A Post-Quantum Non-Interactive Key-Exchange Protocol from Coding Theory

Jean-François Biasse[1], Giacomo Micheli[1], Edoardo Persichetti[2], and Paolo Santini[2,3]

[1] University of South Florida
[2] Florida Atlantic University
[3] Universitá Politecnica delle Marche
{biasse, gmicheli}@usf.edu, epersichetti@fau.edu, p.santini@pm.univpm.it

Abstract. This work introduces a new non-interactive key-exchange protocol, based on the hardness of the Code Equivalence Problem, a staple problem in coding theory. The protocol is modelled on the Diffie-Hellman framework. The novelty of the construction resides in the use of the code equivalence problem as the sole hardness assumption.
To the best of our knowledge, our construction represents the first code-based non-interactive key-exchange protocol, and in fact, the first post-quantum scheme of this kind which is not built upon supersingular isogenies. Our scheme provides significantly better performance than its isogeny counterparts in terms of execution time (at the cost of larger keys). This performance trade-off is favorable to users in most of the cases where the bandwidth is not severely constrained.

1 Introduction

The Diffie-Hellman key-exchange protocol is one of the milestones of modern cryptography. In its current instantiations, this protocol is at the basis of present-day communication mechanisms, such as, for instance, the (ephemeral) Diffie-Hellman handshake performed in TLS, and many other applications. It is therefore of great importance to have an efficient *post-quantum* version of Diffie-Hellman readily available, to serve as "drop-in" replacement if (and when) needed. In fact, due to Shor's algorithm [25], the vast majority of cryptographic primitives based on traditional problems, such as factoring or computing discrete logarithms, will be insecure once a quantum computer with sufficient stability and computational power is available. Historically, this is something that was missing in the post-quantum scenario, and devising a non-interactive key-exchange protocol offering good performance was regarded by the community as a challenging task. This is reflected in the recent Post-Quantum Standardization initiative by the National Institute of Standards and Technology (NIST) [18], which features no such schemes among its submissions.

An important development in this regard has been the recent appearance of schemes based on isogenies between supersingular elliptic curves [14, 6], which offer competitive performance, especially with respect to sizes. The isogeny-based key exchange protocol is the first example of a Post-Quantum non-interactive key exchange protocol. While the area of isogeny-based cryptography is considered very promising, it is also fairly young, in cryptographic terms, and still far from maturity, both concerning security and efficiency aspects. Until now, there has been a considerable gap in lattice-based and code-based cryptography, the two most prominent areas in the post-quantum setting, which do not offer non-interactive key-exchange protocols. It is therefore extremely important to the cryptography community to provide post-quantum alternatives to the isogeny-based non-interactive key exchange protocols, especially if new proposals can offer better performance.

Our contribution. In this work, we introduce an entirely new paradigm to build a post-quantum non-interactive key-exchange protocol. Our paradigm is based on an original use of a long-standing problem: that of determining the equivalence between two linear codes. Often mentioned in cryptography for its relation to the McEliece cryptosystem, the Code Equivalence Problem is considered, for the most part, simply as a (crucial) step to be able to invoke the well-known hardness of the Syndrome Decoding Problem, following the traditional approach in code-based cryptography. To the best of our knowledge, it is the first time a cryptosystem whose security relies solely on code equivalence is proposed.

The idea of our scheme is simple and derives from the Diffie-Hellman protocol. In this paper, we demonstrate its soundness, give a security proof and analyze the scheme's performances. We also provide a discussion on the computational difficulty of the underlying hardness assumption. Below is a high level description of our protocol:

Let $\mathfrak{C}$ be a random (public) linear code.

- Alice chooses a monomial map μ_a at random and sends $\mu_a(\mathfrak{C})$ to Bob
- Bob chooses a monomial map μ_b at random sends $\mu_b(\mathfrak{C})$ to Alice
- Alice computes the subcode $\mathfrak{C}_a := \mathfrak{C} \cap \mu_a^{-1}(\mu_b(\mathfrak{C}))$
- Bob computes the subcode $\mathfrak{C}_b := \mathfrak{C} \cap \mu_b^{-1}(\mu_a(\mathfrak{C}))$
- Since one can prove that the two subcodes are equivalent, Alice and Bob can take their weight enumerator $\mathsf{WEF}(\mathfrak{C}_a) = \mathsf{WEF}(\mathfrak{C}_b)$ as the shared secret

Note that computing the weight enumerator of a code is, in general, a hard task: for purely random codes, this essentially boils down to enumerating the codewords, with a complexity that, therefore, scales like q^k. This becomes feasible, however, for the instances considered by our protocol, since, as we will show, we are able to control the dimension of the resulting subcodes, by carefully choosing our parameters.

The paper is organized as follows. We begin by recalling some preliminary notions about coding theory and key-exchange protocols in Section 2. We then proceed to describe the code equivalence problem in Section 3, along with a discussion on its hardness. Our protocol is presented in Section 4, followed by a security assessment in Section 5. An extensive analysis of known attacks against the scheme is provided in Sections 6 and 7, the former discussing the best existing algorithms that can be used to solve the code equivalence problem, and the latter taking into account the eventuality of an attacker with access to quantum technology. In Section 8, we make some important considerations about the efficiency of our scheme, and present some sample instances, along with the results of a proof-of-concept implementation. Finally, we give some conclusions in Section 9.

2 Preliminaries

2.1 Notation

We will use the following conventions throughout the rest of the paper:

a	a constant
A	a set
$\boldsymbol{a}$	a vector
$\boldsymbol{A}$	a matrix
a	a function or relation
$\mathcal{A}$	an algorithm
I_n	the $n \times n$ identity matrix
$[a; b]$	the set of integers $\{a, a+1, \ldots, b\}$
$\mathbb{U}(A)$	the uniform distribution over the set A
$\xleftarrow{\$} A$	sampling uniformly at random from A
$E[x]$	expected value of a random variable x

We denote with $\mathbb{F}_q$ the finite field of order q, as customary. We will write $\mathrm{GL}_k(q)$, with $k \leq n$, to indicate the set of invertible $k \times k$ matrices with elements in $\mathbb{F}_q$. Let S_n be the group of permutations over n elements. These can equivalently be described as functions $\pi : \mathbb{F}_q^n \to \mathbb{F}_q^n$ or in matrix form as $n \times n$ matrices with exactly one 1 per row and column. By analogy, we denote with $\mathrm{M}_n(q)$ the set of monomial matrices with elements in $\mathbb{F}_q$, i.e. all the matrices of the form $\boldsymbol{Q} = \boldsymbol{P}\boldsymbol{D}$ where $\boldsymbol{P}$ is an $n \times n$ permutation matrix and $\boldsymbol{D} = \{d_{ij}\}$ is an $n \times n$ diagonal matrix such that $d_i := d_{ii} \in \mathbb{F}_q^*$ for all $i \leq n$. Given a vector $\boldsymbol{x} = (x_1, \cdots, x_n) \in \mathbb{F}_q^n$ and a permutation $\pi \in S_n$, we write the action of π on $\boldsymbol{x}$ as $\pi(\boldsymbol{x}) = (x_{\pi^{-1}(1)}, \cdots, x_{\pi^{-1}(n)})$.

2.2 Coding Theory

An $[n, k]$-*linear code* $\mathfrak{C}$ of length n and dimension k over $\mathbb{F}_q$ is a k-dimensional vector subspace of $\mathbb{F}_q^n$; we will sometimes refer to the dimension of a code $\mathfrak{C}$ as $\mathrm{Dim}(\mathfrak{C})$. For convenience, we denote with $\mathcal{L}_{n,k}$ the set of all $[n, k]$-linear codes over $\mathbb{F}_q$. Every linear code can be represented by a matrix $\boldsymbol{G} \in \mathbb{F}_q^{k\times n}$, called *generator matrix*, whose rows form a basis for the vector space.Then, the generator matrix defines the code as a mapping between vectors $\boldsymbol{u} \in \mathbb{F}_q^k$ and the corresponding codewords $\boldsymbol{uG}$. Obviously, there exist more than one generator matrix for the same code, corresponding to different choices of basis. It follows that all generator matrices are connected via a change-of-basis matrix, i.e. an invertible matrix $\boldsymbol{S} \in \mathrm{GL}$ such that $\boldsymbol{G}' = \boldsymbol{SG}$. Alternatively, a linear code can be represented as the kernel of a matrix $\boldsymbol{H} \in \mathbb{F}_q^{(n-k)\times n}$, known as *parity-check matrix*, i.e. $\mathfrak{C} = \{\boldsymbol{x} \in \mathbb{F}_q^n : \boldsymbol{Hx}^T = 0\}$. Once again, the parity-check matrix of a code is not unique. With a little abuse of notation, we sometimes write $\boldsymbol{G} \cap \boldsymbol{G}'$ when referring to the intersection of the two codes $\mathfrak{C}$ and $\mathfrak{C}'$ generated by $\boldsymbol{G}$ and $\boldsymbol{G}'$, respectively. In particular, we use $\boldsymbol{G} \cap \boldsymbol{G}'$ to indicate a generator matrix of the code $\mathfrak{C} \cap \mathfrak{C}'$. The weight enumerator function of a code is the bivariate polynomial defined as $W(x, y) = \sum_{w=0}^{n} A_w x^w y^{n-w}$, where A_w is the number of weight-w codewords in $\mathfrak{C}$. Throughout the rest of the paper, we will denote with WEF the function that, on input a code (represented through its generator matrix), returns its weight enumerator function.

