Structural Cryptanalysis of Polar-KEM

Direct Recovery of the Secret Isometry

Yuyang Xiao Long Chen Zhenfeng Zhang

Institute of Software, Chinese Academy of Sciences
Beijing, China

{xiaoyuyang2023,chenlong,zhenfeng}@iscas.ac.cn

Abstract

Polar-KEM is a lattice-based key encapsulation mechanism submitted to the Next-generation Commercial Cryptographic Algorithms Program. Its security is claimed to rely on a lattice isomorphism problem over polar-code-defined Construction-D lattices. In its core key-generation algorithm, a polar lattice basis is constructed from a polar-code chain, reduced using LLL, and left-multiplied by a secret orthogonal matrix. The resulting matrix is published as the public key.

We show that the key-generation procedure does not instantiate a general lattice isomorphism problem. The reduced reference basis is generated through a public chain of computations: public parameters determine frozen sets, the frozen sets determine a polar-code chain, the code chain determines a Construction-D basis, and the basis is then reduced by the public LLL procedure. Under the natural executable interpretation of the specification, an adversary can reproduce the same reduced basis.

If the public key is $B_{\mathsf{pk}} = OB_{\mathsf{red}}$, the secret orthogonal matrix is recovered by the single linear-algebra operation

$$O = B_{\mathsf{pk}} B_{\mathsf{red}}^{-1}.$$

The attack is polynomial time and recovers the exact secret isometry, or its numerical representation, without solving a shortest-vector problem or a general lattice isomorphism problem.

We also examine the erasure parameter used by the frozen-set selection algorithm. The specification neither assigns it a concrete value nor defines it as a sampled secret with a distribution, encoding, entropy requirement, or storage mechanism. Treating this unspecified value or unspecified LLL tie-breaking as secret entropy would define a different scheme and would not repair the published construction.

The present paper focuses on this direct basis-alignment weakness. Additional structural and correctness analyses are left to an extended version.

Keywords. Post-quantum cryptography, key encapsulation, lattice isomorphism, polar codes, Construction D, key recovery, structural cryptanalysis.

1 Introduction

1.1 NGCC and public cryptanalysis

The Institute of Commercial Cryptography Standards launched the Next-generation Commercial Cryptographic Algorithms Program, or NGCC, to solicit and publicly evaluate new cryptographic algorithms. Public evaluation is especially important for post-quantum public-key algorithms, where a design may combine a familiar primitive with a new hardness assumption or a new structured family of mathematical objects.

Independent analysis of an algorithm submission has two complementary goals. The first is to evaluate the assumed hard problem in the distribution actually generated by the scheme. The second is to check whether the formal algorithms correctly instantiate that distribution. Even if a general computational problem is hard, a concrete key-generation algorithm may accidentally output an easy special case.

1.2 Polar-KEM at a glance

Polar-KEM is presented as a post-quantum key encapsulation mechanism based on the lattice isomorphism problem over polar-code-defined lattices [Pol26]. The intended construction consists of the following components:

1. A chain of nested polar codes is generated from a chain of frozen sets.
2. Construction D is applied to the code chain to produce an integer lattice basis.
3. The basis is reduced using LLL.
4. A secret orthogonal matrix is sampled.
5. The orthogonal image of the reduced basis is published as the public key.
6. Encapsulation embeds a message in the public lattice and adds a short error.
7. Decapsulation applies the inverse secret isometry and then decodes in the original polar lattice.

In the notation of the specification, the central key-generation equation is

$$B_{\mathsf{pk}} = OB_{\mathsf{red}},$$

where $O \in \mathrm{O}_N(\mathbb{R})$ is the secret key and $B_{\mathsf{red}}$is the LLL-reduced polar-lattice basis.

1.3 Our result

We show that the matrix relation above exposes the secret isometry. The basis $B_{\mathsf{red}}$is not sampled as secret data. It is produced by the following chain:

$$(N, L, \varepsilon) \longrightarrow (F_0, \ldots, F_{L-1}) \longrightarrow (C_0, \ldots, C_{L-1}) \longrightarrow B \longrightarrow B_{\mathsf{red}}.$$

Each arrow corresponds to a public algorithm in the Polar-KEM specification:

- Algorithm 10 selects frozen sets from N, L, ε.
- The frozen sets select rows of the public polar transform and thereby define the polar-code chain.
- Section 4.5.2 applies Construction D to that code chain.
- Algorithm 1 applies LLL to the resulting basis.

