The security strength that is satisfied when the suggested parameters are used has 158-bit security for key recovery attack and 162-bit security for plaintext recovery attack. However, the parameters used in the version implemented so far are as shown in [Table 3], and we plan to optimize them so that the proposed parameters can be used in the future.

Table 2. The parameter set satisfied 128-bit security

p	#graph	n per graph	k per subpolynomial	$\lvert I\rvert$ per graph
65521 ($2^{16} - 15$)	2	200	4	5

Table 3. The parameter set satisfied 80-bit security

p	#graph	n per graph	k of subpolynomial	$\lvert I\rvert$ per graph
65521 ($2^{16} - 15$)	2	200	2 / 3	3

p : public parameters. A prime number defining the message space $\mathbf{Z}_p$; not only the message, but also an arbitrary coefficient c_{I_i} is defined in the message space.

n : public parameters. The size of the set of vertices V in graph G. The vertex set size of the 3-regular graph is divisible by 4, so $n = 4 * n_0$ for any integer n_0. And the public key is a set of edges of the graph; the number of the edge set of the 3-regular graph is $6 * n_0$, and each edge has two vertices. According to the above parameter set, the size of the public key is $2 * 2 * 6 * n_0 = 1200$ when using two graphs with $n_i = 200$.

k : public parameters. Each k_i can be adjusted such that the sum of the maximum degrees of the subpolynomials generated with the maximum degree of the ciphertext is k. For example, if two graphs are used when $k = 4$, the degree of the subpolynomial generated by each graph can be selected from among $(1, 3), (2, 2)$, and $(3, 1)$. In the case of the above parameters, the maximum degree of the ciphertext is 8 since the maximum degree of the subpolynomials generated by each graph is 4.

$|I|$: secret parameters. The size of the $P^*(V)$ subset I selected to generate the subpolynomial. As the size of I increases and the number of polynomial terms increases, the reduction occurence is also increases, then the coefficient c_i corresponding to $I_i (\in I)$ is spread evenly and ciphertext factoring becomes more difficult. On the other hand, when $|I|$ increases, the encryption speed becomes very slow and the size of the ciphertext tends to increase rapidly. Therefore, it is necessary to select parameters considering efficiency. $|I|$ suggested above is also a value selected considering the ratio of the reducted term and the encryption speed.

The value of each parameter were determined by analyzing the results of the graph below.

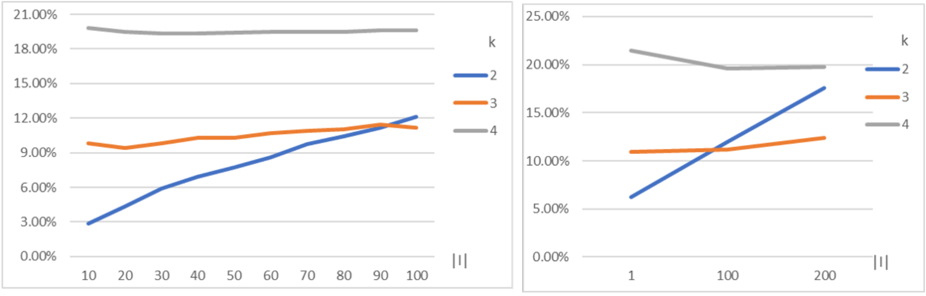


Fig. 9. Reduction rate

[Figure 9] shows the ratio of terms reduced according to $|I|$ and the maximum degree of a polynomial made with a graph where n is 200. The larger the reduction ratio, the lower the probability of knowing the variables configured for encryption, and the higher the probability of spreading the coefficients containing the information of the message. In the case of $k = 4$, the rate of increase of the reduction ratio is slower than the rate of increase of the number of terms, and thus the form is shown in [Figure]. Nevertheless, the parameter set was decided the maximum degree to 4 because the reduction ratio is higher than when $k = 2$ and it generates terms of various degrees. And also, in future research, we plan to expand the parameter set in the direction that $|I|$ increases when $k = 2$.