For every linear code $\mathfrak{C}$, we can define its *dual code* $\mathfrak{C}^\perp$ as the set of words that are orthogonal to $\mathfrak{C}$, i.e. $\mathfrak{C}^\perp = \{\boldsymbol{y} \in \mathbb{F}_q^n \ : \ \boldsymbol{x} \cdot \boldsymbol{y}^T = 0, \forall \boldsymbol{x} \in \mathfrak{C}\}$. It is then easy to see that a parity-check matrix of a linear code is a generator of its dual, and vice versa. In fact, it must be that $\boldsymbol{G} \cdot \boldsymbol{H}^T = \boldsymbol{0}_{k\times(n-k)}$. Codes that are contained in their dual, i.e. $\mathfrak{C} \subseteq \mathfrak{C}^\perp$, are called *weakly self-dual*, and codes that are equal to their dual, i.e. $\mathfrak{C} = \mathfrak{C}^\perp$, are called simply *self-dual*.

2.3 Key Exchange Protocols

We recall here, in a succinct manner, some standard cryptographic notions about key-exchange protocols, and the relevant security definitions. We begin with a sketch of the classic Diffie-Hellman (DH) framework, which illustrates perfectly the generic structure of non-interactive key-exchange protocols.

For the original Diffie-Hellman protocol, the shared secret is the group element g^{ab} which, thanks to the commutativity property, can be computed by both parties in the most straightforward way. A shared key can then be derived by standard means (e.g. through a key-derivation function). Note that commutativity is not strictly necessary to achieve such a scheme, as long as the two parties are able to agree on a common value: this is the case, for instance, of modern instantiations based on isogenies [14], where the parties obtain two different curves with a common j-invariant.

Table 1. Diffie-Hellman Key Exchange.

Public Data	Parameter $r \in \mathbb{N}$, group G and element $g \in G$ of order r.	
Alice		Bob
Choose $a \xleftarrow{\$} [1; r-1]$.		Choose $b \xleftarrow{\$} [1; r-1]$.
Set $g_a = g^a$.		Set $g_b = g^b$.
	$\xrightarrow{g_a}$	
	$\xleftarrow{g_b}$	
$\mathsf{K} = g_b^a$.		$\mathsf{K} = g_a^b$.

It is easy to show that the Diffie-Hellman protocol is secure against passive attackers. This is done explicitly, for example, in [5], where this kind of security is modeled formally as *Session-Key Security* (SK Security) in the *Authenticated-links Model* (AM). An outline is given below.

Definition 1 (SK Security in the AM). *We define an adversary for the key-exchange protocol in the AM as a polynomial-time algorithm $\mathcal{A}$ which is able to observe (but not modify or interfere with) an honest execution of the protocol. After the execution, the adversary is given a value K_b, for $b \xleftarrow{\$} \{0, 1\}$, which is the genuine shared key when $b = 0$, and a randomly generated object (coming from the same distribution) when $b = 1$, and answers with a bit b^*. A key-exchange protocol is said to be* SK-secure in the AM *if the probability of $\mathcal{A}$ guessing correctly (i.e. $b^* = b$) is sufficiently close to $1/2$, that is, $\mathcal{A}$'s* advantage *is negligible in the security parameter.*

Note that we have omitted the "correctness" requirement, which is formally included in the security notion in [5, Definition 4], as we take this for granted when designing a scheme in the first place. Also, we do not digress here on more advanced security notions, such as the Unauthenticated-links Model (UM) or Perfect Forward Secrecy (PFS), as these can normally be achieved from the basic protocol via standard workarounds.

The security of the Diffie-Hellman protocol is based first and foremost on the Discrete Logarithm Problem (DLP) in the group G, as an adversary able to solve this problem can obviously recover the secret keys of Alice and Bob. As for the actual SK security, this relies on the two very famous problems known as CDH and DDH, which we summarize as follows.

Problem 1 (Computational Diffie-Hellman (CDH)) *Consider a triple (g, g^a, g^b) of elements of a group G. Compute the element g^{ab}.*

Problem 2 (Decisional Diffie-Hellman (DDH)) *Consider a quadruple (g, g^a, g^b, g^c) of elements of a group G. Determine whether or not $g^c = g^{ab}$.*

The problems are, obviously, closely related. There is an elementary chain of reductions, in one direction:

$$\text{DDH} \leq \text{CDH} \leq \text{DLP}.$$

The situation is rather more complex regarding the other direction. Indeed, it has been shown that equivalence holds in certain special cases [7, 17], but it is still an open problem to determine whether this is true in general.

3 The Code Equivalence Problem

The security of our construction of a code-based analogue of the DH protocol is connected to the hardness of the Code Equivalence Problem in a similar way that the security of the DH protocol is related to the hardness of the DLP. In this section, we introduce the Code Equivalence Problem and we present the prior work on its resolution.

Definition 2 (Permutation Code Equivalence). *We say that two codes* $\mathfrak{C}$ *and* $\mathfrak{C}'$ *are* permutationally equivalent*, and write* $\mathfrak{C} \overset{\text{PE}}{\sim} \mathfrak{C}'$*, if there is a permutation* $\pi \in S_n$ *that maps* $\mathfrak{C}$ *into* $\mathfrak{C}$*, i.e.*

$$\mathfrak{C}' = \{\pi(\boldsymbol{x}),\ \ \boldsymbol{x} \in \mathfrak{C}\}\,.$$

The previous notion of code equivalence can be extended using linear isometries. Indeed, let $\mu = (\boldsymbol{v}, \pi) \in \mathbb{F}_q^{*n} \rtimes S_n$ be an isometry μ, such that

$$\mu(\boldsymbol{x}) = (v_1 x_{\pi^{-1}(1)}, \cdots, v_n x_{\pi^{-1}(n)}).$$

We can then generalize the previous definition as follows.

Definition 3 (Linear Code Equivalence). *We say that two codes* $\mathfrak{C}$ *and* $\mathfrak{C}'$ *are* linearly equivalent*, and write* $\mathfrak{C} \overset{\text{LE}}{\sim} \mathfrak{C}'$*, if there is a linear isometry* $\mu = (\boldsymbol{v}; \pi) \in \mathbb{F}_q^{*n} \rtimes S_n$ *such that* $\mathfrak{C}' = \mu(\mathfrak{C})$*, i.e.* $\mathfrak{C}' = \{\mu(\boldsymbol{x}),\ \ \boldsymbol{x} \in \mathfrak{C}\}\,.$

The previous definitions can equivalently be stated in terms of generator (or parity-check) matrices; furthermore, the application of a permutation (resp. linear isometry) corresponds to the right multiplication by a permutation matrix $\boldsymbol{P}$ (resp. monomial matrix $\boldsymbol{Q}$).

Let $\mathfrak{C}$ and $\mathfrak{C}'$ be two codes with respective generator matrices $\boldsymbol{G}$ and $\boldsymbol{G}'$: we then have

$$\mathfrak{C} \overset{\text{PE}}{\sim} \mathfrak{C}' \iff \exists (\boldsymbol{S}, \boldsymbol{P}) \in \text{GL}_k(q) \times S_n \ \text{ s.t. } \ \boldsymbol{G}' = \boldsymbol{SGP},$$
$$\mathfrak{C} \overset{\text{LE}}{\sim} \mathfrak{C}' \iff \exists (\boldsymbol{S}, \boldsymbol{Q}) \in \text{GL}_k(q) \times M_n(q) \ \text{ s.t. } \ \boldsymbol{G}' = \boldsymbol{SGQ}.$$

Another notion of code equivalence (using *semilinear* isometries) is often found in the literature; however, it is not needed for our protocol, and therefore we do not present it here. We instead refer the interested reader to [22] for further details, and move on to present some hard problems connected to the notions we just described.

Problem 3 (Permutation Code Equivalence (PE)) *Let $\mathfrak{C}$ and $\mathfrak{C}'$ be two linear codes of length n and dimension k, defined over $\mathbb{F}_q$. Determine whether the two codes are permutationally equivalent.*

Problem 4 (Linear Code Equivalence (LE)) *Let $\mathfrak{C}$ and $\mathfrak{C}'$ be two linear codes of length n and dimension k, defined over $\mathbb{F}_q$. Determine whether the two codes are linearly equivalent.*

The two problems above are clearly two different flavors of the same problem, namely, deciding whether two codes are equivalent, which differ according to which notion of code equivalence is considered. However, as we will see, the connection between the two is not as obvious as it seems.

Hardness at a Glance

The hardness of the code equivalence problem has been studied extensively in several works. In here, we provide an overview of the topic. A more detailed analysis will be given in Section 6, where we discuss the run time of the various existing algorithms aimed at solving the code equivalence problems, that pertain to the cryptanalysis of our scheme.