The only object explicitly sampled as a secret in the core key-generation algorithm is the orthogonal matrix O. Once an adversary reproduces the basis B_{red}used by key generation, the secret isometry follows from

$$\widehat{O} = B_{\mathsf{pk}} B_{\mathsf{red}}^{-1} = O.$$

This is not an algorithm for the general lattice isomorphism problem. In a general instance, two bases may satisfy

$$B_2 = OB_1U$$

for an unknown unimodular change of basis $U \in \mathrm{GL}_N(\mathbb{Z})$. Polar-KEM Algorithm 1 instead publishes the aligned image

$$B_{\mathsf{pk}} = OB_{\mathsf{red}}$$

without an unknown right-hand change of basis. The public key therefore reveals the action of the secret linear transformation on a complete, publicly reproducible basis.

1.4 Contributions

Our contributions are as follows.

1. We give a step-by-step dependency analysis of Polar-KEM key generation, from public parameters through frozen-set selection, polar-code generation, Construction D, and LLL reduction.
2. We identify the exact conditions under which the reduced reference basis can be reproduced by an adversary. These are the natural conditions required for an executable and interoperable specification.
3. We give a polynomial-time key-recovery algorithm that computes the secret orthogonal matrix using one public-basis reconstruction and one linear solve.
4. We prove the correctness of the recovery and explain why it attacks an aligned-basis special case rather than the general lattice isomorphism problem.
5. We analyze the status of the erasure parameter ε. The specification uses this value in frozen-set selection but does not provide a concrete value, a secret sampling procedure, an entropy requirement, or a private-key representation.
6. We explain why the actual key-generation distribution does not instantiate the hard distribution claimed by the polar-LIP assumption.

1.5 Scope

This paper focuses on direct recovery of the secret isometry from Algorithm 1. It does not rely on the minimum distance of the specified polar lattice, the size of its automorphism group, or a Gram-matrix reduction attack. These questions are independent of the direct basis-alignment weakness and may be incorporated into an extended version.

1.6 Organization

Section 2 reviews lattices, LIP, polar codes, and Construction D. Section 3 describes the relevant parts of Polar-KEM. Section 4 proves that the reference basis is publicly reproducible under the natural interpretation of the specification. Section 5 gives and proves the key-recovery attack. Section 6 discusses the security claims, specification ambiguity, possible counterarguments, and the scope of the result.

2 Background

2.1 Lattices and basis conventions

A full-rank lattice in $\mathbb{R}^N$ is the integer span of N linearly independent vectors. Using the column-basis convention, a basis matrix is

$$B = [b_1 \mid b_2 \mid \cdots \mid b_N] \in \mathbb{R}^{N \times N},$$

and the corresponding lattice is

$$\mathcal{L}(B) = \{Bz : z \in \mathbb{Z}^N\}.$$

Two basis matrices generate the same lattice if and only if they differ by a right-hand unimodular transformation:

$$\mathcal{L}(B) = \mathcal{L}(BU)$$

for

$$U \in \mathrm{GL}_N(\mathbb{Z}), \qquad \det U \in \{-1, +1\}.$$

The Gram matrix of a basis is

$$G(B) = B^{\mathsf{T}} B.$$

It records all pairwise inner products between the ordered basis vectors.

We use the column-basis convention because it is the convention used in Polar-KEM Definition 2.1, Definition 2.6, Algorithm 1, and the decapsulation equation. The specification also contains a row-stacking description in its Construction-D section. A consistent row-basis interpretation transposes the relevant equations but does not remove the direct-recovery phenomenon.

2.2 Orthogonal transformations

A real matrix O is orthogonal if

$$O^{\mathsf{T}} O = O O^{\mathsf{T}} = I.$$

Consequently,

$$O^{-1} = O^{\mathsf{T}}.$$

An orthogonal matrix preserves inner products and Euclidean norms:

$$\langle Ox, Oy \rangle = \langle x, y \rangle$$

and

$$\|Ox\|_2 = \|x\|_2.$$

If B is a basis of a lattice Λ, then OB is a basis of the isometric lattice $O\Lambda$.