4 Performance analysis

It is enhanced by some improvements of preceding implementations of Perfect Code Cryptosystems encryption schemes[3]. The currently implemented IPCC is focused on increasing the operation speed. The main difference is that the dynamic array implementation is converted to a static array type for fast operation. And in some schemes, a table using lexical ordering is used for vertex operation.
Section 4.3 describes the algorithm that applies Table to the algorithm of Section 3.2. Although the operation time was dramatically reduced through this process, the size of the memory used during the operation increased significantly.

4.1 Description of platform

All benchmarks were obtained on one core of an Intel Core i7-9700K CPU @3.60HZ processor. The benchmarking PC has 16GB of RAM, and compiled with gcc version 8.3.0. All results reported are the median of the results of 10 000 executions of the respective measurements. It has not been currently optimized for memory usage allocated during the measurement and will be optimized for the stack memory in the future.

4.2 Performance of reference implementation

* 1round performance goals (PC environment, 80-bit security)

Table 4. Performance goals (PC environment, 80-bit security)

key size	cipher size	keygen time	enc time	dec time
$pk = 600$-byte, $sk = 150$-byte	$\leq$ 1MB	$\leq$ 0.1ms	$\leq$ 500ms	$\leq$ 1ms

* Actual performance (PC environment, satisfied 80-bit security)

Table 5. Performance result (PC environment, 80-bit security)

key size	cipher size	keygen time	enc time	dec time
$pk = 4800$-byte, $sk = 400$-byte	9.2×10^4-byte	1.06ms	0.35ms	0.33ms

IPCC generates 4800 bytes of public key, 400 bytes of private key, and 9.2×10^4 bytes of ciphertext when using the parameter that provides 109-bit security in Section 3.3 [Table 3]. At this time, the average key generation speed is 1.06ms, the average encryption speed is 0.35ms, and the average decryption speed is 0.33ms.
Comparing the performance of IPCC with other cryptographic algorithms is shown in the [Figure 10].

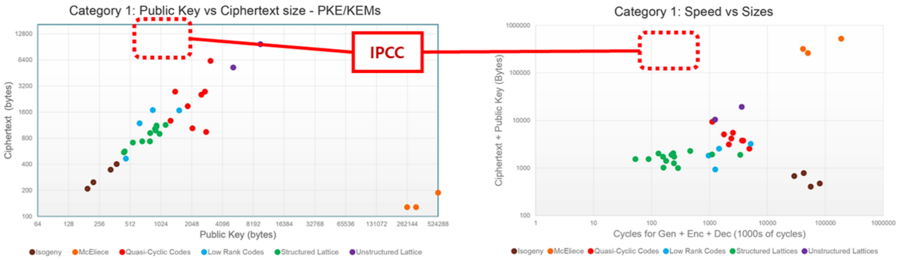


Fig. 10. Comparison with NIST PQC 3round candidate algorithms [7]

The results of comparing the performance of IPCC with the performance of the existing Perfect Code Cryptosystems are shown in the following [Table]. This is the result of measuring the speed of the cryptographic system by inputting $|I|$ to generate a ciphertext of a similar size when the number of graph vertices is 200 and the maximum degree of the ciphertext is 5 [5].

Table 6. Performance comparing (PC environment, 80-bit security)

	key size (byte)	number of terms	keygen time	enc time	dec time
PDF	$pk = 4800,\ sk = 400$	24,352	1ms	84781ms	4ms
IPCC	$pk = 4800,\ sk = 400$	25,555	0.028ms	2.411ms	3.363ms

