IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 18 December 2023, accepted 28 December 2023, date of publication 4 January 2024,
date of current version 11 January 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3349704

THEORY

IPCC7: Post-Quantum Encryption Scheme Based on a Perfect Dominating Set in 3-Regular Graph

JIEUN RYU[1], YONGBHIN KIM[2], SEUNGTAI YOON[3], JU-SUNG KANG[1,2], AND YONGJIN YEOM[1,2]

[1]Department of Financial Information Security, Kookmin University, Seoul 02707, Republic of Korea
[2]Department of Information Security Cryptography Mathematics, Kookmin University, Seoul 02707, Republic of Korea
[3]Cold Spring Harbor Laboratory, Laurel Hollow, NY 11724, USA

Corresponding author: Jieun Ryu (ofryuji@kookmin.ac.kr)

This work was supported by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government through Ministry of Science and Information & Communications Technology (MSIT) under Grant 2021M1A2A2043893.

ABSTRACT Post-quantum cryptography (PQC) has been actively explored to meet the requirements arising with the rapid development of quantum computers. The National Institute of Standards and Technology (NIST) conducted a competition to establish the next-generation cryptographic standards. While previous competitions selected a single cryptographic standard, this competition aimed to standardize several algorithms based on various mathematical problems since the security of PQC has not been studied as extensively as that of legacy cryptosystems. The recent exclusion of the isogeny-based key-establishment algorithm, SIKE, from the competition emphasizes the necessity of exploring cryptographic algorithms based on various fundamental problems. In this study, we propose the Improved Perfect Code Cryptosystem 7 (IPCC7), a new post-quantum encryption scheme, as an improved version of the perfect code cryptosystem (PCC) based on combinatorics conceptualized by Koblitz. The security of our cryptosystem relies on the intractability of finding the perfect dominating set in a given graph. A PCC proposed previously by Koblitz did not receive much attention because of its low efficiency for handling higher-order polynomials. To overcome these drawbacks, we used the product of low-degree polynomials and demonstrated the feasibility of a graph-based encryption scheme. IPCC7 has some limitations for use as a general-purpose PQC. However, considering its relatively small key size (768 bytes public-key and 64 bytes secret key), fast decryption speed (2.0 Gbps), and usable encryption speed (8.6Mbps), IPCC7 is particularly suitable for environments with low-memory constraints, such as white-box encryptions.

INDEX TERMS Graph-based encryption, perfect code cryptosystems, perfect dominating function, perfect dominating set, post-quantum cryptography.

I. INTRODUCTION

Public-key cryptosystems are indispensable to communication services, such as online payments, key agreements, and message encryptions. However, the rapid development of quantum computers in recent years has threatened the security of the existing public-key systems [1], [2], [3]. Consequently, post-quantum cryptography (PQC) has been actively studied owing to its quantum resistance. Recently, post-quantum cryptographic algorithms based on various mathematical problems were refined and optimized [4].

The National Institute of Standards and Technology (NIST) in the USA has been running a program for PQC standardization since 2016 in response to the development of quantum computers. In addition, numerous institutes and companies have conducted research on post-quantum technology. According to the NIST, PQC must satisfy IND-CCA2 security against attackers and provide sets of

The associate editor coordinating the review of this manuscript and approving it for publication was Sedat Akleylek.

parameters for various security strengths, such as the 128, 192, and 256 bits security level options of the standard block cipher AES [5]. Currently, the NIST has selected four algorithms, including CRYSTALS-KYBER, as the standard through competition, and yet other four algorithms are still in contention as candidate algorithms [6]. Notably, the candidate algorithms use code-based problems that are different from the selected algorithms, which are based on lattice and hash problems [7].

Unlike the traditional public-key cryptography, which has been proven and analyzed from the perspective of underlying problems for decades, PQC algorithms are too recent to determine whether they are secure enough. Therefore, the decision of the NIST decision to standardize multiple PQCs that use different underlying problems is intended to prepare for future unexpected attacks [8]. In fact, SIKE, an isogeny-based key-establishment algorithm, was one of the final candidates up to the third round of competition before it succumbed to mathematical attacks and was eliminated [9]. SIKE was vulnerable probably because it has undergone a shorter research period than the other third-round candidates. Considering the possibility that PQC algorithms will be compromised later, long-term and continuous R&D on cryptographic algorithms based on various fundamental problems is necessary.

The cryptosystem investigated in this study is a public-key cryptosystem based on a novel fundamental problem, namely, a combinatorial theory that has never been considered in NIST's competitions thus far. It can be used as a PQC based on the conjecture that finding a particular subset within a graph is an NP-hard problem [10], [11], [12], [13], [14].

The perfect code cryptosystem (PCC) proposed by Koblitz in 1992 is a public-key algorithm based on the intuitive concept of constructing a ciphertext polynomial using variables of graph vertices and constants mapped to vertices [15], [16]. An encrypted message is a polynomial generated using a graph as a public-key, and the ciphertext can be decrypted by evaluating the polynomial using a special subset of vertices as a corresponding private-key. The security of this cryptosystem depends on the randomness of the ciphertext polynomial. That is, it is difficult to distinguish the ciphertext from a randomly selected polynomial. However, to make it difficult to distinguish, the ciphertext to be extremely large and the encryption speed to be low. To overcome these difficulties and use the cryptosystems, several researchers have studied these systems, including their underlying problems [17], [18], [19], [20], [21], [23], [24], [25], [26], [27], [28]. The simplest form of the polynomial that this cryptosystem can generate allows for relatively rapid generation of the ciphertext [15], [29]. However, its highly constrained and simplified form renders it vulnerable to simple attacks [30].

In this paper, we propose a post-quantum encryption scheme named Improved PCC 7 (IPCC7) as an improved PCC. To satisfy the 128-bit security level, we used a graph with 256 vertices as a public-key and generated a ciphertext as a sparse polynomial of degree 7. Thus, the size of the public-key is 768 bytes representing a 3-regular graph. The private-key, i.e., the subset information of the graph, occupies much less memory than its corresponding public-key. Since a ciphertext is a sparse but high-degree polynomial, only nonzero terms are stored in a table. Thus, the size of a ciphertext is not constant and it occupies a large amount of memory, about 280 kB on average. Encryption is relatively slow because of the heavy polynomial operations in the process of generating ciphertext. In contrast, the decryption process merely requires a simple evaluation of the polynomial. Our experiments showed that the decryption time was less than 1 ms, and the scheme can be implemented faster through parallalization.

Fig. 1 compares the key and ciphertext sizes of the algorithms included in the NIST competition. In the figure, most candidates are located on the diagonal, implying that in general, the size of ciphetext is proportional to that of the key. One exception is the McEliece cryptosystem, which is based on the traditional coding theory and uses a large generating matrix of Goppa code as a public-key. The advantage of the McEliece scheme is high-speed encryption (less than 1 ms) and decryption (about 20 ms) once a huge public-key of hundreds of kilobytes is shared. Our IPCC7 lies far away from the McEliece system in Fig. 1. The advantages of IPCC7 are small key size and fast decryption, while its disadvantages are large ciphertext and slow encryption.

The unique properties of IPCC7 make it suitable for cryptographic applications that allow for large memory, such as white-box implementations. Although ciphertext of IPCC requires a lot of memory, in environments without memory limitations, the size of the ciphertext is not an issue. White-box cryptography uses memory ranging from a few megabytes to several gigabytes to protect the encryption key from attacks such as memory dumps and code lifting [31], [32]. In this environments without memory limitations, the large ciphertexts of IPCC7 are expected to be especially efficient when applied to a trapdoor one-way function in white-box encryption for external encoding to strengthen the one-wayness [33]. IPCC is a primitive suitable for providing such one-way encoding with a quantum-safe security level.

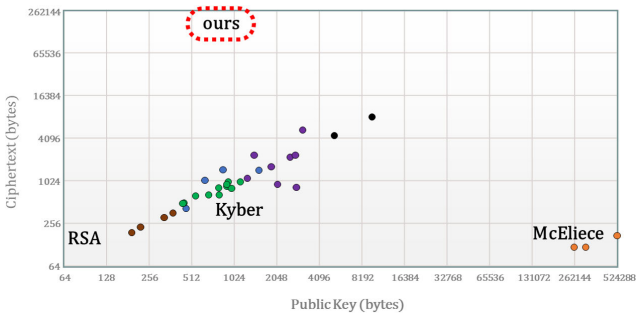

FIGURE 1. Algorithms included in the NIST competition: the size of the public-key and ciphertext [34].

A. OUR CONTRIBUTION

Based on the concept of Koblitz's PCC, we propose a new post-quantum encryption scheme. Through quantitative security analysis, we redesigned the construction algorithm for invariant polynomials by combining low-degree polynomials and determined the parameters for the algorithm with 128-bit security, including the graph size and the degree of the polynomial. Further, our implementation demonstrated the practical potential of IPCC7 as an alternative to post-quantum algorithms.

II. PRELIMINARY

This section introduces 3-regular graph and their properties. Then we discuss previous studies on cryptosystems based on the graph theory. The abbreviations used in our paper are summarized in Appendix A.

A. MATHEMATICAL BACKGROUND

Let $G = (V, E)$ be a graph with a set of vertices V and a set of edges $E(\subseteq V \times V)$.

Definition 1 (Closed Neighborhood): For a vertex $v \in V$ of graph $G = (V, E)$, the *closed neighborhood* $N[v]$ of v is a set of vertices consisting of v and its adjacent vertices. The closed neighborhood of vertex v is expressed as follows:

$$N[v] = \{u \in V \mid uv \in E\} \cup \{v\}.$$

Example 1: Fig. 2 shows a graph with vertices $V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$. In this graph, all the closed neighborhoods are as follows:

$$\begin{aligned} &N[v_1] = \{v_1, v_2, v_5\}, \quad N[v_2] = \{v_1, v_2, v_3, v_6, v_7\},\\ &N[v_3] = \{v_2, v_3, v_4\}, \quad N[v_4] = \{v_3, v_4, v_5\},\\ &N[v_5] = \{v_1, v_4, v_5, v_6\}, \quad N[v_6] = \{v_2, v_5, v_6, v_7\},\\ &N[v_7] = \{v_2, v_6, v_7\}. \end{aligned}$$

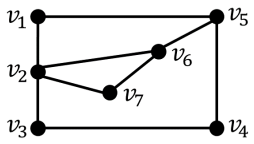

FIGURE 2. Graph with seven vertices.

Definition 2 (r-Regular Graph): For some positive integer r, a simple graph $G = (V, E)$ is called an *r-regular graph* if it satisfies the following condition.

$$\forall v \in V,\ |N(v)| = r + 1.$$

Example 2: Fig. 3 shows a graph with six vertices $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$. For each vertex $v \in V$, the closed neighborhood $N[v]$ has exactly four vertices as follows:

$$\begin{aligned} &N[v_1] = \{v_1, v_2, v_4, v_6\}, \quad N[v_2] = \{v_1, v_2, v_3, v_6\},\\ &N[v_3] = \{v_2, v_3, v_4, v_5\}, \quad N[v_4] = \{v_1, v_3, v_4, v_5\},\\ &N[v_5] = \{v_3, v_4, v_5, v_6\}, \quad N[v_6] = \{v_1, v_2, v_5, v_6\}. \end{aligned}$$

Therefore, the graph is called a 3-regular graph.

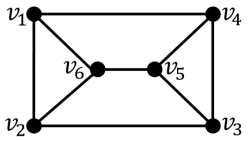

FIGURE 3. 3-Regular graph with six vertices.

Fact 1: The number of edges in an r-regular graph consisting of n vertices is $nr/2$.

Definition 3 (Perfect Dominating Set, PDS): For a given graph $G = (V, E)$, a subset $D \subseteq V$ is called a *perfect dominating set* (*PDS*) of G if $N[v]$ contains exactly one element of D for each vertex $v \in V$ (i.e., each vertex has exactly one neighbor in D). We say $D \in PDS(G)$ if

$$|N[v] \cap D| = 1 \text{ for all } v \in V,$$

where PDS(G) is the collection of PDS in graph G.

Example 3: Fig. 4 shows a graph with eight vertices $V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$. In this graph, the closed neighborhood of each vertex $v \in V$ is defined as

$$\begin{aligned} &N[v_1] = \{v_1, v_2, v_4, v_6\}, \quad N[v_2] = \{v_1, v_2, v_3, v_7\},\\ &N[v_3] = \{v_2, v_3, v_4, v_8\}, \quad N[v_4] = \{v_1, v_3, v_4, v_5\},\\ &N[v_5] = \{v_4, v_5, v_6, v_8\}, \quad N[v_6] = \{v_1, v_5, v_6, v_7\},\\ &N[v_7] = \{v_2, v_6, v_7, v_8\}, \quad N[v_8] = \{v_3, v_5, v_7, v_8\}. \end{aligned}$$

Let $D = \{v_1, v_8\}$. Then, for each $v \in V$, the intersection of $N[v]$ and D has exactly one element as

$$\begin{aligned} &N[v_1] \cap D = \{v_1\}, \quad N[v_2] \cap D = \{v_1\},\\ &N[v_3] \cap D = \{v_8\}, \quad N[v_4] \cap D = \{v_1\},\\ &N[v_5] \cap D = \{v_8\}, \quad N[v_6] \cap D = \{v_1\},\\ &N[v_7] \cap D = \{v_8\}, \quad N[v_8] \cap D = \{v_8\}. \end{aligned}$$

Therefore, D is a PDS of graph G.

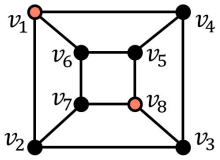

FIGURE 4. 3-Regular graph with $\{v_1, v_8\} \in V$ as a perfect dominating set (PDS).

A graph may or may not have a PDS; if it does, there may be one or more PDSes. Notably, the graph in Fig. 3 cannot have a PDS.

Fact 2: If an r-regular graph G has a PDS at least, then the number of PDSes is $r + 1$. If the number of vertices $|G|$ is n, then the size of each PDS is $\frac{n}{r+1}$.

Proposition 1: The distance between distinct vertices belonging to a PDS is always at least three.

Example 4: Fig. 5 shows two different parts of a 3-regular graph. We assume that the vertices colored in red belong to

the same PDS. In the graph on the left, if we check the closed neighborhood set for all vertices, there is always only one red vertex in this set. This implies that the set consisting of red vertices does not violate the definition of PDS.

However, in the graph on the right, two red vertices appear in the closed neighborhood set of the central black vertex. This implies that the red vertices cannot form PDS. Similarly, two vertices that are at a distance less than or equal to two cannot be elements of the same PDS.

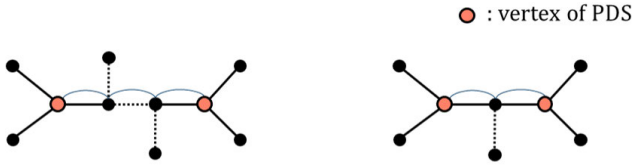

FIGURE 5. Relation between the vertex distance and PDS.

Definition 4 (Perfect Dominating Function, PDF): Given a graph $G = (V, E)$, a function $\chi : V \to \{0, 1\}$ is called a *perfect dominating function (PDF)* if it satisfies the following condition.

$$\sum_{u \in N[v]} \chi(u) = 1 \quad \text{for each } v \in V.$$

In this case, we write $\chi \in \mathrm{PDF}(G)$ where $\mathrm{PDF}(G)$ is the set of PDFs in a graph G.