The complexity of the problem was analyzed by Petrank and Roth, who proved a very interesting result. In fact, in [20], the authors showed that, on one hand, the permutation equivalence problem is unlikely to be NP-complete (or else the polynomial hierarchy would collapse) while, on the other hand, the problem is at least as hard as Graph Isomorphism (GI). The GI problem was recently proved to be solvable in quasi-polynomial time by Babai [1] without implying the weakness of permutation code equivalence. This comparison with permutation code equivalence was later generalized by Grochow [10] to the q-ary case. Following this line of work, Sendrier and Simos [23, 22] focused on analyzing the practical hardness of the problem, concluding that there exist many instances that are indeed intractable.

Leon's algorithm [16], introduced in 1982, is probably the earliest method that can be used to solve the code equivalence problem, and specifically the permutation version. The algorithm reconstructs the secret permutation from its action on the set of codewords with fixed weight. When this set is not too large, the permutation can efficiently be recovered. The bottleneck is in the codeword search, whose time complexity is $nq^{O(k)}$. Thus, as noted in [2], Leon's algorithm is impractical, unless considering codes of small dimension defined over small finite fields.

A second approach, also aimed at solving the permutation version, is that of the Support Splitting Algorithm (SSA), initially described by Sendrier [21]. The algorithm, which represents a strong improvement over Leon's for the task at hand, is based on the concept of the hull, which is simply the intersection between a code and its dual. Computing the hull is easy and requires only simple linear algebra, and the time complexity of the whole algorithm essentially grows as q^h, where h is the hull's dimension. For random codes, this dimension is, with high probability, equal to a small constant [24], de facto making SSA a polynomial-time solver for many instances of the permutation equivalence problem. The algorithm was later generalized in [22] to attack the linear equivalence problem. In fact, the problem of establishing the linear equivalence between two codes can always be reduced to that of finding a permutation equivalence between their *closures*. It follows that constructing the closures (as we explain in Section 6) and applying SSA is enough to solve the linear equivalence.

It is important to mention that SSA fails when the codes considered only have a trivial (i.e. empty) hull. However, an efficient treatment of this situation has recently been provided [2] through a reduction, running in time $O(n^\omega)$ (with $2 \leq \omega \leq 3$), from permutation equivalence to an instance of the graph isomorphism problem between undirected weighted graphs. Another case which cannot efficiently be solved through SSA is that of codes with a large hull. In fact, since the time complexity is dominated by q^h, SSA becomes quickly unfeasible as h grows. This is, for instance, the case of self-dual (or weakly self-dual codes), for which $h = k$: for such codes, SSA can be made arbitrarily hard by choosing codes with a sufficiently large dimension. The hardness of such instances is corroborated by the reduction to graph isomorphism in [2] which, for non-trivial hulls, runs in time $O(hn^{\omega+h+1})$ and, as expected, becomes quickly unfeasible for large values of h.

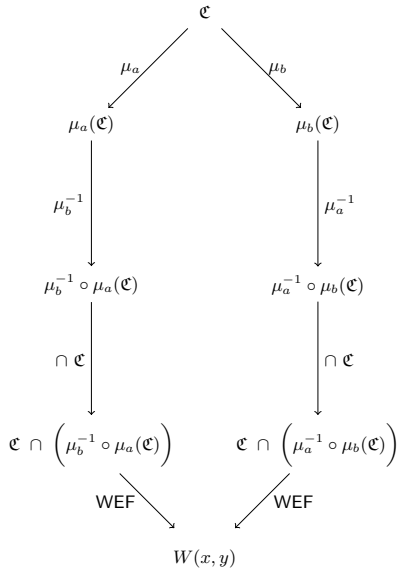
4 Protocol Description

Our protocol is based on a procedure which, given a public code, allows two parties to agree on subcodes that are linearly equivalent. In particular, this means also that they have the same weight enumerator function. This common weight enumerator function is taken as the shared secret. The operations which are performed in our protocol, to obtain such an equivalence, are depicted in Figure 1. In the figure, $\mathfrak{C}$ denotes an arbitrary linear code defined over $\mathbb{F}_q$, while μ_a and μ_b are linear isometries. We have used μ_a^{-1} (resp. μ_b^{-1}) to denote the inverse of μ_a (resp. μ_b), i.e., the isometry such that its composition with μ_a (resp. μ_b) returns the identity.

There is an obvious isomorphism between linear isometries and monomial matrices, such that each linear isometry can be mapped into a monomial matrix. We have

$$\mu \mapsto \boldsymbol{Q} \iff \mu^{-1} \mapsto \boldsymbol{Q}^{-1},$$
$$\mu \mapsto \boldsymbol{Q},\ \ \mu' \mapsto \boldsymbol{Q}' \iff \mu' \circ \mu \mapsto \boldsymbol{Q} \cdot \boldsymbol{Q}'.$$

Fig. 1. Diagram of our Scheme.



It is then easily seen that all code operations can be represented through linear algebra, so that codes are described through their generator matrices, and isometries through the corresponding monomial matrices. Because of this fact, a possible practical realization of the flow described in Figure 1 is provided in Table 2. We have denoted with $\boldsymbol{Q}_a$ and $\boldsymbol{Q}_b$ the monomial matrices associated to the isometries μ_a and μ_b, while their inverses are associated to μ_a^{-1} and μ_b^{-1}, respectively. The matrices $\boldsymbol{S}_a$ and $\boldsymbol{S}_b$ are non-singular and they act through a change of basis of the codes.

Table 2. Description of a Practical Instantiation.

Public Data Parameters $q, n, k \in \mathbb{N}$, matrix $\boldsymbol{G} \in \mathbb{F}_q^{k \times n}$ and hash function H.

Alice		Bob
Choose $\boldsymbol{S}_a \xleftarrow{\$} \mathrm{GL}_k(q)$, $\boldsymbol{Q}_a \xleftarrow{\$} \mathrm{M}_n(q)$.		Choose $\boldsymbol{S}_b \xleftarrow{\$} \mathrm{GL}_k(q)$, $\boldsymbol{Q}_b \xleftarrow{\$} \mathrm{M}_n(q)$.
Set $\boldsymbol{G}_a = \boldsymbol{S}_a \boldsymbol{G} \boldsymbol{Q}_a$.		Set $\boldsymbol{G}_b = \boldsymbol{S}_b \boldsymbol{G} \boldsymbol{Q}_b$.
	$\xrightarrow{\boldsymbol{G}_a}$	
	$\xleftarrow{\boldsymbol{G}_b}$	
Compute $\tilde{\boldsymbol{G}}_a = \boldsymbol{G}_b \boldsymbol{Q}_a^{-1}$.		Compute $\tilde{\boldsymbol{G}}_b = \boldsymbol{G}_a \boldsymbol{Q}_b^{-1}$
Obtain $\boldsymbol{G}'_a = \boldsymbol{G} \cap \tilde{\boldsymbol{G}}_a$.		Obtain $\boldsymbol{G}'_b = \boldsymbol{G} \cap \tilde{\boldsymbol{G}}_b$.
Compute $\mathsf{K} = \mathsf{WEF}(\boldsymbol{G}'_a)$.		Compute $\mathsf{K} = \mathsf{WEF}(\boldsymbol{G}'_b)$.

4.1 Correctness

In this section, we show that the protocol is correct by proving that the operations described in Figure 1 actually lead to codes having the same weight enumerator function. The main property which we will use is expressed in the following Lemma.

Lemma 1. *Let $\mathfrak{C}$ be a linear code of length n and dimension k, defined over $\mathbb{F}_q$, μ be a linear isometry and $\mathfrak{C}' = \mu(\mathfrak{C})$. Let $\boldsymbol{G}$ and $\boldsymbol{H}$ be a generator and a parity-check matrix for $\mathfrak{C}$, respectively. Then, $\mathfrak{C}'$ admits a parity-check matrix in the form $\boldsymbol{H}(\boldsymbol{V}^{-1})^\top$, where $\boldsymbol{V}$ is the monomial matrix associated to μ and $^\top$ denotes transposition.*

Proof. Clearly, $\mathfrak{C}'$ is generated by $\boldsymbol{G}' = \boldsymbol{G}\boldsymbol{V}$. The matrix $\boldsymbol{H}' = \boldsymbol{H}(\boldsymbol{V}^{-1})^\top$ has the same row rank of $\boldsymbol{H}$ (that is, $n-k$) and furthermore

$$\begin{aligned}\boldsymbol{G}'\boldsymbol{H}'^\top &= \boldsymbol{G}\boldsymbol{V}\left((\boldsymbol{H}\boldsymbol{V}^{-1})^\top\right)^\top \\ &= \boldsymbol{G}\boldsymbol{V}\boldsymbol{V}^{-1}\boldsymbol{H}^\top \\ &= \boldsymbol{G}\boldsymbol{H}^\top \\ &= \boldsymbol{0}_{k\times(n-k)},\end{aligned}$$

since $\boldsymbol{G}\boldsymbol{H}^\top = \boldsymbol{0}_{k\times(n-k)}$ by definition. Then, $\boldsymbol{H}'$ is a valid parity-check matrix for $\mathfrak{C}'$. □