4.3 Implemetation technic for performance

This algorithm utilizes a table using lexical ordering (so that we can calculate by substituting the vertex number to a bit digit) on the sub-encryption and decryption functions to improve implementation speed.

```
Algorithm 8 Sub-encryption function SubEnc with table
Input: message $m$, max degree $k$, graph $G = (V, E)$
Output: polynomial $f_G^k$
```

```
Step 1:  Same as Alogrithm2 upto step 3
Step 2:  For each vertex in graph G, create a table by defining the list of
   neighboring vertices as follows
        for $i = 1$ to $|V|$ do
            for $i = v_j$ to $N[v_i]$ do
                $Table(v_i) \leftarrow 2^{v_j - 2}$
            end for
        end for
Step 3:  After expanding the invariant polynomial $f(x_{v_1}, x_{v_2}, ...)$, set empty
   term list $T$ represented by lexical ordering and calculate each term as
   follows
        for $i = 1$ to $|\sharp terms(f)|$ do
            for $i = 1$ to $|term_i|$ do
                for $i = 1$ to $|j - 1|$ do
                    if $Table(v_{ij})$ & $Table(v_{jk}) > 0$ then
                        remove the $term_i$
                        goto next $i$
                    end if
                end for
                $T_i \leftarrow T_i | 2^{v_{ij} - 1}$
            end for
        end for
Step 4:  Convert the value of undeleted $T_i$ into polynomial form as follows
        for $i = 1$ to $|T_i|$ do
            for $i = 1$ to tablecolsize do
                if $\{T_i \gg (j - 1)\} \& 1 = 1$ then
                    $term_i \leftarrow term_i * v_j$
                end if
            end for
            $term_i \leftarrow term_i * coefficient(term_i)$
        end for
Step 5:  $f$ is an invariant polynomial $f_G^k$ in which the variable in the
   polynomial corresponds to the vertex of $G$ and has the highest degree $k$
        $f_G^k \leftarrow f = \sum term_i$
Step 6:  return $f_G^k$
```

The if statement of step 5 of [Algorithm 5] is a process of checking whether two or mode vertices that are neighboring are multiplied. If the i does not go to the next$(i + 1)$ by the if statement, vertex multiplication is performed through OR operation. This ensures that the order does not increase when the same vertex is multiplied. Furthermore, step 4 can be pre-computed after receiving the public key before proceeding with encryption.

Algorithm 9 Decryption function Dec with table

```
Input: cipher $ct$, private function PDF
Output: message $m$
Step 1: For each variable $x_{v_i}$ of the $PDS$ element, it is converted as follows
        for $i = 1$ to $|PDS|$ do
            $y_{v_i} \leftarrow 2^{v_i - 1}$
        end for
Step 2: Generate $LoPDS$ by XORing all $y_{v_i}$
        for $i = 1$ to $|PDS|$ do
            $LoPDS \leftarrow y_{v_{i1}} \bigoplus y_{v_{i2}} \bigoplus ...$
        end for
Step 3: Convert all terms $T_j$ of the ciphertext as follows
        for $j = 1$ to $|\sharp terms(ct)|$ do
            for $k = 1$ to $dim(T_j)$ do
                $y_{v_{jk}} \leftarrow 2^{v_{jk} - 1}$
            end for
            $T_j \leftarrow y_{v_{j1}} \bigoplus y_{v_{j2}} \bigoplus ...$
        end for
Step 4: If the result of AND operation with $T_j$ and $LoPDS$ is $T_j$, the
    coefficient of the $T_j$ term is added to $pt$; otherwise, check the next term
        for $j = 1$ to $|\sharp terms(ct)|$ do
            if $T_j$ & $LoPDS = T_j$ then
                $pt \leftarrow pt + coefficient(T_j)$
            end if
        end for
Step 5: Return message $m$, which is the decryption result $pt$
        $m \leftarrow pt$
Step 6: return $m$
```

Step 4 of [Algorithm 6] is a procedure of comparing the two values of the term and the PDS converted by lexical ordering at once, rather than comparing the vertices corresponding to each variable of the term with the PDS elements one by one.

5 Security

This section describes the security strength according to the complexity of exhaustive key search on IPCC, then explains the known attack technique and the security decreasing. And the rationale for determining the security parameters is also explained.

