Code-Based Non-Interactive Key Exchange Can Be Made

Zhuoran Zhang[1,2] and Fangguo Zhang[1,2] ⋆

[1] School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510006, China
[2] Guangdong Province Key Laboratory of Information Security Technology, Guangzhou 510006, China

Abstract. Code-based cryptography plays an important role in post-quantum cryptography. While many crypto-primitives such as public-key encryption, digital signature have been proposed from codes, there is no non-interactive code-based key exchange protocol. We solve the opening problem of constructing a non-interactive key exchange protocol from coding theory in this work. To prove the security of this protocol, we propose a new hard problem called sub-LE problem, which is a sub-problem of code equivalence problem. We prove its hardness by reducing the well-known code linearly equivalence problem to the sub-LE problem. This new hard problem provides many good properties such as the partial commutativity. This excites us most because it allows not only the construction of key exchange protocol, but also many other primitives such as a new public-key encryption scheme.

Keywords: Post-quantum, Code-based, Key exchange, Public-key encryption

1 Introduction

Key exchange protocols are mechanisms by which two parties that communicate over an adversarially controlled network can generate a common secret key. This kind of protocols are essential for enabling the use of shared-key cryptography to protect transmitted data over insecure networks. Since the most well-known Diffe-Hellman protocol's proposing in 1976 [14], there are lots of researches focus on it. However, with the fast development of quantum computing, the cryptosystems whose security are based on traditional mathematical problems are facing attacks from quantum algorithms.

Nowadays, post-quantum cryptosystems, i.e. cryptosystems that can resist attacks from quantum computing are the most popular research direction. Although many wonderful post-quantum encryption systems, digital signature systems, and other cryptographic applications such as secret handshake protocols and function encryption systems have been proposed, there are only a few works

⋆ Corresponding author, email: isszhfg@mail.sysu.edu.cn

on key exchange protocols. The most famous post-quantum key exchange protocol is the supersingular isogenies-based SIDH [12]. In 2012, Ding *et.al* [15] proposed a lattice-based key exchange protocol with the help of robust extractors. Lattice-based authenticated key agreement protocols are also widely researched [32, 33]. But these lattice-based protocols need interaction resources or can be viewed as a key encapsulation mechanism.

Coding theory is an important tool to construct post-quantum cryptosystems. The most widely used hard problems in coding theory to build cryptosystem are syndrome decoding problem, code equivalence problem, and some other problems under rank metric. Unlike public-key encryption systems [21, 23], digital signature systems [10, 11], identification systems [29, 30] which have been well-studied from codes, how to construct key exchange protocol from coding theory has been an open problem for decades of years. In 2017, several key exchange protocols from coding theorem are proposed, including CAKE [3] and Ouroboros [13]. They both use MDPC codes and are derived from the McEliece encryption algorithm. And no exception, they are interactive protocols. As a consequence, how to build non-interactive post-quantum key exchange protocols from different hard problems other than supersingular isogenies remains a challenging problem.

We are full of interest and curiosity about how to solve this problem. When it comes to the construction of a code-based cryptosystem, A natural idea is to consider the syndrome decoding problem, which is the most widely used hard problem in code-based cryptosystems. However, we failed because the syndrome decoding problem is a trapdoor problem and is difficult to be applied without the trapdoor, which is exactly the case in a key exchange protocol. Then we turn our attention to the hard problems based on the code space, and code equivalence problem is one of them. Actually, the code equivalence problem has played a significant role in code-based cryptosystems. For example, in the McEliece public-key encryption system [21], the secret key includes a $k \times k$ invertible matrix $\mathbf{S}$, a Goppa code with generator matrix $\mathbf{G}$ and a permutation matrix $\mathbf{P}$, while the public key is $\mathbf{SGP}$. The onewayness from the public key to the secret key is indeed a code permutation equivalence problem. Recently, digital signature schemes from code equivalence problem are proposed as well. In 2020, Biasse *et. al* built a signature called LESS [8] and this scheme was further improved in 2021 by Barenghi *et. al* which is named as LES-FM [2]. LESS is achieved by applying Fiat-Shamir transformation on a code equivalence based identification scheme.

The hardness of code (permutation) equivalence problem was first studied in [24]. After that, lots of works did deeper research on the hardness of code equivalence and its application in cryptography [2, 8, 27]. There exist many works [6, 19, 26] targeting solving the code equivalence problem, but none of them are efficient enough (i.e. in polynomial time). As a result, the cryptosystems whose security relies on the code equivalence problem are viewed as post-quantum cryptosystems.

Unfortunately, the traditional code equivalence problem is still unsuitable to construct key exchange protocol directly. The reason is that the operations between matrices are not commutatively, which is a disadvantageous property for building non-interactive key exchange protocols. Although commutativity is not strictly necessary since the communicating participants may share some common value as is the case for isogenies-based protocol [12], it is not easy to find such properties in coding theory. Biasse *et. al* made a trail in constructing such key exchange protocols from coding [7]. In their construction, Alice and Bob send $\mathbf{G}_a = \mathbf{S}_a\mathbf{G}\mathbf{Q}_a$ and $\mathbf{G}_b = \mathbf{S}_b\mathbf{G}\mathbf{Q}_b$ to each other respectively, where $\mathbf{G}$ is a public matrix. Then Alice calculates $\mathbf{G}'_a = \mathbf{G}_b\mathbf{Q}_a$ and Bob calculates $\mathbf{G}'_b = \mathbf{G}_a\mathbf{Q}_b$. The session key is defined as the weight enumerator function (WEF) of $\mathbf{G}'_a \cap \mathbf{G}$ or $\mathbf{G}'_b \cap \mathbf{G}$. Unfortunately, their work has been withdrawn. As far as we are considered, their session key can be recovered from the public transcripts, i.e. $\mathsf{WEF}(\mathbf{G}'_a \cap \mathbf{G}) = \mathsf{WEF}(\mathbf{G}'_b \cap \mathbf{G}) = \mathsf{WEF}(\mathbf{G}_a \cap \mathbf{G}_b)$. In order to construct a Diffie-Hellman like protocol, we find a sub-problem of code equivalence problem, which plays an important role in our construction.

Contributions. Unlike mostly code-based cryptosystems whose security are built on the well-known syndrome decoding problem, our protocol takes the (sub-problem of the) code equivalence problem as the basic hard problem. The code equivalence problem asks to decide whether the given two code are equivalent or not. Generally speaking, the code equivalence problem is believed to be a candidate to build post-quantum cryptosystems.

We aimed at constructing a non-interactive key exchange protocol, which is usually based on some commutative operations. However, most operations in coding theory do not support commutative laws. Thus, we propose a new hard problem in coding theory called the sub-LE problem. This is a sub-problem of the Linear code Equivalence (LE) problem, and we proved that the sub-LE problem can be reduced to the LE problem. Informally, given two codes with generator matrix $\mathbf{G}$ and $\mathbf{G}'$, the LE problem asks whether there exists an invertible matrix $\mathbf{S}$ and a monomial matrix $\mathbf{Q}$ such that $\mathbf{G}' = \mathbf{S}\mathbf{G}\mathbf{Q}$. If we put a restriction on the monomial matrix $\mathbf{Q}$ such that its action only affects the odd or even columns, then it will be called a sub-LE problem. Namely, in a sub-LE problem, the monomial matrix $\mathbf{Q} = \mathbf{D}\mathbf{P}$ where $\mathbf{D}$ is a diagonal matrix with elements on odd or even position all equal to 1, and $\mathbf{P}$ is a permutation matrix which only permutes the odd or even (on the contrary to the positions of 1 in matrix $\mathbf{D}$) columns. If $\mathbf{P}$ only affects the odd (even) columns, we denote the corresponding monomial matrix as $\mathbf{Q}_{\text{odd}}$ ($\mathbf{Q}_{\text{even}}$).

Intuitively, $\mathbf{Q}_{\text{odd}}$ and $\mathbf{Q}_{\text{even}}$ effect on exactly disjoint columns of $\mathbf{G}$, thus the action of them can be commutative, i.e. $\mathbf{Q}_{\text{odd}}\mathbf{Q}_{\text{even}} = \mathbf{Q}_{\text{even}}\mathbf{Q}_{\text{odd}}$. With this property, a Diffie-Hellmen type key exchange protocol from coding theory can be established easily. The main idea is that the two participants Alice and Bob choose a monomial matrix with different parity respectively, thus the commutative property holds between these two monomial matrices (covered by the

sub-LE problem). This key exchange protocol directly inspiring an El Gamal like public encryption scheme.

Our contributions can be summarizes as follows:

- We construct the first code-based non-interactive key exchange protocol KECE. We proved that KECE is SK-security in the Authenticated Model (AM).
- To complete the security proof of KECE, we defined a new problem in coding theory called sub-LE problem. We proved that the computational version of the sub-LE problem (CLE problem) is as hard as that of the LE problem by reducing the LE problem to the sub-LE problem. As the sub-LE problem provides some useful properties, this new problem can be applied in many areas in building cryptosystems.
- We present a new code-based public-key encryption scheme and proved that it is IND-CPA secure. This new scheme has a smaller key size and does not need any decoding algorithm in the decryption step, which indicates a better performance. The proof of the security also builds on the hardness of the sub-LE problem.

Organization. The remainder of this paper is organized as follows: In Section 2, we recall some preliminaries on coding theory together with the definitions and security properties for key exchange protocol and public-key encryption scheme. In Section 3, our key exchange protocol is presented. To prove its security, we define the sub-LE problem at the beginning of the security proof. The hardness of the sub-LE problem is analysed as last. In Section 4, we show a public key encryption scheme. Based on the sub-DLE assumption, we proved it is IND-CPA secure. Lastly, Section 5 concludes the paper and discusses the future works.

2 Preliminaries

In this section, we present the notions of coding theory that are prerequisite for the following chapters as well as basic knowledge about code-based cryptography.

2.1 Notation and Conventions

Let $\mathbb{N}$ denote the set of natural numbers. If $n \in \mathbb{N}$, then $\{0,1\}^n$ stands for the set of n-bit strings, and $\{0,1\}^*$ for the set of all bit strings. For $a, b \in \mathbb{N}$, $[a;b]$ denotes the set of integers $\{i \in \mathbb{N}, a \leq i \leq b\}$. Let $\mathbb{F}_q$ be the finite field of q elements, and then $\mathbb{F}_q^n$ denotes the set of vectors with length n over $\mathbb{F}_q$, $\mathbb{F}_q^{m\times n}$ denotes the set of $m \times n$ matrices over $\mathbb{F}_q$. The Hamming weight of a vector $\mathbf{v}$ is written as $\mathsf{wt}(\mathbf{v})$. The group of all $k \times k$ invertible matrices over $\mathbb{F}_q$ is denoted as $\mathrm{GL}_k(q)$. We write Σ_n for the group of n-order permutations. For any $\sigma \in \Sigma_n$, it is associated with an $n \times n$ matrix $\mathbf{P}_\sigma$ with $P_{i,j} = 1$ if $\sigma(i) = j$ and $P_{i,j} = 0$ otherwise. This matrix $\mathbf{P}$ is called a permutation matrix, and all the permutation matrices of order n consists set P. Let $\mathrm{Mono}_n(q)$ stands for the set of $n \times n$ monomial matrices with

elements in $\mathbb{F}_q$, i.e. all matrices of form $\mathbf{Q} = \mathbf{DP}$ where $\mathbf{D} = \mathrm{diag}(d_1, \ldots, d_n)$ is a diagonal matrix with $D_{i,i} = d_i$ and $D_{i,j} = 0$ if $i \neq j$, $\mathbf{P}$ is a permutation matrix. We use the symbol $\sim$ to denote the equivalence between two matrices, i.e. $\mathbf{A} \sim \mathbf{B} \iff \exists \mathbf{S} \in \mathrm{GL}_k(q)$ s.t. $\mathbf{A} = \mathbf{SB}$. If S is a set, then $x \leftarrow_{\$} S$ is the assignment to x of an element chosen uniformly at random from S. If $\mathcal{A}$ is an algorithm, then $y \leftarrow \mathcal{A}(x)$ denotes the assignment to y of the input x. We say an algorithm is PPT, if it runs in probabilistic polynomial-time.