Let us denote with $\boldsymbol{V} = \boldsymbol{Q}_b\boldsymbol{Q}_a^{-1}$ the monomial corresponding to $\mu_a^{-1}\circ\mu_b$, i.e., $\mu_a^{-1}\circ\mu_b \mapsto \boldsymbol{V}$. Since $\mu_b^{-1}\circ\mu_a = (\mu_a^{-1}\circ\mu_b)^{-1}$, we additionally have

$$\mu_b^{-1}\circ\mu_a \mapsto \boldsymbol{V}^{-1}.$$

Consider now the actual codes that are derived by Alice and Bob. Alice obtains $\tilde{\mathfrak{C}}_a = \mu_a^{-1}\circ\mu_b(\mathfrak{C})$, which is generated by $\tilde{\boldsymbol{G}}_a = \boldsymbol{G}\boldsymbol{Q}_b\boldsymbol{Q}_a^{-1} = \boldsymbol{G}\boldsymbol{V}$. Alice then intersects the public code $\mathfrak{C}$ with $\tilde{\mathfrak{C}}_a$: the resulting code will be formed by codewords that simultaneously belong to both codes. Let $\boldsymbol{H}$ and $\tilde{\boldsymbol{H}}_a$ be arbitrary parity-check matrices for, respectively, $\mathfrak{C}$ and $\tilde{\mathfrak{C}}_a$; then

$$\mathfrak{C}'_a = \mathfrak{C}\cap\tilde{\mathfrak{C}}_a = \left\{\boldsymbol{c}\in\mathbb{F}_q^n \text{ s.t. } \begin{cases}\boldsymbol{H}\boldsymbol{c}^\top = \boldsymbol{0}_r,\\ \tilde{\boldsymbol{H}}_a\boldsymbol{c}^\top = \boldsymbol{0}_r.\end{cases}\right\}. \tag{1}$$

Because of Lemma 1, we know that, for each $\boldsymbol{H}$, we will have a parity-check matrix for $\tilde{\mathfrak{C}}_a$ in the form $\tilde{\boldsymbol{H}}_a = \boldsymbol{H}(\boldsymbol{V}^{-1})^\top$. Then, (1) can be rewritten as

$$\mathfrak{C}'_a = \mathfrak{C}\cap\tilde{\mathfrak{C}}_a = \left\{\boldsymbol{c}\in\mathbb{F}_q^n \text{ s.t. } \begin{cases}\boldsymbol{H}\boldsymbol{c}^\top = \boldsymbol{0}_r,\\ \boldsymbol{H}(\boldsymbol{V}^{-1})^\top\boldsymbol{c}^\top = \boldsymbol{0}_r.\end{cases}\right\}. \tag{2}$$

On Bob's side, we have

$$\tilde{\mathfrak{C}}_b = \mu_b^{-1}\mu_a(\mathfrak{C}).$$

We apply again Lemma 1, and thus know that $\tilde{\mathfrak{C}}_b$ admits a parity-check matrix in the form $\boldsymbol{HV}^\top$; then

$$\mathfrak{C}'_b = \mathfrak{C} \cap \tilde{\mathfrak{C}}_b = \left\{ \boldsymbol{c} \in \mathbb{F}_q^n \text{ s.t. } \begin{cases} \boldsymbol{H}\boldsymbol{c}^\top = \boldsymbol{0}_r, \\ \boldsymbol{H}\boldsymbol{V}^\top \boldsymbol{c}^\top = \boldsymbol{0}_r. \end{cases} \right\}. \tag{3}$$

We now show that $\mathfrak{C}'_a$ and $\mathfrak{C}'_b$ are linearly equivalent. This straightforwardly comes from the fact that replacing $\boldsymbol{c}$ with $\boldsymbol{cV}$ turns (2) into (3); then, for each codeword $\boldsymbol{c} \in \mathfrak{C}'_a$, there is a codeword in $\boldsymbol{C}'_a$ in the form $\boldsymbol{cV}$. Since $\mu_a^{-1} \circ \mu_b \mapsto \boldsymbol{V}$, we have

$$\mathfrak{C}'_b = \mu_a^{-1} \circ \mu_b(\mathfrak{C}'_a).$$

Since the two codes are equivalent, they share the same weight distribution function.

5 Underlying Problems and Complexity Assumptions

The security of our non-interactive key exchange can be examined in way analogous to what done for the existing protocols of the same kind, such as the original Diffie-Hellman, and its supersingular isogeny version. By this, we mean that, for instance, the secret key of each user is hidden behind a well-known and well-understood hard problem (such as DLP and, for isogenies, the SSI problem), while the actual key being exchanged is protected by a dedicated complexity assumption (respectively CDH/DDH and SSCDH/SSDDH).

In our case, the former is represented by the code equivalence problem. It is in fact obvious that, if an adversary is able to solve Problem 4, he is able to recover the secret keys of Alice and Bob. On the other hand, to address the latter, we introduce the following problems.

Problem 5 (Code Equivalence CDH (CECDH)) *Consider a triple* $(\mathfrak{C}, \mathfrak{C}_a = \mu_a(\mathfrak{C}), \mathfrak{C}_b = \mu_b(\mathfrak{C}))$ *of linearly equivalent* $[n, k]$ *linear codes over* $\mathbb{F}_q$. *Compute either* $\mu_b^{-1}(\mathfrak{C}_a)$ *or* $\mu_a^{-1}(\mathfrak{C}_b)$.

Compared to the usual formulation of the CDH assumption, we have a few differences. In fact, since the group action given by a linear isometry is not commutative, it is not true that $\mu_b^{-1}(\mathfrak{C}_a) = \mu_a^{-1}(\mathfrak{C}_b)$. Instead, the two codes are merely equivalent and therefore an attacker, in principle, has two distinct targets, either of which is sufficient for his purposes. This is in contrast with Diffie-Hellman, where clearly $(g^a)^b$ and $(g^b)^a$ are one and the same, but not unlike the situation for other variants of the paradigm, such as the isogeny version. Note that, however, $\mu_b^{-1}(\mathfrak{C}_a)$ and $\mu_a^{-1}(\mathfrak{C}_b)$ are not two *any* linearly equivalent codes, but they are tied together by the fact that they are both obtained by the composition of the same two group actions (or, to be precise, one action with the inverse of the other) and this is what makes it possible to perform the reconciliation. In fact, it is obviously not true in general that $\mathfrak{C} \cap \mathfrak{C}'$ is equivalent to $\mathfrak{C} \cap \mathfrak{C}''$ if $\mathfrak{C}, \mathfrak{C}'$ and $\mathfrak{C}''$ are generic linearly equivalent codes. Finally, note that,

in our scheme, we omitted the multiplication by the invertible matrices $\boldsymbol{S}_a^{-1}$ and $\boldsymbol{S}_b^{-1}$, since that is not meaningful in terms of security and not necessary for reconciliation.

The decisional version of the problem is presented below.

Problem 6 (Code Equivalence DDH (CEDDH)) *Consider a quadruple* $(\mathfrak{C}, \mathfrak{C}_a = \mu_a(\mathfrak{C}), \mathfrak{C}_b = \mu_b(\mathfrak{C}), \mathfrak{C}_c)$ *of* $[n, k]$ *linear codes over* $\mathbb{F}_q$*. Determine whether* $\mathfrak{C}_c$ *is equal to* $\mu_b^{-1}(\mathfrak{C}_a)$ *or* $\mu_a^{-1}(\mathfrak{C}_b)$*, or neither.*

We conjecture that the above problems are computationally infeasible, and that any polynomial solver for either of the problems only has a negligible advantage in the security parameter n. With regards to this, the situation is as follows. Clearly, if an attacker had access to an oracle capable of solving the code equivalence problem, it could trivially solve CECDH, and in turn, an efficient solver for CECDH is an obvious method to solve CEDDH. Thus we have

$$\text{CEDDH} \leq \text{CECDH} \leq \text{LE}.$$

No reduction is known in the other direction, and thus we can only plausibly assume that the scenario is indeed similar to the one of Diffie-Hellman; in the spirit of [14], we deem that the question of the equivalence between the above problems is at least hard to fully resolve, and treat the problems as being indeed equivalent for the remainder of this work.

We conclude this section by mentioning that it is indeed easy to show that our protocol satisfies the notion of session-key security defined in Section 2.3, based on the assumptions we just defined. The claim is summarized in the next result.

Proposition 1. *Under the CEDDH assumption, the key-exchange protocol of Table 2 is SK secure in the authenticated-links adversarial model.*

The above proposition can be proved straightforwardly following the proof of [5, Theorem 8]. A detailed treatment of the known attacks against code equivalence, explaining why and how we are able to generate hard instances for our scheme, will be given in the next section.

6 Analysis of Known Attacks

In this section we discuss various existing algorithms and techniques that can be used to attack our scheme. In particular, we focus on algorithms that are designed to solve the linear equivalence problem, which is at the base of our construction. By the end of the section, we will be able to derive precise conditions under which a given instance of the linear equivalence problem can be considered difficult.