Proposition 2: Let x_v denote the variable assigned to vertex $v \in V$. For a 3-regular graph $G = (V, E)$ with PDS, χ_D is a PDF if it maps the vertices within D to 1 and all the remaining vertices to 0, where D is a PDS.

Example 5: Consider the graph in Fig. 4 containing PDSes. Let x_{v_i} be a variable for $v_i \in V$, and $u \in N[v_i]$ for each v_i. Furthermore, let the function χ map vertices in $\{v_1, v_8\} \in PDS(G)$ to 1 and all other vertices to 0.

For each vertex v_i, define $e_i = \sum_{u \in N[v_i]} x_u$ which is written explicitly as follows:

$$\begin{aligned} e_1 &= x_{v_1}+x_{v_2}+x_{v_4}+x_{v_6}, & e_2 &= x_{v_1}+x_{v_2}+x_{v_3}+x_{v_7},\\ e_3 &= x_{v_2}+x_{v_3}+x_{v_4}+x_{v_8}, & e_4 &= x_{v_1}+x_{v_3}+x_{v_4}+x_{v_5},\\ e_5 &= x_{v_4}+x_{v_5}+x_{v_6}+x_{v_8}, & e_6 &= x_{v_1}+x_{v_5}+x_{v_6}+x_{v_7},\\ e_7 &= x_{v_2}+x_{v_6}+x_{v_7}+x_{v_8}, & e_8 &= x_{v_3}+x_{v_5}+x_{v_7}+x_{v_8}. \end{aligned}$$

Choose a PDS and consider its corresponding PDF. Then we observe that each e_i has exactly one nonzero term. If we choose $\{v_1, v_8\}$ as a PDS, then x_{v_1} and x_{v_8} take the value 1 and remaining terms are zero. Hence, each e_i equals one. Formally, we write

$$e_i(\chi(v_1), \cdots, \chi(v_8)) = 1 \quad \text{for any } \chi \in \mathrm{PDF}(G). \tag{1}$$

Note that this happens regardless of the choice of PDF χ. We define a vector valued function $\boldsymbol{\chi} : V^n \to \{0, 1\}^n$ by

$$\boldsymbol{\chi}(v_1, \ldots, v_n) = (\chi(v_1), \ldots, \chi(v_n)).$$

Then equation (1) is simplified as $e_i \circ \boldsymbol{\chi} = 1$ which means e_i is transformed into a constant function under any PDF χ.

Example 5 leads us to define an invariant polynomial.

Definition 5 (Invariant Polynomial): Let $G = (V, E)$ be a graph with a PDS whose vertices are ordered as $v_1, \ldots, v_n$. For a nonnegative integer p (not necessarily a prime), consider a polynomial ring $\mathbb{Z}_p[x_{v_1}, x_{v_2}, \ldots, x_{v_n}]$. As a monomial, x_{v_i} is interpreted as the i-th coordinate function:

$$x_{v_i}(a_1, a_2, \ldots, a_n) \mapsto a_i.$$

A polynomial $f \in \mathbb{Z}_p[x_{v_1}, \ldots, x_{v_n}]$ is called an *invariant polynomial* if

$$f \circ \boldsymbol{\chi} = c \quad \text{for any } \chi \in \mathrm{PDF}(G),$$

where $c \in \mathbb{Z}_p$ is a constant independent of χ.

Note that $(f \circ \boldsymbol{\chi})(v_1, \ldots, v_n)$ is explicitly evaluated as

$$f(x_{v_1}(\chi(v_1), \cdots, \chi(v_n)), \cdots, x_{v_n}(\chi(v_1), \cdots, \chi(v_n))).$$

We easily to see that e_1 in Example 5 is an invariant polynomial. For the PDF χ_D corresponding to PDS $D = \{v_1, v_8\}$, $e_1 \circ \boldsymbol{\chi}_D = 1$. For,

$$\begin{aligned} &(e_1 \circ \boldsymbol{\chi}_D)(v_1, \ldots, v_8)\\ &\quad = x_{v_1}(\boldsymbol{\chi}_D(v_1, \ldots, v_8)) + \cdots + x_{v_6}(\boldsymbol{\chi}_D(v_1, \ldots, v_8))\\ &\quad = x_{v_1}(\chi_D(v_1), \ldots, \chi_D(v_8))\\ &\qquad + \cdots + x_{v_6}(\chi_D(v_1), \ldots, \chi_D(v_8))\\ &\quad = \chi_D(v_1) + \chi_D(v_2) + \chi_D(v_4) + \chi_D(v_6)\\ &\quad = 1 + 0 + 0 + 0 = 1. \end{aligned}$$

In Example 5, $e_1, \ldots, e_8$ are invariant polynomials of degree 1. Our definition of the invariant polynomial follows that of Koblitz, leading to the following proposition [15].

Proposition 3: Given a graph $G = (V, E)$ with PDS, choose a vertex v. Then for $i \in 1, 2, \cdots, |N[v]|$, if g_i is an arbitrary invariant polynomial of degree $k-1$, and all $g_i + c_i$ are evaluated to have the same value for the adjusted constant c_i, then the following is an invariant polynomial of degree k:

$$\sum_{u \in N[v]} (g_u + c_u) x_u.$$

Proposition 4: Given a graph $G = (V, E)$ with PDS, consider an invariant polynomial f over G, where x_v, x_u are the variables for vertices $v, u \in V$. If these variables appear in the same term of f, then the following hold:

(a) $x_v x_u = x_v$ if $v = u$, and
(b) $x_v x_u = 0$ if $v \neq u$ and the distance between the two vertices is ≤ 2.

We can directly establish these from Proposition 1. Throughout this paper, such a transformation process of polynomials according to Proposition 4 is called *"reduction."* The following example illustrates the process of generating an invariant polynomial with reduction.

Example 6: Consider a 3-regular graph G with $D = \{v_1, v_8\} \in \mathrm{PDS}(G)$ as shown in Fig. 4. We can construct an invariant polynomial of degree 2 for this graph as follows.

Select a vertex $v_i \in V$ and let $N[v_i] = \{u_{i,1}, u_{i,2}, u_{i,3}, u_{i,4}\}$ be a closed neighborhood.

Suppose f_j is an arbitrary invariant polynomial of degree 1, such as $e_1, \ldots, e_8$ in Example 5. Adjust constants c_j so that $(f_j \circ \boldsymbol{\chi}_D) + c_j$ has the same value for $j = 1, 2, 3, 4$. We can then generate an invariant polynomial f of degree 2 as

$$f = \sum_{j=1}^{4} (f_j + c_j) x_{u_{i,j}}.$$

If we choose a vertex v_1 and four invariant polynomials,

$$f_1 = 4e_2, \quad f_2 = e_1 + 2e_5, \; f_3 = 5e_8, \; f_4 = 3e_6,$$

then, we can adjust $c_1 = -1$, $c_2 = 0$, $c_3 = -2$, and $c_4 = 0$ so that

$$f_j + c_j = 3 \quad \text{for } j = 1, 2, 3, 4.$$

Now, f is an invariant polynomial of degree 2 with a value 3. For,

$$\begin{aligned} f &= \sum_{j=1}^{4} (f_j + c_j) x_{u_{1,j}} \\ &= (4e_2 - 1)x_{v_1} + (e_1 + 2e_5)x_{v_2} + (5e_8 - 2)x_{v_4} + (3e_6)x_{v_6} \\ &= \left\{4(x_{v_1} + \not{x}_{v_2} + \not{x}_{v_3} + \not{x}_{v_7}) - 1\right\} x_{v_1} \\ &\quad + \left\{(\not{x}_{v_1} + x_{v_2} + \not{x}_{v_4} + \not{x}_{v_6}) + 2(\not{x}_{v_4} + x_{v_5} + \not{x}_{v_6} + \not{x}_{v_8})\right\} x_{v_2} \\ &\quad + \left\{5(\not{x}_{v_3} + \not{x}_{v_5} + x_{v_7} + \not{x}_{v_8}) - 2\right\} x_{v_4} \\ &\quad + 3(\not{x}_{v_1} + \not{x}_{v_5} + x_{v_6} + \not{x}_{v_7}) x_{v_6} \\ &= 3x_{v_1} + x_{v_2} - 2x_{v_4} + 3x_{v_6} + 2x_{v_2}x_{v_5} + 5x_{v_4}x_{v_7}. \end{aligned}$$

Similarly, another example of an invariant polynomial g of degree 2 can be obtained by choosing the vertex v_5 as

$$g = 5x_{v_4} + 3x_{v_6} - x_{v_8} + 4x_{v_1}x_{v_8} + 3x_{v_2}x_{v_5} - 2x_{v_4}x_{v_7}.$$

It is easy to check that for any $\chi \in \mathrm{PDF}(G)$, $g \circ \boldsymbol{\chi} = 3$.

B. PCC

Determining whether a given graph $G = (V, E)$ has a PDS is an NP-hard problem. Furthermore, it has been conjectured that finding any PDS is also NP-hard, provided that a given graph is known to have PDSes [20]. These properties can be used to construct a public-key cryptosystem, where a graph with PDS serves as the public-key, thus ensuring the one-wayness of the cryptosystem. Moreover, one of the PDSes is used as a private-key to establish a trapdoor for the cryptosystem.

As explained in Definition 6, the PCC scheme consists of three procedures: key generation, encryption, and decryption. Using invariant polynomials generated from a graph G with PDS, one can produce the ciphertext polynomial of a message m. It can be decrypted by evaluating the ciphertext polynomial using a PDF as the corresponding private-key.

Definition 6 (PCC Scheme): Let m be a message to be encrypted.

(a) **[Key generation]** Construct a graph G that has a perfect dominating set $D \in \mathrm{PDS}(G)$. Then the public-key is the graph G itself, and the private-key is its PDS D.

(b) **[Encryption]** Construct an invariant polynomial g such that

$$g \circ \boldsymbol{\chi} = m \quad \text{for any } \chi \in \mathrm{PDF}(G).$$

Note that the polynomial g as the ciphertext of m can be constructed without knowing D or χ.

(c) **[Decryption]** Using the private-key D, define a perfect dominating function $\chi_D \in \mathrm{PDF}(G)$ by $\chi_D(v) = 1$ for $v \in D$ and $\chi_D(v) = 0$ elsewhere. Then one can evaluate $g \circ \boldsymbol{\chi}_D$ to obtain the decrypted message.

Because of the property of invariant polynomials, the evaluation of the ciphertext polynomial during decryption always yields the same value as the original message.

The overall flow of secure communication using PCC cryptosystem is shown in Fig. 6. Firstly, receiver Bob generates graph G with n vertices having PDSes and selects one of the PDSes, say D, as the private-key. Then, Bob shares the graph G with sender Alice. After obtaining a public-key, Alice encrypts the message m in $\mathbb{Z}_p$ as follows: the encryption algorithm generates an invariant polynomial $ct \in \mathcal{F}_k$ as its ciphertext, where $\mathcal{F}_k$ is the set of all invariant polynomials of degree k or less. With invariant polynomials g_u in $\mathcal{F}_{k-1}$, Alice can recursively generate one in $\mathcal{F}_k$ in the form $\sum_{u \in N[v]} (g_u + c_u) x_u$ for a randomly selected vertex $v \in V$, where c_u are selected so that $g_u + c_u = m$. Finally, the encryption process is completed by applying the reductions described in Proposition 4 to obtain the ciphertext that can be hardly distinguished from a randomly selected polynomial in $\mathcal{F}_k$.

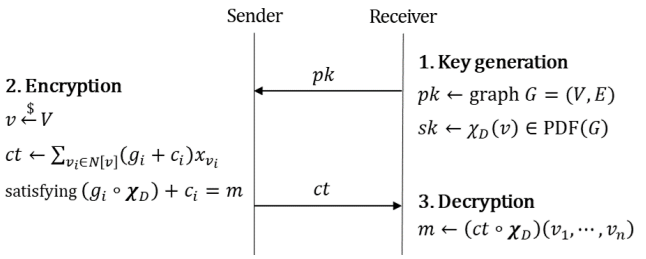

FIGURE 6. Secure communication with a perfect code cryptosystem (PCC).

After the encryption process is completed, Alice sends the ciphertext polynomial ct to Bob. Then, for decryption, Bob simply evaluates the polynomial $ct \circ \boldsymbol{\chi}_D$ to recover the original message m.

Example 7: As in Example 6, suppose that Alice has a message $m = 3$ and wants to encrypt m into an invariant polynomial of degree 2. Alice has the public key G shown in Fig. 4. Example 5 says that $e_i (i = 1, \ldots, 8)$ are invariant polynomials of degree 1 for the graph G. Although Alice does not know any $\chi \in \mathrm{PDF}(G)$, she knows that $e_i \circ \boldsymbol{\chi} = 1$ is always satisfied for all i, as shown in (1) of Example 5.

To generate the ciphertext, Alice randomly selects a vertex, say v_5. Then, the closed neighborhood $N[v_5] = \{v_4, v_5, v_6, v_8\}$. Alice selects four invariant polynomials of

degree 1:

$$g_{v_4} = 4e_1 - 2e_7, \quad g_{v_5} = 3e_7,$$
$$g_{v_6} = 5e_1, \quad g_{v_8} = 6e_2 - 2e_4.$$

Note that a linear combination of invariant polynomials is also invariant. Next, Alice determines constants c_u so that

$$(g_u \circ \boldsymbol{\chi}) + c_u = 3 = m.$$

In fact, constants are calculated as

$$c_{v_4} = 3 - (4 - 2) = 1, \quad c_{v_5} = 3 - (3) = 0,$$
$$c_{v_6} = 3 - (5) = -2, \quad c_{v_8} = 3 - (6 - 2) = -1.$$

Using these elements in $\mathcal{F}_1$, the ciphertext polynomial ct in $\mathcal{F}_2$ is constructed as follows:

$$\begin{aligned} ct &= (g_{v_4} + c_{v_4})x_{v_4} + (g_{v_5} + c_{v_5})x_{v_5} \\ &\quad + (g_{v_6} + c_{v_6})x_{v_6} + (g_{v_8} + c_{v_8})x_{v_8} \\ &= (4e_1 - 2e_7 + 1)x_{v_4} + (3e_7)x_{v_5} \\ &\quad + (5e_1 - 2)x_{v_6} + (6e_2 - 2e_4 - 1)x_{v_8} \\ &= 5x_{v_4} + 3x_{v_6} - x_{v_8} + 4x_{v_1}x_{v_8} + 3x_{v_2}x_{v_5} - 2x_{v_4}x_{v_7}. \end{aligned}$$

The final line is obtained after the reduction.