2.2 Linear Codes

We now recall some basic definitions for linear codes.

An $[n, k]_q$ linear error-correcting code $\mathcal{C}$ is a linear subspace of a vector space $\mathbb{F}_q^n$, where $\mathbb{F}_q$ denotes the finite field of q elements, and k denotes the dimension of the subspace. The generator matrix for a linear code is a $k \times n$ matrix with rank k which defines a linear mapping from $\mathbb{F}_q^k$ (called the message space) to $\mathbb{F}_q^n$. Namely, the code $\mathcal{C}$ is

$$\mathcal{C} = \mathcal{C}(\mathbf{G}) = \{\mathbf{xG} \mid \mathbf{x} \in \mathbb{F}_q^k\}.$$

For one code space $\mathcal{C}$, there exists more than one generator matrix corresponding to different choice of basis. That is to say, if $\mathbf{G}$ is a generator matrix of code $\mathcal{C}$ and $\mathbf{S} \in \mathrm{GL}_k(q)$, then $\mathbf{G}' = \mathbf{SG}$ is a generator matrix of $\mathcal{C}$ as well. Alternatively, the linear code can be defined by parity check matrix. If $\mathcal{C}$ is the kernel of a matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$, we call $\mathbf{H}$ a parity check matrix of $\mathcal{C}$, i.e.

$$\mathcal{C} = Ker(\mathbf{H}) = \{\mathbf{y} \in \mathbb{F}_q^n \mid \mathbf{Hy} = \mathbf{0}\}.$$

Obviously, the parity check matrix is not unique as the generator matrix. In fact, the parity check matrix $\mathbf{H}$ must satisfies that $\mathbf{GH}^T = \mathbf{0}_{k\times(n-k)}$. Furthermore, the parity check matrix can also be viewed as a generator matrix of another code $\mathcal{C}^\perp = \{\mathbf{xH} \mid \mathbf{x} \in \mathbb{F}_q^{n-k}\} = \{\mathbf{y} \in \mathbb{F}_q^n \mid \mathbf{x} \cdot \mathbf{y}^T = 0, \forall \mathbf{x} \in \mathcal{C}\}$. This code $\mathcal{C}^\perp$ is called the dual code of $\mathcal{C}$ and it is orthogonal of $\mathcal{C}$. We call a vector in $\mathcal{C}$ a codeword.

2.3 Code Equivalence Problem

In this subsection, we introduce the code equivalence problem. The code equivalence problem asks for the equivalent relationship between two codes, and this relationship can be expressed by matrices. We first define the equivalent relationship between codes as follows.

Definition 1 (Permutation Code Equivalence) *Two linear codes $\mathcal{C} \in \mathbb{F}_q^n$ and $\mathcal{C}' \in \mathbb{F}_q^n$ are said to be permutationally equivalent and we denote it by $\mathcal{C} \overset{PE}{\sim} \mathcal{C}'$, if there exists a permutation $\sigma \in \mathrm{P}_n$ that maps $\mathcal{C}$ into $\mathcal{C}'$, i.e. $\mathcal{C}' = \sigma(\mathcal{C}) = \{\sigma(\mathbf{x}), \mathbf{x} \in \mathcal{C}\}$.*

This definition can be generalized using linear isometries. Let $\tau = (\mathbf{v}; \sigma) \in \mathbb{F}_q^{*n} \rtimes \mathcal{P}_n$ be as isometry such that

$$\tau(\mathbf{x}) = (\mathbf{v}; \sigma)(x) = (v_1 x_{\sigma^{-1}(1)}, \ldots, v_n x_{\sigma^{-1}(n)}),$$

then the previous definition can be extended as follows:

Definition 2 (Linear Code Equivalence) *Two linear codes $\mathcal{C} \in \mathbb{F}_q^n$ and $\mathcal{C}' \in \mathbb{F}_q^n$ are said to be linearly equivalent and we denote it by $\mathcal{C} \overset{LE}{\sim} \mathcal{C}'$, if there exists a linear isometry $\tau = (\mathbf{v}; \sigma) \in \mathbb{F}_q^{*n} \rtimes \mathrm{P}_n$ that maps $\mathcal{C}$ onto $\mathcal{C}'$, i.e. $\mathcal{C}' = \tau(\mathcal{C}) = \{\tau(\mathbf{x}), \mathbf{x} \in \mathcal{C}\}$.*

The definitions above can be stated in terms of generator matrices or parity check matrices equivalently. Let $\mathcal{C}$ and $\mathcal{C}'$ be two linear codes with generator matrices $\mathbf{G}$ and $\mathbf{G}'$, then we have

$$\mathcal{C} \overset{PE}{\sim} \mathcal{C}' \Longleftrightarrow \exists (\mathbf{S}, \mathbf{P}) \in \mathrm{GL}_k(q) \times \mathrm{P}_n \text{ s.t. } \mathbf{G}' = \mathbf{SGP},$$
$$\mathcal{C} \overset{LE}{\sim} \mathcal{C}' \Longleftrightarrow \exists (\mathbf{S}, \mathbf{Q}) \in \mathrm{GL}_k(q) \times \mathrm{Mono}_n(q) \text{ s.t. } \mathbf{G}' = \mathbf{SGQ}.$$

The decisional versions of these problems are defined as follows.

Problem 1 (Decisional Permutation Code Equivalence (DPE) Problem) *Given two codes $\mathcal{C}$ and $\mathcal{C}'$ with generator matrices $\mathbf{G}, \mathbf{G}' \in \mathbb{F}_q^{k \times n}$ respectively, determine whether the two codes are permutationally equivalent. Namely, does there exist $\mathbf{S} \in \mathrm{GL}_k(q)$ and a permutation matrix $\mathbf{P}$ such that $\mathbf{G}' = \mathbf{SGP}$.*

Problem 2 (Decisional Linear Code Equivalence (DLE) Problem) *Given two codes $\mathcal{C}$ and $\mathcal{C}'$ with generator matrices $\mathbf{G}, \mathbf{G}' \in \mathbb{F}_q^{k \times n}$ respectively, determine whether the two codes are linearly equivalent. Namely, does there exist $\mathbf{S} \in \mathrm{GL}_k(q)$ and a monomial matrix $\mathbf{Q} \in \mathrm{Mono}_n(q)$ such that $\mathbf{G}' = \mathbf{SGQ}$.*

The two problems above are similar except that different notions of code equivalence are considered. The DPE problem over binary field was introduced in [24]. Then [17] generalized this problem onto any field $\mathbb{F}_q$ and discussed its hardness. Since DPE is a sub-problem of DLE problem, it is clear that the DLE problem is not easier than the DPE problem.

2.4 Model and Security Properties

In this subsection, the model and security definitions for key exchange protocol and public-key encryption scheme are reviewed.

Key Exchange. A key exchange protocol enables the participants to communicate in an open channel, and finally share a secret session key. Usually, a non-interactive key exchange protocol has a paradigm as showed in Fig.1.

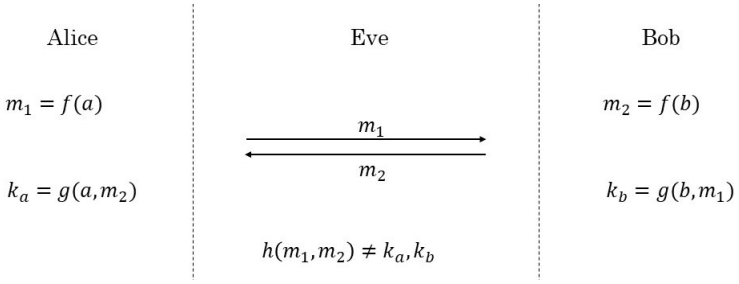

Fig. 1. Non-Interactive Key Exchange Protocol.

Canetti *et. al.* give formally description on the key exchange protocol in [9]. An unauthenticated model (UM) allows the adversary to have full control of the communication channel, i.e. the adversary not only can listen to all the transmitted information, but also can operate the messages, like change some bits or inject information. In an authenticated model (AM), the adversary is restricted to only deliver the messages truly generated by the parties without any change or addition to them. Sometimes we also call this passive attackers. Note that the first point of the following definition is actually the requirement of "correctness", which is a fundamental demand for a key exchange protocol.

Definition 3 (SK secure in the AM) *An adversary $\mathcal{A}$ is a polynomial-time algorithm. Given a value K_b for $b \in \{0,1\}$ where K_0 is the genuine shared key and K_1 is a randomly generated object, $\mathcal{A}$ will output $b' \in \{0,1\}$ to distinguish K_0 or K_1 is given. A KE protocol Π is called SK-secure (SK is the short for Session Key) if the following properties hold for any adversary $\mathcal{A}$ in the AM.*

1. *Protocol Π satisfies the property that if two uncorrupted parties complete matching sessions then they both output the same key;*
2. *the probability that $\mathcal{A}$ guesses correctly the bit b (i.e. outputs $b' = b$) is no more than $1/2 + \mathsf{negl}$.*

There are also some advanced security notions for key exchange protocols, such as under unauthenticated channels or with perfect forward secrecy. A protocol with SK-secure under AM can be transformed into protocols with these advanced security notions by universal transformation [9].

Public Key Encryption. The conception of public-key encryption was introduced in 1976 [14]. A public-key encryption scheme provides two different keys for encryption and decryption respectively. The key used for encryption is public and the key for decryption is kept secret.

Definition 4 *An public-key encryption scheme is a triple of PPT algorithms (***KeyGen***,* **Encrypt***,* **Decrypt***):*

- **KeyGen** *is the key generation algorithm.* **KeyGen**(param) *outputs a pair of keys* (pk, sk)*, where* pk *is the public key for encryption and* sk *is the secret key for decryption.*
- **Encrypt** *is the encryption algorithm. Given a plaintext* m *from a message space* $\mathcal{M}$*, algorithm* **Encrypt** (pk, m) *outputs a ciphertext* c*, which is the encrypted* m *under the public key* pk*.*
- **Decrypt** *is the decryption algorithm.* **Decrypt**(sk, c) *uses the secret key* sk *to decrypt the ciphertext* c*, and outputs a plaintext* m *or a symbol* $\perp$ *representing incorrect decryption. Incorrect decryption can happen for example a valid ciphertext.*

The public parameters param is generated by a **Setup** algorithm which takes the security parameter 1^λ as input. The encryption scheme has to satisfy both Correctness and Security properties. The correctness means that for all $m \in \mathcal{M}$ and all possible (pk, sk) output by **KeyGen**, we have **Decrypt**(**Encrypt**$(m)) = m$.