6.1 Brute Force and Other Obvious Attacks

The most naive attack to the scheme would probably consist of trying to solve the linear equivalence problem by simple enumeration of all of the possible linear isometries of vectors over $\mathbb{F}_q^n$. This would result in searching through a space of dimension $|S_n| \cdot |\mathbb{F}_q^{*n}| = n!(q-1)^n = O(n^n q^n)$, which is clearly beyond any hope of being feasible.

Let $\mathfrak{C}$ be a code such that

$$\exists i, j, \;\; i \neq j \;\text{ s.t. }\; c_i = c_j, \;\; \forall \boldsymbol{c} \in \mathcal{C}.$$

In such a case, all generator matrices $\boldsymbol{G}$ for $\mathfrak{C}$ are such that the i-th and j-th columns are identical. All such columns will remain identical even after left multiplication by a non singular matrix. Let $\mathfrak{C}' = \mu(\mathfrak{C})$, where $\mu = (pi, \boldsymbol{v})$ is a linear isometry; we denote with $\boldsymbol{G}'$ a generator matrix for such a code. We denote with $\boldsymbol{g}_i$ and $\boldsymbol{g}_j$ the i-th and j-th columns in $\boldsymbol{G}$; through action of the isometry, these columns are mapped in $\boldsymbol{g}'_{i'}$ and g'_j, where $i' = \pi^{-1}(i)$ and $j' = \pi^{-1}(j)$. Furthermore, the element-wise division between $\boldsymbol{g}'_{i'}$ and $\boldsymbol{g}'_{j'}$ will return a vector in the form $[a, a, \cdots, a]$, with $a = v_{i'}/v_{j'} \in \mathbb{F}_q^*$. To find such columns in $\boldsymbol{G}'$, it is enough to consider all $\binom{n}{2}$ couples of columns $(\boldsymbol{g}_u, \boldsymbol{g}_\ell)$; for each couple, we compute the element-wise division and check whether it is a constant vector.

The original linear equivalence problem can then be solved by targeting the code obtained by puncturing $\mathfrak{C}$ and $\mathfrak{C}'$ in the positions (i, j). We remark the fact that, despite this is just a method to reduce its complexity, designing a scheme with clear and known ways to simplify known attacks is completely pointless. Thus, in our protocol, we impose that $\boldsymbol{G}$ has all different columns.

6.2 Leon's Algorithm

Leon's algorithm [16] is the first dedicated technique that can be used to solve permutation equivalence. The attack, in fact, consists simply of analyzing the action of the algorithm on the subset of codewords with fixed weight ω. 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 ω is the actual bottleneck of the algorithm. Usually ω is set as the minimum distance of the code; for random codes, this can be estimated with the the GV bound. If this set does not have sufficient structure, then ω is slightly increased. We now briefly describe how the codeword enumeration can be performed. Let $\boldsymbol{G}$ be the generator matrix of a code $\mathfrak{C}$ of length n and dimension k, and call $\boldsymbol{G}_{\mathsf{SF}}$ its systematic form. For $\delta \leq w$, and $i \leq k - \delta$, we define

$$U(\delta, i) = \left\{ \boldsymbol{u} \in \mathbb{F}_q^k \;\text{ s.t. }\; \mathrm{wt}(\boldsymbol{u}) = \delta, \; u_i = 1, \;\; u_j = 0 \;\; \forall j < i \right\}.$$

It can then be easily seen that, when $\omega \leq k$ (which is the case we consider in this paper) we have

$$\{\boldsymbol{c} \in \mathfrak{C} \text{ s.t. } \mathrm{wt}(\boldsymbol{c}) = \omega\} \subseteq \left\{ a(\boldsymbol{u}\boldsymbol{G}_{\mathsf{SF}}), \;\; a \in \mathbb{F}_q^* \setminus \{1\}, \;\; \boldsymbol{u} \in \bigcup_{\delta=1}^{\omega} \bigcup_{i=0}^{k-\delta} U(\delta, i) \right\}.$$

From a practical point of view, the codeword search can be performed by testing all codewords of the form $\boldsymbol{u}\boldsymbol{G}_{\mathsf{SF}}$. Once a codeword of weight ω 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-ω codewords to be relatively small; thus, we can neglect the computational cost of this step. For each candidate $\boldsymbol{u}$, we need to compute $n - k$ codeword symbols; when $\boldsymbol{u}$ has weight δ, this can be done with $\delta - 1$ multiplications (since the first non null entry of $\boldsymbol{u}$ 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 $\boldsymbol{u}$ that are tested is $\sum_{\delta=1}^{\omega} \binom{k}{\delta}(q-1)^{\delta-1}$. Then, by neglecting the cost of the partitioning step, we have

$$C_{LEON} = O\left(4(n-k)\sum_{\delta=1}^{\omega}(\delta-1)\binom{k}{\delta}(q-1)^{\delta-1}\right). \tag{4}$$

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 quickly becomes unfeasible. This step cannot be avoided, as the algorithm requires to find *all* the codewords of weight ω. We do not exclude the possibility of strong improvements in Leon's algorithm, leading to the possibility of operating with just a subset of all such codewords. In such a case, codewords of some weight ω can be efficiently determined by means of ISD algorithms which, at each call, randomly pick a codeword of the desired weight. Thus, multiple ISD calls can be used to find the required number of codewords of the desired weight. If this scenario ever became a concern, the issue could be entirely avoided by choosing code parameters such that even a single ISD call is computationally too expensive.

6.3 The Support Splitting Algorithm

The basis on which SSA is built is the concept of *signature function*, introduced in [21], which is defined in the following way.

Definition 4. *Let $\mathfrak{C}$ be a linear code of length n; we say that a function S is a* signature function *over a set F if it maps $\mathfrak{C}$ and a position $i \in [0; n-1]$ to F and is such that*

$$\mathsf{S}(\mathfrak{C}, i) = \mathsf{S}\big(\pi(C), \pi(i)\big), \;\; \forall \pi \in S_n.$$

We say a signature function is fully discriminant *if $\mathsf{S}(\mathfrak{C}, i) \neq \mathsf{S}(\mathfrak{C}, j), \;\; \forall i \neq j$.*

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 π can immediately be recovered, since

$$\mathsf{S}(\mathfrak{C}, i) = \mathsf{S}(\mathfrak{C}', j) \iff j = \pi(i). \tag{5}$$

Assuming that such a fully discriminant function S is available, then SSA corresponds to the trivial algorithm that searches for collisions between the sets of values $\mathsf{S}(\mathfrak{C}, i)$ and $\mathsf{S}(\mathfrak{C}', j)$, for $(i, j) \in [1; n] \times [1; n]$. We point out that the existence of such a function (without, additionally, requiring unfeasible computations) is clearly not guaranteed for all pairs of codes. In such cases, SSA makes use of signatures refining, that is, new computations and combinations of signatures, that proceed until a fully discriminant function is obtained. In this paper, with a conservative choice, we assume that the chosen signature function is fully discriminant for the pair of codes considered, and that the refining of signatures is never required. In this way, we are guaranteed to provide a lower bound on the actual complexity of SSA.

Sendrier in [21] proposes to build signatures from the hull space that, for a code $\mathfrak{C}$, is defined as

$$\mathrm{Hull}\,(\mathfrak{C}) = \mathfrak{C} \cap \mathfrak{C}^{\perp}.$$

For each $\pi \in S_n$, we have $\mathrm{Hull}\,(\pi(\mathfrak{C})) = \pi(\mathrm{Hull}\,(\mathfrak{C}))$. Let $\mathfrak{C}_i$ be the code obtained by puncturing $\mathfrak{C}$ in position i: then, a valid signature can be obtained by letting $\mathsf{S}(\mathfrak{C}, i)$ be the weight enumerator function of $\mathrm{Hull}\,(\mathfrak{C}_i)$. With this choice of signature function, the complexity of SSA can be estimated with some simple observations.

1. As mentioned in [21], the hull of a punctured code can be conveniently obtained from the hull of the original code. Thus, as an initialization, the Hulls of $\mathfrak{C}$ and $\mathfrak{C}'$ are computed. A basis for the Hull can be computed with a simple Gaussian elimination: thus, this step requires $O(n^3)$ operations.
2. For each position $i \in [1; n]$, the Hull of $\mathfrak{C}_i$ can be obtained with negligible computation from $\mathrm{Hull}\,(\mathfrak{C})$. Then, to evaluate the weight enumerator function, all codewords of $\mathrm{Hull}\,(\mathfrak{C}_i)$ must be generated. This requires $O(nq^{d_{\mathrm{Hull}}} \log n)$ operations in $\mathbb{F}_q$, where d_{Hull} denotes the dimension of $\mathrm{Hull}\,(\mathfrak{C}_i)$.
3. In the conservative assumption that such a signature is fully discriminant, it is sufficient to evaluate it for $2n$ times (for codes $\mathfrak{C}$ and $\mathfrak{C}'$, and for each $i \in [1; n]$).

Thus, the overall complexity of SSA is given by

$$C_{SSA} = O\big(n^3 + n^2 q^h \log n\big), \tag{6}$$

where h corresponds to the hull dimension.