Suppose that Bob possesses the private-key $D = \{v_1, v_8\} \in \mathrm{PDS}(G)$. The corresponding $\chi_D \in \mathrm{PDF}(G)$ is defined as

$$\chi_D(v_1) = \chi_D(v_8) = 1, \quad \text{and} \quad \chi_D(v) = 0 \text{ elsewhere.}$$

Bob recovers the message m by evaluating $ct \circ \boldsymbol{\chi}_D$ as

$$\begin{aligned} &(ct \circ \boldsymbol{\chi})(v_1, v_2, \ldots, v_8) \\ &= ct(\chi_D(v_1), \ldots, \chi_D(v_8)) \\ &= ct(1, 0, 0, 0, 0, 0, 0, 1) \\ &= 5 \cdot 0 + 3 \cdot 0 - 1 + 4 \cdot 1 \cdot 1 + 3 \cdot 0 \cdot 0 - 2 \cdot 0 \cdot 0 \\ &= 3. \end{aligned}$$

To summarize, Koblitz shows the possibility of building a PCC, but it cannot be efficiently implemented [15]. Intuitively, higher-degree ciphertexts make it more difficult to mount an attack but take longer to encrypt messages. For example, consider a PCC using a 3-regular graph with 100 vertices. If we choose degree $k = 7$, the number of nonzero monomials in a randomly selected invariant polynomial in $\mathcal{F}_7$ will be larger than 10^8. Thus, sparse polynomials of high degree are desirable for ciphertexts, assuming that they do not compromise the security of the cryptosystem. Koblitz's PCC suggests an algorithm for constructing invariant polynomials in $\mathcal{F}_k$ to generate the ciphertext in time $O(n(r+1)^k)$ using r-regular graph with n vertices. Within $n(r+1)^k = 100 \times 4^7 \approx 10^6$ operations or so, one can encrypt a message with a sparse polynomial. However, Koblitz's scheme is feasible but not practical.

III. PROPOSED CRYPTOSYSTEM: IPCC7

In this section, we propose a post-quantum encryption scheme called IPCC7, the improved PCC with polynomials of degree 7. We also explain a quantitative security analysis with efficient implementation and performance analysis.

We will first clarify the notations. Here, m denotes the message to be encrypted; ct represents the ciphertext that takes the form of an invariant polynomial; f_k denotes an arbitrary invariant polynomial of degree k or less.

A. PARAMETERS

IPCC7 uses a 3-regular graph G with PDS as a public-key. By Property 2, G has four distinct PDSes: D_1, D_2, D_3, and D_4, one of which can be selected as the corresponding private-key. The PDFs $\chi_{D_i} : V \to \{0, 1\}$ are defined by

$$\chi_{D_i}(v) = \begin{cases} 1, & \text{if } v \in D_i, \\ 0, & \text{otherwise,} \end{cases} \tag{2}$$

for $i = 1, \ldots, 4$.

In addition to public and private-keys, the following security parameters should be determined:

- p is a parameter that determines the message space $\mathbb{Z}_p$, and it can be a composite number.
- n is the number of vertices in the graph (i.e., the size of the vertex set $n = |V|$).
- k is the degree of ciphertext polynomials.
- k' is the maximum degree of temporary subpolynomials generated in the encryption procedure.
- n_e is the parameter used for generating invariant polynomials of degree 1.

The parameter set summarized in Table 1 is chosen to ensure 128-bit security. The parameter set $S_{para} = \{p, n, k, k', n_e\}$ are preshared or made public beforehand. The rationale for these specific parameter choices is discussed in Section IV.

TABLE 1. Parameters for 128 bit security.

parameter	p	n	k	k'	n_e
value	2^{32}	256	7	2, 3, 4	3

B. ALGORITHMS

We use the notation $\xleftarrow{\$}$ for the random selection of an element from a set. For example, $v \xleftarrow{\$} V$ where v is an element of V. Similarly, $U \xleftarrow[i]{\$} V$ implies that the set U is a randomly selected subset of set V with size i, i.e., $U \subseteq V, |U| = i$. Further, we frequently use the fact that the number of closed neighborhoods in a vertex is four for all vertices in a 3-regular graph.

The cryptosystem comprises key generation, encryption, and decryption algorithms. The key generation algorithm creates a public-key graph and a set of private keys. The encryption algorithm generates a polynomial ciphertext for the message, and the decryption algorithm derives the message from the polynomial ciphertext.

1) KEY GENERATION

First, the key generation algorithm generates a 3-regular graph with PDSes. Since the 3-regular graph used as the public key is a simple and regular graph, it can be constructed by assigning six random one-to-one correspondences to each pair of PDSes.

Algorithm 1 (KeyGen) Key Generation

Input: Security parameter n
Output: A pair of keys (pk, sk)

1: $V \leftarrow \{v_1, v_2, \ldots, v_n\}$
2: Divide a set V into four disjoint subsets D_1, D_2, D_3, and D_4 such that
$$|D_1| = |D_2| = |D_3| = |D_4| = \frac{n}{4}$$
3: Randomly create six one-to-one correspondences between sets and connect the related vertices. The result is then generated as a graph $G = (V, E)$ with four PDSes.
4: Check whether the generated graph is connected. If not, repeat step 3.
5: $pk \leftarrow G = (V, E)$ and $sk \leftarrow D_1$
6: **return** pk, sk

Example 8: The procedure used to generate the graph shown in Fig. 4 using Algorithm 1 is as follows: First, we divided the set $V = \{v_1, v_2, \ldots, v_8\}$ into four subsets as shown in Fig. 7.

$$D_1 = \{v_1, v_8\}, \quad D_2 = \{v_2, v_5\},$$
$$D_3 = \{v_3, v_6\}, \quad D_4 = \{v_4, v_7\}.$$

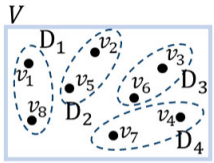

FIGURE 7. Example of key generation step 2.

Subsequently, six random one-to-one correspondences are randomly created between subsets as shown in Fig. 8.

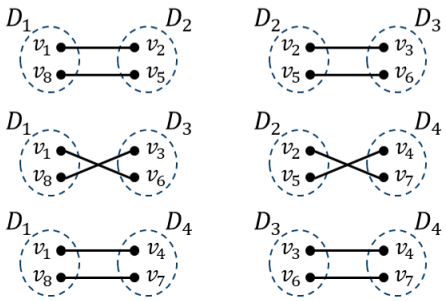

FIGURE 8. Example of key generation step 3.

We confirm that the graph generated by these correspondences is connected. Therefore, the sender uses the graph as a public key, as shown in Fig. 9.

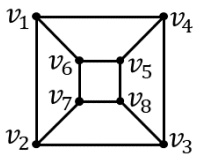

FIGURE 9. Example of key generation step 4.

Finally, we select a private key $D \xleftarrow{\$} \{D_1, D_2, D_3, D_4\}$. If we choose D_1, the corresponding PDF χ_{D_1} is depicted as shown on the right side in Fig. 10.

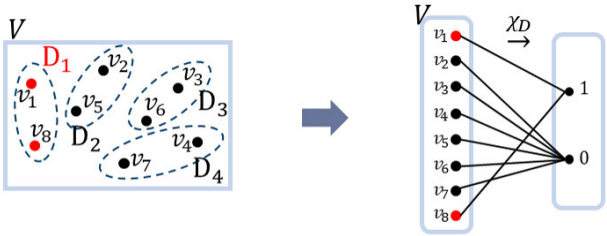

FIGURE 10. Example of key generation step 5.

2) ENCRYPTION

In the encryption process, Algorithm 2 generates an invariant polynomial as shown in Fig. 11.

Algorithm 2 (Enc) Encryption

Input: graph $G = (V, E)$, message m, and parameter set S_{para}
Output: invariant polynomial f of degree k

1: $m' \xleftarrow{\$} \mathbb{Z}_p$
2: Find m'', and c_m such that $m = m'm'' + c_m \pmod p$
3: $k' \leftarrow \lfloor k/2 \rfloor$
4: $k'' \leftarrow k - k'$
5: $f'_{k'} \leftarrow \text{EncDegK}(G, k', m')$
6: $f''_{k''} \leftarrow \text{EncDegK}(G, k'', m'')$
7: $f_k \leftarrow f'_{k'} \times f''_{k''} + c_m$
8: Reduction(f_k)
9: **return** f_k

In Fig. 11, *GIP* is a function that generates an invariant polynomial. In 12, a, b, c, d, e, and f in the gray or black boxes represent distinct vertex variables x_v. The "hiding phase" consists of five steps: *RD* (reduce degree), *RT* (reduce terms), and *SC* (sum coefficients) are the functions that organize the reduction process, and *SV* (sort variables) and *ST* (shuffle terms) are processes that prevent tracing of the polynomial generation process. Thus, this phase makes it impossible to perform attacks that factorize the ciphertext. The *hiding phase* is described in detail in APPENDIX D.

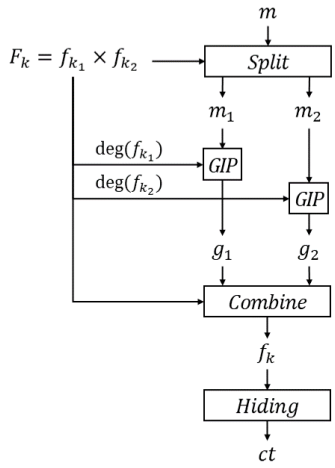

FIGURE 11. Encryption flow.

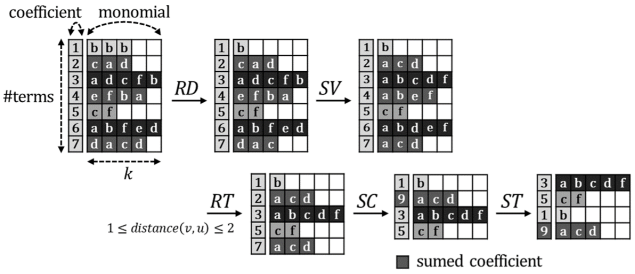

FIGURE 12. Hiding phase.

Algorithm 3 (EncDeg1) Generate an Invariant Polynomial of Degree 1

Input: graph $G = (V, E)$, message m
Output: invariant polynomial f of degree 1

1: Initialize the polynomial f_1 and temporary value sum.
$$f_1 \longleftarrow 0, \quad sum \longleftarrow 0.$$
2: Select the number of vertices n_v and distinct n_v vertices.
$$n_v \xleftarrow{\$} \{1, \ldots, n_e\},$$
$$\{v_1, \ldots, v_{n_v}\} \xleftarrow[n_v]{\$} V$$
3: For each v_j in $v_1, \ldots, v_{n_v-1}$ except for the last one, choose a random constant value $const_j \in \mathbb{Z}_p$.
$$sum \leftarrow sum + const_j \pmod p,$$
$$f_1 \leftarrow f_1 + const_j \sum_{u \in N[v_j]} x_u.$$
4: For the last vertex v_{n_v},
$$const_{n_v} \longleftarrow m - sum \pmod p,$$
$$f_1 \leftarrow f_1 + const_{n_v} \sum_{u \in N[v_{n_v}]} x_u$$
5: **return** f_1

Algorithm 4 (EncDegK) Degree k Invariant Polynomial Generation

Input: graph $G = (V, E)$, degree k, message m
Output: invariant polynomial f_k of the degree k

1: If $k = 1$, **return** $f_k \leftarrow$ EncDeg1(G, m)
2: Initialize the polynomial $f_k \longleftarrow 0$
3: Select a random vertex $v \xleftarrow{\$} V$
4: **for** For each $u \in N[v]$ **do**
5: Choose the random fragment $m' \xleftarrow{\$} \mathbb{Z}_p$
6: $f_{k-1} \leftarrow$ EncDegK$(G, k-1, m')$
7: Set $const \longleftarrow m - m' \pmod p$
8: $f_k \leftarrow f_k + (f_{k-1} + const)x_u$
9: **end for**
10: **return** f_k

3) DECRYPTION

The decryption algorithm is remarkably simple. The receiver obtains the message m by substituting the variables x_v in ct by 0 or 1 with the knowledge of $\chi_D \in$ PDF(G). As mentioned earlier in Examples 5 and 7, we can evaluate this algorithm using equation (2) as follows:

$$\left(ct \circ \boldsymbol{\chi}_D\right)(v_1, \ldots, v_n) = ct\left(\chi_D(v_1), \ldots, \chi_D(v_n)\right) = m.$$

Algorithm 5 (Dec) Decryption

Input: Private key D, ciphertext ct
Output: message m

1: Evaluate the polynomial ct using the private key D
$$m \longleftarrow \left(ct \circ \boldsymbol{\chi}_D\right)(v_1, \ldots, v_n) \pmod p$$
2: **return** m

C. DESIGN RATIONALE

In the last decade, various efforts have been made to design secure public-key cryptography for cryptanalysis using quantum computers, particularly in the NIST PQC standardization project. However, no candidate has yet been confirmed secure against quantum computing attacks. Therefore, cryptographic algorithms with new underlying problems still need to be considered. Many NP problems exist in combinatorics. In the early stage of public-key cryptography, combinatorics-based cryptosystems were attempted, but they were not practical because of their low efficiency and high complexity of implementation. We intend to improve the efficiency of combinatorics-based cryptography, quantitatively analyze its security, and propose a new quantum resistant algorithm.

Koblitz's PCC scheme in Definition 6 uses an invariant polynomial of a graph to generate a ciphertext. In fact, it should be almost randomly selected from a pool of invariant polynomials. The problem is that generating the necessary high-order polynomials requires much time. In designing a new cryptographic algorithm, we aim to increase the encryption speed. Since the encryption speed decreases to almost its square root when the maximum degree is halved, we tried to enhance the encryption speed by using low-degree polynomials to generate a high-degree polynomial. We quantitively analyzed the security and concluded that attackers must exert an effort equivalent to breaking a high-degree polynomial while the ciphertext is generated at a speed similar to that of a low-degree polynomial. In contrast to the encryption speed, the key generation and decryption speeds are remarkably higher.

1) GENERATING INVARIANT POLYNOMIALS

A k-degree invariant polynomial on a 3-regular graph with PDSes can be generated using $(k-1)$-degree invariant polynomials, as described in Proposition 3. Theoretically, to cover all invariant polynomials, we can pregenerate and store all invariant polynomials of degree less than k in a set $\mathfrak{F}_{k-1}$. However, the random selection of a polynomial from this set for each encryption is inefficient in terms of memory utilization. Instead, an encryption algorithm may use a recursive approach to generate an invariant polynomial of degree k in the form

$$\sum_{i=1}^{|N[u]|}(g_i + c_i)x_i, \quad u \in V.$$

where, g_i are invariant polynomials of degree $k-1$ generated recursively and do not contain any variables related to vertex $v \in N[u]$. Note that c_i belong to $\mathbb{Z}_p$.

When thus generating ciphertext polynomials, it is impossible to cover all conceivable invariant polynomials over graph G. Because of the existence of $O\left(n^{3k}/k^{2k}\right)$ (where $n \gg k$) invariant polynomials of degree k that can be generated recursively, this is expected to be a large burden for an attacker.

However, even with a recursive approach, the computational complexity of encryption increases drastically as the maximum degree of the polynomial increases. To ensure the usability of the algorithm, we did not just recursively generate a high-degree polynomial. Instead, we performed encryption by recursively generating several low-degree subpolynomials and combining them to form a high-degree polynomial.

2) COMBINING SUBPOLYNOMIALS

The primary concept of the proposed encryption algorithm is to generate a low-degree polynomial by fragmenting a message, as shown in Algorithm 9. Our goal is to randomize the combinatorial methods for splitting and reassembling messages. This concept can be summarized in the following algorithm, where, $\mathcal{F}_k$ is a family of methods that combine subpolynomials of degree less than k, such that the resulting polynomial is guaranteed to have degree k. Here, F_k denotes a polynomial combination method randomly selected from $\mathcal{F}_k$, represented as $F_k(f_1, f_2, \cdots)$. We choose F_7 for IPCC7 as

$$F_7(f_1, f_2) = f_1 \cdot f_2 + c,$$

where f_1 and f_2 are invariant polynomials of degree 4 or less, and c is a constant depending on the message to be encrypted.