There are different type of security definitions, and here we use the Indistinguishability under Chosen Plaintext Attack (IND-CPA). In a CPA scenario, the adversary has access to the public key pk, and hence can encrypt any messages she wants. A formal definition of IND-CPA is as follows.

Definition 5 *A public key encryption scheme* (**KeyGen**, **Encrypt**, **Decrypt**) *is IND-CPA secure (or semantically secure) if for all PPT algorithms* $\mathcal{A}$*, the following is negligible:*

$$\left| \Pr \left[\begin{array}{c} (pk, sk) \leftarrow \textbf{KeyGen}(\mathsf{param}); (m_0, m_1) \leftarrow \mathcal{A}(pk); \\ b \leftarrow \{0,1\}; c \leftarrow \textbf{Encrypt}(m_b); b' \leftarrow \mathcal{A}(pk, c) \end{array} : b = b' \right] - \frac{1}{2} \right| \quad (1)$$

IND-CPA secure not only means that the adversary can not obtain the plaintext from the knowledge of the public key and the ciphertext, but also that an adversary can not obtain any partial information about it. Stronger notions of security such as security under non-adaptive chosen-ciphertext attack (CCA1) or security under adaptive chosen-ciphertext attack (CCA2) can be achieved from universal transformations like [16].

3 A Key Exchange Protocol from Coding Theory

In this section, we are going to describe our key exchange protocol. This is a non-interactive key exchange protocol, i.e. the two participants send messages to each other independently. The correctness of the KECE protocol relies on the partly commutatively between the sub-monomial matrices. And the security of the KECE protocol relies on the hardness of the sub-LE problem.

3.1 Presentation of The Protocol

Assume Alice and Bob want to establish a session key through an eavesdropped channel.

– **Setup:** Input security parameter λ, output public parameters $param = (n, k, \mathbf{G})$. Here k, n are integers and $k > n/2$. $\mathbf{G} \in \mathbb{F}_q^{k \times n}$ is a generator matrix of a random linear code.

With the public parameters,

– Alice $\xrightarrow{\mathbf{G}_a}$ Bob:
  1. $\mathbf{Q}_a \leftarrow_\$ \mathrm{OM}_n(q)$, where $\mathrm{OM}_n(q) \subset \mathrm{Mono}_n(q)$ contains all the matrices in $\mathrm{Mono}_n(q)$ such that the corresponding linear isometry keeps the even coordinates unchanged, i.e. $\mathrm{OM}_n(q) = \{\mathbf{M} \in \mathrm{Mono}_n(q) \wedge M_{i,i} = 1 \text{ if } i = 0 \mod 2\}$;
  2. $\mathbf{S}_a \leftarrow_\$ \mathrm{GL}_k(q)$;
  3. $\mathbf{G}_a = \mathbf{S}_a \mathbf{G} \mathbf{Q}_a$.

– Bob $\xrightarrow{\mathbf{G}_b}$ Alice:
  1. $\mathbf{Q}_b \leftarrow_\$ \mathrm{EM}_n(q)$, where $\mathrm{EM}_n(q) = \{\mathbf{M} \in \mathrm{Mono}_n(q) \wedge M_{i,i} = 1 \text{ if } i = 1 \mod 2\}$ contains all the matrices in $\mathrm{Mono}_n(q)$ such that the corresponding linear isometry keeps the odd coordinates unchanged.
  2. $\mathbf{S}_b \leftarrow_\$ \mathrm{GL}_k(q)$;
  3. $\mathbf{G}_b = \mathbf{S}_b \mathbf{G} \mathbf{Q}_b$;
  4. $\mathbf{G}_{kb} = \mathbf{G}_b \mathbf{Q}_a$, and transforms it into a systematic form $\mathbf{G}_{kb} \sim [I \mid K_b]$. Taking K_b as Bob's final session key.

– Alice calculates $\mathbf{G}_{ka} = \mathbf{G}_a \mathbf{Q}_b$, and transforms it into a systematic form $\mathbf{G}_{ka} \sim [I \mid K_a]$. Taking K_a as Alice's final session key.

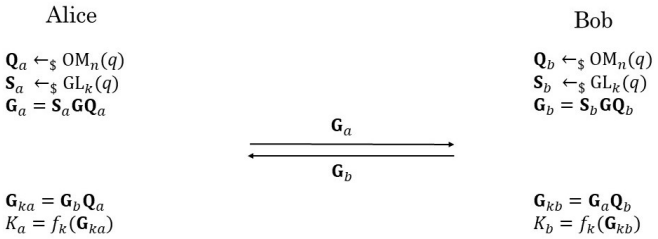

Fig. 2. Our Key Exchange Protocol

As we will see in Theorem 1, if Alice and Bob run the protocol honestly, they will share a same code space. Actually, they can choose any (unique) property of the code as the output session key, e.g. the minimum weight codewords, the weight distribution function. However, this property must be different to the code generated by $\mathbf{G}$. Otherwise the session can be calculated from the public knowledge, and the protocol will be insecure, which is the case in [7]. In our description, we choose to use the systematic generator matrix of this code as the final session key. The reason is that to calculate the systematic form only needs linear operations by Gaussian elimination.

Correctness. We show that if Alice and Bob run the protocol honestly, they will share an identical session key $K_a = K_b$. Since $\mathbf{G}_{ka} = \mathbf{G}_b\mathbf{Q}_a = \mathbf{S}_b\mathbf{G}\mathbf{Q}_b\mathbf{Q}_a$ and $\mathbf{G}_{kb} = \mathbf{G}_a\mathbf{Q}_b = \mathbf{S}_a\mathbf{G}\mathbf{Q}_a\mathbf{Q}_b$, we have $K_a = K_b \iff \mathbf{Q}_b\mathbf{Q}_a = \mathbf{Q}_a\mathbf{Q}_b$. Thus, we only have to prove that $\mathbf{Q}_a$ and $\mathbf{Q}_b$ are commutative to show the correctness.

Lemma 1 $\mathbf{AB} = \mathbf{BA}$ *if* $\mathbf{A} \in \mathrm{OM}_n(q)$ *and* $\mathbf{B} \in \mathrm{EM}_n(q)$.

Proof. $\forall \mathbf{A} \in \mathrm{OM}_n(q) \subset \mathrm{Mono}_n(q)$, there exists a permutation matrix $\mathbf{P}_a$ and a diagonal matrix $\mathbf{D}_a$ such that $\mathbf{A} = \mathbf{D}_a\mathbf{P}_a$. By the definition of $\mathrm{OM}_n(q)$, which is a subset of $\mathrm{Mono}_n(q)$, we have $\mathbf{A}_{i,i} = 1$ if $i = 0 \mod 2$, and this implies $\mathbf{A}_{i,j} = 0$ for all $i = 0 \mod 2, i \neq j$ because there is only one 1 in each column and each row. In more detail, the corresponding linear isometry τ_a has the following property:

For $\mathbf{x} = (x_1, \ldots, x_n)$, $\tau_a(\mathbf{x}) = (v_1 x_{\sigma^{-1}(1)}, x_2, v_3 x_{\sigma^{-1}(3)}, x_4, \ldots, v_{n-1} x_{\sigma^{-1}(n-1)}, x_{n-1})$. Namely, $x_i = \tau_a(\mathbf{x})_i$ for $i = 0 \mod 2$. Similarly, the linear isometry τ_b corresponding to $\mathbf{B} \in \mathrm{EM}_n(q) \subset \mathrm{Mono}_n(q)$ has property that $x_i = \tau_b(\mathbf{x})_i$ for all $i = 1 \mod 2$.

Since $\mathbf{AB} \in \mathrm{Mono}_n(q)$ also corresponds to a linear isometry $\tau_{ab} = \tau_a \circ \tau_b$, we have

$$\tau_{ab}(\mathbf{x})_i = \begin{cases} \tau_b(\mathbf{x})_i, i = 1 \mod 2; \\ \tau_a(\mathbf{x})_i, i = 0 \mod 2. \end{cases} \tag{2}$$

Similarly, for $\mathbf{BA} \in \mathrm{Mono}_n(q)$ which corresponds to a linear isometry $\tau_{ba} = \tau_b \circ \tau_a$, Equation.(2) holds as well.

As a result, $\tau_{ba} = \tau_{ab}$, and thus $\mathbf{AB} = \mathbf{BA}$. □

Theorem 1 (Correctness). *The key exchange protocol KECE described in Section 3.1 is correct.*

Proof. From Lemma 1, for $\mathbf{Q}_a \in \mathrm{OM}_n(q)$ and $\mathbf{Q}_b \in \mathrm{EM}_n(q)$, we have $\mathbf{Q}_a\mathbf{Q}_b = \mathbf{Q}_b\mathbf{Q}_a$. Since $\mathbf{S}_a, \mathbf{S}_b \in \mathrm{GL}_k(q)$, there exists $\mathbf{A} \in \mathrm{GL}_k(q)$ such that $\mathbf{S}_b = \mathbf{A}\mathbf{S}_a$, e.g. $\mathbf{A} = \mathbf{S}_b\mathbf{S}_1^{-1}$. Hence we have

$$\begin{aligned} \mathbf{G}_{ka} &= \mathbf{G}_a\mathbf{Q}_b = \mathbf{S}_a\mathbf{G}\mathbf{Q}_a\mathbf{Q}_b = \mathbf{S}_a\mathbf{G}\mathbf{Q}_b\mathbf{Q}_a \\ &\sim \mathbf{A}\mathbf{S}_a\mathbf{G}\mathbf{Q}_b\mathbf{Q}_a \\ &= \mathbf{S}_b\mathbf{G}\mathbf{Q}_b\mathbf{Q}_a = \mathbf{G}_b\mathbf{Q}_a = \mathbf{G}_{kb} \end{aligned}$$

Thus $\mathbf{G}_{ka}$ and $\mathbf{G}_{kb}$ generate a same code space, and they share the same systematic form. □

3.2 Security Analysis

3.2.1 Sub-LE Problem

Before given the security proof, we first define a new problem named sub-LE problem, which is a sub-problem of the LE problem. Informally, the decisional

version of sub-LE problem (i.e. sub-DLE problem) asks whether the two given codes (given by their generator matrices $\mathbf{G}$ and $\mathbf{G}'$ respectively) are linearly equivalent, i,e, $\mathbf{G}' = \mathbf{SGQ}$. The computational version sub-CLE problem requires to give a pair of matrices $\mathbf{S}, \mathbf{Q}$ if the answer of the sub-DLE problem is yes. The difference between the sub-DLE problem and the DLE problem is that in the DLE problem, we choose the matrix $\mathbf{Q} \in \mathrm{Mono}_n(q)$ and in the sub-DLE problem we choose $\mathbf{Q} \in \text{sub-M}_n(q)$. We define the set of sub-$\mathrm{M}_n(q)$ as follows:

Definition 6 *Define* $\mathrm{OM}_n(q) = \{\mathbf{Q} \in \mathrm{Mono}_n(q) \wedge Q_{i,i} = 1 \text{ if } i = 0 \mod 2\}$ *and* $\mathrm{EM}_n(q) = \{\mathbf{Q} \in \mathrm{Mono}_n(q) \wedge Q_{i,i} = 1 \text{ if } i = 1 \mod 2\}$. *Namely,* $\mathbf{Q} \in \mathrm{OM}_n(q)$ *(resp.* $\mathbf{Q} \in \mathrm{EM}_n(q)$*) only affects the odd (resp. even) columns of* $\mathbf{G}$ *when do the right multiplication. Then sub-*$M_n(q) = \mathrm{OM}_n(q) \cup \mathrm{EM}_n(q) = \{\mathbf{Q} \mid \mathbf{Q} \in \mathrm{Mono}_n(q) \wedge (Q_{i,i} = 1 \text{ for all } i = 0 \mod 2 \vee Q_{i,i} = 0 \text{ for all } i = 1 \mod 2)\}$.