2.3 The general lattice isomorphism problem

The Polar-KEM specification describes the lattice isomorphism problem in the following form. Given bases B_1 and B_2 of isometric lattices, find matrices

$$O \in \mathrm{O}_N(\mathbb{R})$$

and

$$U \in \mathrm{GL}_N(\mathbb{Z})$$

such that

$$B_2 = O B_1 U.$$

The matrix U represents the fact that an isometry maps a lattice to a lattice, but the supplied basis of the target lattice need not be the direct image of the supplied source basis.

The corresponding Gram matrices satisfy

$$B_2^{\mathsf{T}} B_2 = U^{\mathsf{T}} B_1^{\mathsf{T}} B_1 U.$$

Thus, in a general LIP instance, the ordered columns of B_1 and B_2 are not known to correspond under the secret orthogonal map.

2.4 The aligned-basis special case

We isolate the special case used by Polar-KEM Algorithm 1.

Proposition 2.1 (Recovery from aligned bases) *Let $B_1 \in \mathbb{R}^{N\times N}$ be invertible and let $O \in \mathrm{O}_N(\mathbb{R})$. If*

$$B_2 = OB_1$$

and both B_1 and B_2 are known, then O is uniquely determined by

$$O = B_2 B_1^{-1}.$$

Proof. Right-multiplying the equality $B_2 = OB_1$ by B_1^{-1} gives

$$B_2 B_1^{-1} = OB_1 B_1^{-1} = O.$$

Because B_1 is a full-rank basis, its inverse exists. □

The distinction between

$$B_2 = OB_1$$

and

$$B_2 = OB_1 U$$

is central to our analysis. The former reveals the value of the orthogonal map on a known complete basis and is solved by ordinary linear algebra. The latter additionally requires recovery of an unknown integer change of basis.

2.5 Polar codes

Let

$$F = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}.$$

For block length $N = 2^m$, the polar transform is

$$G_N = B_N F^{\otimes m},$$

where B_N is the bit-reversal permutation matrix.

A polar code is specified by a frozen set

$$F_\ell \subseteq \{0, \ldots, N-1\}.$$

The information set is its complement:

$$I_\ell = \{0, \ldots, N-1\} \setminus F_\ell.$$

A generator matrix for the corresponding code is obtained by selecting from G_N the rows indexed by I_ℓ. Hence, once N and F_ℓ are fixed, the code C_ℓ is fixed.

Polar-KEM uses a nested chain

$$F_0 \subseteq F_1 \subseteq \cdots \subseteq F_{L-1},$$

which corresponds to a nested code chain

$$C_0 \supseteq C_1 \supseteq \cdots \supseteq C_{L-1}.$$

2.6 Construction D

Given the nested binary-code chain, Polar-KEM defines a Construction-D lattice of the form

$$\Lambda = C_0 + 2C_1 + \cdots + 2^{L-1}C_{L-1} + 2^L\mathbb{Z}^N.$$

At the implementation level, the specification constructs scaled generators from the selected polar-code rows and appends the generators of $2^L\mathbb{Z}^N$. A basis-extraction or normal-form step is implicitly necessary because the displayed stack may contain more than N generators, whereas Algorithm 1 requires a square full-rank basis.

For the present attack, the exact basis-extraction convention is not important provided that it is part of the public key-generation algorithm and can be reproduced from the same public inputs.

2.7 LLL reduction

The LLL algorithm takes an integer or rational lattice basis and returns a reduced basis of the same lattice [LJL82]. If

$$B_{\mathsf{red}} = \mathsf{LLL}(B),$$

then there exists a unimodular matrix V such that

$$B_{\mathsf{red}} = BV.$$

An LLL-reduced basis is not mathematically unique. Its concrete output may depend on reduction parameters, arithmetic conventions, tie-breaking, and implementation details. This fact must not be confused with cryptographic secret randomness.

For a fixed executable implementation, fixed input, fixed reduction parameters, and fixed tie-breaking rules, the execution and output are reproducible. If a specification does not define these choices, it has an interoperability ambiguity; it does not automatically obtain a secret key from that ambiguity.