Algorithm 6 (GeneralEnc) Generalized Encryption

Input: graph $G = (V, E)$, parameter set S_{para}, message m
Output: ciphertext ct

1: Select the combination method F_k from $\mathcal{F}_k$
2: Split the message m into random message pieces $m_1, m_2, \ldots$ so that $F(m_1, m_2, \ldots) = m$
3: For each message m_i, execute the algorithm 4 and store the result as g_i, i.e.,

$$g_i \leftarrow \text{EncDegK}(G, k_i, m_i)$$

where k_i are slected so that the degree of polynomial $F_k(g_1, g_2, \ldots)$ equals to k.
4: Combine the g_i into F_k as

$$\tilde{F}_k \leftarrow F_k(g_1, g_2, \ldots)$$

5: Apply the hiding procedure to $\tilde{F}_k$, including reduction.

$$ct \leftarrow Hiding(\tilde{F}_k)$$

6: **return** ct

The Algorithm 6 is illustrated in Fig. 13.

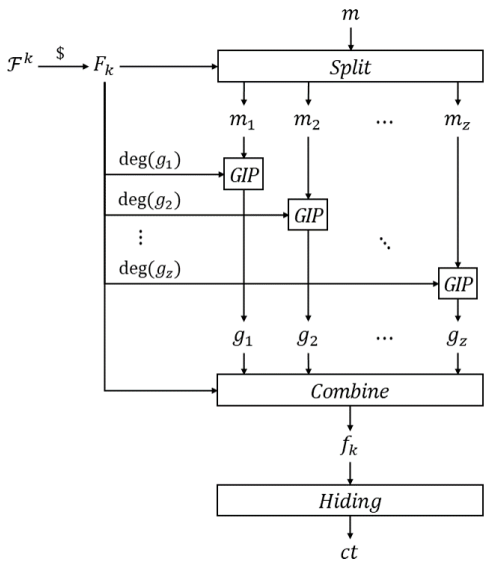

FIGURE 13. Encryption flow.

The message m is divided into $m_1, m_2, \ldots$ according to the combinatorial scheme F_k selected from $\mathcal{F}_k$. After recursively generating subpolynomials $g_1, g_2, \ldots$ of degree $k_1, k_2, \ldots$, we combine them into a k-degree polynomial $F_k(g_1, g_2, \ldots)$ to form the ciphertext.

When constructing a high-degree ciphertext polynomial from low-degree polynomials, fewer computations are required to generate the ciphertext than those required by an attacker to recover the plaintext, as discussed in Section IV.

In this scenario, an attacker can attempt an attack based on a low-degree polynomial. However, owing to the difficulty of polynomial factorization, it is not feasible for an attacker to divide the ciphertext polynomial into low-degree polynomials as designed [35], [36]. In addition, the reduction procedure at the end of encryption makes it harder for an attacker to restore the subpolynomials.

The receiver can decrypt the ciphertext without knowing how the sender generated it. However, an attacker cannot factorize the ciphertext without knowing F_k. Thus, the attacker cannot benefit from low-degree polynomials and is based on a high-degree polynomial.

Example 9: Suppose the sender wants to generate a ciphertext polynomial of degree k by choosing the following

combination scheme.

$$F = f_{k_1} \cdot f_{k_2} + f_{k_3} \cdot f_{k_4} \xleftarrow{\$} \mathcal{F}_k$$

When an attacker eavesdrops on a ciphertext and mounts an attack to recover the plaintext, they must guess the coefficients of the ciphertext polynomial. Knowledge of the combination of subpolynomials can drastically reduce the number of unknowns that need to be guessed. The attacker may attempt to decompose F into $f_{k_1} \cdot f_{k_2} + f_{k_3} \cdot f_{k_4}$. However, merely by intercepting the ciphertext, the attacker cannot determine the combination for F including the following methods:

$$\begin{aligned} &f_{k_1} \cdot f_{k_2} \\ &f_{k_1} + f_{k_3} \cdot f_{k_4} \\ &f_{k_1} \cdot f_{k_2} + f_{k_3} \cdot f_{k_4} \\ &\vdots \end{aligned}$$

Therefore, an attacker must try to recover the plaintext by guessing a polynomial of degree k without decompositon.

IV. SECURITY ANALYSIS

The primary attack strategies against PCC are exhaustive key search and plaintext recovery attacks. The main security parameters that directly affect both attacks are the size n of the graph G and the degree k of the ciphertext polynomial. In addition, a guessing attack in an exhaustive key search is influenced by the security parameter n_e used to generate invariant polynomials. The key recovery attack of an exhaustive key search is only affected by n. Security can be balanced by increasing k to ensure a security level similar to that provided against plaintext recovery attacks.

Before exploring various attack techniques, we note that our cryptosystem implements the reduction procedure when the distance between vertices is two or less. Therefore, considering a situation that an attack is executed, it is more efficient to select vertices at a distance of three or more.

Typically, when generating a polynomial, reduction is necessitated by the *hiding phase*. Predicting the output ciphertext after this phase becomes challenging, making it difficult to prove the security level. However, by selecting vertices in a manner such that they are not affected by the reduction, we can prove security more efficiently.

Fact 3 presents the number of combinations required to select j vertices in a 3-regular graph such that not all selected vertices are reduced. This corresponds to the case where $r_1 = 4$ and $r_2 = 10$ in Proposition 6.3(a) of Koblitz [15].

Proposition 5 (In the Proof of Koblitz's Proposition 6.3): Let G be an arbitrary graph with n vertices, and let u_0 be a vertex such that $|N[u_0]| = r_1$ where $r_1 = \min_{u \in V} \#\{v \in V : v \in N[u]\}$ and $r_2 = \max_{u \in V} \#\{v \in V : distance(u, v) \leq 2\}$. The number of monomials of degree j, composed of variables x_u, corresponds to the number of sets of distinct vertices $u \neq N[u_0]$ such that no two of the u are at a distance ≤ 2 from one another. Consequently, the number of subsets of j vertices is

$$\begin{aligned} &\geq \frac{(n - r_1)(n - r_1 - r_2) \cdots (n - r_1 - (j-1)r_2)}{j!} \\ &= r_2^j \binom{(n - r_1)/r_2}{j}. \end{aligned}$$

Fact 3: In a 3-regular graph with n vertices, the minimum number of methods to select distinct j vertices that generate the same monomial before and after applying the reduction is

$$10^j \binom{(n-4)/10}{j}.$$

A. KEY RECOVERY ATTACK

A key recovery attack is a method to find a PDS with $n/4$ vertices in a 3-regular graph G with n vertices to determine the secret key of the receiver. Determining whether an arbitrary graph has PDSes is NP-complete. It is conjectured that the problem of finding vertices in a graph with PDSes presents a comparable security level [20]. Even if the attacker fails to precisely identify D, which is a PDS selected from among the four PDSes by the receiver during key generation, the attack is considered successful if the attacker discovers at least one PDS. PCC counters the key recovery attacks by leveraging NP-hardness, where the computation of a key recovery attack is proportional to the size of graph n.

Theorem 1: The complexity of a naive exhaustive key search is

$$O(2^{n/2}) \leq O(\binom{n}{n/4}) \leq O(2^n).$$

Proof: Because a PDS of a 3-regular graph with n vertices consists of $n/4$ vertices, an attacker checks every subset with $n/4$ vertices to confirm whether it is a PDS. The number of trials is bounded by

$$\binom{n}{n/4} = \frac{n(n-1)\cdots(n-(n/4)+1)}{(n/4)!}.$$

Stirling's approximation can be applied to obtain the upper and lower bounds of the following expression:

$$\frac{(3n/4)^{n/4}}{(\pi n/2)^{1/2}\,(n/4e)^{n/4}} \leq \binom{n}{n/4} \leq \frac{n^{n/4}}{(\pi n/2)^{1/2}\,(n/4e)^{n/4}}.$$

This inequality can be simplified as

$$6^{n/4} \leq (3e)^{n/4} \leq \binom{n}{n/4} \leq (4e)^{n/4} \leq 12^{n/4}.$$

Since $2^{n/2} < 6^{n/4}$ and $12^{n/4} < 2^n$, we finally have

$$2^{n/2} < \binom{n}{n/4} < 2^n.$$

□

The security of our algorithm against key recovery attacks increases exponentially with the graph size n. A better

algorithm than the naive version, such as backtracking, can slightly reduce the time complexity of exhaustive PDS search, but the time complexity still increase exponentially. From Theorem 1, we may set $n = 256$ for 128-bit security.

B. CIPHERTEXT SEARCH ATTACK

Another exhaustive search strategy is the ciphertext search attack, which aims to identify a message corresponding to a given ciphertext. Each invariant polynomial has a unique value for any PDF in a given graph. According to Proposition 3, anyone can generate an invariant polynomial with the desired value without knowledge of the PDF. Let $\mathfrak{F}$ be a set of all possible ciphertext polynomials. Then, this set can be expressed as a disjoint union of the set $\mathfrak{F}(m)$ of ciphertext polynomials for every message m.

$$\mathfrak{F} = \bigcup_{m\in\mathbb{Z}_p} \mathfrak{F}(m).$$

Suppose that an attacker eavesdrops the ciphertext ct of a sender. If the attacker identifies $\mathfrak{F}(m)$, which contains the ciphertext polynomial ct corresponding to the message m, then they succeed in message recovery. Consequently, the cryptosystem should handle a sufficiently large number of invariant polynomials to ensure that the attacker cannot identify the set $\mathfrak{F}(m)$ to which the ciphertext belongs.

Fact 4: The computational cost of a ciphertext search attack depends on the number t_k of ciphertext polynomials of degree k or less.

The security level in Fact 4 is highly dependent on the parameters n, k, and n_e. Using Theorem 2, the size of ciphertext space $\mathfrak{F}(m)$ can be estimated.

Lemma 1: The number t_k of ciphertext polynomials of degree k or less satisfies

$$t_1 = p^{n_e}\binom{n}{n_e},$$
$$t_k \geq \left\{p^{n_e+1}\binom{n}{n_e}\right\}^{4^{k-1}} \quad \text{for } k > 1.$$

Proof: We use the security parameters such as p, n, and n_e defined in Table 1. Note that n_e is the number of polynomials of degree 1 whose linear combination forms an invariant polynomial f_1 of degree 1 during the encryption process, which is depicted in Algorithm 3 (EncDeg1).

When $k = 1$, polynomial f_1 is constructed as follows: Select n_e distinct vertices and define elementraty invariant polynomials e_i $(i = 1, 2, \ldots, n_e)$ for the vertices. Choose coefficients α_i in $\mathbb{Z}_p$ for $i = 1, 2, \ldots, n_e$. Then, generate linear combinations of the form

$$\alpha_1 e_1 + \alpha_2 e_2 + \cdots + \alpha_{n_e} e_{n_e}.$$

Considering the number of possible combinations, we have

$$t_1 = p^{n_e}\binom{n}{n_e}.$$

For $k > 1$, according to Proposition 5, polynomial f_k is generated by selecting one vertex from $n-4-10(n_e+k-3)$ *previously unselected* and *irreducible* vertices.

According to the step 8 in Algorithm 4, we can estimate t_k by the recursive formula as

$$t_k \geq (t_{k-1}p)^4\{n - 4 - 10(n_e + k - 3)\}.$$

Let $r(k) = p^4\{n - 4 - 10(n_e + k - 3)\}$. Then, we have

$$t_k = t_{k-1}^4 \cdot r(k),$$
$$t_{k-1} = t_{k-2}^4 \cdot r(k-1),$$
$$\vdots$$
$$t_2 = t_1^4 \cdot r(2).$$

By mathematical induction, it follows from $r(i) \geq p^4$ for all i that

$$t_k = t_1^{4^{k-1}} \prod_{i=2}^{k} r(i)^{4^{k-i}} \geq t_1^{4^{k-1}} \cdot p^{4^{k-1}}.$$

Consequently,

$$t_k \geq \left\{p^{n_e+1}\binom{n}{n_e}\right\}^{4^{k-1}}.$$

□

Theorem 2: If the encryption parameter n_e is chosen so that $n_e \ll n$ and $n_e \ll p$, then the number t_k of ciphertext polynomials of degree k or less satisfies

$$t_k \geq (np)^{n_e 4^{k-1}}.$$

Proof: The computational cost of a ciphertext search attack depends on the number of ciphertext terms, denoted as t_k.

First, let us estimate the number of ciphertext polynomials of order one. From Lemma 1,

$$t_1 = p^{n_e}\binom{n}{n_e} = p^{n_e}\frac{n(n-1)\cdots(n-n_e+1)}{n_e!}.$$

Stirling's approximation can be used to obtain the lower bound of the expression.

$$t_1 \geq \frac{\{p(n-n_e)\}^{n_e}}{(2n_e\pi)^{1/2}(n_e/e)^{n_e}} \geq \left(\frac{p(n-n_e)}{n_e}\right)^{n_e}.$$

The final representation is simplified by the relation $(2n_e\pi)^{1/2}e^{-n_e} < 1$ for $n_e = 1, 2, \ldots$. Thus, the following expression for t_k is obtained.

$$t_k \geq p^{4^{k-1}}\left\{p^{n_e}\binom{n}{n_e}\right\}^{4^{k-1}}$$
$$\geq p^{4^{k-1}}\left(\frac{p(n-n_e)}{n_e}\right)^{n_e 4^{k-1}}$$
$$\approx p^{(n_e+1)4^{k-1}}\left(\frac{n}{n_e}\right)^{n_e 4^{k-1}}$$

$$= (np)^{n_e 4^{k-1}} \left(\frac{p}{n_e^{n_e}}\right)^{4^{k-1}}$$
$$\approx (np)^{n_e 4^{k-1}}.$$

□

Theorem 2 shows that it is infeasible for an attacker to guess the ciphertext if we choose parameters n_e and k properly. In fact, given parameters $n = 256$ and $n_e = 3$ proposed by our algorithm, the security against ciphertext search attacks increases exponentially with the degree k of the ciphertext. For a given parameter $k = 7$, more than $2^{6\times 10^4}$ invariant polynomials can be generated. Therefore, it can be sufficiently challenging to determine whether for message m, ciphertext is contained in $\mathcal{F}(m)$.

C. PLAINTEXT RECOVERY ATTACK

A plaintext recovery attack compares the coefficients of an arbitrary invariant polynomial with those of the intercepted ciphertext to determine an equivalent polynomial that evaluates the same value. This method does not directly identify the key (i.e., find a PDS) or the ciphertext polynomials; instead, by using Gaussian elimination, it derives the message information dispersed among the coefficients.

In this attack, the attacker first generates an arbitrary polynomial to recover a message. They then compare the coefficients of the terms in the intercepted ciphertext with those of the terms in the arbitrarily generated polynomial. With the equivalent coefficients as unknowns, the attacker generates a system of linear equations. Then, by applying Gaussian elimination, the last column of the reduced row echelon form (RREF) matrix is summed at $\mathbb{Z}_p$ to retrieve the message. The procedure of this attack is detailed in APPENDIX B and a toy example is described in APPENDIX C.