5.1 Security definition

IPCC is the PKE algorithm having configured the parameter set to satisfy 80-bit security. The main parameters that determine the security strength of this cryptosystem are the size n of each graph and the maximum order k of the cryptogram. In addition, the size of set I(the number of c_i) used to generate the ciphertext affects the attack on the cryptosystem.

5.2 Security strength categories

The complexity of key recovery attack on IPCC is $O\binom{|V_1|}{|V_1|/4} + O\binom{|V_2|}{|V_2|/4} + ... = O\binom{|V_i|}{|V_i|/4}$ $(|V_i| \geq |V_j| \; for\, \forall j)$ where $\mathcal{G} = \{G_1, G_2, ...\}$. For IPCC using multiple graphs, the complexity of key recovery attack for the largest graph is the complexity of exhaustive key search because each graphs are public and key recovery attacks can be performed on each. So, when the size of the set of graph vertices is 200, the complexity of key recovery attack is $O(2^{158})$. [Table 7] shows the security level presented by NIST [8]. The strength of the cryptographic algorithm according to the size of the security parameters proposed in this proposal corresponds to security level 1.

Table 7. strength of the cryptographic algorithm

Level	Security Description
1	At least as hard to break as AES128 (exhaustive key search)
2	At least as hard to break as SHA256 (collision search)
3	At least as hard to break as AES192 (exhaustive key search)
4	At least as hard to break as SHA384 (collision search)
5	At least as hard to break as AES256 (exhaustive key search)

5.3 Cost of plaintext recovery attacks

The goal of this attack is to recover the plaintext corresponding to the ciphertext. In a situation where the (I_S, c_S) pair is unknown, the attacker generates $\widetilde{ct}$ including terms in all cases and applies Gaussian-Jordan elimination to $ct = \widetilde{ct}$

to obtain $m = \widetilde{m}$. The attack complexity is $O(t^3)$ when the number of terms in the arbitrary ciphertext generated by the attacker is $t = \sum_{i=1}^{k} \binom{n}{k}$. That is, along with the size of the public key, the maximum degree k of the ciphertext has a great influence on complexity.

algorithm of plaintext recovery attacks

Algorithm 10 Plaintext recovery attack

```
Input: ciphertext $ct$, graph $G = (V, E)$
Output: message $m$
Step 1: Set an adversary's set $\widetilde{I}$ as following
            $\widetilde{I} \subseteq P(V)$ such that for all $S \in \widetilde{I}, |S| \leq k$ and $|\widetilde{I}| = \sum_{j=1}^{k} \binom{n}{k}$
Step 2: The adversary leaves the set $C$ selected by the sender as an unknown
    set $\widetilde{C}$
            $\widetilde{C} = \left\{ \widetilde{c_1}, \widetilde{c_2}, ..., \widetilde{c}_{|\widetilde{I}|} \right\}$
Step 3: Generate a polynomial $\widetilde{ct}$ in the same way as the Algorithm2
            $\widetilde{ct} = \widetilde{f}(x_1, x_2, ..., x_n) = \sum_{i=1}^{|\widetilde{I}|-1} \widetilde{c_i} \prod_{u \in \widetilde{I_i}} \sum_{u \in N[u]} x_u$
Step 4: From the condition $\widetilde{ct}$ equals to the given ciphertext $ct$ as a
    polynomial, we have the system of linear equations with unknowns $\{\widetilde{c_s}\}$ by
    comparing their coefficients
Step 5: Applying Gaussian Jordan Elimination to the linear equations $\widetilde{ct} = ct$;
    if the result of adding each column vector of the reduced row echelon form
    matrix on $Z_p$ is $\left\{ r_1, r_2, ..., r_{|\widetilde{I}|}, r_{|\widetilde{I}|+1} \right\}$, then $\left\{ r_1, r_2, ..., r_{|\widetilde{I}|}, r_{|\widetilde{I}|+1} \right\} =$
    $\{1, 1, ..., 1, \widetilde{m}\}$
Step 6: Return $\widetilde{m}$ as $m$
Step 7: Return $m$
```

example of plaintext recovery attacks We show an example of plaintext recovery attack for Perfect Code Cryptosystem where $n = 8, k = 1, p = 11$. The sets of vertices are $V = v_1, v_2, ..., v_8$. The receiver Bob first generates a key pair and publishes the graph [Figure 11] with the public key.