6.4 Solving the Linear Equivalence Problem

Given a couple of permutation equivalent codes $\mathfrak{C}$ and $\mathfrak{C}'$, both Leon's algorithm and SSA can be used to find a permutation mapping a code into the other. Leon's time complexity can be estimated through Equation (4), by setting ω equal to the GV distance, while SSA time complexity is given by Equation (6). We point out that, as explained in the previous sections, both choices lead to a conservative estimate.

We now describe how such algorithms can be used to tackle the liner equivalence problem. We first introduce the following additional notion, which is crucial in solving linear equivalence.

Definition 5. *Let* $\mathbb{F}_q = \{a_0 = 0, a_1, \cdots, a_{q-1}\}$*, and* $\boldsymbol{a} = (a_1, \cdots, a_{q-1})$*. We define the* closure *of a linear code* $\mathfrak{C}$*, defined over* $\mathbb{F}_q$*, as*

$$\tilde{\mathfrak{C}} = \{\boldsymbol{c} \otimes \boldsymbol{a}, \ \ \boldsymbol{c} \in \mathfrak{C}\}.$$

In other words, for a code $\mathfrak{C}$ of length n and dimension k, the closure is a code of the same dimension and length $(q-1)n$, and is obtained by multiplying the codewords of $\mathfrak{C}$ by all non-null elements of $\mathbb{F}_q$. The following result [22] states a relation between the closures of equivalent codes.

Proposition 2. *Let* $\mathfrak{C}, \mathfrak{C} \subseteq \mathbb{F}_q^n$*; then,* $\mathfrak{C} \overset{\text{LE}}{\sim} \mathfrak{C}'$ *if and only if* $\tilde{\mathfrak{C}} \overset{\text{PE}}{\sim} \tilde{\mathfrak{C}}'$*.*

To decide if two codes $\mathfrak{C}$ and $\mathfrak{C}'$ are linearly equivalent, and to retrieve the corresponding isometry, we can then solve the permutation equivalence problem between their closures $\tilde{\mathfrak{C}}$ and $\tilde{\mathfrak{C}}'$. Indeed, finding a permutation mapping $\tilde{\mathfrak{C}}$ into $\tilde{\mathfrak{C}}'$ trivially allows to retrieve the isometry between the initial codes $\mathfrak{C}$ and $\mathfrak{C}'$.

To use Leon's algorithm, all low weight codewords of the closures need to be enumerated. In particular, it can be easily shown that, for each word $\boldsymbol{c} \in \mathfrak{C}$ of weight ω, there is a word $\boldsymbol{c}' \in \mathfrak{C}'$ of weight $\omega' = (q-1)\omega$, obtained as $\boldsymbol{c}' = \boldsymbol{c} \otimes \boldsymbol{a}$. Then, it is enough to enumerate low weight codewords in the initial code. From an operational point of view, we can set ω as the GV distance computed with parameters n, k and q of the given code $\mathfrak{C}$, apply the technique described in Section 6.2, multiply through Kronecker product by $\boldsymbol{a}$ to build the set of codewords of weight $\omega' = (q-1)\omega$ and then apply Leon's algorithm. Then, the time complexity can be estimated through Equation (4) by setting $\omega = d_{\mathsf{GV}}(n, k, q)$.

To use SSA, we just need to apply the algorithm on the closures. However, as shown in [21], for $q \geq 5$ the closure of a code is always weakly self-dual, i.e., is always contained in its dual. It follows that its hull has then always (maximal) dimension k. As a consequence, to solve the Linear Equivalence Problem over $\mathbb{F}_q$, when $q \geq 5$, the algorithmic complexity of SSA is estimated through Equation (6) with $h = k$.

6.5 Considerations on the Dual Code

It is well-known [19, Prop. 1.5.16] that, if two codes are linearly equivalent, then so are their duals. The following proposition describes how a solution to the Linear Equivalence Problem on a pair of codes $\mathfrak{C}$ and $\mathfrak{C}'$ can be easily obtained from a solution obtained for their duals $\mathfrak{C}^\perp$ and $\mathfrak{C}'^\perp$.

Proposition 3. *Let $\boldsymbol{G}, \boldsymbol{G}'$ be the generator matrices of two linear equivalent codes $\mathfrak{C}$ and $\mathfrak{C}'$, i.e. $\boldsymbol{G}' = \boldsymbol{SGQ}$ for some unknown $\boldsymbol{S} \in GL$ and $\boldsymbol{Q} \in M_n(q)$. Then, solving the Linear Equivalence Problem on $\mathfrak{C}^\perp$ and $\mathfrak{C}'^\perp$ is equivalent to finding a solution to the original problem on $\mathfrak{C}$ and $\mathfrak{C}'$.*

The proof of Proposition 3 is fairly intuitive, and therefore omitted in the interest of space. A consequence of this result is that we should amend the complexity estimates given in (4) and (6) and consider a code dimension of $n - k$ instead of k. This turns in a simple criterion: when $k > n/2$, both Leon's algorithm and SSA are optimized by attacking the dual code (which can easily be computed with linear algebra).

7 A Quantum Scenario

In this section, we discuss various techniques to devise possible quantum attacks against the code equivalence problem.

7.1 Grover Search

To begin, it makes sense to consider the use of Grover's search algorithm [11], which plays a major role in quantum cryptanalysis. The algorithm allows us to search an unsorted database X of N entries for an element $x \in X$, with $f(x) = 1$, at a cost in $O(\sqrt{N}C_f)$, where $f : X \to \{0, 1\}$ and where C_f is the cost of implementing f. By "cost" here we indicate either the number of gates or the execution time (i.e. circuit depth). With regards to Leon's algorithm, it can indeed be expected that an application of Grover can improve the search part of the algorithm, leading to the usual speedup which corresponds, in the worst case, to roughly halving the complexity exponent (if one ignores the remaining part). Interestingly, though, a Grover search over all possible secrets (i.e. $P \in S_n$) would not outperform the classical SSA because of the size of S_n.

In principle, it is also possible to use Grover's algorithm *within* SSA. Indeed, for each $i \in [1; n]$, the search for $j \in [1; n]$ such that $j = \pi(i)$ corresponds to finding $j \in [1; n]$ such that $f(j) = 1$, where $f : [1; n] \to \{0, 1\}$ is defined as

$$f(j) = \begin{cases} 1 \text{ if } \mathsf{S}(\mathfrak{C}', j) = \mathsf{S}(\mathfrak{C}, i) \\ 0 \qquad \text{otherwise} \end{cases}$$

for a fully discriminant function S. Following the application of Bennett's generic method [3] (which converts any algorithm taking time T and space S into a reversible algorithm taking time $T^{1+\varepsilon}$ and space $O(S \log T)$), the cost of a quantum circuit evaluating f is that of S, which is in $\tilde{O}(nq^{d_{\mathrm{Hull}}} \log n)$. Thus, the search for $j \in [1; n]$ such that $j = \pi(i)$ costs

$$O(\sqrt{|[1; n]|} C_f) = \tilde{O}(n^{3/2} q^{d_{\mathrm{Hull}}} \log n).$$

This process needs to be repeated $n/2$ times. Every time a pair $(i, \pi(i))$ is found, both elements can be removed from the search space. This means that, in the previous formulas, we replace $[1; n]$ with $[1; n]^{(k)}$, where $n-2k \leq |[1; n]^{(k)}| \leq n-k$ (at each stage we remove either 1 or 2 elements depending on whether $\pi(i) = i$). Our total cost is

$$O\left(\left(\sum_{k \leq n/2} \sqrt{|[1; n]^{(k)}|}\right) C_f\right).$$

We can bound this using the fact that

$$\underbrace{\sum_{k=1}^{n/2} \sqrt{2k}}_{\Omega(n^{3/2})} \leq \sum_{k \leq n/2} \sqrt{|[1; n]^{(k)}|} \leq \underbrace{\sum_{k=n/2}^{n} \sqrt{k}}_{\Omega(n^{3/2})}.$$

In the end, the complexity of the overall procedure is $\tilde{O}(n^{5/2} q^{d_{\mathrm{Hull}}} \log n)$, which does not outperform the classical method consisting in $2n$ evaluations of S followed by a matching of the values obtained.

7.2 Other Approaches

The other famous family of algorithms for quantum cryptanalysis is based on quantum Fourier sampling. These algorithms can be seen as generalizations of Shor's algorithm for factoring and solving the Discrete Logarithm Problem [25]. The general approach is to rephrase a problem as the search for a secret subgroup H within a known "control group" G. The Quantum Fourier Transform (QFT) over G allows us to create a state whose measurement (hopefully) yields an element in $\hat{H}$. By repeating this operation and using ad-hoc methods depending on H, one can recover H and solve the problem. In [8] and in the follow up work [9], Dinh, Moore and Russell show that to use a similar approach for solving the Permutation Equivalence Problem, one would have to choose $G = (\mathrm{GL} \times S_n) \rtimes \mathbb{Z}_2$. A criterion is given in Corollaries 1 and 2 of [9] for linear codes to be *HSP-hard*, meaning that it does not reveal any information about $\hat{H}$. The criterion asks that the code has very high rate, namely, that $q^{k^2} \leq n^{0.2n}$, and that the automorphism group of the code has very small degree.