The following figure Fig.3 shows the relationship between $\mathrm{Mono}_n(q)$ and sub-$\mathrm{M}_n(q)$. Then we can define the sub-LE problem.

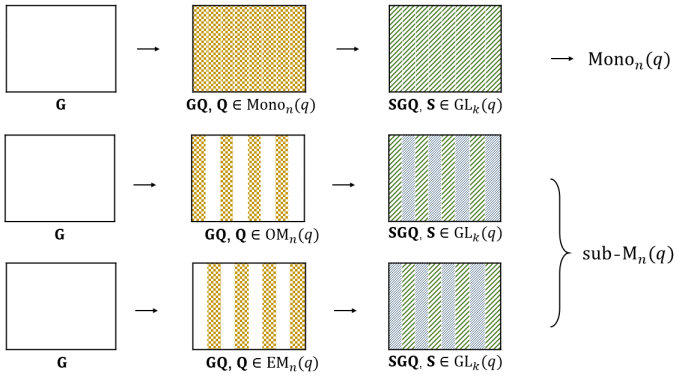

Fig. 3. The relationship between $\mathrm{Mono}_n(q)$ and sub-$\mathrm{M}_n(q)$.

Problem 3 (sub-CLE Problem) *Given two codes* $\mathcal{C}$ *and* $\mathcal{C}'$ *with generator matrices* $\mathbf{G}, \mathbf{G}' \in \mathbb{F}_q^{k \times n}$ *respectively, the sub-CLE*(n, k, q) *problem asks for a pair of matrices* $\mathbf{S} \in \mathrm{GL}_k(q), \mathbf{Q} \in \text{sub}-M_n$ *such that* $\mathbf{G}' = \mathbf{SGQ}$.

Before given a decisional version of the sub-CLE problem, we define the R-sub-LE distribution at first.

Definition 7 (R-sub-LE distribution) *We say a pair of matrices* $(\mathbf{G}_1, \mathbf{G}_2)$ *is in R-sub-LE distribution, if the sub-matrices of* $\mathbf{G}_1$ *and* $\mathbf{G}_2$ *that consist by their odd (or even) columns are equivalent.*

Namely, for matrices $\mathbf{G}_1 = [\mathbf{g}_{1,1}, \mathbf{g}_{1,2}, \cdots, \mathbf{g}_{1,n}]$ *and* $\mathbf{G}_2 = [\mathbf{g}_{2,1}, \mathbf{g}_{2,2}, \cdots, \mathbf{g}_{2,n}]$, *if* $\mathbf{G}_{1,\text{odd}} \sim \mathbf{G}_{2,\text{odd}}$ *or* $\mathbf{G}_{1,\text{even}} \sim \mathbf{G}_{2,\text{even}}$, *then* $(\mathbf{G}_1, \mathbf{G}_2)$ *are in R-sub-LE distribution. Here*

$$\begin{aligned}
\mathbf{G}_{1,\text{odd}} &= \left[\mathbf{g}_{1,1}, \mathbf{g}_{1,3}, \cdots, \mathbf{g}_{1,2\lceil n/2\rceil-1}\right]; \\
\mathbf{G}_{2,\text{odd}} &= \left[\mathbf{g}_{2,1}, \mathbf{g}_{2,3}, \cdots, \mathbf{g}_{2,2\lceil n/2\rceil-1}\right]; \\
\mathbf{G}_{1,\text{even}} &= \left[\mathbf{g}_{1,2}, \mathbf{g}_{1,4}, \cdots, \mathbf{g}_{1,2\lfloor n/2\rfloor}\right]; \\
\mathbf{G}_{2,\text{even}} &= \left[\mathbf{g}_{2,2}, \mathbf{g}_{2,4}, \cdots, \mathbf{g}_{2,2\lfloor n/2\rfloor}\right].
\end{aligned}$$

We have to define this distribution because two random matrices can be distinguished from two sub-linearly equivalent codes easily. The hardness of distinguishing whether the given two codes are linearly equivalent mainly relies on the action of the monomial matrix $\mathbf{Q}$. Since $\mathbf{Q} \in \text{sub-M}_n(q)$ only affects the odd or even columns, the unchanged columns of two sub-linearly equivalent matrices must be equivalent. This distribution can be sampled by first choose a full rank random matrix $\mathbf{G}_1$, and then replacing the odd (or even) columns of $\mathbf{G}_1$ into random vectors.

Problem 4 (sub-DLE Problem) *Given two codes* $\mathcal{C}$ *and* $\mathcal{C}'$ *with generator matrices* $\mathbf{G}, \mathbf{G}' \in \mathbb{F}_q^{k\times n}$ *respectively. Here the given two generator matrices consist a matrix pair* $(\mathbf{G}, \mathbf{G}')$ *that is in R-sub-LE distribution. Then the sub-DLE*(n, k, q) *problem asks to determine whether there exists* $\mathbf{S} \in \text{GL}_k(q)$ *and a monomial matrix* $\mathbf{Q} \in$ *sub-*$M_n(q)$ *such that* $\mathbf{G}' = \mathbf{SGQ}$.

Assumption 1 *There is no PPT algorithm that can solve the sub-DLE problem.*

3.2.2 Security Proof of Our Key Exchange Protocol

Now we prove the SK-secure of our KECE protocol under AM.

Theorem 1 *The KECE protocol is SK-secure under the AM, if the sub-LE problem is hard.*

Proof. The requirement of correctness has been proved in Section 3.2. In the following proof we focus on the second statement in Def.3.

If there exists an adversary $\mathcal{A}$ who can break the KECE scheme with non-negligible probability, then we can build a distinguisher $\mathcal{D}$ to break the sub-DLE assumption, i.e. solve the sub-DLE problem. Given a pair $(\mathbf{G}_1, \mathbf{G}_2)$, the distinguisher $\mathcal{D}$ invokes $\mathcal{A}$ in a series of games. The first game **Game 0** is the real game which the adversary gets the real session key K, while the last game **Game 4** outputs a uniformly random K to the adversary.

An outline of the proof is illustrated in Fig.4.

- **Game 0**: This is the real game between the protocol challenger and the adversary $\mathcal{A}$. In this game, the adversary obtains the transcripts of $\mathbf{G}_a, \mathbf{G}_b$ and the real session key K between Alice and Bob. Then $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$.

- **Game 1**: In this game, $\mathcal{D}$ set the public matrix as $\mathbf{G} = \mathbf{G}_1$. Since our scheme sets the public matrix as the generator matrix of a random code, i.e. the public matrix can be viewed as a random full rank matrix, this game is indistinguishable with **Game 0**.
- **Game 2**: In this game, $\mathcal{D}$ simulates Alice and generates her $\mathbf{G}_a$ by change the odd columns of $\mathbf{G}_1$ into random vectors and then multiple $\mathbf{S}_a \in \mathrm{GL}_n$. In Lemma 2 we prove that under the sub-DLE assumption, the views in **Game 1** and **Game 2** are computationally indistinguishable for PPT adversaries.
- **Game 3**: In this game, Bob generates his $\mathbf{G}_b$ by change the odd columns of $\mathbf{G}_1$ into random vectors and then multiple $\mathbf{S}_b \in \mathrm{GL}_n$. In Lemma 3 we prove that under the sub-DLE assumption, the views in **Game 2** and **Game 3** are computationally indistinguishable for PPT adversaries.
- **Game 4**: In this game, $\mathcal{D}$ sets the output agreement key as $\mathbf{G} = \mathbf{G}_2$. In Lemma 4 we prove that under the sub-DLE assumption, the views in **Game 3** and **Game 4** are computationally indistinguishable for PPT adversaries.

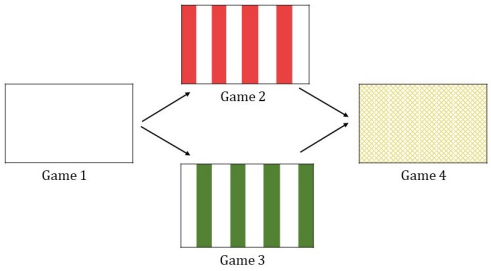

Fig. 4. An outline of the proof.

To summarize, when **Game 4** terminates, the adversary $\mathcal{A}$ outputs b' which indicates that $\mathbf{G}_2$ is a random random object for $b' = 0$ and is the real session key for $b' = 1$. Thus the dinstinguisher $\mathcal{D}$ can determine that the given matrices pair $(\mathbf{G}_1, \mathbf{G}_2)$ are linearly equivalent if $\mathcal{A}$ outputs 1. As a result, the advantage for $\mathcal{D}$ to win a sub-DLE problem is

$$Adv_{\mathcal{D}} = Succ_{\mathcal{A}},$$

which means that if the advantages for any PPT adversaries $\mathcal{A}$ to break the SK secure in the AM of KECE is non-negligible, then we can build an algorithm $\mathcal{D}$ to solve the sub-DLE problem with non-negligible probability. □

Lemma 2 ***Game 1*** *and* ***Game 2*** *are indistinguishable for any PPT adversary, if the sub-DLE assumption holds.*

Proof. The only difference between **Game 1** and **Game 2** is the generation of the matrix $\mathbf{G}_a$. Thus the indistinguishability between **Game 1** and **Game 2** is

the indistinguishability between two $\mathbf{G}_a$. In fact, if there exists an adversary $\mathcal{A}$ distinguishes **Game 1** and **Game 2** with non-negligible advantage, we can sole the sub-DLE problem as follows: Given a sub-DLE instance $\mathbf{G}_1, \mathbf{G}_2$, we set the public matrix as $\mathbf{G}_1$ and the matrix in **Game 2** as $\mathbf{G}_a = \mathbf{G}_2$. Then invoke $\mathcal{A}$. If $\mathcal{A}$ output '**Game 1**', then we say the instance is sub-linearly equivalent. And if $\mathcal{A}$ output '**Game 2**', then we say the instance is not sub-linearly equivalent. □

Lemma 3 ***Game 2*** *and* ***Game 3*** *are indistinguishable for any PPT adversary, if the sub-DLE assumption holds.*

Proof. The proof of this lemma is similar as the proof of lemma 2. □

Lemma 4 ***Game 3*** *and* ***Game 4*** *are indistinguishable for any PPT adversary, if the sub-DLE assumption holds.*

Proof. In the real game, the session key is sub-linearly equivalent with $\mathbf{G}_a$ and $\mathbf{G}_b$, and is linearly equivalent with the public matrix $\mathbf{G}$. Hence the indistinguishability between **Game 3** and **Game 4** is the indistinguishability between $\mathbf{G}_b\mathbf{Q}_a$ and $\mathbf{G}_2$. In fact, if there exists an adversary $\mathcal{A}$ distinguishes **Game 3** and **Game 4** with non-negligible advantage, we can sole the sub-DLE problem as follows: Given a sub-DLE instance $\mathbf{G}'_1, \mathbf{G}'_2$,

1. set the public matrix as $\mathbf{G}$ by change the even columns of $\mathbf{G}'_1$ into random vectors and then multiple by $\mathbf{S} \in \mathrm{GL}_k(q)$;
2. Set $\mathbf{G}_b = \mathbf{G}_1$ in **Game 3**;
3. Set the output session key in **Game 4** as $\mathbf{G}_2 = \mathbf{G}'_2$.
4. Invoke $\mathcal{A}$(**Game 3**, **Game 4**).