3 The Polar-KEM Construction

This section summarizes only the components needed for the direct key-recovery attack. Our description follows the Polar-KEM draft specification [Pol26].

3.1 Public parameters

The specification gives three parameter sets. The parameters relevant to constructing the polar lattice are shown in Table 1.

Table 1: Polar-KEM parameters relevant to lattice construction.

Parameter set	N	q	L
PolarKEM-128	512	12289	5
PolarKEM-256	1024	12289	5
PolarKEM-512	2048	18433	6

The dimension N, modulus q, number of Construction-D levels L, rate progression, error distribution, and algorithm descriptions are public scheme parameters.

3.2 Frozen-set selection

For each level ℓ, the specification sets the code dimension according to the rate progression

$$R_\ell = 2^{-\ell}$$

and

$$K_\ell = \left\lfloor \frac{N}{2^\ell} \right\rfloor .$$

Algorithm 10, denoted here by $\mathsf{SelectFrozenSets}(N, L, \varepsilon)$, computes reliability or Bhattacharyya values for a channel parameterized by ε, orders the indices, selects the prescribed number of frozen positions, and enforces the nesting relation

$$F_0 \subseteq F_1 \subseteq \cdots \subseteq F_{L-1}.$$

For fixed input values and fixed tie-breaking, these operations contain no sampling step. Thus

$$(F_0, \ldots, F_{L-1}) = \mathsf{SelectFrozenSets}(N, L, \varepsilon)$$

is deterministic.

The status of εrequires separate discussion. The specification describes it as an erasure probability corresponding to the error distribution, but the core key-generation algorithm lists only N, q, L as required parameters. The parameter tables do not assign a concrete value to ε. No algorithm samples εas secret data, and neither the core secret key nor the full secret-key format stores it. We return to this issue in Section 4.6.

3.3 From frozen sets to the code chain

Once the frozen set F_ℓ is known, the corresponding polar code is obtained from the public polar transform

$$G_N = B_N F^{\otimes m}.$$

The information set is

$$I_\ell = \{0, \ldots, N-1\} \setminus F_\ell,$$

and the generator matrix G_ℓ is formed by selecting the rows of G_N indexed by I_ℓ.

Therefore,

$$F_\ell \longrightarrow I_\ell \longrightarrow G_\ell \longrightarrow C_\ell$$

is a public deterministic computation. No secret seed, secret permutation, or secret change of basis appears in this step.

3.4 Construction-D basis generation

Section 4.5.2 of the specification constructs the lattice from the scaled code generators

$$G_0, \quad 2G_1, \quad \ldots, \quad 2^{L-1}G_{L-1},$$

together with

$$2^L I_N.$$

Abstracting over the row-versus-column formatting issue in the displayed stack, we denote the basis produced by the specified public basis-construction procedure by

$$B = \mathsf{ConstructionD}(C_0, \ldots, C_{L-1}).$$

Algorithm 1 treats B as an $N \times N$ integer basis. The construction takes only the code chain and public level scalings as input. It does not sample a random unimodular matrix or any other secret basis representation.

3.5 Basis reduction

Algorithm 1 next computes

$$B_{\mathsf{red}} = \mathsf{LLL}(B).$$

No secret randomness is specified for this invocation. The reduced basis is not included in the secret key because the specification treats it as an intermediate value derived from the constructed polar lattice.

For a fixed public implementation, an adversary can invoke the same LLL routine on the same basis B, with the same parameters and tie-breaking, and reproduce the same B_{red}.

3.6 Core key generation

The core key-generation algorithm can be summarized as follows.

Listing 1: Core key generation in Polar-KEM.

```
Input: public parameters (N, q, L)

1. Construct the nested polar-code chain.
2. Construct an integer basis B using Construction D.
3. Compute B_red <- LLL(B).
4. Sample a secret orthogonal matrix O.
5. Compute B_pk <- O * B_red.
6. Return pk = B_pk and sk = O.
```

The essential output relation is

$$\boxed{B_{\mathsf{pk}} = OB_{\mathsf{red}}.}$$

The only explicit secret sampling step in the core algorithm is the sampling of O. In particular, Algorithm 1 does not sample a secret unimodular matrix U and does not publish $OB_{\mathsf{red}}U$.