The authors give some concrete examples of families of codes that satisfy the criterion. This is the case, for instance, of Alternant codes and Goppa codes. For

these families, it is possible to give explicit bounds on the size of the automorphism group. Moreover, since these codes are subfield subcodes of Generalized Reed-solomon codes, the criterion can be satisfied by considering a generator matrix over the extension field and referring to the dimension of the "parent" code. This makes it so that the resulting code does not need to have the very high rate mentioned above, thus generating practical cryptographic instances.

The results just presented naturally extend to the Linear Equivalence Problem via the use of the closure. We note that these conditions, as interesting as they are from a theoretical point of view, are not necessary for our codes to offer quantum resistance. Indeed, no attack relying on the quantum Fourier sampling has been described so far in literature. Interestingly, the conditions are also not sufficient to claim post-quantum resistance since other attacks not based on quantum Fourier sampling might exist. This is for example the case of certain Goppa codes which satisfy the conditions described in [8] showing the impossibility of using the quantum Fourier sampling method, despite being attacked by the classical SSA because their hull has a small dimension.

8 Efficiency Considerations

In the previous section we have derived some conditions that the system parameters must satisfy, in order to guarantee a sufficiently large time complexity for the state-of-the-art algorithms that aim to solve the linear equivalence problem. In this section, moved by a different motive, we derive conditions which guarantee a low algorithmic complexity for our proposed protocol. Our analysis is based on the next assumption, which essentially follows from the assumed hardness of the Liner Equivalence Problem.

Assumption 1 ***Indistinguishability under Linear Isometry***
Let $\boldsymbol{G} \in \mathbb{F}_q^{k\times n}$ *with full row rank* k *and all different columns,* $\boldsymbol{S} \xleftarrow{\$} GL_k(q)$ *and* $\boldsymbol{Q} \xleftarrow{\$} M_n(q)$*. Then,* $\boldsymbol{G'} = \boldsymbol{SGQ}$ *is indistinguishable from a random* $k \times n$*, full row rank matrix over* $\mathbb{F}_q$*.*

The assumption can equivalently be re-written in terms of codes; however, the formulation we chose additionally captures the entity of the hardness of the linear equivalence problem, since the left multiplication by a full rank matrix $\boldsymbol{S}$ is fundamental in "hiding the structure" of $\boldsymbol{G}$ into that of $\boldsymbol{G'}$. However, $\boldsymbol{G}$ and $\boldsymbol{SG}$ are two generators of the same subspace of $\mathbb{F}_q^n$, since left multiplication by $\boldsymbol{S}$ just corresponds to a change of basis. All the properties we consider in this section are invariant with respect to the basis which is used as a representation; thus, for the sake of simplicity, we will omit the matrix $\boldsymbol{S}$ and simply denote $\boldsymbol{G'} = \boldsymbol{GQ}$.

8.1 Dimension of the Secret Equivalent Subcodes

As stated in the previous section, for a given code $\mathfrak{C}$ and a random isometry μ, hardness of linear equivalence implies that $\mathfrak{C}' = \mu(\mathfrak{C})$ can be treated as a random code. Then, for the properties we derive in this section, we will always consider $\mathfrak{C}' \xleftarrow{\$} \mathcal{L}_{n,k}(q)$, where $\mathcal{L}_{n,k}(q)$ denotes the set of linear codes of length n and dimension k, defined over some finite field $\mathbb{F}_q$.

For two arbitrary codes $\mathfrak{C} \in \mathcal{L}_{n,k}(q)$ and $\mathfrak{C}' \in \mathcal{L}_{n,k'}(q)$, we have $\mathrm{Dim}\,(\mathfrak{C} \cap \mathfrak{C}') \leq \min\{k, k'\}$. When $k + k' > n$, the codes necessarily intersect in a non-trivial (i.e., null) space. This is easily seen by first considering that, since linear codes are subspaces of $\mathbb{F}_q^n$, we have

$$\mathrm{Dim}\,(\mathfrak{C} \cup \mathfrak{C}') \leq \mathrm{Dim}\left(\mathbb{F}_q^n\right) = n, \;\; \forall \mathfrak{C} \in \mathcal{L}_{n,k}(q), \; \mathfrak{C}' \in \mathcal{L}_{n,k'}(q). \tag{7}$$

Let $\mathfrak{C} \in \mathcal{L}_{n,k}(q)$ and $\mathfrak{C}' \in \mathcal{L}_{n,k'}(q)$ with respective dimensions k and k' such that $k + k' > n$; if we assume that $\mathfrak{C} \cap \mathfrak{C}' = \varnothing$, we end up with

$$\mathrm{Dim}\,(\mathfrak{C} \cup \mathfrak{C}') = \mathrm{Dim}\,(\mathfrak{C}) + \mathrm{Dim}\,(\mathfrak{C}') = k + k' > n,$$

which contradicts (7). Then, it must be $\mathfrak{C} \cap \mathfrak{C}' \neq \varnothing$. With analogous arguments, it can be shown that $\mathrm{Dim}\,(\mathfrak{C} \cap \mathfrak{C}') \geq \max\{0 \; , \; k + k' - n\}$.

When $\mathfrak{C}'$ is sampled from $\mathcal{L}_{n,k}(q)$ according to the uniform distribution, we can study the dimension of the intersection code as a discrete random variable, whose probability distribution is derived next [12, Proposition 3.2].

Proposition 4. *Let $\mathfrak{C} \in \mathcal{L}_{n,k}(q)$ and $\mathfrak{C}' \xleftarrow{\$} \mathcal{L}_{n,k'}(q)$; let $k^* = \mathrm{Dim}\,(\mathfrak{C} \cap \mathfrak{C}')$. Then, k^* is a discrete random variable over $[\max\{0 \; , \; k + k' - n\} \; ; \; \min\{k \; , \; k'\}]$, with probability distribution given by*

$$\Pr\left[\mathrm{Dim}\,(\mathfrak{C} \cap \mathfrak{C}') = k^*\right] = \frac{\begin{bmatrix} k \\ k^* \end{bmatrix}_q \begin{bmatrix} n-k \\ k'-k^* \end{bmatrix}_q q^{(k-k^*)(k'-k^*)}}{\begin{bmatrix} n \\ k' \end{bmatrix}_q},$$

where $\begin{bmatrix} a \\ b \end{bmatrix}_q$ denotes the Gaussian binomial coefficient, defined as

$$\begin{bmatrix} a \\ b \end{bmatrix}_q = \begin{cases} \prod_{i=0}^{b-1} \frac{1-q^{a-i}}{1-q^{b-i}} & \text{if } a \leq b, \\ 0 & \text{if } b > a. \end{cases}$$

We now study the dimension of the secret subcodes which are obtained by Alice and Bob which, is fundamental in determining the algorithmic complexity of our proposed key exchange protocol. First of all, as shown in Section 4.1, Alice and Bob are in possession of two equivalent codes. For simplicity, then, we will only consider Alice's side, and denote with $\mathfrak{C}$ the public code (i.e., the one generated by the public matrix $\boldsymbol{G}$) and with $\mathfrak{C}'$ the code generated by $\tilde{\boldsymbol{G}}_a$, of dimension k'; since $\tilde{\boldsymbol{G}}_a$ has full row-rank, we have $k' = k$. For obvious reasons,

we want $\mathfrak{C} \cap \mathfrak{C}'$ to have trivial intersection with low probability. If we choose $k \leq n/2$, then this probability is not null, but it rapidly decreases as k gets lower than $n/2$. To completely avoid the issue, we impose $k > n/2$.

Despite the intersection being non-trivial, it is also crucially important that the dimension of the subcode not be too large. Indeed, the complexity of computing the weight enumerator of a code grows with the code dimension. With a naive approach, all codewords can be enumerated by computing all possible linear combinations of the rows of a generator matrix. If we denote the code dimension with k^*, then, using the schoolbook multiplication algorithm, will lead to a cost of $O\left(nk^* \cdot q^{k^*}\right)$, as there are a total of q^{k^*} linear combinations and the cost of computing each one of them is $O(kn)$. It is clear that this cost can be reduced; for example, computation of codewords that are scalar multiples can be avoided. Furthermore, improved multiplication algorithms and parallelization can be used, as well as ad hoc strategies (for instance, sums of previously found codewords). In any case, it is very likely that, regardless of the particular technique employed, the cost of computing the weight enumerator function will be proportional to the number of codewords, which is exactly equal to q^{k^*}. It follows that, to guarantee a low complexity, k^* needs to be sufficiently low. We can calculate the expected value of k^* as

$$E[\mathrm{Dim}\left(\mathfrak{C} \cap \mathfrak{C}'\right)] = \sum_{k^*=2k-n}^{k} k^* \cdot \Pr\left[\mathrm{Dim}\left(\mathfrak{C} \cap \mathfrak{C}'\right) = k^*\right]. \tag{8}$$

We now must ensure that the probability of ending up with subcodes of large dimension is an extremely rare event. To do this, we first define a threshold dimension $\bar{k}$, such that the complexity of enumerating a code of dimension $\bar{k}$ is not too high. Then, we use again Proposition 4 and design the code parameters to keep $\Pr\left[\mathrm{Dim}\left(\mathfrak{C} \cap \mathfrak{C}'\right) > \bar{k}\right]$ reasonably low. As a conservative choice, we will propose parameters n, k, q and $\bar{k}$ such that $q^{\bar{k}}$ is sufficiently low and, at the same time,

$$\Pr\left[\mathrm{Dim}\left(\mathfrak{C} \cap \mathfrak{C}'\right) > \bar{k}\right] = \sum_{\bar{k}^*=\bar{k}}^{k} \Pr\left[\mathrm{Dim}\left(\mathfrak{C} \cap \mathfrak{C}'\right) = k^*\right] < 2^{-\lambda}, \tag{9}$$

where λ is the security parameter.