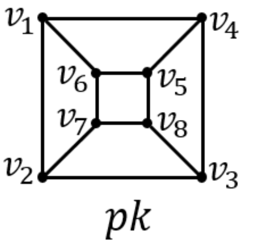


Fig. 11. public key graph *pk*

The sender Alice generated the ciphertext ct to deliver the message $m = 5$. At this time, the I selected by Alice is $I = \{v_1, v_6, v_7\}$, and coefficients set is $C = \{c_{v_1} = 7, c_{v_6} = 3, c_{v_7} = 6\}$.
Assume that Alice is sending a ciphertext to the recipient Bob, and the attacker eavesdrops on it. The ciphertext that the attacker obtained is $ct = 10x_{v_1} + 2x_{v_2} + 7x_{v_4} + 3x_{v_5} + 5x_{v_6} + 9x_{v_7} + 6x_{v_8}$.
First, the attacker constructs a set $\widetilde{I}$ so that all possible cases where $k = 1$.

$$\widetilde{I} = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$$

And let the integer corresponding to each element of $\widetilde{I}$ be the unknown $\widetilde{c}_i$.

$$\widetilde{C} = \{\widetilde{c}_{v_1}, \widetilde{c}_{v_2}, \widetilde{c}_{v_3}, \widetilde{c}_{v_4}, \widetilde{c}_{v_5}, \widetilde{c}_{v_6}, \widetilde{c}_{v_7}, \widetilde{c}_{v_8}\}$$

Then, an attacker generates an arbitrary ciphertext ct for attacky referring to the public key.

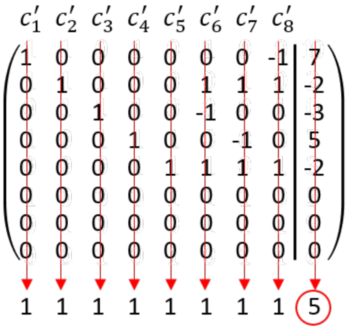
$$\begin{aligned}
\widetilde{ct} = {} & \widetilde{ct}_{v_1}(x_{v_1} + x_{v_2} + x_{v_4} + x_{v_6}) + \widetilde{ct}_{v_2}(x_{v_1} + x_{v_2} + x_{v_3} + x_{v_7}) \\
& + \widetilde{ct}_{v_3}(x_{v_2} + x_{v_3} + x_{v_4} + x_{v_8}) + \widetilde{ct}_{v_4}(x_{v_1} + x_{v_3} + x_{v_4} + x_{v_5}) \\
& + \widetilde{ct}_{v_5}(x_{v_4} + x_{v_5} + x_{v_6} + x_{v_8}) + \widetilde{ct}_{v_6}(x_{v_1} + x_{v_5} + x_{v_6} + x_{v_7}) \\
& + \widetilde{ct}_{v_7}(x_{v_2} + x_{v_6} + x_{v_7} + x_{v_8}) + \widetilde{ct}_{v_8}(x_{v_3} + x_{v_6} + x_{v_7} + x_{v_8}) \\
= {} & x_{v_1}(\widetilde{ct}_{v_1} + \widetilde{ct}_{v_2} + \widetilde{ct}_{v_4} + \widetilde{ct}_{v_6}) + x_{v_2}(\widetilde{ct}_{v_1} + \widetilde{ct}_{v_2} + \widetilde{ct}_{v_3} + \widetilde{ct}_{v_7}) \\
& + x_{v_3}(\widetilde{ct}_{v_2} + \widetilde{ct}_{v_3} + \widetilde{ct}_{v_4} + \widetilde{ct}_{v_8}) + x_{v_4}(\widetilde{ct}_{v_1} + \widetilde{ct}_{v_3} + \widetilde{ct}_{v_4} + \widetilde{ct}_{v_5}) \\
& + x_{v_5}(\widetilde{ct}_{v_4} + \widetilde{ct}_{v_5} + \widetilde{ct}_{v_6} + \widetilde{ct}_{v_8}) + x_{v_6}(\widetilde{ct}_{v_1} + \widetilde{ct}_{v_5} + \widetilde{ct}_{v_6} + \widetilde{ct}_{v_7}) \\
& + x_{v_7}(\widetilde{ct}_{v_2} + \widetilde{ct}_{v_6} + \widetilde{ct}_{v_7} + \widetilde{ct}_{v_8}) + x_{v_8}(\widetilde{ct}_{v_3} + \widetilde{ct}_{v_6} + \widetilde{ct}_{v_7} + \widetilde{ct}_{v_8})
\end{aligned}$$