The computational complexity of the attack depends on the size of the matrix used for Gaussian elimination. The computational complexity for an $a \times a$ matrix is about $O(a^3)$.

Fact 5: Let a be the minimum number of coefficients in an arbitrarily chosen ciphertext. Then an attacker can succeed in the plaintext recovery attack with complexity $O\left(a^3\right)$ for Gaussian elimination.

The a is proportional to n and k and are independent of n_e.

Lemma 2: The minimum number of monomials appearing in an arbitrary ciphertext polynomial generated by an attacker is

$$\sum_{i=1}^{k} \frac{10^{i-1}n}{i}\binom{\frac{n-4}{10}}{i-1}.$$

Proof: A set of monomials to be included in any ciphertext generated by an attacker must include all monomials that can appear in the intercepted ciphertext of the sender.

The encryption scheme of the sender can represent all the terms that appear after reduction over graph G. From Fact 3, the number of combinations for selecting j vertices from vertex set V to generate polynomials of degree j is

$$\geq 10^j \binom{\frac{n-4}{10}}{j},$$

where $\binom{x}{j}$ is assumed zero if $x \leq j-1$.

From Koblitz's proof for Proposition 5, in a 3-regular graph that includes cases in which the number of vertices at distance ≤ 2 is at most 10, the number of ways to select j distinct vertices unaffected by reduction is as follows.

$$\frac{n(n-4)(n-4-10)(n-4-2\cdot 10)\cdots(n-4-(j-1)10)}{j!} = \frac{10^{j-1}n}{j}\binom{\frac{n-4}{10}}{j-1}$$

Because the arbitrary polynomials generated for our attack cover cases $i = 1, 2, \cdots, k$, the minimum number of monomials that must be present in an arbitrary ciphertext polynomial is

$$\sum_{i=1}^{k} \frac{10^{i-1}n}{i}\binom{\frac{n-4}{10}}{i-1}.$$

□

Theorem 3: In an environment where an attacker prepares polynomials of degree k required for an attack in advance, the computational cost of a plaintext recovery attack is

$$O\left((2n)^{3k}/k^{3k}\right).$$

Proof: The computational complexity of a plaintext recovery attack depends on the size of the matrix used for Gaussian elimination. The size of the row space of the matrix required for an attack is $\sum_{i=1}^{k} \frac{10^{i-1}n}{i}\binom{\frac{n-4}{10}}{i-1}$.

Let the size of the row space be a. Then,

$$a^3 = \left\{\sum_{i=1}^{k} \frac{10^{i-1}n}{i}\binom{\frac{n-4}{10}}{i-1}\right\}^3$$
$$\geq \left\{\sum_{i=1}^{k} 10^i \binom{n/10}{i}\right\}^3.$$

Using Stirling's approximation, the lower bound is

$$a^3 \geq \left\{\sum_{i=1}^{k} \frac{(n-10i)^i}{(2i\pi)^{1/2}(i/e)^i}\right\}^3$$

Thus, the inequality can be simplified as follows:

$$a^3 \geq \left\{\sum_{i=1}^{k}\left\{\frac{e(n-10i)}{i}\right\}^i\right\}^3$$
$$\geq \left\{\frac{e(n-10k)}{k}\right\}^{3k}$$
$$\geq \left\{2^3\left(\frac{n^3}{k^3} - 30\frac{n^2}{k^2} + 300\frac{n}{k} - 1000\right)\right\}^k$$

$$\approx \left(\frac{2n}{k}\right)^{3k}.$$

□

From Theorem 3, given that the parameters of IPCC7 are $n \gg k$, the security of our algorithm against plaintext recovery attacks increases exponentially with the degree k of the ciphertext. From Theorem 3, it is seen that we achieve 128-bit security for plaintext recovery attack with $n = 256, k = 7$.

Remark 1: The lower bound of the computational cost incurred by an attacker for each attack can be summarized as follows:

- Key recovery attack: $O\left(2^{n/2}\right)$
- Ciphertext search attack: $O\left((np)^{n_e 4^{k-1}}\right)$
- Plaintext recovery attack: $O\left((2n)^{3k}/k^{3k}\right)$

Here, the computational cost for each attack grows exponentially with the security parameters n, k, and n_e.

Moreover, for a plaintext recovery attack assuming an arbitrary ciphertext polynomial with unknown coefficients, the computation required by an attacker to generate a single polynomial differs from that required by the sender to generate a ciphertext by at most $O(n_e^k)$. Therefore, the security level can be adjusted by increasing or decreasing the values of the relevant security parameters. For the parameters proposed in Table 1, IPCC7 achieves 128 bits and 149 bits of security against key recovery attacks and plaintext attacks, respectively. Moreover, the number of invariant polynomials that an attacker needs to search during a ciphertext search attack is greater than $2^{6\times 10^4}$, providing a sufficiently large target set for the attack.

D. RATIONALE FOR PARAMETER SELECTION

The security corresponding to the value of each parameter for each attack is expressed as follows:

TABLE 2. Complexity of key recovery attack for the number of vertices n (bit).

$n = \|V\|$	60	120	180	240	300
complexity	30	60	90	120	150

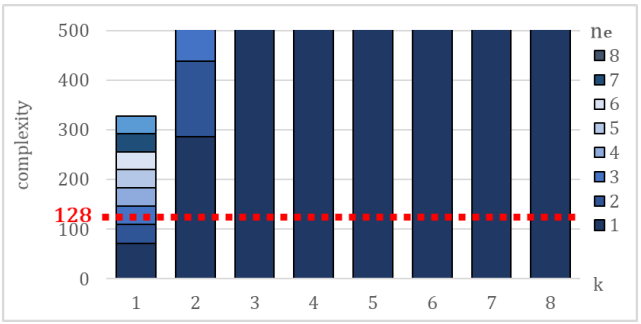

FIGURE 14. Complexity of ciphertext search attack for the degree k and parameter n_e (unit:bit, $n = 256$).

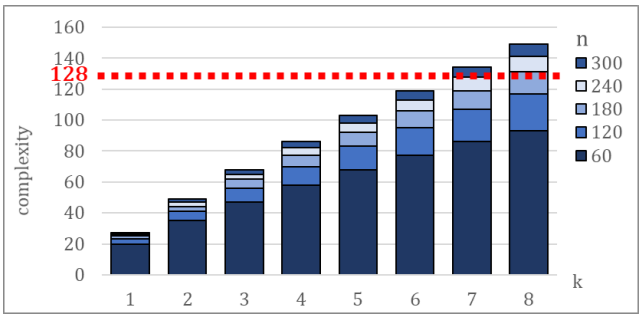

FIGURE 15. Complexity of plaintext recovery attack for the number of vertices n and degree k (unit: bit).

TABLE 3. Complexity of plaintext recovery attack for the degree k (unit: bit, $n = 256$).

k	2	3	4	5	6	7	8
complexity	44	62	80	96	112	128	144

From Table 2, to ensure 128-bit security against a key exhaustive search attack, the graph size must be $n \geq 256$. Thus, the parameter $n = 256$ proposed in Table 1 satisfies the 128-bit security requirements.

As the size of the graph increases, IPCC7 becomes more secure against key recovery attacks. However, we must consider memory usage to overcome implementation limitations. Before selecting parameter n, we considered the data type to represent the graph vertices from the perspective of implementation. This consideration is related to the memory size required for intermediate operations during the algorithm execution and the size of the ciphertext. We observed that an increase in the graph size for the same data type did not drastically affect the algorithm speed or memory usage. Hence, we chose a graph size of $n = 256$, which represents ≥ 256 vertices, to satisfy the 128-bit security level using a data type char (1 byte). This choice facilitated the generation of a 3-regular graph. The number 256 is a multiple of 4; thus, a 3-regular graph with $n = 256$ can be generated, with each byte representing 256 vertices (0-255).

According to Fig. 14, to ensure 128-bit security against ciphertext recovery attacks when $n_e = 1$, the maximum degree of the polynomial should be $k \geq 2$. The parameters k and k' proposed in Table 1 satisfy the 128-bit security level. Because the 128-bit security level is achieved when starting with $n_e = 1$, if there is a preference to use higher values to improve security, this can be made possible by considering the memory and speed. The proposed parameters were determined considering the implementation speed and ciphertext size.

According to Fig. 15, a plaintext recovery attack generates an arbitrary polynomial based on the maximum degree of the eavesdropped ciphertext. This attack can be counteracted by increasing k. According to Table 3, when the graph size is 256, $k \geq 7$ is necessary to ensure 128-bit security, where the proposed parameter $k = 7$ satisfies 128-bit security.

E. DEDICATED ATTACKS FOR SPECIFIC CASE

The security of PCC is based on two main impracticabilities: the difficulty of finding a PDS in a given graph and the difficulty of guessing ciphertext polynomials. Instead of the complicated recursive process adopted in IPCC7, one may expect that it is possible to construct a ciphertext polynomial as a simple combination of elementary invariant polynomials:

$$ct = \sum_{U_i \subset U} c_i \prod_{u \in U_i} \sum_{v \in N[u]} x_v, \tag{3}$$

where U is the power set of V which consists of all subsets of V, and c_i is constant in $\mathbb{Z}_p$. Then, it is easy to see that the corresponding plaintext is $\sum c_i$. However, this results in a dedicated attack [30].

In the encryption process, we increase the degree of the invariant polynomial from $k-1$ to k using $\sum(g_i + c_i)x_i$. Meanwhile, equation (3) can be considered as $(g+c)\sum x_i$. Thus constant c may appear repeatedly even in the final polynomial particularly when using a sparse polynomial for efficiency. Lange noted that in a sparse ciphertext constructed using (3), the message can be recovered by only adding different coefficients that appear individually [30].

To prevent such a coefficient summation attack, polynomials should be chosen more diversely, as $\sum(g_i + c_i)x_i$, at each stage during ciphertext generation. Then constants c_i are mixed up in the recursive process even in a sparse case. This method increases the computational effort required by the sender, but only by a constant factor.

V. IMPLEMENTATION

The encryption speed of a cryptosystem is drastically affected by the degree k. We proposed a method for constructing high-degree polynomials from low-degree polynomials. This method allows rapid encryption while retaining security against known attacks.

In this section, we describe the performance of the program using the proposed parameters and algorithms. The pseudocode of the cryptosystem is described in APPENDIX D, and an example of the program execution is described in APPENDIX E. Our cryptosystem operates under the assumption that the randomness used for vertex selection and coefficients are achieved using a cryptographically secure random number generator. To implement the proposed algorithm, we used the random number generator AES-CTR-DRBG.

A. IMPLEMENTATION CONSIDERATIONS

1) POLYNOMIAL STRUCTURE

In speed-sensitive cryptosystems, it is necessary to devise designs for static memory allocation, rather than for dynamic memory allocation. However, static memory allocation may lead to memory wastage owing to several reasons. Fortunately, when designing the implementation of the invariant polynomial in IPCC7, unlike in the case of general polynomials, a suitable range of upper and lower limits can be measured based on the degree of the polynomial. Therefore, we used static memory allocation, considering the upper limit in the design of a polynomial, coefficients, and vertices for a term.

This process consumes four bytes for a coefficient and one byte for each vertex. The coefficient is designed as an unsigned int data type to calculate the coefficients without BigNumber operations, where the vertex is represented as a byte data type to indicate the index of a vertex in a cryptosystem with 128-bit security.

2) LOOKUP TABLE

The encryption speed is greatly affected by the hiding procedure, which consumes 90% of the total encryption time. However, hiding is essential because it increases the complexity of factoring in the ciphertext by an attacker. To enhance the speed of hiding operation, we can prepare a precomputed lookup table. The public key, which is the output of the key generation algorithm, stores edge information. By reorganizing the table into a 256×4 table of closed neighborhoods for each vertex, we can improve the operation to check whether the distance between two vertices in the same term is less than or equal to two in the *RD* of the hiding process. Thus, the complexity can be reduced from $O(n)$ to $O(1)$. This enhancement leads to a considerable improvement in encryption performance.

Decryption is the process of obtaining a message using a PDS as the private key. The private key generated by the key generation algorithm is implemented as a byte array containing 256/4 PDS vertices. For decryption, the array is reconstructed into a 256×1 table. This is achieved by storing ''1'' for the vertex belonging to the PDS and ''0'' for the other vertices. Each nonzero term in the ciphertext contains the vertices from the PDS. Bitwise AND operations can be perrformed to determine whether all vertices in the term belong to the PDS.

3) FUTURE WORK

Currently, in our cryptosystem, most of the encryption time is consumed by the *hiding phase* during the *SC* process, with a time complexity of $O(n^2)$. However, by defining terms based on a lexical ordering according to the vertices, the speed was improved by approximately five times. It is predicted that for an appropriate application of hashing to lexical ordering, similar to the dictionary of Python or hashmap of Java, the time complexity will decrease from $O(n^2)$ to $O(n)$.

B. PERFORMANCE ANALYSIS

IPCC7, which is known to output very large ciphertexts, assumes an environment with abundant memory capacity. Therefore, when analyzing time and memory complexity, we first examined time complexity.

All benchmarks were obtained using an Intel(R) Core i5-7360U CPU @2.3GHz processor. The benchmarking PC was equipped with 8GB RAM and compiled using Apple Clang

version 13.0.0. All measured results were the averages of 1,000 iterations.

As observed in the previous analysis, the security strength of IPCC7 is determined by two main security parameters. Here, n is the number of vertices, and k is the maximum degree of the ciphertext polynomial. For a cryptosystem to be adopted in the real world, a balance must be established between these parameters in terms of security and performance. From the perspective of implementation, the speed of key generation is influenced by n. For a graph with $n = 256$, key generation takes 2.32 ms.

Before evaluating the improved encryption performance of IPCC7, we analyzed the speed of each function in the encryption algorithm and found that the *SC* procedure in the *hiding phase* consumes most of the total encryption time.

To address this issue, we measured the performance difference when applying the *SC* to only a portion of the terms in the ciphertext. The result indicated that the computational burden of *SC* can be reduced from $O(n^2)$ to $O(rn)$ depending on the ratio r of terms to which *SC* is applied. On increasing r from 0 to 1 in increments of 0.2, we found a marked rise in encryption time, particularly, as r shifted from 0.2 to 0.4 (see Fig. 16). Meanwhile, there was no noticeable decline in the number of terms during shifting. Consequently, we fixed the algorithm to execute the *SC* process on just 20% of the terms after the *RT* process of the *hiding phase* is finished.

Eliminating the *SC* process poses the risk of exposing the selection of random coefficients. However, by applying the *SC* for 20% of the terms, subsequent *ST* processes obscure the terms for which the *SC* was applied. This makes it challenging for attackers attempting to reverse engineer on the basis of the coefficients.

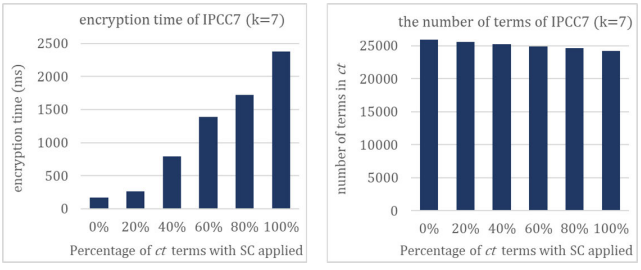

FIGURE 16. Encryption speed according to the percentage of terms in *ct* to which *SC* is applied in IPCC7 with $k = 7$.