3.7 Encapsulation and decapsulation

The core encapsulation algorithm embeds a message in a lattice vector $v \in \mathcal{L}(B_{\mathsf{pk}})$, samples a short error e, and forms

$$c = v + e \pmod{q}.$$

Core decapsulation applies the secret inverse isometry:

$$c' = O^{-1}c.$$

It then invokes the polar-lattice decoder in the original coordinate system.

The full KEM stores the core secret together with an implicit-rejection seed and the public key. The Fujisaki–Okamoto-style transformation does not change the core equation

$$B_{\mathsf{pk}} = OB_{\mathsf{red}}$$

or hide O after it has been recovered.

4 Public Reconstructibility of the Reference Basis

The key question is whether an adversary can obtain the exact reduced basis B_{red}used in

$$B_{\mathsf{pk}} = OB_{\mathsf{red}}.$$

We analyze every dependency in its construction.

4.1 Dependency chain

The construction can be represented as

$$(N, L, \varepsilon) \xrightarrow{\text{Alg. 10}} (F_0, \ldots, F_{L-1}) \longrightarrow (C_0, \ldots, C_{L-1}) \xrightarrow{\text{Construction D}} B \xrightarrow{\text{LLL}} B_{\mathsf{red}}.$$

Table 2 records the input and randomness of each step.

Table 2: Dependency chain for the reduced reference basis.

Object	Specification step	Inputs	Secret sampling	Reproducible
K_ℓ	Rate progression	N, L, ℓ	None	Yes
F_ℓ	Algorithm 10	N, L, ε	None specified	Yes for fixed inputs
C_ℓ	Polar-code construction	G_N, F_ℓ	None	Yes
B	Construction D	Code chain and L	None	Yes for a fixed basis convention
B_{red}	Algorithm 1	B	None specified	Yes for a fixed LLL implementation
O	Algorithm 1	Orthogonal sampling	Yes	Attack target
B_{pk}	Algorithm 1	O, B_{red}	Only through O	Published

4.2 Reconstructing the frozen sets

For each level, the dimension K_ℓ is fixed by the public rate progression. Algorithm 10 computes channel-reliability values, sorts positions by those values, and selects the prescribed number of frozen indices.

For fixed N, L, ε and fixed tie-breaking, this is an ordinary deterministic computation:

$$(F_0, \ldots, F_{L-1}) = \mathsf{SelectFrozenSets}(N, L, \varepsilon).$$

There is no instruction of the form "sample a random frozen set" and no secret seed is supplied to the selection algorithm.

If reliability values are equal, the implementation must resolve the tie in some way. A public reference implementation may, for example, use the natural index order or the behavior of a fixed stable sorting routine. Reproducing that behavior is sufficient for the attack.

An unspecified tie-breaking rule is a specification ambiguity. It does not create a cryptographically analyzed secret because no tie-breaking key is sampled, stored, or passed to decapsulation.

4.3 Reconstructing the polar-code chain

For each recovered frozen set, the attacker computes

$$I_\ell = \{0, \ldots, N-1\} \setminus F_\ell$$

and reconstructs the public polar transform

$$G_N = B_N F^{\otimes m}.$$

Selecting the rows indexed by I_ℓ gives the same generator matrix used by key generation:

$$G_\ell = G_N[I_\ell, :].$$

Consequently, the attacker obtains the entire nested chain

$$C_0 \supseteq C_1 \supseteq \cdots \supseteq C_{L-1}.$$

This step uses no information derived from O.

4.4 Reconstructing the Construction-D basis

The attacker applies the Construction-D basis procedure to the reconstructed code chain, with the same public level scalings:

$$B = \mathsf{ConstructionD}(C_0, \ldots, C_{L-1}).$$

If the displayed generating set contains more than N vectors, the executable key-generation algorithm must select or compute a full-rank square basis. The attacker uses the same public basis extraction, normal form, or linear-algebra procedure.

No secret random change of basis appears in Algorithm 1. In particular, the specification does not contain a step

$$U \leftarrow \mathrm{GL}_N(\mathbb{Z})$$

followed by

$$B \leftarrow BU.$$

Therefore, the concrete basis representation entering LLL is reproducible from the same inputs.