Knowing $ct = 10x_{v_1} + 2x_{v_2} + 7x_{v_4} + 3x_{v_5} + 5x_{v_6} + 9x_{v_7} + 6x_{v_8}$, an attacker can generate a system of equations by comparing the coefficients of each term. As shown in [Figure 12] below, Gauss-Jordan elimination is applied by expressing the system of $\widetilde{ct} = ct$ equations as a matrix.

$$\begin{array}{cccccccc|c} c'_1 & c'_2 & c'_3 & c'_4 & c'_5 & c'_6 & c'_7 & c'_8 & \\ \hline 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 10 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 2 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 7 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 3 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 5 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 9 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 6 \end{array} \Rightarrow \begin{array}{cccccccc|c} c'_1 & c'_2 & c'_3 & c'_4 & c'_5 & c'_6 & c'_7 & c'_8 & \\ \hline 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 7 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & -2 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & -3 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 5 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & -2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

Fig. 12. apply the Gauss-Jordan elimination

The message m can be recovered by adding the column vector of the matrix calculated in the Reduced Row Echelon Form over $\mathbb{Z}_p$. An attacker cannot compute the unique $\widetilde{C}$, but can obtain the message.



$$\begin{array}{cccccccc|c} c'_1 & c'_2 & c'_3 & c'_4 & c'_5 & c'_6 & c'_7 & c'_8 & \\ \hline 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 7 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & -2 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & -3 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 5 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & -2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 5 \end{array}$$

Fig. 13. Recover the message on $\mathbf{Z}_p$

Algorithm 11 Plaintext recovery attack against IPCC

Input: ciphertext ct, private function PDF

Output: message m

```
Step 1: Check whether ct can be decomposed into $\widetilde{F}$
Step 2: Execute Algorithm7 for each $\widetilde{f_i}$ of $\widetilde{F}$
Step 3: Compute $\widetilde{m}$ as m by composing $\widetilde{m}_j$ calculated for each $\widetilde{f_j}$ into $\widetilde{F}$
Step 4: return $m$
```