If $\mathcal{A}$ output '**Game 3**', then we say the instance is sub-linearly equivalent. And if $\mathcal{A}$ output '**Game 4**', then we say the instance is not sub-linearly equivalent. □

3.2.3 Hardness of The Sub-LE Problem

Now let us discuss the hardness of the sub-DLE problem. An intuition is that if the sub-DLE problem between matrices $\mathbf{G}$ and $\mathbf{G}'$ has a 'YES' answer, then the matrix $\mathbf{G}$ only differs from $\mathbf{G}'$ in the odd or even columns. As a consequence, their identical columns may leak some information about the linear equivalence relationship. To observe this problem deeply, we do another permutation denoted by matrix $\mathbf{P}$ on $\mathbf{G}'$ such that the action of $\mathbf{Q}$ only affects the first $n/2$ columns of $\mathbf{SG}$, and depart this part into two $k/2 \times n/2$ matrices as showed in Fig.5. Then solving the sub-DLE problem can be depart in to two questions:

1. Solve $\mathbf{S}$ by solving the linear system $\mathbf{SG}_2 = \mathbf{G}'_2$, which needs to solve k^2 variables to determine $\mathbf{S}$. Since there are $nk/2$ equations in the linear system, the solution will contain $(2k-n)k/2$ free variants. Each of these free variants can be evaluated in $\mathbb{F}_q$ and thus the solution space has $q^{(2k-n)k/2}$ elements and only one of them is the correct $\mathbf{S}$. As a result, it needs $q^{(2k-n)k/2}$ trails to determine the matrix $\mathbf{S}$.

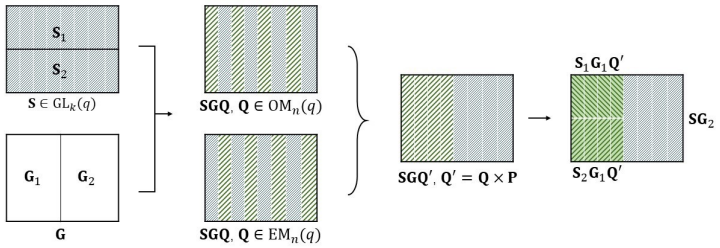

Fig. 5. Depart the matrix $\mathbf{M} \in \text{sub-M}_n(q)$ into three sub-matrices.

2. Solving the DLE problem between $\mathbf{S}_i\mathbf{G}_i\mathbf{Q}'$ and $\mathbf{G}_i$ for $i \in \{1, 2\}$.

If we depart $\widetilde{\mathbf{G}} = \mathbf{SGQ}$ into two parts, i.e. the odd columns as $\widetilde{\mathbf{G}}_1$ and the even columns as $\widetilde{\mathbf{G}}_2$, an observation is that $\widetilde{\mathbf{G}}_2$ does not contain any information of $\mathbf{Q}$, and may let out the matrix $\mathbf{S}$. To prevent this leakage, we put a limitation on the parameters n, k such that recover $\mathbf{S}$ by solving the linear system invoke $\widetilde{\mathbf{G}}_2$ is difficult. Moreover, the length of $\widetilde{\mathbf{G}}_1$ is cut off from $\widetilde{\mathbf{G}}$, and the entropy of $\mathbf{Q}$ is decreased from the LE problem. As a result, we have to choose lager parameter size to ensure the hardness of the sub-DLE problem.

Based on these observations, we can make a reduction from the sub-DLE problem to the DLE problem. Namely, if there exists an algorithm which can solve the sub-DLE problem between two given $[2n, 2k]$ codes with non-negligible advantage, we can call it to solve the DLE problem between two $[n, k]$ codes. We prove the following theorem to support the sub-DLE assumption.

Theorem 2 *If there exist a PPT algorithm $\mathcal{A}$ can solve the sub-LE problem between two $[2n, 2k]$ code, we can build a PPT algorithm $\mathcal{B}$ to solve the LE problem.*

Proof. Suppose $\mathcal{A}$ is an algorithm can solve the sub-LE problem. Given two codes (described by their generator matrices) to $\mathcal{A}$, it will output $b = 1$ if the two codes are sub-LE and $b = 0$ otherwise. If $b = 1$, $\mathcal{A}$ will also output a pair of matrices $\mathbf{S}^*, \mathbf{Q}^*$ such that $\mathbf{G}_2 = \mathbf{S}^*\mathbf{G}_1\mathbf{Q}^*$.

Given a LE instance $\mathbf{G}_1, \mathbf{G}_2 \in \mathbb{F}_q^{k\times n}$, we construct two matrix $\widetilde{\mathbf{G}}_1, \widetilde{\mathbf{G}}_2 \in \mathbb{F}_q^{2k\times 2n}$, and then use the algorithm $\mathcal{A}$ to solve the LE instance. A sketch of the reduction algorithm is illustrated in Fig.6.

More precisely, we depart the generator matrix $\widetilde{\mathbf{G}}_1, \widetilde{\mathbf{G}}_2$ into four $n \times k$ sub-matrices at first. Then put the given DLE instance $\mathbf{G}_1$ and a random full rank matrix $\mathbf{R}$ on the main diagonal sub-matrices, i.e.

$$\widetilde{\mathbf{G}}_1 = \begin{bmatrix} \mathbf{G}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_1 \end{bmatrix}.$$

Afterwards we permute them by a permutation

$$\sigma = \begin{cases} i/2 + n, & i = 0 \mod 2 \\ (i+1)/2, & i = 1 \mod 2 \end{cases}$$

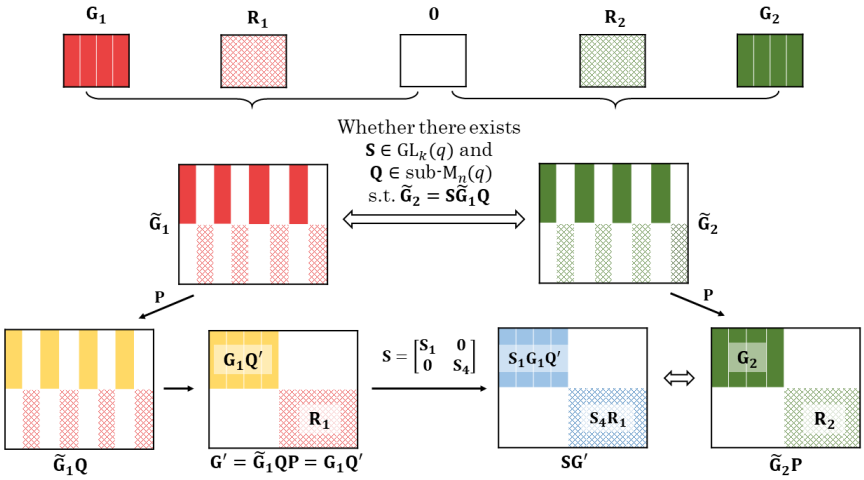

Fig. 6. A sketch of the reduction.

$\widetilde{\mathbf{G}}_2$ is constructed by the same way except that $\mathbf{R}_1$ is replaced by $\mathbf{R}_2 = \mathbf{S}_2\mathbf{R}_1$ where $\mathbf{S}_2$ is a full rank $k \times k$ matrix. Thus if there exists

$$\widetilde{\mathbf{S}} = \begin{bmatrix} \mathbf{S}_1 & \mathbf{S}_3 \\ \mathbf{S}_4 & \mathbf{S}_2 \end{bmatrix}$$

and $\widetilde{\mathbf{Q}}_{\text{odd}}$ or $\widetilde{\mathbf{Q}}_{\text{even}}$ such that $\widetilde{\mathbf{G}}_2 = \widetilde{\mathbf{S}}\widetilde{\mathbf{G}}_1\widetilde{\mathbf{Q}}$, we have $\mathbf{G}_2 = \mathbf{S}_1\mathbf{G}_1\mathbf{Q}$. Here $\mathbf{Q}$ denotes the first sub-matrix in $\sigma(\widetilde{\mathbf{Q}})$, i.e.

$$\widetilde{\mathbf{Q}} = \sigma^{-1}(\mathbf{D}) \cdot \sigma^{-1}(\mathbf{P}) \begin{bmatrix} \mathbf{Q}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix},$$

where $\mathbf{I}$ is an identity matrix. We illustrate the full reduction algorithm in Algorithm 1 as follows:

As a result, our algorithm will output $b \in \{0, 1\}$ to answer the DLE problem between $\mathbf{G}_1$ and $\mathbf{G}_2$. A pair of matrices $\mathbf{S}, \mathbf{Q}$ will also be output iff $b = 1$ such that $\mathbf{G}_2 = \mathbf{S}\mathbf{G}_1\mathbf{Q}$. Hence the theorem holds. □

4 A New Code-based Public-key Encryption Scheme

4.1 Presentation of The Scheme

- **Setup:** Input the security parameter λ, output public parameter $\mathsf{param} = (q, n, k, \mathbf{G}, \mathcal{H})$, where $\mathbf{G}$ is a full rank $n \times k$ matrix over $\mathbb{F}_q$, and $\mathcal{H}$ is a secure hash function which output a string $s \in \{0, 1\}^{512}$.
- **KeyGen**: Input the public parameter param, output the secret key sk and corresponding public key pk.

Algorithm 1: Reduction from LE problem to sub-LE problem

Input: $\mathbf{G}_1, \mathbf{G}_2, \mathcal{A}$.
/* $\mathbf{G}_1$ and $\mathbf{G}_2$ is a given LE instance; $\mathcal{A}$ is an algorithm which can solve the sub-LE problem. */
Output: $b \in \{0, 1\}$.
If $b = 1$, also outputs matrices $\mathbf{S}$ and $\mathbf{Q}$ such that $\mathbf{G}_2 = \mathbf{SGQ}$.