4.5 Reproducing the reduced basis

The attacker invokes the same LLL implementation as key generation:

$$\widehat{B}_{\mathsf{red}} = \mathsf{LLL}(B).$$

Under a fixed implementation,

$$\widehat{B}_{\mathsf{red}} = B_{\mathsf{red}}.$$

It is important to state the condition correctly. An "LLL-reduced basis" is not unique as a mathematical object, so the claim is not that all possible LLL implementations return the same matrix. The claim is that a concrete public key-generation procedure must execute a concrete reduction algorithm. If that implementation is public, the attacker can reproduce it.

If the specification permits incompatible implementations to use different LLL parameters or tie-breaking behavior, then the specification does not uniquely define key generation. Such underspecification cannot be treated as hidden entropy unless the scheme explicitly introduces, samples, stores, and analyzes it as secret key material.

4.6 The status of the erasure parameter

The parameter $\varepsilon$is the only apparent missing input in the public dependency chain. We distinguish four interpretations.

Fixed public design parameter. If $\varepsilon$is a fixed channel-design parameter, then it is part of the public parameter set and the attack directly reproduces the frozen sets.

Derived from the public error distribution. Section 4.5.1 describes the channel erasure probability as corresponding to the error distribution. If $\varepsilon$is deterministically derived from the public error parameters, it remains public and reproducible.

Unspecified parameter. If no value or derivation is supplied, then key generation is underspecified. Independent implementations cannot be expected to construct the same code chain or produce reproducible test vectors. For every implementation that fixes or hard-codes a value, the attack applies to that implementation.

Retrospectively secret parameter. The specification does not define εas a secret. In particular, it does not specify:

1. a finite sampling domain for ε;
2. a probability distribution;
3. an encoding precision;
4. a minimum-entropy requirement;
5. a collision analysis for distinct values producing the same frozen-set chain;
6. a private-key field storing εor the frozen sets;
7. a decapsulation procedure obtaining the secret frozen sets; or
8. a security reduction involving this additional secret.

Declaring εsecret after the fact would therefore define a modified scheme rather than interpret the published Algorithm 1.

Moreover, the relevant effective secret would not be the bit length of an arbitrary floating-point encoding of ε. It would be the number and distribution of distinct frozen-set chains induced by the map

$$\varepsilon \longmapsto (F_0, \ldots, F_{L-1}).$$

The specification gives no security analysis of this map.

4.7 Reconstruction proposition

Proposition 4.1 (Public reconstruction of the reduced basis) *Assume that:*

1. *the polar-code design inputs are public or publicly derived;*
2. *frozen-set selection uses the public specified algorithm;*
3. *Construction D uses a fixed public basis convention; and*
4. *key generation uses a fixed public LLL implementation.*

Then a polynomial-time adversary can reproduce the exact matrix B_{red}used by Polar-KEM Algorithm 1.

Proof. The adversary first computes the dimensions K_ℓ from the public rate progression. It runs Algorithm 10 on the same inputs to obtain the frozen sets. It reconstructs the public polar transform and selects the same information rows, obtaining the same code chain. It then runs

the public Construction-D basis procedure, obtaining the same basis B. Finally, it executes the same LLL implementation on the same input and obtains

$$\widehat{B}_{\mathsf{red}} = \mathsf{LLL}(B) = B_{\mathsf{red}}.$$

Every step is polynomial time in the dimension and the bit length of the represented matrices. □

Remark 4.2 *The proposition describes the natural executable interpretation of the specification. If any listed convention is not fixed, the result is an underspecified key-generation algorithm, not a documented cryptographic secret.*

5 Direct Recovery of the Secret Isometry

We now combine public basis reconstruction with the aligned public-key equation.

5.1 Attack algorithm

The adversary receives the public parameters and public-key matrix B_{pk}. It executes the following procedure.