Expansion of plaintext recovery attack to IPCC In the case of IPCC, it is not necessary for the encrypting user to randomly select$F$ and not to disclose it. Therefore, if each subpolynomial cannot be attacked by factoring the ciphertext, the security against plaintext recovery attacks is maintained.
For example, suppose that Alice uses the graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ with $|V_1| = |V_2| = 200$ to generate a ciphertext with a maximum degree of 2. The Fchosen by Alice to make the ciphertext is $f^2_{G_1} + f^1_{G_1} f^1_{G_2} + f^2_{G_2}$, and $|I| = 5$for each polynomial. Then, the time required for encryption is 0.186 ms, and the cipher text consists of 531 terms. If Alice uses a graph with $|V| = 400$ to encrypt a ciphertext with a maximum degree of 2 and $|I| = 35(= 5{+}5{*}5{+}5)$, a ciphertext consisting of 345 terms is obtained by consuming 4.3463ms of encryption time.
Attacker Eve can scan the ciphertext and separate the terms generated using only one graph, G_1 or G_2, and both graphs. Then, for ciphertext generation, the number of selectable vertex sets is ${}_{200}C_1 +_{200} C_2$ from G_1, ${}_{200}C_1 +_{200} C_2$ from G_2, and ${}_{200}C_1 *_{200} C_1$ when two graphs are used. However, it is the same as the number of possible vertex combination when plaintext recovery is performed where the maximum degree is 2 and graph size is 400. In both cases, the number of terms in the random ciphertext that Eve needs to generate is 80200.
As the order increases, the difference in encryption speed increases when dividing the two graphs, and the difference between the size of the generated ciphertext and the number of random ciphertexts that an attacker must generate also increases sharply. Therefore, with respect to the parameters proposed in Section 3.3, it can be expected that the IPCC will be safe against plaintext recovery attacks. Above all, it is important not to affect the complexity of attack when the attacker separates even a part of F. The goal is to generate F so that the attacker's factoring ability has no effect on the attack complexity.
Since the current implementation uses a selecting one of four list in F as a secret, there is a possibility of being attacked with a computational ability similar to that of a user who encrypts by operating each F directly, but in future implementations, we plan to generate F be configured randomly.

Table 8. complexity according to k (n=400)

k	size of $\widetilde{ct}$[6]	complexity
2	8.02×10^4	49
3	1.06×10^7	71
4	1.06×10^9	90
5	8.42×10^{10}	109
6	5.56×10^{12}	127
7	3.13×10^{14}	145
8	1.54×10^{16}	162

If the proposed parameter set is followed, the number of vertices that can appear in the ciphertext is 400 and the maximum degree is 8, so it has 162-bit safety against plaintext recovery attacks. However, in this parameter setting, follows the key recovery attack safety because the key recovery attack for each graph has 158-bit security, unlike generally following the security against plaintext recovery attack since the complexity of a key recovery attack is greater than that of a plaintext recovery attack.

6 Conclusion

In public key cryptography, Perfect Code Cryptosystems has not received much attention due to its inefficiency in terms of speed and memory. However, the IPCC proposes a way to improve encryption/decryption speed using multiple graphs. Designing a cryptosystem based on combinatorics is not a mature technology. We expect that our proposal will be an initial step toward practical cryptosystems based on an NP-problem in graph theory. In this paper, we try to determine appropriate parameters such as the size of each graph, the number of graphs, and the number of terms in ciphertext polynomial, the degree of polynomials and so on. However, there is still room for improvement. In future studies, we have to provide precise security analysis and optimal implementation.

References

1. Fellows, M., Koblitz, N.: Kid krypto. pp. 371–389 (2021(1993))
2. Fellows, M.R., Koblitz, N.: Combinatorially based cryptography for children (and adults). Congressus Numerantium pp. 9–9 (1994)
3. Haynes, T.W., Hedetniemi, S., Slater, P.: Fundamentals of domination in graphs (2013)
4. Kratochvíl, J.: Regular codes in regular graphs are difficult. Discrete Mathematics **133**(1), 191–205 (1994)
5. Kwon, S., Kang, J.S., Yeom, Y.: Analysis of public-key cryptography using a 3-regular graph with a perfect dominating set. pp. 1–6 (2021)
6. Livingston, M., Stout, Q.: Perfect dominating sets. Congressus Numerantium **79** (1997)
7. Moody, D.: The 2nd Round of the NIST PQC Standardization Process (2019)
8. NIST: Submission requirements and evaluation criteria for the post-quantum cryptography standardization process (2016)
9. Yoon, S.: (1,-1, 0)-perfect minus dominating function and its application to the public key cryptosystem (2001)

[9]