Encryption speed is affected by both n and k, and the effect of k is prominent. Meanwhile, although decryption speed is also affected by n and k, its dependence on these variables is drastically less pronounced than that of the encryption speed.

For $n = 256$, the operating speeds of the existing method (PCC) and the proposed approach (IPCC7) are compared in Fig. 17. Comparison of the performances of generating degree-k ciphertext using the IPCC7 and PCC algorithms are shown in Fig. 17. We measured the performance of the PCC algorithm using its variant, namely, 4. For simplicity in subsequent references, Algorithm 4 will be referred to as PCC.

According to Fig. 17, the encryption speed of IPCC7 is approximately five times that of PCC. Conversely, the decryption speed of IPCC7 is relatively lower than that of PCC because the former generates more terms in the ciphertext than the latter. Nevertheless, given that the decryption speed of IPCC7 is inherently high, the difference in decryption speed between the two algorithms is negligible.

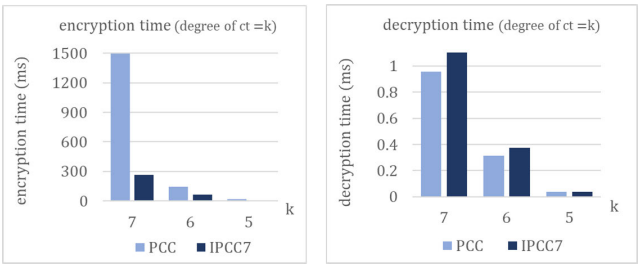

FIGURE 17. Algorithm performance for each degree k ($n = 256$, $n_e = 3$).

The differences in encryption speed between the PCC and the IPCC7 are summarized in Table 4. As explained in Section III-C, IPCC7 creates two lower-degree polynomials of degrees 3 and 4 and multiplies them to generate a ciphertext polynomial of degree 7. It takes about 1.5 s to create a polynomial of degree 7, but only 0.8 ms and 3.4 ms are sufficient for generating polynomials of degrees 3 and 4, respectively. Considering this, while the PCC spends a long time in the encryption to create the targeted degree of polynomial, IPCC7 enjoys a speed advantage in that it only takes about twice the time to achieve the target degree by generating low-degree polynomials. According to Table 4, as the degree of the ciphertext increases from 4 to 7, PCC requires approximately 440 times encryption time. The time required for generating a degree-7 ciphertext with IPCC7 is not simply the sum of the times needed to generate degree-3 and degree-4 polynomials with the PCC; it requires more encryption time than that required for the *hiding phase*. However, the time required to generate a degree-7 ciphertext polynomial is only about 80 times that needed for a degree-4 polynomial, making the increase in degree much less burdensome compared to the PCC.

TABLE 4. Performance of encryption algorithm for k when $n = 256$.

k	speed (ms)	
	PCC	**IPCC7**
3	0.77	0.84
4	3.40	3.22
5	18.33	4.88
6	141.13	63.71
7	**1498.83**	**262.34**

For all the provided parameters, after averaging the results of 1,000 runs for each scheme, the operational times were

2.32 ms for key generation, 262.34 ms for encryption, and 1.12 ms for decryption (see Table 5). The sizes of the public-key and secret-key are $(3n/2)2 = 768$ bytes and $n/4 = 64$ bytes, respectively, and the average size of the ciphertext is 235,818 bytes.

TABLE 5. Implementation speed with the proposed parameters.

Key Generation	Encryption	Decryption
3.33 Mbps	8.59 Mbps	2.01 Gbps

C. COMPARISON OF IPCC7 WITH OTHER PQC ALGORITHMS

In our implementation, IPCC7 generated a ciphertext of 235,818 bytes for a 768-byte public-key with 128-bit security. The size of the data transmitted over the network was 236,586 bytes, and the algorithm execution time was 2,338.71 ms (including key generation, encryption, and decryption). Fig. 18 and 19 compare the performances of IPCC7 and several other PQC algorithms, all run with the parameters required to ensure 128-bit security [34].

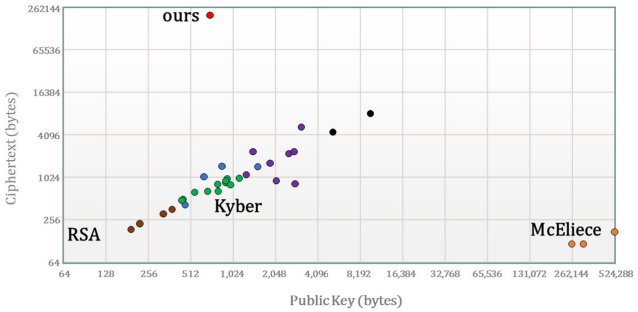

FIGURE 18. Comparison of IPCC7 and several other PQC algorithms in terms of public key and ciphertext size.

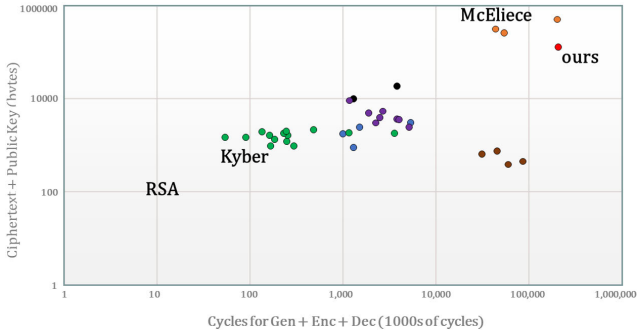

FIGURE 19. Comparison of IPCC7 and several other PQC algorithms in terms of operation speed and data size.

IPCC7 has the advantage of fast key generation and fast decryption. Its encryption speed, although relatively slow, is realistically usable. The unique features of this cryptosystem include its reliance on novel problems that are distinct from those proposed during the NIST PQC algorithm standardization process: small public key and large ciphertext sizes. Considering these characteristics, it is expected to outperform other PQC algorithms in various applications such as one-way functions and white-box encryption, particularly in environments where the size of the ciphertext is not an issue. White-box cryptography is a technology that protects encryption keys from threats such as memory dumps while encrypting messages. To achieve this purpose, this cryptography method uses several megabytes to several gigabytes of memory; hence, memory limitations are not a major concern. Consequently, IPCC7 is expected to be efficient when applied to the one-way external encoding required in the white-box encryption process.

Current research on IPCC7 has revealed its potential for use as a trapdoor one-way function in cryptographic primitives. Future studies will aim to reduce the performance gap with the general PQC algorithms by optimizing the recursively implemented code and offering parameter options for various security levels. Additionally, progressive research into the security model against general adversaries is expected to yield more reasonable and specific performance compared with the other PQC algorithms.

VI. CONCLUSION

In the domain of public-key cryptosystems, encryption schemes based on the graph theory have been mostly neglected for a long time. This is mainly because of the extreme inefficiency of such schemes in terms of speed and memory efficiency. However, by integrating the concept of polynomial combinatorics, we achieved a drastic improvement in encryption speed.

In this study, we determined the appropriate parameters for 128-bit security, including the size of the graph, maximum degree of ciphertext, and the number of terms in the ciphertext polynomial. We then demonstrated that an algorithm based on the graph theory can be implemented practically.

IPCC7 is still under study as a primitive. At present, it does not satisfy the conditions of the PQC scheme required by NIST because of the lack of analysis on IND-CCA2 security and the absence of variable security strengths based on different parameters. However, this study is meaningful in that it demonstrates that a trapdoor one-way function based on graph theory can actually be used as a PQC primitive. By analyzing and adapting this primitive to meet the construction requirements of the NIST, we can expect it to be usable as an actual PQC in environments with suitable resource requirements.

Cryptosystem design based on graph theory and combinatorics is not yet mature. Nonetheless, numerous concepts can be studied and integrated for the effective use of cryptosystems. For instance, subpolynomials can be generated using multiple graphs sharing a PDS, or a PDF can be extended to a perfect minus dominating function to enhance security against key recovery attacks.

In future work, we will analyze more general security, improve the algorithm, and optimize the implementation of the algorithm. We anticipate that these studies will bolster the field of combinatorics-based cryptosystems and contribute to the diversification of the problems underlying PQC.

APPENDIX A
LIST OF ABBREVIATIONS

TABLE 6. List of abbreviations.

PQC	Post-quantum cryptography
NIST	National Institute of Standards and Technology
PCC	Perfect code cryptosystem
IPCC7	Improved perfect code cryptosystem 7
IND-CCA2	Indistinguishability under adaptive chosen ciphertext attack
PDS	Perfect dominating set
PDF	Perfect dominating function
KeyGen	Key generation
Enc	Encryption
EncDeg1	Encryption for an invariant polynomial of degree 1
EncDegK	Encryption for an invariant polynomial of degree k
RD	Reduce degree
RT	Reduce terms
SC	Sum coefficients
SV	Sort variables
ST	Shuffle terms
Dec	Decryption

APPENDIX B
ALGORITHM OF THE PLAINTEXT RECOVERY ATTACK

Algorithm 7 (PRA) Plaintext Recovery Attack

Input: ciphertext $ct = f_k$

Output: plaintext pt

1: To generate an arbitrary polynomial $\widehat{f_k}$ that encompasses all the terms that appear in the ciphertext, the attacker may generate and collect the vertices required to generate the ciphertext, along with their corresponding pairs of unknowns.
2: These organized pairs are used to generate an arbitrary polynomial $\widehat{f_k}$. Note that $\widehat{f_k}$ is an invariant polynomial, generated by a simple recursive method without using $\mathcal{F}_k$.
3: Then, $f_k = \widehat{f_k}$ is applied to obtain a linear system for the sum of the unknowns, which are the coefficients of the monomials of $\widehat{f_k}$.
4: The resultant linear system is transformed and Gauss-Jordan elimination is applied to the matrix.
5: The values in the last column of the RREF matrix are added over $\mathbb{Z}_p$ to derive the plaintext pt.
6: **return** pt

APPENDIX C
EXAMPLE OF EXECUTING A PLAINTEXT RECOVERY ATTACK

This example illustrates how Gauss-Jordan elimination can be used in a plaintext recovery attack. Suppose that an attacker Eve wants to recover a message by intercepting the ciphertext. The security parameters

$$p = 11, n = 8, k = 1, k' = 1, n_e = 2$$

shared by the principle of cryptographic communication, Alice (sender) and Bob (receiver) are known, and Bob's public key graph is shown in Fig. 4.

Eve is also aware of this public security parameter and the public key. The ciphertext that has been eavesdropped upon after being generated by Alice is as follows:

$$ct = 4x_{v_1} + 9x_{v_3} + 2x_{v_5} + 4x_{v_6} + 2x_{v_7} + 9x_{v_8}.$$

To perform Gauss-Jordan elimination by generating a linear equation, Eve constructs an arbitrary polynomial containing all the possible combinations that Alice could have chosen to generate an invariant polynomial of degree 1. Given that $k = 1$, ${}_8C_1 = 8$ sets of vertices can be selected to generate a polynomial. The (vertex, coefficient) pairs that map a random coefficient $\widehat{c}_i$ onto each set are as follows:

$$(v_1, \widehat{c}_1), (v_2, \widehat{c}_2), (v_3, \widehat{c}_3), (v_4, \widehat{c}_4),$$
$$(v_5, \widehat{c}_5), (v_6, \widehat{c}_6), (v_7, \widehat{c}_7), (v_8, \widehat{c}_8).$$

By using these pairs to generate a ciphertext, Eve gets the following arbitrary invariant polynomial of degree 1:

$$\begin{aligned}\widehat{ct} = {} & \widehat{c}_1(x_{v_1} + x_{v_2} + x_{v_4} + x_{v_6}) \\ & + \widehat{c}_2(x_{v_1} + x_{v_2} + x_{v_4} + x_{v_6}) \\ & + \cdots \\ & + \widehat{c}_7(x_{v_2} + x_{v_6} + x_{v_7} + x_{v_8}) \\ & + \widehat{c}_7(x_{v_3} + x_{v_5} + x_{v_7} + x_{v_8}).\end{aligned}$$

At present, this polynomial is arranged for unknown coefficients $\widehat{c}_i$; however, it can be rearranged for vertex variables x_v. The solved polynomial is then compared with the stolen ciphertext to generate the following system of equations:

$$\begin{cases}\widehat{c}_1 + \widehat{c}_2 + \widehat{c}_4 + \widehat{c}_6 = 4 & (\text{coefficient of } x_{v_1}) \\ \widehat{c}_1 + \widehat{c}_2 + \widehat{c}_3 + \widehat{c}_7 = 0 & (\text{coefficient of } x_{v_2}) \\ \widehat{c}_2 + \widehat{c}_3 + \widehat{c}_4 + \widehat{c}_8 = 9 & (\text{coefficient of } x_{v_3}) \\ \widehat{c}_1 + \widehat{c}_3 + \widehat{c}_4 + \widehat{c}_5 = 0 & (\text{coefficient of } x_{v_4}) \\ \widehat{c}_4 + \widehat{c}_5 + \widehat{c}_6 + \widehat{c}_8 = 2 & (\text{coefficient of } x_{v_5}) \\ \widehat{c}_1 + \widehat{c}_5 + \widehat{c}_6 + \widehat{c}_7 = 4 & (\text{coefficient of } x_{v_6}) \\ \widehat{c}_2 + \widehat{c}_6 + \widehat{c}_7 + \widehat{c}_8 = 2 & (\text{coefficient of } x_{v_7}) \\ \widehat{c}_3 + \widehat{c}_5 + \widehat{c}_7 + \widehat{c}_8 = 9 & (\text{coefficient of } x_{v_8})\end{cases}$$

This system of equations can be represented as a matrix, as shown in Fig. 20. By applying Gauss-Jordan elimination to this matrix, we obtain an RREF matrix, as shown in Fig. 21. The message can be recovered by summing all values in the last column over $\mathbb{Z}_p$.

The values in the last column of Fig. 21 are {2, 2, 7, 0, 2, 0, 0, 0}, and their sum yields $13 = 2 \mod p$. This result is consistent with $4 + 9 = 2 \mod p$. which is obtained by

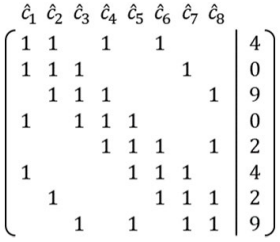

FIGURE 20. $ct = \hat{ct}$ matrix generated to perform a plaintext recovery attack.

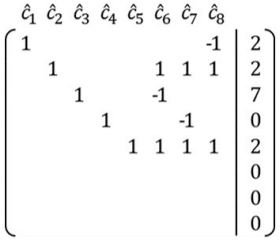

FIGURE 21. PPEF matrix of $ct = \hat{ct}$ with Gauss-Jordan Elimination.

substituting the value of the secret-key PDF corresponding to PDS $\{v_1, v_8\}$ in the public-key graph (Fig. 4) into the intercepted ciphertext ct.

APPENDIX D
PSEUDOCODE FOR IMPLEMENTATION

Each vertex $v \in V$ is represented by a 1-byte index number. Polynomial f_k generated for the input message (32-bit plaintext) is stored in a $t \times (k+1)$ two-dimensional array, where t is the number of polynomial terms. Here, cryptographic algorithms are described in the order of their execution for implementation.