Remark The previous criteria naturally allow for the possibility of producing a constant-time implementation. Indeed, the computation of the weight enumerator function may be designed to always run in constant time, corresponding to that of enumerating a code of dimension $\bar{k}$. Another option would be to introduce an abort criterion, such that the protocol returns a failure if $\mathrm{Dim}\left(\mathfrak{C} \cap \mathfrak{C}'\right) > \bar{k}$. This would introduce a probability of failure which, however, can be controlled by properly tuning the system parameters. For instance, a reasonable trade-off between performances and failure rate may be reached by satisfying (9). A more comprehensive discussion of this topic is deferred to future work, where implementation aspects are addressed in full detail.

8.2 A Practical Instance

To showcase the performance of the scheme, we have designed practical instances of the scheme with a security parameter of $\lambda = 128$, i.e., reaching 128 bits of security. To do this, we have carefully chosen the parameters n, k and q such that all the attack procedures described in Section 6 require more than 2^{128} (classical) operations. Among the wide range of secure instances, we have selected the following one

$$(n,\ k,\ q) = (68,\ 32,\ 23),$$

leading to a public key of $12,240$ bits, that is, just over 1.5 kB. This quantity also corresponds to the size of the messages exchanged by Alice and Bob.

With the above choices, we have $2k - n = 4$, and the expected dimension of the secret subcode (computed through Equation (8)) is roughly equal to 4. The distribution of the subcode dimension, which can be estimated through Proposition 4, is strongly centered around 4, and the probability of having a subcode of dimension 7 is already negligible (i.e., smaller than 2^{-128}).

The bottleneck in the time complexity is represented by the computation of the weight enumerator function computation, which dominates the other elementary linear algebra operations. This allows us to obtain a rough upper bound on the required algorithmic cost, by taking into account the naive approach we have explained in the previous section, yielding $68 \cdot 36 \cdot 23^4 \approx 2^{29}$ operations in $\mathbb{F}_q$. We point out that, with ad hoc techniques such as those described in the previous section, the complexity of this step may be significantly reduced.

To provide a concrete example on the practicality of our proposed scheme, we have considered a proof-of-concept implementation using the Magma Computational Algebra system, running on a 3.2 GHz Quad-Core Intel Core i5 processor, with 16 GB of RAM. Our timing measurements have been obtained through $10,000$ runs of the protocol. The subcode dimension has always resulted to be equal to 4, while the average execution time for a complete run of the protocol, i.e., considering both Alice's and Bob's operations, is 5.8 milliseconds. This is to be compared with the implementation of the CSIDH non-interactive key-exchange protocol, at a security level comparable to NIST Category 1, which runs in 100 milliseconds and has a key of 64 bytes [13, Table 1].

9 Conclusion

In this work, we have introduced a post-quantum non-interactive key-exchange protocol based on problems coming from coding theory. The protocol follows the Diffie-Hellman framework, but uses an entirely new paradigm for post-quantum schemes, relying solely on the hardness of the Code Equivalence problem. This is in contrast, for example, with the traditional method used in code-based cryptography, which revolves around the hardness of syndrome decoding. Since security

does not depend on error-correction capability and other decoding-related notions, we are able to use entirely random codes, and choose parameters resulting in very small lengths. This is again a substantial difference: where traditional code-based schemes have codes of lengths in the order of thousands, our suggested instance has only $n = 68$. As a consequence, we obtain small key sizes and very fast timings.

To the best of our knowledge, our protocol is the first instantiation of a post-quantum Diffie-Hellman key-exchange, that is not based on supersingular isogenies. Compared to isogeny-based schemes, we benefit from a very efficient arithmetic (essentially just linear algebra) which results in a noticeable computational advantage. On the other hand, our keys are obviously quite far from the extremely small sizes typical of isogeny schemes, thus presenting a significant trade-off. Note that, unless the bandwidth allocated for the protocol is extremely limited, this trade-off works in our favor, and the sizes of all the objects involved in the protocol are of the same order of magnitude (if not smaller) as the existing post-quantum, and especially code-based, alternatives (e.g. [4, 15]).

In the end, we see great potential in this work, not only for the protocol here described, but also as an avenue for building other types of schemes, based on our original construction. Future work naturally includes the study and development of our paradigm, as well as many optimizations, and especially a high-performance implementation, that can further highlight the strengths of our approach.

References

1. L. Babai. Graph Isomorphism in Quasipolynomial Time [extended abstract]. In D. Wichs and Y. Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 684–697. ACM, 2016.
2. M. Bardet, A. Otmani, and M. Saeed-Taha. Permutation Code Equivalence is Not Harder Than Graph Isomorphism When Hulls Are Trivial. In *2019 IEEE International Symposium on Information Theory (ISIT)*, pages 2464–2468, July 2019.
3. C. H. Bennett. Time/Space Trade-offs for Reversible Computation. *SIAM Journal on Computing*, 18(4):766–776, 1989.
4. https://bikesuite.org/.
5. R. Canetti and H. Krawczyk. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 453–474. Springer, 2001.
6. W. Castryck, T. Lange, C. Martindale, L. Panny, and J. Renes. Csidh: An Efficient Post-Quantum Commutative Group Action. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 395–427. Springer, 2018.
7. B. Den Boer. Diffie-Hellman is as Strong as Discrete Log for Certain Primes. In *Conference on the Theory and Application of Cryptography*, pages 530–539. Springer, 1988.

8. H. Dinh, C. Moore, and A. Russell. McEliece and Niederreiter Cryptosystems That Resist Quantum Fourier Sampling Attacks. In P. Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, pages 761–779, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
9. H. Dinh, C. Moore, and A. Russell. Limitations of Single Coset States and Quantum Algorithms for Code Equivalence. *Quantum Info. Comput.*, 15(3-4):260–294, Mar. 2015.
10. J. A. Grochow. Matrix Lie Algebra Isomorphism. In *IEEE Conference on Computational Complexity (CCC12)*, pages 203–213, 2011.
11. L. Grover. A Fast Quantum Mechanical Algorithm for Database Search. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 212–219, New York, NY, USA, 1996. ACM.
12. A. Hauteville. Décodage en Métrique Rang et Attaques sur un Système de Chiffrement à Base de Codes LRPC. Master's thesis, Université de Limoges, France, Sept. 2014.
13. A. Jalali, R. Azarderakhsh, M. M. Kermani, and D. Jao. Towards Optimized and Constant-Time CSIDH on Embedded Devices. In I. Polian and M. Stöttinger, editors, *Constructive Side-Channel Analysis and Secure Design - 10th International Workshop, COSADE 2019, Darmstadt, Germany, April 3-5, 2019, Proceedings*, volume 11421 of *Lecture Notes in Computer Science*, pages 215–231. Springer, 2019.
14. D. Jao and L. De Feo. Towards Quantum-Resistant Cryptosystems from Supersingular Elliptic Curve Isogenies. In *International Workshop on Post-Quantum Cryptography*, pages 19–34. Springer, 2011.
15. https://www.ledacrypt.org/.
16. J. Leon. Computing Automorphism Groups of Error-Correcting Codes. *IEEE Transactions on Information Theory*, 28(3):496–511, May 1982.
17. U. M. Maurer. Towards the Equivalence of Breaking the Diffie-Hellman Protocol and Computing Discrete Logarithms. In *Annual international cryptology conference*, pages 271–281. Springer, 1994.
18. https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization.
19. R. Pellikaan, X.-W. Wu, and S. Bulygin. *Codes, cryptology and curves with computer algebra*. Cambridge University Press, 2017.
20. E. Petrank and R. M. Roth. Is Code Equivalence Easy to Decide? *IEEE Transactions on Information Theory*, 43(5):1602–1604, Sep. 1997.
21. N. Sendrier. The Support Splitting Algorithm. *Information Theory, IEEE Transactions on*, pages 1193 – 1203, 08 2000.
22. N. Sendrier and D. E. Simos. The Hardness of Code Equivalence over $\mathbb{F}_q$ and Its Application to Code-Based Cryptography. In P. Gaborit, editor, *Post-Quantum Cryptography*, pages 203–216, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
23. N. Sendrier and D. E. Simos. How Easy Is Code Equivalence over $\mathbb{F}_q$? 2013.
24. N. Sendrier and P. Symbolique. On the Dimension of the Hull. *SIAM Journal on Discrete Mathematics*, 10, 11 1995.
25. P. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.