$\mathbf{R}_1 \leftarrow \text{FullRankRandomMatrix}(\mathbb{F}_q, k, n)$;
$\mathbf{S}_2 = \text{GL}_k(q)$;
$\mathbf{R}_2 = \mathbf{S}_2\mathbf{R}_1$;
$\widetilde{\mathbf{G}}_1 \leftarrow \text{ZeroMatrix}(\mathbb{F}_q, 2k, 2n)$;
$\widetilde{\mathbf{G}}_2 \leftarrow \text{ZeroMatrix}(\mathbb{F}_q, 2k, 2n)$;
for $i = 1$ *to* $2k$ **do**
    $r = (i + (i \mod 2))/2$;
    for $i = 1$ *to* n **do**
        if $i \mod 2 == 1$ **then**
            $\widetilde{G}_1[j][i] = G_1[j][r]$;
            $\widetilde{G}_2[j][i] = G_2[j][r]$;
        else
            $\widetilde{G}_1[j+n][i] = R_1[j][r]$;
            $\widetilde{G}_2[j+n][i] = R_2[j][r]$;
$b \leftarrow \mathcal{A}(\widetilde{\mathbf{G}}_1, \widetilde{\mathbf{G}}_2)$;
if $b == 1$ **then**
    /* In this case $\mathcal{A}$ outputs $\widetilde{\mathbf{S}}$ and $\widetilde{\mathbf{Q}}$ s.t. $\widetilde{\mathbf{G}}_2 = \widetilde{\mathbf{S}}\widetilde{\mathbf{G}}_1\widetilde{\mathbf{Q}}$. */
    $\sigma = \text{PermutationSequence}(2n)$;
    $\mathbf{S}_1 = \text{ZeroMatrix}(\mathbb{F}_q, k, k)$;
    for $i = 1$ *to* n **do**
        $\sigma(2 * i - 1) = i$;
        $\sigma(2 * i) = n + i$;
    $\mathbf{P} = \text{PermutationMatrix}(\sigma)$;
    $\mathbf{Q} = \widetilde{\mathbf{Q}}\mathbf{P}$;
    for $i = 1$ *to* k **do**
        for $j = 1$ *to* k **do**
            $S[i][j] = \widetilde{S}[i][j]$;
Return $b, \mathbf{S}, \mathbf{Q}$;

1. $\mathbf{S} \leftarrow_{\$} \mathrm{GL}_k(q)$;
2. $\mathbf{Q} \leftarrow_{\$} \mathrm{OM}_n(q)$;
3. $\mathbf{G}_p = \mathbf{SGQ}$;
4. $sk = \mathbf{Q}$, $pk = (\mathbf{G}, \mathbf{G}_p)$;

– **Encrypt**: Input the public parameter param, the public key pk, and a plaintext m, output the ciphertext $\mathbf{c}$.
1. $\mathbf{S}_e \leftarrow_{\$} \mathrm{GL}_k(q)$;
2. $\mathbf{Q}_e \leftarrow_{\$} \mathrm{EM}_n(q)$;
3. $c_1 = \mathbf{S}_e\mathbf{GQ}_e$;
4. $\mathbf{G}_E = \mathbf{G}_p\mathbf{Q}_e$;
5. $[I \mid \mathbf{G}_e] = \mathsf{GE}(\mathbf{G}_E)$;
   $\backslash *$ here GE denotes the Gaussian elimination algorithm, which takes a matrix as input and output a systematic matrix $*\backslash$
6. $c_2 = \mathcal{H}(\mathbf{G}_e) + m$;
7. $\mathbf{c} = (c_1, c_2)$;

– **Decrypt**: Input the public parameter param, the secret key sk, and a ciphertext $\mathbf{c} = (c_1, c_2)$, output the plaintext m.
1. $\mathbf{G}_D = c_1 \cdot sk = \mathbf{S}_e\mathbf{GQ}_e\mathbf{Q}$;
2. $[I \mid \mathbf{G}_d] = \mathsf{GE}(\mathbf{G}_D)$;
3. $m = c_2 - \mathcal{H}(\mathbf{G}_d)$;

Correctness. It is obvious that $\mathbf{Decrypt}\left(\mathbf{Encrypt}(pk, m)\right) = (\mathbf{G}_e + m) - \mathbf{G}_d = m + (\mathbf{G}_e - \mathbf{G}_d)$. Hence the correctness holds that if $\mathbf{G}_e = \mathbf{G}_d$, i.e. there exists $\hat{\mathbf{S}} \in \mathrm{GL}_k(q)$ such that $\mathbf{G}_E = \hat{\mathbf{S}}\mathbf{G}_D$.

$$\mathbf{G}_E = \mathbf{G}_p\mathbf{Q}_e = \mathbf{SGQQ}_e \rightarrow \mathbf{GQQ}_e \rightarrow \mathbf{S}_e\mathbf{GQQ}_e = \mathbf{S}_e\mathbf{GQ}_e\mathbf{Q}$$

Thus we have $\mathbf{G}_E = \mathbf{S}_e\mathbf{S}^{-1}\mathbf{G}_D$, i.e. $\hat{\mathbf{S}} = \mathbf{S}_e\mathbf{S}^{-1}$, which proves that the correctness holds.

4.2 Security Proof

Theorem 3 *The public key encryption scheme described above is IND-CPA under the sub-DLE assumption.*

Proof. If there exists a PPT adversary $\mathcal{A}$ which breaks the IND-CPA secure of the underline scheme, we can build an algorithm $\mathcal{B}$ which solves the sub-DLE problem. To see this, $\mathcal{B}$ build a series of games as follows:

– **Game 1**: This is the real IND-CPA game as described in definition. In this game, the plaintext m_0 is encrypted.

– **Game 2**: This game is the same as **Game 1** except that we forget the secret key, i.e. we put random vectors on the odd columns of the public matrix $\mathbf{G}$. We claim that **Game 1** and **Game 2** are indistinguishable under $\mathcal{A}$'s view. More precisely, if $\mathcal{A}$ can distinguish this two games, then for a given sub-DLE

instance $\widetilde{\mathbf{G}}_1, \widetilde{\mathbf{G}}_2$, $\mathcal{B}$ can invoke $\mathcal{A}$ by setting the public matrix as $\widetilde{\mathbf{G}}_1$, and the public key in **Game 2** as $\widetilde{\mathbf{G}}_2$. If $\mathcal{A}$ outputs '**Game 1**' then $\mathcal{B}$ decides $\widetilde{\mathbf{G}}_1$ and $\widetilde{\mathbf{G}}_2$ are sub-DLE, otherwise are not. In fact, since the only difference between **Game 1** and **Game 2** is that the public key is sub- linearly equivalent with the public matrix in **Game 1**, and in not sub-linearly equivalent in **Game 2**, we have the advantage for $\mathcal{B}$ to solve the sub-DLE instance is equal to the advantage for $\mathcal{A}$ to distinguish the two games.

– **Game 3**: In this game, we start by taking the first part of the ciphertext as another matrix by changing the even columns of $\mathbf{G}_e$ as random vectors, the other parts remain the same as **Game 2**. As a result, $\mathbf{G}_e$ has no relationship with $\mathbf{Q}_e$ now.

  Then we claim that **Game 2** and **Game 3** are indistinguishable under $\mathcal{A}$'s view. In more detail, if $\mathcal{A}$ can distinguish this two games, then for a given sub-DLE instance $\widetilde{\mathbf{G}}_1, \widetilde{\mathbf{G}}_2$, $\mathcal{B}$ can invoke $\mathcal{A}$ by setting the public matrix as $\widetilde{\mathbf{G}}_1$, and c_1 in **Game 3** as $\widetilde{\mathbf{G}}_2$. If $\mathcal{A}$ outputs '**Game 2**' then $\mathcal{B}$ decides $\widetilde{\mathbf{G}}_1$ and $\widetilde{\mathbf{G}}_2$ are sub-DLE, otherwise are not. This is because that the only difference between **Game 2** and **Game 3** is that c_1 is sub linearly equivalent with the public matrix in **Game 2**, and is not sub linearly equivalent in **Game 3**, we have the advantage for $\mathcal{B}$ to solve the sub-DLE instance is equal to the advantage for $\mathcal{A}$ to distinguish the two games.

– **Game 4**: This game is the same as **Game 3**, except that we take $\mathbf{G}_E$ as a random matrix, and this will lead to the second part of the ciphertext c_2 as a random matrix since $c_2 = \mathbf{G}_E + m_0$.

  As before, we show that **Game 3** and **Game 4** are indistinguishable under $\mathcal{A}$'s view. More precisely, in **Game 3** $\mathbf{G}_E$ is in fact linearly equivalent to the public matrix $\mathbf{G}$, and sub linearly equivalent to $\mathbf{G}_p$ and $\mathbf{G}_e$. In **Game 4**, given a ciphertext $\mathbf{c}$, the adversary $\mathcal{A}$ can calculate $\mathbf{G}' = c_2 - m_0$. If the encrypted plaintext is m_0, $\mathbf{G}'$ will be sub linearly equivalent to both $\mathbf{G}_p$ and $\mathbf{G}_e$. Otherwise, $\mathbf{G}'$ is not sub linearly equivalent to them. Thus if $\mathcal{A}$ can distinguish this two games, then for a given sub-DLE instance $\widetilde{\mathbf{G}}_1, \widetilde{\mathbf{G}}_2$, $\mathcal{B}$ can invoke $\mathcal{A}$ by setting the public key $\mathbf{G}_p$ as $\widetilde{\mathbf{G}}_1$, and c_2 in **Game 4** as $\widetilde{\mathbf{G}}_2 + m_0$. If $\mathcal{A}$ outputs '**Game 3**' then $\mathcal{B}$ decides the sub-DLE problem between $\widetilde{\mathbf{G}}_1$ and $\widetilde{\mathbf{G}}_2$ is true, otherwise is false. We have the advantage for $\mathcal{B}$ to solve the sub-DLE instance is equal to the advantage for $\mathcal{A}$ to distinguish the two games.

– **Game 5**: Now we begin to encrypt the other plaintext m_1, and other parts remain the same as **Game 4**.

  Since $\mathbf{G}_E$ is a random matrix, the second part of the ciphertext c_2 is uniformly distributed over $\mathbb{F}_q^{(n-k)\times k}$. As a result, **Game 4** and **Game 5** is indistinguishable from the information theory knowledge.

– **Game 6**: This game is the same as **Game 5** except that $\mathbf{G}_E$ is generated honestly from $\mathbf{G}_p\mathbf{Q}_e$. The indistinguishability between **Game 5** and **Game 6** is similar to that between **Game 3** and **Game 4**.

- **Game 7**: This game is the same as **Game 6** except that c_1 is generated honestly from the public matrix $\mathbf{G}$. The indistinguishability between **Game 6** and **Game 7** is similar to that between **Game 2** and **Game 3**.
- **Game 8**: This game is the same as **Game 7** except that the public key $\mathbf{G}_p$ is generated honestly from the public matrix $\mathbf{G}$. The indistinguishability between **Game 7** and **Game 8** is similar to that between **Game 1** and **Game 2**. Moreover, **Game 8** is the real IND-CPA game in which the plaintext m_1 is encrypted.

To summarize, every two adjacent games in among the above eight games are indistinguishable, and thus **Game 1** and **Game 8** are indistinguishable. Namely, the real IND-CPA games in which the plaintext m_0 and m_1 are encrypted respectively are indistinguishable, i.e. the scheme is IND-CPA secure if the sub-DLE assumption holds. □

Here we remove the effect caused by the hash function in the second part of the ciphertext. The invoke of a secure hash function will ensure a better randomness of c_2, and thus the scheme will be more secure.

4.3 Performance

The main attacks toward the code equivalence problem includes Leon's attack and the The Support Splitting Algorithm (SSA) attack. We give a glance of them and then present a set of parameters.