A. KEY GENERATION

The key generation algorithm closely follows Algorithm 1, except for the storage process of the public-key and secret-key.

A public-key graph comprises a set of vertices V and a set of edges E, which can be collectively represented through neighborhood relationships for each vertex. Because the edges of a graph carry information on the vertices at both ends, we can ascertain all vertices from E without knowing V.

Thus, the graph public-key is stored as a $\frac{3n}{2} \times 2$ two-dimensional array, and the edge information is stored in a randomized order to prevent the inference of the PDS identity. The secret key is stored as an $\frac{n}{4}$ one-dimensional array, selected from one of the PDSes in the graph, without generating a PDF.

In the key generation algorithm, the notation $A[i]$ signifies the i th element of array A. When arrays A and B are of the same size, $A[a] \xleftrightarrow{\$} B[b]$ denotes a random one-to-one correspondence between elements $A[a]$ and $B[b]$.

Algorithm 8 (KeyGen) Key Generation Implementation

Input: The number of vertices n
Output: A pair of keys (pk, sk)

$V \leftarrow \{1, 2, \cdots, n\}$
$E \leftarrow \emptyset$
$L_1 \leftarrow$ Shuffling(V)
$D_1, D_2, D_3, D_4 \leftarrow \emptyset$
for $i = 1$ to $n/4$ ▷ generate PDS
  $D_1[i] \leftarrow L_1[4i-3]$
  $D_2[i] \leftarrow L_1[4i-2]$
  $D_3[i] \leftarrow L_1[4i-1]$
  $D_4[i] \leftarrow L_1[4i]$
for $i = 1$ to 3 ▷ generate graph that has PDS
  for $j = i+1$ to 4
    $L_2 \leftarrow$ Shuffling($\{1, 2, \cdots, \frac{n}{4}\}$)
    for $k = 1$ to $n/4$
      $E \leftarrow E \cup \{D_i[k] \xleftrightarrow{\$} D_j[L_2[k]]\}$
$sk \xleftarrow{\$} \{D_1, D_2, D_3, D_4\}$ ▷ Select PDS as the sk
$pk \leftarrow$ Shuffling(E) ▷ G has a vertex information potentially
return sk, pk

B. ENCRYPTION

The encryption algorithm is a function that calls Algorithms 11 and 10 to generate a k-degree invariant polynomial as a ciphertext. The distribute function generates message fragments and lower-degree invariant polynomials that comprise F, where q is the number of polynomials that constitute F. The function "Combine" combines the generated subpolynomials $g_1, g_2, \cdots, g_q$ into a polynomial of degree k, following the format F.

Algorithm 9 (Enc) Encryption Implementation

Input: Graph $G = (V, E)$, parameter set D, message m
Output: Ciphertext ct

$F \xleftarrow{\$} \mathcal{F}_k$
$\{m_1, m_2, \cdots, m_q\} \leftarrow$ Distribute(F, m)
$\{k_1, k_2, \cdots, k_q\} \leftarrow$ Distribute(F, k)
for $i = 1$ to q
  $g_i \leftarrow$ GenPolyDk(G, GRT, $\emptyset$, k_i, m_i)
$f_k \leftarrow$ Combine($g_1, g_2, \cdots, g_q$)
Hiding(f_k)
$ct \leftarrow f_k$
return ct

When an invariant polynomial is generated, an encryption algorithm is designed to avoid scenarios in which a polynomial with a maximum degree $< k$ is returned as a ciphertext. This implementation ensures that if any term of the polynomial satisfies the maximum degree k, then the ciphertext degree also satisfies k, guaranteeing that at least one term matches the k-degree.

For a randomly selected vertex v, if vertex v' selected during the recursive process does not belong to $N[v]$; the

term that includes v and v' is neither removed nor its degree decreased. To preserve the maximum degree, the algorithm passes the set of vertices to be considered in the reduction, that is, the set $\widetilde{V} \subset V$ comprises the vertices selected in the previous step.

The input parameter "GRT" is a flag that indicates whether the maximum degree of generated term satisfies k.

Algorithm 10 (GenDeg1) Degree 1 Invariant Polynomial Generation Implementation

Input: Graph $G = (V, E)$, selected vertices set $\widetilde{V}$, message m
Output: Invariant polynomial f of the degree 1

$n_e \xleftarrow{\$} \{1, 2, 3\}$
$U \xleftarrow[n_e]{\$} V \setminus \widetilde{V}$ ▷ U is a set consisted by n_e distinct randomly selected vertices
for $i = 1$ to n_e
  $value \xleftarrow{\$} \mathbb{Z}_p^\times$
  if $i = n_e$
    $value = m - sum$
  $sum \leftarrow sum + value$
  $f_1 \leftarrow f_1 + value \cdot \Sigma_{v \in N[u_i]} x_v$ ▷ $u_i \in U$
return f_1

Algorithm 11 (GenDegK) Degree k Invariant Polynomial Generation Implementation

Input: Graph $G = (V, E)$, guarantee gua, selected vertices set $\widetilde{V}$, degree $\widetilde{k}$, message m
Output: Invariant polynomial f of the degree $\widetilde{k}$

if $\widetilde{k} = 1$
  if $gua =$ GRT ▷ guarantees maximum degree
    $f_{\widetilde{k}} \leftarrow$ GenPolyD1(G, $\widetilde{V}$, m)
  else
    $f_{\widetilde{k}} \leftarrow$ GenPolyD1(G, $\emptyset$, m)
  return $f_{\widetilde{k}}$
$v \xleftarrow{\$} V$
if $gua =$ GRT
  while break ▷ choose and cheak a vertex not in $\widetilde{V}$
    if $v \notin \widetilde{V}$
      break
    $v \xleftarrow{\$} V \setminus \widetilde{V}$
$s \leftarrow 0$ ▷ $|N[v]| = 4$
if $gua =$ GRT
  $s \xleftarrow{\$} \{1, 2, 3, 4\}$
for $i = 1$ to 4
  $value \xleftarrow{\$} \mathbb{Z}_p$ ▷ value evaluated for $f_{\widetilde{k}-1}$
  if $i = s$ ▷ term that guarantees maximum degree
    $\widetilde{V} \leftarrow \widetilde{V} \cup \{u : u \in N[v[s]]\}$
    $f_{\widetilde{k}-1} \leftarrow$ GenPolyDk(G, GRT, $\widetilde{V}$, $\widetilde{k} - 1$, $value$)
  else
    $f_{\widetilde{k}-1} \leftarrow$ GenPolyDk(G, NOT, $\emptyset$, $\widetilde{k} - 1$, $value$)
  $const = m - value \pmod p$
  $f_{\widetilde{k}} \leftarrow f_{\widetilde{k}} + (f_{\widetilde{k}-1} + const) x_{v[s]}$
return $f_{\widetilde{k}}$

The *hiding phase* operates on the polynomial returned by Algorithm 11.

Algorithm 12 (Hiding) Reduction and Ordering Implementation

Input: Graph $G = (V, E)$, invariant polynomial f_k, maximum degree k
Output: Reducted invariant polynomial f'_k

$g_0 \leftarrow f_k$
$g_1 \leftarrow$ReduceDegree(G, g_0, k)
$g_2 \leftarrow$SortVariable(g_1, k)
$g_3 \leftarrow$ReduceTerms(G, g_2, k)
$g_4 \leftarrow$SumCoefficients(g_3, k)
$f'_k \leftarrow$ShufflingTerms(g_4, k)
return f'_k

The following notation is used to describe the subalgorithms within the *hiding phase*.

- $|f|$: the number of terms in f
- $f[\alpha]$: αth term of f
- $|f[\alpha]|$: the number of variables in the αth term of f
- $f[\alpha][\beta]$: vertex for the βth variable of the αth term in f
- $y_{\alpha,\beta}$: the βth variable of the αth term of f, i.e., $x_{f[\alpha][\beta]}$
- c_α: the coefficient of the αth term in f.

Detailed algorithms for the SortVariable and ShufflingTerms functions are not included. The full program code is available at https://github.com/KMURASEofficial/ipcc/tree/master/ipcc7.

Algorithm 13 (RD) Reducing Polynomial Degree

Input: Graph $G = (V, E)$, invariant polynomial f_k, maximum degree k
Output: Reduced invariant polynomial f'_k

$f'_k \leftarrow 0$
for $\alpha = 1$ to $|f_k|$
  $T \leftarrow \{y_{\alpha,1}\}$
  term $\leftarrow c_\alpha y_{\alpha,1}$
  for $\beta = 2$ to $k - 1$
    for $\gamma = \beta + 1$ to k
      if $y_{\alpha,\beta} \neq y_{\alpha,\gamma}$
        check $\leftarrow 0$
        if $y_{\alpha,\beta} \in T$
          check $\leftarrow 1$
        if check $= 0$
          $T \leftarrow T \cup \{y_{\alpha,\beta}\}$
          term $\leftarrow$ term $\cdot\, y_{\alpha,\beta}$
        if $|T| = k$
          break
        check $\leftarrow 0$
        if $y_{\alpha,\gamma} \in T$
          check $\leftarrow 1$
        if check $= 0$
          $T \leftarrow T \cup \{y_{\alpha,\gamma}\}$
          term $\leftarrow$ term $\cdot\, y_{\alpha,\gamma}$
        if $|T| = k$

```
23:                 break
24:     f'_k ← f'_k + term
25: return f'_k
```

Algorithm 14 (RT) Reducing Terms in Polynomial

Input: Graph $G = (V, E)$, invariant polynomial f_k, maximum degree k
Output: Reduced invariant polynomial f_k

```
1: for α = 1 to |f_k|
2:   for β = 1 to k − 1
3:     for γ = β to k
4:       for i = 1 to 4
5:         u_i ∈ N[f_k[α][β]]
6:         for j = 1 to 4
7:           w_j ∈ N[f_k[α][γ]]
8:           if N[u_i] ∩ N[w_j] ≠ ∅
9:             f_k[α] ← 0
10:            break
11: return f_k
```

Algorithm 15 (SC) Sum up the Coefficients of Terms Having Same Monomial

Input: Invariant polynomial f_k, maximum degree k
Output: Reducted invariant polynomial f'_k

```
1: f'_k ← f_k[1]
2: for α = 2 to |f_k|
3:   for β = 1 to |f'_k|
4:     dup ← 0
5:     for γ = 1 to k
6:       if f_k[α][γ] = f'_k[β][γ]
7:         dup ← dup + 1
8:     if dup = k
9:       c'_β ← c'_β + c_α
10:      goto line 2
11:  f'_k ← f'_k + f_k[α]
12: return f'_k
```

C. DECRYPTION

The decryption algorithm compares the variables of the terms in a ciphertext polynomial with those corresponding to the vertices stored in the PDS. Let U denote a set of vertex variables that form a monomial for any term in ciphertext. If $|U \cap sk| = |U|$, then the coefficients of that term are meaningful; otherwise, they are considered meaningless. In the algorithm below, *tmp* is a temporary variable that stores information on whether monomial variables are meaningful vertex variables.

In line 6, u denotes the jth vertex variable of the ith term. In line 7, $(if\ u \in PDS\ ?\ 1 : 0)$ is a ternary operator that returns "1" if u is an element of the PDS and "0" otherwise. In the iteration in line 5, $tmp = 1$ if all vertex variables in the term are in PDS, and $tmp = 0$ if none of them are, thus allowing us to decide whether to add c_i to pt in line 8. $|ct|$ is the number of terms in the ciphertext ct.

Algorithm 16 (Dec) Decryption Implementation

Input: Private key $sk(PDS)$, ciphertext ct
Output: message m

```
1: pt = 0
2: for i = 1 to |ct|
3:   if c_i ≠ 0
4:     tmp = 1
5:     for j = 1 to k
6:       tmp ← tmp × (if u ∈ PDS ? 1 : 0)
7:     pt ← pt + c_i × tmp
8: m ← pt
9: return m
```

APPENDIX E
EXAMPLE OF CIPHERTEXT

The example ciphertext ct presented below is a degree 7 invariant polynomial resulting from the encryption of message $pt = 4410$ where $pt \in \mathbb{Z}_{2^{32}}$, over a 3-regular graph of 256 vertices with $n_e = 3$. The total number of terms in ct is 22,641, and the size of the ciphertext polynomial is 249,051 bytes. In this study, we present 50 randomly chosen terms that constitute 0.2% of the total number of terms in the ciphertext. The full transcript is available at https://github.com/KMURASEofficial/ipcc/tree/master/ipcc7.

$$
\begin{aligned}
ct = {} & 485730577\, x_{v_{53}}x_{v_{119}}x_{v_{134}}x_{v_{169}}x_{v_{218}}x_{v_{229}} \\
& + 1185214412\, x_{v_{16}}x_{v_{31}}x_{v_{38}}x_{v_{63}}x_{v_{93}}x_{v_{193}}x_{v_{220}} \\
& + 1187133452\, x_{v_{53}}x_{v_{56}}x_{v_{120}}x_{v_{134}}x_{v_{169}}x_{v_{250}}x_{v_{255}} \\
& + 252775020\, x_{v_{53}}x_{v_{73}}x_{v_{164}}x_{v_{166}}x_{v_{206}}x_{v_{235}}x_{v_{250}} \\
& + 1280749315\, x_{v_{7}}x_{v_{60}}x_{v_{83}}x_{v_{105}}x_{v_{191}}x_{v_{217}} \\
& + 1393233314\, x_{v_{18}}x_{v_{53}}x_{v_{58}}x_{v_{85}}x_{v_{131}}x_{v_{206}}x_{v_{211}} \\
& + 832293056\, x_{v_{51}}x_{v_{60}}x_{v_{63}}x_{v_{196}}x_{v_{213}}x_{v_{247}} \\
& + 484242184\, x_{v_{63}}x_{v_{93}}x_{v_{108}}x_{v_{183}}x_{v_{206}} \\
& + 93293340\, x_{v_{31}}x_{v_{63}}x_{v_{65}}x_{v_{130}}x_{v_{146}}x_{v_{228}}x_{v_{229}} \\
& + 1926696176\, x_{v_{1}}x_{v_{38}}x_{v_{53}}x_{v_{68}}x_{v_{134}}x_{v_{141}} \\
& + 1442193836\, x_{v_{31}}x_{v_{63}}x_{v_{67}}x_{v_{124}}x_{v_{125}}x_{v_{166}} \\
& + 2017528070\, x_{v_{6}}x_{v_{24}}x_{v_{31}}x_{v_{124}}x_{v_{212}}x_{v_{225}}x_{v_{227}} \\
& + 1582964880\, x_{v_{31}}x_{v_{53}}x_{v_{193}}x_{v_{229}}x_{v_{236}}x_{v_{255}} \\
& + 403276964\, x_{v_{31}}x_{v_{47}}x_{v_{120}}x_{v_{122}}x_{v_{124}}x_{v_{152}}x_{v_{191}} \\
& + 1866835366\, x_{v_{53}}x_{v_{139}}x_{v_{140}}x_{v_{183}}x_{v_{206}}x_{v_{239}}x_{v_{250}} \\
& + 79727103\, x_{v_{3}}x_{v_{17}}x_{v_{38}}x_{v_{120}}x_{v_{134}}x_{v_{191}}x_{v_{193}} \\
& + 1569908464\, x_{v_{53}}x_{v_{60}}x_{v_{73}}x_{v_{78}}x_{v_{229}}x_{v_{255}} \\
& + 733183104\, x_{v_{5}}x_{v_{38}}x_{v_{48}}x_{v_{105}}x_{v_{134}}x_{v_{191}}x_{v_{236}} \\
& + 1335896384\, x_{v_{5}}x_{v_{38}}x_{v_{48}}x_{v_{105}}x_{v_{134}}x_{v_{190}}x_{v_{191}} \\
& + 1531249596\, x_{v_{31}}x_{v_{51}}x_{v_{63}}x_{v_{169}}x_{v_{181}}x_{v_{193}}x_{v_{247}} \\
& + 1597157840\, x_{v_{25}}x_{v_{60}}x_{v_{63}}x_{v_{122}}x_{v_{146}}x_{v_{202}} \\
& + 272552863\, x_{v_{78}}x_{v_{134}}x_{v_{143}}x_{v_{157}}x_{v_{171}}x_{v_{191}} \\
& + 1034371758\, x_{v_{63}}x_{v_{125}}x_{v_{134}}x_{v_{159}}x_{v_{166}}x_{v_{169}}
\end{aligned}
$$