Listing 2: Direct recovery of the Polar-KEM secret isometry.

```
Algorithm RecoverIsometry(params, B_pk)

Input:
    Public Polar-KEM parameters params
    Public basis matrix B_pk

1. Determine the channel-design parameter epsilon from the
   public parameter set or the public implementation.

2. Reconstruct the frozen-set chain:
       (F_0, ..., F_{L-1})
           <- SelectFrozenSets(N, L, epsilon).

3. Reconstruct the polar-code chain:
       C_l <- PolarCode(G_N, F_l)
   for l = 0, ..., L-1.

4. Reconstruct the Construction-D basis:
       B <- ConstructionD(C_0, ..., C_{L-1}).

5. Reproduce the key-generation reduction:
       B_red <- LLL(B).

6. Solve the matrix equation:
       O_hat <- B_pk * inverse(B_red).

7. Verify:
       O_hat^T * O_hat = I,
       O_hat * B_red = B_pk.

8. Return O_hat.
```

In an implementation, Step 6 need not explicitly compute a matrix inverse. The attacker may instead solve

$$XB_{\mathsf{red}} = B_{\mathsf{pk}}$$

for X. This is numerically and computationally preferable in many linear-algebra libraries.

5.2 Correctness

Theorem 5.1 (Direct recovery of the secret isometry) *Suppose Polar-KEM Algorithm 1 produces*

$$B_{\mathsf{pk}} = OB_{\mathsf{red}},$$

where $O \in \mathrm{O}_N(\mathbb{R})$, and suppose the adversary reconstructs the same full-rank matrix B_{red}. Then RecoverIsometry *outputs the exact core secret key O.*

Proof. The algorithm computes

$$\widehat{O} = B_{\mathsf{pk}} B_{\mathsf{red}}^{-1}.$$

Substituting the key-generation equation gives

$$\widehat{O} = OB_{\mathsf{red}} B_{\mathsf{red}}^{-1} = OI = O.$$

Thus the recovered matrix is exactly the matrix sampled by core key generation. □

5.3 Public verification

The attacker can verify the result without access to any additional secret data. Exact arithmetic permits the checks

$$\widehat{O}^{\mathsf{T}} \widehat{O} = I$$

and

$$\widehat{O} B_{\mathsf{red}} = B_{\mathsf{pk}}.$$

The first check confirms orthogonality. The second confirms that the recovered matrix maps the reconstructed reference basis to the published public basis.

If matrices are serialized at finite precision, the corresponding checks become

$$\left\| \widehat{O}^{\mathsf{T}} \widehat{O} - I \right\| \leq \delta_1$$

and

$$\left\| \widehat{O} B_{\mathsf{red}} - B_{\mathsf{pk}} \right\| \leq \delta_2,$$

where the tolerances depend on the public-key encoding and arithmetic precision.

Finite precision does not provide a cryptographic defense by itself. The legitimate decapsulator must also retain sufficient precision to apply O^{-1} and decode correctly. A complete implementation-level analysis should use the exact serialization rules, which the draft specification does not fully define.

5.4 Complexity

The attack consists of:

1. reconstructing the frozen sets;
2. reconstructing the polar-code generator matrices;
3. constructing the same Construction-D basis;
4. executing the same LLL reduction as key generation; and
5. solving one dense matrix equation.

All these operations are polynomial time in the represented input size. The final linear solve requires

$$O(N^\omega)$$

field operations, where ω is the matrix multiplication exponent and ordinary Gaussian elimination gives $\omega = 3$.

The attack does not enumerate orthogonal matrices and does not invoke SVP, CVP, BKZ, or an automorphism-group algorithm. Its cost is comparable to publicly re-executing the deterministic part of key generation, followed by one linear solve.

5.5 Experimental confirmation

The accompanying structural analysis used exact rational arithmetic and constructed orthogonal matrices from rational 3-4-5 rotations. A basic two-dimensional block has the form

$$R = \begin{pmatrix} 3/5 & -4/5 \\ 4/5 & 3/5 \end{pmatrix},$$

for which

$$R^{\mathsf{T}} R = I.$$

Placing such rotations in disjoint coordinate planes produces higher-dimensional rational orthogonal matrices. Exact arithmetic allows entry-by-entry verification of

$$\widehat{O} = O.$$

The reported recovery results are summarized in Table 3.

Table 3: Exact recovery reported for the direct attack.

Instance	N	L	Exact recovery	Linear-solve time
Small instance	32	5	Yes	0.001 s
Small instance	128	5	Yes	0.010 s
PolarKEM-128	512	5	Yes	0.235 s
PolarKEM-256	1024	5	Yes	1.169 s

For the $N = 1024$ experiment, most of the reported total time was spent constructing public matrices and checking determinants, rather than recovering the secret isometry. The final recovery step was ordinary exact linear algebra.