Leon's Attack [19]. Leon [19] proposed an algorithm for computing the automorphism group of an error correcting code. The automorphism of a code $\mathcal{C}$ is a monomial permutation σ such that $\sigma(\mathcal{C}) = \mathcal{C}$. Leon's attack [19] consists simply of analysing the actions of the algorithm on the subset of codewords with fixed weight w. Once such a set is computed, it gets partitioned into smaller subsets, which are then used to retrieve the permutation mapping one code to the other. The partitioning phase has very low complexity, while finding all codewords of weight w is the actual bottleneck of the algorithm. Usually w is set as the minimum distance of the code. And for random codes, this can be estimated with the GV bound. If this set does not have sufficient structures, then w is slightly increased. We now briefly describe how the codeword enumeration can be performed. Let $\mathbf{G}$ be the generator matrix of a code $\mathcal{C}$ of an $[n, k]$ code with systematic form $\mathbf{G}_s$. For $\delta \leq w$ and $i \leq k - \delta$, we define $U(\delta, i) = \{\mathbf{u} \in \mathbb{F}_q^k \text{ s.t. } \mathsf{wt}(\mathbf{u}) = \delta, u_i = 1, u_j = 0 \forall j < i\}$. It can then be easily seen that when $w < k$ (which is the case we consider in this paper) we have

$$\{\mathbf{c} \in \mathcal{C} \text{ s.t. } \mathsf{wt}(\mathbf{c} = w)\} \subset \{a(\mathbf{u}\mathbf{G}_s), a \in \mathbb{F}_q^* \setminus \{1\}, \mathbf{u} \in \cup_{\delta=1}^{w} \cup_{i=0}^{k-\delta} U(\delta, i)\}.$$

From a practical point of view, the codeword search can be performed by testing all codewords of the form $\mathbf{u}\mathbf{G}_s$. Once a codeword of weight w is found, then all of its scalar multiples are computed. In particular, few scalar multiples will

be computed with respect to the whole number of tested codewords, since we expect the set of weight w codewords to be relatively small. Thus we can neglect the computational cost of this step. For each candidate $\mathbf{u}$, we need to compute $n-k$ multiplications (since the first non zero entry of $\mathbf{u}$ is 1) and $\delta - 1$ sums in $\mathbb{F}_q$. Since all sets $U(\delta, i)$ are disjoint, it can be straightforwardly shown that the number of vectors $\mathbf{u}$ that are tested is $\sum_{\delta=1}^{w} \binom{k}{\delta}(q-1)^{\delta-1}$. Then, by neglecting the cost of partitioning step, we have

$$\mathcal{O}_{\text{Leon}} = \mathcal{O}\left(4(n-k)\sum_{\delta=1}^{w}(\delta-1)\binom{k}{\delta}(q-1)^{\delta-1}\right). \tag{3}$$

One final remark is about eventual future developments regarding Leon's algorithm. Indeed, the algorithm is inefficient for large codes, or for large finite fields, since the codeword enumeration becomes infeasible. This step can not be avoid as the algorithm requires to find all the codeword of weight w.

SSA Attack [26]. The SSA was introduced in [26]. This algorithm aims to distinguish whether the two given linear code are permutation equivalent or not, and recover this permutation furthermore. The algorithm employs the concept of invariants and signatures, where invariants are mappings such that any two permutationally equivalent codes take the same value, and signature depends on the code and one of its positions. The key point of SSA is to choose a suitable signature, whose formal definition is presented as follows:

Definition 8 *Let $\mathcal{C}$ be an $[n,k]$ linear code. S is called a signature function over a set F if it maps $\mathcal{C}$ and a position $i \in [0; n-1]$ into F and satisfies*

$$\mathsf{S}(\mathcal{C}, i) = \mathsf{S}\left(\sigma(\mathcal{C}), \sigma(i)\right), \forall \sigma \in S_n.$$

Moreover, a signature function is fully discriminant if $\mathcal{C}, \mathsf{i} \neq \mathsf{S}(\mathcal{C}, j), \forall i \neq j$. Form this property, we have that signature functions can be used to recover information about the permutation that is acting on the code. In particular, once in possession of a fully discriminant signature, the permutation σ can immediately be recovered because $\mathsf{S}(\mathcal{C}, i) = \mathsf{S}(\mathcal{C}', j) \Leftrightarrow j = \sigma(i)$. The fundamental idea of SSA is first try to find a distinct property for the code and one of its positions, and thus by labelling them accordingly it is possible to recover the permutation between equivalent codes. Considering the difficulty of calculating the weight enumerator of the code, [26] chooses the weight enumerator of the hull, i.e. its intersection with its dual $\text{Hull}(\mathcal{C}) = \mathcal{C} \cap \mathcal{C}^{\perp}$. For each $\sigma \in S_n$, we have $\text{Hull}(\sigma(\mathcal{C})) = \sigma(\text{Hull}(\mathcal{C}))$. Let $\mathcal{C}_i$ be the code obtained by puncturing $\mathcal{C}$ in position i, then a valid signature can be obtained by letting $\mathsf{S}(\mathcal{C}, i)$ be the weight enumerator function of a $\text{Hull}(\mathcal{C}_i)$. With this choice of signature function, the complexity of SSA can be estimated with some simple observations.

- As mentioned in [26], the hull of a punctured code can be conveniently obtained from the hull of the original code. Thus, as an initialization, the Hulls of $\mathcal{C}$ and $\mathcal{C}'$ are computed. A basis for the Hull can be computed with a simple Gaussian elimination. Thus this step requires $\mathcal{O}(n^3)$ operations.

- For each position $i \in [1; n]$, the Hull of $\mathcal{C}_i$ can be obtained with negligible computation from Hull($\mathcal{C}_i$) must be generated. This requires $\mathcal{O}(nq^h \log n)$ operations in $\mathbb{F}_q$, where d denotes the dimension of Hull($\mathcal{C}_i$).
- In the conservative assumption that such a signature is fully discriminant, it is sufficient to evaluate it for $2n$ times (for codes $\mathcal{C}$ and $\mathcal{C}'$, and for each $i \in [1; n]$).

Thus the overall complexity of SSA is given by

$$\mathcal{O}_{\mathrm{SSA}} = \mathcal{O}(n^3 + n^2 q^h \log n). \tag{4}$$

Improved Leon's Attack [6]. In [6], a new attack which can be viewed as an improved Leon's algorithm was proposed to attack cryptosystems based on code equivalence problem. This algorithm works best with large finite fields and hulls. Their improvement is based on the observation that if the size of the finite field is large enough then the implication also holds in the other direction with large probability. That is to say, if $\mathbf{x} \in \mathcal{C}_1$ and $\mathbf{y} \in \mathbf{C}_2$ are low-weight codewords with the same support and entries, then with large probability $\sigma(\mathbf{x}) = \mathbf{y}$. For example, if $\mathbf{x} = (1, 0, 0, 5, 0, 23, 0, 7, 0) \in \mathcal{C}_1$ and $\mathbf{y} = (0, 7, 0, 23, 1, 0, 0, 0, 5) \in \mathbf{C}_2$, then we know $\sigma(1) = 5$, $\sigma(4) = 9$, $\sigma(6) = 4$ and $\sigma(8) = 2$.

The case becomes a little complexity for linearly-equivalence. The strategy above does not work because the linear isometries τ do not preserve the entries. [6] proposed to use a pair of 2-dimensional subspaces (U, V) with small support instead of the codeword pairs $(\mathbf{x}, \mathbf{y})$ above. With the help of a function $\mathsf{lex}(\cdot)$, codewords that have a same scale in the two given codes can be found. Here $\mathsf{lex}(V)$ is defined to be the lexicographically first basis of a 2-space in the orbit of linear isometries of V. If $\tau(V) = U$, then the orbit of linear isometries of V and U will be the same and hence $\mathsf{lex}(V) = \mathsf{lex}(U)$. By generating a list L that contains enough $(V, \mathsf{lex}(v))$ pairs, the attack algorithm then try to find $U \in \mathcal{C}_2$ such that $\mathsf{lex}(V) = \mathsf{lex}(U)$. If such a pair is found, then add (V, U) into a list P until P has $2\log(n)$ elements. At last, all the linear isometries τ will be iterated to find the unique one such that $\tau(V) = U$ for all $(V, U) \in P$. This τ is considered to satisfies $\tau(\mathcal{C}_1) = \mathcal{C}_2$. The complexity of this algorithm can be estimated as

$$\text{Beullens} = \mathcal{O}\left(\frac{\sqrt{\binom{n}{w} \cdot \log n}}{\binom{n-k}{w-2}} q^{-w+2+n-k} \right). \tag{5}$$

Parameters. We simulate our public key encryption scheme by online MAGMA [20]. The most cost in encryption and decryption is the Gaussian elimination algorithm and the matrix multiplications. Our simulation is only a craft implementation to test the correctness, and there is a lot of room for optimization. Roughly speaking, the generation of a random permutation matrix and a diagonal matrix needs $\mathcal{O}(n)$ operations, the multiplication among $k \times k$, $k \times n$ and $n \times n$ matrices need $\mathcal{O}(k^2 n + k n^2)$ operations, and the Gaussian elimination algorithm costs $\mathcal{O}(n^3)$ operations. As a result, our **KeyGen** step needs $n + k^2 n + k n^2 + n^2$

operations over $\mathbb{F}_q$, **Enc** step needs $n + k^2n + kn^2 + n^2 + n^3$ operations over $\mathbb{F}_q$, and **Dec** step needs $k^2n + n^3$ operations over $\mathbb{F}_q$. Our scheme does not need any decoding algorithms, and this is a breakthrough in constructing code-based public-key encryption schemes.

1^λ	q	n	k	**KeyGen**	**Encrypt**	**Decrypt**
2^{128}	31	400	194	22.56 ms	95.48 ms	63.92 ms
2^{192}	31	600	294	55.75 ms	244.40 ms	184.47 ms
2^{256}	31	800	394	128.40 ms	489.17 ms	360.96 ms

Table 1. Parameters and Timings

Moreover, our scheme also performs well in the key size. We compared the key size of our scheme with the Classic McEliece scheme [1] and HQC scheme [22] where the former is in the round-3 finalist of NIST post-quantum cryptography standardization and the latter is in the alternate candidates. Classic McEliece provides ten sets of parameters, and we choose kem/mceliece348864, kem/mceliece460896, kem/mceliece6688128 for security level 128, 192, 256 respectively.

1^λ	Instance	Public key	Secret key	Cipher text	KEM message
2^{128}	Our Scheme	24978	250	25042	25106
	Classic McEliece	261120	6492	96	128
	HQC Scheme	2761	218	2761	2825
2^{192}	Our Scheme	56228	375	56292	56356
	Classic McEliece	524160	13608	156	188
	HQC Scheme	6227	385	6227	6291
2^{256}	Our Scheme	99978	500	100042	100105
	Classic McEliece	1044992	13932	208	240
	HQC Scheme	10807	509	10807	10871

Table 2. Theoretical Key Size (in Bytes)

The public key of our scheme is $pk = (\mathbf{G}, \mathbf{G}_p)$. The first matrix $\mathbf{G}$ is a full rank random matrix, which can be generated by a seed. In our simulation, we use a 256 bits seed to generate a string with length nk, thus the size of this part is 32 bytes. The second matrix can be presented in a systematic form and takes $(n-k)k\lceil\log(q)\rceil$ bits. The secret key is $sk = \mathbf{Q} \in \mathrm{OM}_n(q)$ where $\mathbf{Q} = \mathbf{DP}$. Here $\mathbf{D}$ is a diagonal matrix with $D_{i,i} = 1$ for $i = 0 \mod 2$ and $\mathbf{P}$ is a permutation matrix related to a permutation function σ with $\sigma(i) = i$ for $i = 0 \mod 2$. Thus $\mathbf{D}$ and $\mathbf{P}$ can be stored by two strings in $\mathbb{F}_q$ with length $n/2$, and the size

of secret is hence $n\lceil\log(q)\rceil$. The cipher text of our scheme contains two parts, where the first part is a $k \times n$ matrix (can be presented in systematic form) and the second part is a string of 512 bits. This cost a size of $(n-k)k\lceil\log(q)\rceil + 512$ bits. Our IND-CPA secure public key encryption scheme can be transformed into an IND-CCA2 key encapsulation mechanism (KEM) by Fujisaki-Okamoto transformation technique [16]. The KEM message contains an original ciphertext and a hash value. We takes the SHA-512 function (HQC uses SHA-512 as well, while Classic McEliece uses SHAKE256) to calculate this value. Thus the size of the KEM message is $(n-k)k\lceil\log(q)\rceil + 2 \cdot 512$ bits.