$$+ 891399469\, x_{v_7} x_{v_{83}} x_{v_{85}} x_{v_{131}} x_{v_{191}} x_{v_{206}} x_{v_{231}}$$
$$+ 1943924752\, x_{v_{38}} x_{v_{82}} x_{v_{134}} x_{v_{143}} x_{v_{171}} x_{v_{191}} x_{v_{209}}$$
$$+ 1804266088\, x_{v_{18}} x_{v_{19}} x_{v_{31}} x_{v_{53}} x_{v_{55}} x_{v_{124}}$$
$$+ 1485573648\, x_{v_7} x_{v_{83}} x_{v_{166}} x_{v_{191}} x_{v_{206}} x_{v_{217}} x_{v_{244}}$$
$$+ 777739680\, x_{v_{25}} x_{v_{53}} x_{v_{60}} x_{v_{68}} x_{v_{154}}$$
$$+ 987761874\, x_{v_{53}} x_{v_{68}} x_{v_{134}} x_{v_{152}} x_{v_{169}} x_{v_{248}}$$
$$+ 665067856\, x_{v_{53}} x_{v_{60}} x_{v_{68}} x_{v_{76}} x_{v_{211}}$$
$$+ 2069066654\, x_{v_{16}} x_{v_{31}} x_{v_{63}} x_{v_{166}} x_{v_{193}}$$
$$+ 328203030\, x_{v_{63}} x_{v_{75}} x_{v_{134}} x_{v_{139}} x_{v_{166}} x_{v_{169}}$$
$$+ 1804266088\, x_{v_{18}} x_{v_{31}} x_{v_{53}} x_{v_{55}} x_{v_{124}} x_{v_{157}}$$
$$+ 94944640\, x_{v_{25}} x_{v_{26}} x_{v_{33}} x_{v_{63}} x_{v_{93}} x_{v_{183}} x_{v_{206}}$$
$$+ 1280749315\, x_{v_5} x_{v_7} x_{v_{60}} x_{v_{83}} x_{v_{191}} x_{v_{210}}$$
$$+ 356289050\, x_{v_4} x_{v_{60}} x_{v_{120}} x_{v_{191}} x_{v_{194}} x_{v_{210}}$$
$$+ 615627056\, x_{v_{20}} x_{v_{63}} x_{v_{75}} x_{v_{166}} x_{v_{206}} x_{v_{244}}$$
$$+ 840805087\, x_{v_{18}} x_{v_{53}} x_{v_{58}} x_{v_{69}} x_{v_{126}} x_{v_{131}} x_{v_{206}}$$
$$+ 1422995562\, x_{v_{14}} x_{v_{31}} x_{v_{53}} x_{v_{68}}$$
$$+ 83463576\, x_{v_{16}} x_{v_{31}} x_{v_{53}} x_{v_{54}} x_{v_{193}} x_{v_{225}} x_{v_{229}}$$
$$+ 1301019424\, x_{v_{15}} x_{v_{25}} x_{v_{60}} x_{v_{171}} x_{v_{191}}$$
$$+ 1026991232\, x_{v_{25}} x_{v_{31}} x_{v_{33}} x_{v_{63}} x_{v_{93}} x_{v_{164}} x_{v_{228}}$$
$$+ 1399391380\, x_{v_{53}} x_{v_{68}} x_{v_{134}} x_{v_{154}} x_{v_{169}} x_{v_{255}}$$
$$+ 544586942\, x_{v_{60}} x_{v_{63}} x_{v_{122}} x_{v_{146}} x_{v_{174}}$$
$$+ 737529456\, x_{v_{38}} x_{v_{60}} x_{v_{63}} x_{v_{92}} x_{v_{93}} x_{v_{227}}$$
$$+ 308534452\, x_{v_{53}} x_{v_{134}} x_{v_{169}} x_{v_{200}} x_{v_{218}} x_{v_{229}}$$
$$+ 1928067505\, x_{v_{63}} x_{v_{144}} x_{v_{146}} x_{v_{183}} x_{v_{206}} x_{v_{226}} x_{v_{250}}$$
$$+ 415282332\, x_{v_6} x_{v_{31}} x_{v_{49}} x_{v_{114}} x_{v_{212}} x_{v_{228}} x_{v_{237}}$$
$$+ 84079493\, x_{v_{31}} x_{v_{53}} x_{v_{164}} x_{v_{193}} x_{v_{226}} x_{v_{227}} x_{v_{250}}$$
$$+ 262643568\, x_{v_9} x_{v_{24}} x_{v_{31}} x_{v_{53}} x_{v_{124}} x_{v_{225}} x_{v_{229}}$$
$$+ \cdots$$

REFERENCES

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Rev.*, vol. 41, no. 2, pp. 303–332, Jan. 1999, doi: 10.1137/S0036144598347011.

[2] IBM. *The IBM Quantum Development Roadmap*. Accessed: Jun. 19, 2023. [Online]. Available: https://www.ibm.com/quantum/roadmap

[3] R. Kuang and M. Perepechaenko, "Quantum encryption with quantum permutation pad in IBMQ systems," *EPJ Quantum Technol.*, vol. 9, no. 1, p. 26, Dec. 2022, doi: 10.1140/epjqt/s40507-022-00145-y.

[4] J. Buchmann and J. Ding, "Post-quantum cryptography," in *Proc. 2nd Int. Workshop PQCrypto*, vol. 5299, Cincinnati, OH, USA. Berlin, Germany: Springer, Oct. 2008, pp. 1–14.

[5] NIST. *Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process*. Accessed: Apr. 12, 2023. [Online]. Available: https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf

[6] NIST. *Post-Quantum Cryptography Standardization*. Accessed: May 14, 2023. [Online]. Available: https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization

[7] J. Buchmann, E. Dahmen, and M. Szydlo, "Hash-based digital signature schemes," in *Post-Quantum Cryptography*. Berlin, Germany: Springer, 2009, pp. 25–93.

[8] "Status report on the third round of the NIST post-quantum cryptography standardization process," NIST Interagency, Gaithersburg, MD, USA, Tech. Rep. 8413, Jul. 2022.

[9] C. Wouter and D. Thomas, "An efficient key recovery attack on SIDH," in *Advances in Cryptology—EUROCRYPT 2023* (Lecture Notes in Computer Science), vol. 14008, H. Carmit and S. Martijn, Eds. Cham, Switzerland: Springer, 2023, pp. 423–447.

[10] M. H. Freedman, "P/NP, and the quantum field computer," *Proc. Nat. Acad. Sci. USA*, vol. 95, no. 1, pp. 98–101, Jan. 1998, doi: 10.1073/pnas.95.1.98.

[11] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of Np-Completeness*. San Francisco, CA, USA: Freeman, 1979.

[12] F. Rosamond, "Computational thinking enrichment: Public-key cryptography," *Informat. Educ.*, vol. 17, no. 1, pp. 93–103, Apr. 2018.

[13] J. Kratochvl, "Perfect codes in general graphs," in *Combinatorics* (Colloquia Mathematica Societatis Janos Bolyai), vol. 52. Amsterdam, NY, USA: North-Holland, 1988, pp. 357–364.

[14] F. Arat and S. Akleylek, "Attack path detection for IIoT enabled cyber physical systems: Revisited," *Comput. Secur.*, vol. 128, May 2023, Art. no. 103174, doi: 10.1016/j.cose.2023.103174.

[15] M. Fellows and N. Koblitz, "Combinatorially based cryptography for children (and adults)," *Congr. Numerantium*, vol. 99, pp. 9–41, Aug. 1994.

[16] M. Fellows and N. Koblitz, "Kid krypto," in *Proc. Annu. Int. Cryptol. Conf.* Berlin, Germany: Springer, 1992, pp. 371–389.

[17] J. A. Bondy and U. S. R. Murty, *Graph Theory With Applications*. London, U.K.: Macmillan, 1976.

[18] T. W. Haynes, S. Hedetniemi, and P. Slater, *Fundamentals of Domination in Graphs*. Boca Raton, FL, USA: CRC Press, 1998.

[19] S. Yoon, "(1,-1,0)-perfect minus dominating function and its application to the public-key cryptosystem," M.S. thesis, Dept. Math. Educ., Seoul Nat. Univ., Seoul, South Korea, 2001.

[20] J. Kratochvl, "Regular codes in regular graphs are difficult," *Discrete Math.*, vol. 133, nos. 1–3, pp. 191–205, Oct. 1994, doi: 10.1016/0012-365X(94)90026-4.

[21] S. Kwon, J.-S. Kang, and Y. Yeom, "Analysis of public-key cryptography using a 3-regular graph with a perfect dominating set," in *Proc. IEEE Region Symp. (TENSYMP)*, Jeju, South Korea, Aug. 2021, pp. 1–6, doi: 10.1109/TENSYMP52854.2021.9550868.

[22] D. W. Bange, A. E. Barkauskas, L. H. Host, and P. J. Slater, "Generalized domination and efficient domination in graphs," *Discrete Math.*, vol. 159, nos. 1–3, pp. 1–11, Nov. 1996, doi: 10.1016/0012-365X(95)00094-D.

[23] J. Dunbar, W. Goddard, S. Hedetniemi, A. McRae, and M. A. Henning, "The algorithmic complexity of minus domination in graphs," *Discrete Appl. Math.*, vol. 68, nos. 1–2, pp. 73–84, Jun. 1996, doi: 10.1016/0166-218X(95)00056-W.

[24] X. Bultel, J. Dreier, P. Lafourcade, and M. More, "How to explain modern security concepts to your children," *Cryptologia*, vol. 41, no. 5, pp. 422–447, Sep. 2017, doi: 10.1080/01611194.2016.1238422.

[25] L. Wang and W. Wang, "An approximation algorithm for a variant of dominating set problem," *Axioms*, vol. 12, no. 6, p. 506, May 2023, doi: 10.3390/axioms12060506.

[26] A. Razaq, G. Alhamzi, S. Abbas, M. Ahmad, and A. Razzaque, "Secure communication through reliable S-box design: A proposed approach using coset graphs and matrix operations," *Heliyon*, vol. 9, no. 5, May 2023, Art. no. e15902, doi: 10.1016/j.heliyon.2023.e15902.

[27] A. Razzaque, A. Razaq, S. M. Farooq, I. Masmali, and M. I. Faraz, "An efficient S-box design scheme for image encryption based on the combination of a coset graph and a matrix transformer," *Electron. Res. Arch.*, vol. 31, no. 5, pp. 2708–2732, Jan. 2023.

[28] J. Ryu et al., "KpqC competition round 1," IPCC, KpqC Competition, Tech. Rep., 2023. [Online]. Available: https://www.kpqc.or.kr/competition.html

[29] S. Kwon, "A study on graph-based public-key cryptographic primitives and transition to post-quantum cryptography," M.S. thesis, Dept. Finance Inf. Secur. Kookmin Univ., Seoul, South Korea, 2021.

[30] T. Lange, "Analysis of IPCC," KpqC-Bulletin, KpqC Competition Round 1, Tech. Rep., 2022. [Online]. Available: https://groups.google.com/g/kpqc-bulletin/c/vy0BSMg4cI4/m/RkpbP77fBAAJ

[31] S. Chow, P. Eisen, H. Johnson, and P. C. Van Oorschot, "A white-box DES implementation for DRM applications," in *Proc. ACM Workshop Digit. Rights Manag.* Berlin, Germany: Springer, Nov. 2002, pp. 1–15.

[32] A. Bogdanov and T. Isobe, "White-box cryptography revisited: Space-hard ciphers," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2015, pp. 1058–1069.

[33] T. Isobe, H. Hiwatari, and K. Shibutani, "Encryption device, encryption method, decryption device and decryption method," U.S. Patent 2019 0 103 957, A1, Apr. 4, 2019, p. 1029.
[34] D. Moody, "The 2nd round of the NIST PQC standardization process," in *The 2nd PQC Standardization Conf.* Santa Barbara, CA, USA: University of California, Santa Barbara, 2019, pp. 15–16.
[35] M. T. Dickerson, "The functional decomposition of polynomials," Ph.D. thesis, Dept. Comput. Sci., Cornell Univ., Ithaca, NY, USA, 1989.
[36] J. Ding and D. Schmidt, "Rainbow, a new multivariable polynomial signature scheme," in *Proc. Int. Conf. Appl. Cryptography Netw. Secur.* Berlin, Germany: Springer, 2005, pp. 164–175.

JIEUN RYU received the B.S. degree in information security cryptography mathematics from Kookmin University, Seoul, Republic of Korea, in 2022, where she is currently pursuing the M.S. degree in financial information security. Her research interests include post-quantum cryptography, white-box cryptography, and security protocol.

YONGBHIN KIM is currently pursuing the B.S. degree in information security cryptography mathematics with Kookmin University, Seoul, Republic of Korea. His research interests include post-quantum cryptography, white-box cryptography, and optimization of cryptographic algorithms and their efficient implementation on various IT devices.

SEUNGTAI YOON received the B.S. and M.S. degrees in mathematics education from Seoul National University, Seoul, Republic of Korea, in 1999 and 2001, respectively, and the Ph.D. degree in applied mathematics and statistics from Stony Brook University, New York, NY, USA, in 2006. His research interests include cryptography, abstract algebra, and statistical genetics.

JU-SUNG KANG received the B.S., M.S., and Ph.D. degrees in mathematics from Korea University, Seoul, South Korea, in 1989, 1991, and 1996, respectively. From 1997 to 2004, he was a member of the Technical Staff with the Electronics and Telecommunication Research Institute. He has been a Professor with Kookmin University, since 2004. His research interests include cryptanalysis, protocol, and random number generator.

YONGJIN YEOM received the B.S., M.S., and Ph.D. degrees in mathematics from Seoul National University, Seoul, Republic of Korea, in 1991, 1994, and 1999, respectively. From 2000 to 2011, he was a member of the Engineering Staff and a Principal Researcher with the National Security Research Institute. He has been a Professor with Kookmin University, since 2012. His research interests include cryptanalysis, random number generator, and security system evaluation.