These measurements are implementation-dependent and are not intended as optimized benchmarks. Their purpose is to confirm the algebraic identity at the concrete parameter dimensions.

5.6 Why this is not a general LIP attack

It is essential to distinguish our result from an algorithm for general LIP.

In a general LIP instance, the public bases satisfy

$$B_2 = OB_1U$$

for an unknown

$$U \in \mathrm{GL}_N(\mathbb{Z}).$$

Naively multiplying by B_1^{-1} gives

$$B_2 B_1^{-1} = OB_1UB_1^{-1},$$

which is generally not equal to O. Recovering the unknown integer basis transformation is part of the problem.

Polar-KEM Algorithm 1 instead publishes

$$B_{\mathsf{pk}} = OB_{\mathsf{red}}.$$

This is the special case

$$U = I.$$

Writing

$$B_{\mathsf{red}} = [b_1 \mid \cdots \mid b_N]$$

and

$$B_{\mathsf{pk}} = [b'_1 \mid \cdots \mid b'_N],$$

the public-key equation states that

$$b'_i = Ob_i$$

for every $i \in \{1, \ldots, N\}$.

The attacker therefore knows the image of every vector in a complete basis under the secret linear map. A linear map is uniquely determined by its action on a basis. The attack exploits this exposed column-by-column correspondence; it does not solve the hidden correspondence problem present in general LIP.

5.7 Consequences for encapsulation

The recovered matrix is the core secret key. An attacker can compute

$$\widehat{O}^{-1} = \widehat{O}^{\mathsf{T}} = O^{-1}$$

and execute the same coordinate transformation as the legitimate decapsulator.

Consequently, any confidentiality claim that relies on O remaining unknown fails. A Fujisaki–Okamoto-style transform cannot repair public-key recovery of the complete core decryption key. Such a transform assumes an underlying primitive satisfying its required security properties; it does not hide a secret key that is already a polynomial-time function of the public key and public parameters.

6 Conclusion

We analyzed the core key-generation algorithm of Polar-KEM. The specification publishes

$$B_{\mathsf{pk}} = OB_{\mathsf{red}},$$

where O is the secret orthogonal matrix and $B_{\mathsf{red}}$is obtained from the polar-code chain through Construction D and LLL reduction. Under the natural executable interpretation of the specification, this chain depends only on public parameters and public algorithms. An adversary can therefore reproduce $B_{\mathsf{red}}$and recover the secret key by

$$O = B_{\mathsf{pk}}B_{\mathsf{red}}^{-1}.$$

The attack is a polynomial-time linear-algebra attack and does not solve the general lattice isomorphism problem. It succeeds because the public key is the aligned orthogonal image of a publicly reproducible basis, without an unknown unimodular change of basis. Consequently, Algorithm 1 does not instantiate the hard instance distribution described by the polar-LIP assumption.

The unspecified channel-design parameter $\varepsilon$and the nonuniqueness of LLL output do not constitute defined secret entropy. If fixed by an implementation, they are reproducible; if left unspecified, they make key generation ambiguous rather than secure. Since the attack recovers the core decapsulation key, a higher-level Fujisaki–Okamoto transform cannot restore confidentiality.

Possible design changes and future work. A revised construction must avoid publishing an aligned image of a publicly reproducible basis. One possible direction is to introduce a secret unimodular change of basis and publish

$$B_{\mathsf{pk}} = OB_{\mathsf{red}}U, \qquad U \in \mathrm{GL}_N(\mathbb{Z}).$$

This change blocks the direct recovery formula but does not by itself establish security. Future work should analyze the resulting public-key distribution, Gram-matrix and structural attacks, the exact Construction-D lattice, and decoding correctness.

References

[LJL82] Arjen K. Lenstra, Hendrik W. Lenstra Jr., and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

[Pol26] Polar-KEM Submitters. Polar-KEM: A lattice-based key encapsulation mechanism using polar-code-defined lattices. Algorithm Specification, Version 1.0 Draft, available at `https://www.niccs.org.cn/niccs/R1KeyEncapsulation/pc/list.html`, April 2026. Dated 30 April 2026.