Since our scheme is an El Gamal type public-key encryption scheme, the first part of the ciphertext has a similar structure with the public-key, and thus has a huge size. But we think this is not a serious problem in practice. In a KEM scheme or a hybrid encryption system, the KEM message or a ciphertext only need to be transmitted once to establish a secure channel for subsequent communication. With the arrival of the 5G era, data transmission takes less cost nowadays. A high-definition film with several gigabytes can be download in a few seconds, not to mention a message within 100KB. Thus our new scheme with high communication cost but small computation cost can be applied in many areas.

5 Conclusion

In this paper, we introduce a new hard problem in coding theory called sub-DLE problem. This is a subproblem of the well-known LE (linearly equivalence) problem, which has been widely used in code-based cryptosystems. We proved that the sub-DLE problem can be reduced to the DLE problem. Based on this new hard problem, we construct the first code-based key exchange protocol, KECE. A public-key encryption scheme in a new paradigm is constructed as well. The security of both schemes only relies on the hardness of the sub-DLE problem, which is different from most existed code-based cryptosystems that rely on the syndrome decoding problem. One of the advantages of our construction is a smaller key size compared with the traditional code-based cryptosystems. The reason is that we can make full use of the whole code space instead of only using one codeword. However, this also causes a drawback that our communication cost (ciphertext size) is larger than others.

The construction of the KECE protocol is very similar to the classic Diffie-Hellman key exchange protocol (DH) [14]. Although KECE is the first code-based key exchange protocol, there have existed lots of researches on the DH protocol. Many of these researches are possible to be promoted to KECE. For example, since DH does not provide authentication, it can not resist the men-in-middle attack. Authenticated key exchange is then proposed to solve this problem. In [4], a secure authenticated key exchange protocol from DH is proposed, and a three rounds password-based authenticated key exchange protocol from DH is presented in [18]. As our KECE protocol shares many similar properties with DH, transforming KECE to an authenticated protocol seems to be feasible. However,

the commutative property in the sub-CLE problem is not as strong as the CDH problem, which may cause challenges in the transformation.

Furthermore, our KECE protocol is a two-party key exchange protocol. Since there have existed many multi-parity DH protocols, how to modify our protocol into a multi-parity protocol is worth considering. An observation is that in order to ensure the commutative of matrices $\mathbf{Q}_a$ and $\mathbf{Q}_b$, we depart the columns of them into odd and even parts such that their action affects disjoint columns of $\mathbf{G}$, and also requires $k > n/2$. If more participants are involved, the actions of the sub-monomial matrix have to be departed into more disjoint sets, which indicates a much more large k, i.e. $k > (l-1)/l \cdot n$ for a l-parity protocol. This larger k will cause a smaller dimension k' for its dual code. Since that if two codes are linearly equivalent, then so are their dual, the small k' may consequence in an effective SSA attack. Thus a multi-parity protocol from coding theory is not an easy promotion.

Moreover, our KECE protocol is not only the first code-based non-interactive key exchange protocol, but also the first post-quantum non-interactive key exchange protocol besides SIDH. Since the hardness of super-singular isogenies is not studied throughout, KECE is a very competitive candidate. There also exists some lattice-based key exchange protocols, including [15], authenticated key exchange protocol [33] and [32]. However, these protocols all need interactive between participants. As far as we are considered, the reason is that lattice-based cryptosystems, especially those based on LWE assumptions, are impossible to be clear. Namely, the noise is difficult to be removed and thus further interaction is necessary to eliminate the effects taking by the noise.

The new code-based public-key encryption scheme also needs to follow with interest. One of the main drawbacks of code-based cryptosystems is the so large key size. Nowadays, the main ideas to solve this problem contains (1) referring to the lattice-based scheme like [22] and (2) using codes with special structures and powerful error correction capability like [31]. Our new scheme provides a brand-new solution, i.e. using the full code space instead of one codeword. Base on this new idea, how to build more practical code-based cryptosystems is worthwhile to work on. Since the sub-DLE problem shares a similar property as DDH problem, this provides new hope in constructing a new and efficient code-based signature system.

Acknowledgements

This work is supported by Guangdong Major Project of Basic and Applied Basic Research(2019B030302008) and the National Natural Science Foundation of China (No. 61972429).

References

1. Albrecht M.R., Bernstein D.J., Chou T., et. al.: Classic McEliece, 2017, https://classic.mceliece.org/ Accesses 25 September 2021.

2. Barenghi A., Biasse JF., Persichetti E., Santini P.: LESS-FM: Fine-Tuning Signatures from the Code Equivalence Problem. In: PQCrypto 2021. LNCS vol. 12841, pp.23-43. Springer, Cham. 2021.
3. Barreto P.S.L.M., Gueron S., Guneysu T., et. al.: CAKE: Code-Based Algorithm for Key Encapsulation. In: IMACC 2017. LNCS vol. 10655. pp. 207-226. Springer, Cham. 2017.
4. Bellare M., Pointcheval D., Rogaway P.: Authenticated Key Exchange Secure against Dictionary Attacks. In: EUROCRYPT 2000. LNCS vol. 1807, pp. 139-155. Springer, Berlin, Heidelberg. 2000.
5. Berlekamp E., McEliece,R., van Tilborg H.: On the inherent intractability of certain coding problems. IEEE Transactions on Information Theory. vol. 24(3), pp. 384-386. 1978.
6. Beullens W.: Not Enough LESS: An Improved Algorithm for Solving Code Equivalence Problems over Fq. In: SAC 2020. LNCS vol. 12804, pp. 387-403. Springer, Cham. 2021.
7. Biasse JF., Micheli G., Persichetti E., et. al.: A Post-Quantum Non-Interactive Key-Exchange Protocol from Coding Theory. IACR eprint (withdrawn) https://eprint.iacr.org/2020/206.
8. Biasse JF., Micheli G., Persichetti E., et. al.: LESS is More: Code-Based Signatures Without Syndromes. In: AFRICACRYPT 2020. LNCS vol. 12174, pp. 45-65. Springer, Cham. 2020.
9. Canetti R., Krawczyk H.: Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In: EUROCRYPT 2001. LNCS vol. 2045, pp. 453-474. Springer, Berlin, Heidelberg. 2001.
10. Courtois N., Finiasz M., Sendrier N.: How to achieve a Mceliece-based digital signature scheme. In: ASIACRYPT 2001. LNCS vol. 2248, pp. 157-174. Springer, Berlin, Heidelber.
11. Debris-Alazard T., Sendrier N., Tillich JP.: Wave: A New Family of Trapdoor One-Way Preimage Sampleable Functions Based on Codes. In: ASIACRYPT 2019. LNCS vol. 11921, pp. 21-51. Springer, Cham.
12. De Feo L., Jao D., Plut J.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. Journal of Mathematical Cryptology. vol. 8(3), pp. 209-247. 2014.
13. Deneuville JC., Gaborit P., Zémor G.: Ouroboros: A Simple, Secure and Efficient Key Exchange Protocol Based on Coding Theory. In: PQCrypto 2017. LNCS vol. 10346. pp. 18-34. Springer, Cham. 2017.
14. Diffie W., Hellman M.: New directions in cryptography. IEEE Transactions on Information Theory. vol. 22(6), pp. 644-654. 1976.
15. Ding J., Lin X.: A Simple Provably Secure Key Exchange Scheme Based on the Learning with Errors Problem. Iacr Cryptology Eprint Archive, 2012. https://eprint.iacr.org/2012/688
16. Hofheinz D., Hovelmanns K., Kiltz E.: A modular analysis of the Fujisaki-Okamoto transformation. In: TCC 2017, LNCS vol. 10677, pp. 341-371. 2017.
17. Grochow J.A.: Matrix Isomorphism of Matrix Lie Algebras. In: 2012 IEEE 27th Conference on Computational Complexity. pp. 203-213. IEEE. 2012.
18. Katz J., Ostrovsky R., Yung M.: Efficient and Secure Authenticated Key Exchange Using Weak Passwords. Journal of the ACM. vol. 57(1), pp. 3:1-3:39. 2009.
19. Leon J.: Computing Automorphism Groups of Error-Correcting Codes. IEEE Transactions on Information Theory. vol. 28(3), pp. 496-511. 1982.
20. MAGMA Calculator. http://magma.maths.usyd.edu.au/calc/

21. Mceliece R.J.: A public-key cryptosystem based on algebraic coding theory, DSN Progress Rep. 42(44), pp. 114-116. 1978.
22. Melchor C.A., Aragon N., Bettaieb S., et. al.: Hamming quasi-cyclic (HQC), 2017, https://pqc-hqc.org/ Accesses 25 September 2021.
23. Niederreiter H.: Knapsack-type cryptosystems and algebraic coding theory, Problems of Controland Information Theory. vol. 15(2), pp. 159-166. 1986.
24. Petrank E. and Roth R.M.: Is Code Equivalence Easy to Decide? IEEE Transactions on Information Theory, vol. 43(5), pp. 1602-1604. 1997.
25. Pellikaan R., Wu X., Bulygin S., et al.: Codes, Cryptology and Curves with Computer Algebra. Cambridge University Press, 2017.
26. Sendrier N.: Finding the Permutation Between Equivalent Linear Codes: The Support Splitting Algorithm. IEEE Transactions on Information Theory. vol. 46(4), pp. 1193-1203,08. 2000.
27. Sendrier N., Simos D.E.: The Hardness of Code Equivalence over Fq and Its Application to Code-Based Cryptography. In: PQCrypto 2013. LNCS vol. 7932, pp. 203-216. Springer, Berlin, Heidelberg. 2013.
28. Shor P.: Algorithms for quantum computation: discrete logarithms and factoring. In: FOCS 1994. pp. 124-134. IEEE. 1994.
29. Stern J.: A new identification scheme based on syndrome decoding. In: CRYPTO 1993. LNCS vol. 773, pp. 13-21. Springer, Berlin, Heidelberg. 1993.
30. Stern J.: A new paradigm for public key identification. IEEE Transactions on Information Theory, vol. 42(6), pp. 1757-1768. 1996.
31. Zhang F., Zhang Z., Guan P.: ECC2: Error correcting code and elliptic curve based cryptosystem. Information Sciences. vol. 526, pp. 301-320. 2020
32. Zhang J., Yu Y.: Two-Round PAKE from Approximate SPH and Instantiations from Lattices. In: ASIACRYPT 2017. LNCS vol. 10626, pp. 37-67. Springer, Cham. 2017.
33. Zhang J., Zhang Z., Ding, J., et. al.: Authenticated Key Exchange from Ideal Lattices. In: EUROCRYPT 2015. LNCS vol. 9057, pp. 719-751. Springer, Berlin, Heidelberg. 2015.