

Superposition Key-Recovery Attacks on Dilithium and Fiat-Shamir Signature Schemes

Carlos Cid^{1,2}, David Elkouss¹, Manuel Goulão³

¹Okinawa Institute of Science and Technology Graduate University, Japan

²Simula UiB, Norway

³INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal

carlos.cid@oist.jp, david.elkouss@oist.jp, manuel.goulao@inesc-id.pt

Abstract

In this work, we assess the security of the main signature scheme standardised by NIST, *ML-DSA*, and its precursor identification schemes under an extended quantum adversarial model, one in which the adversary is allowed to use quantum resources to interact with a prover/signer. We begin our analysis by developing new techniques for superposition attacks that realise a relative phase oracle for the LSBs of a classical function evaluated in a quantum computer. We then extend these techniques to enable a relative phase oracle for arbitrary bits. Leveraging these techniques, we demonstrate full key-recovery attacks on several lattice-based identification schemes, exploiting the affine structure of the output. Finally, we extend these attacks to the corresponding Fiat-Shamir signature schemes and obtain key-recovery attacks under reasonable implementation assumptions. As a result, we are able to mount full key-recovery attacks in the Q2 model against the original *CRYSTALS-Dilithium* scheme, as well as the corresponding NIST Standardisation Rounds 1 and 2 proposals. However modifications introduced in the scheme during the NIST process preclude our superposition attacks against the final standardised version *ML-DSA*. We identify and discuss the specific aspects of the algorithm’s execution that affect the applicability of our techniques. Overall, our work expands the body of research on superposition attacks and represents a further step towards establishing full quantum security for cryptographic constructions.

1 Introduction

Emerging quantum technologies are having a major impact on the design and evaluation of cryptographic mechanisms used to protect our digital infrastructure. The expected development of large-scale fault-tolerant quantum computers capable of running Shor’s algorithm [Sho94] to solve the factorisation and discrete logarithm problems will make today’s most popular public-key cryptographic schemes completely ineffective. Although such quantum computers may still be several years away, it is now a standard practice for cryptographic schemes to be developed under this new adversary model, one which assumes adversaries may have access to a Cryptographically Relevant Quantum Computer (CRQC). This has led to the development of the field of *Post-Quantum Cryptography* (PQC). While PQC has attracted the attention of the academic community since at least the 1990s, the field has advanced significantly since 2016, spurred by the National Institute of Standards and Technology (NIST) PQC standardisation project. The project was established to select new standards for public-key encryption and signature schemes [Nat17], culminating in the publication of the first PQC Federal Information Processing Standards (FIPS) in August 2024.

Beyond the development of large-scale quantum computers, quantum technologies will also enable new modes of communication. This in turn may lead to extended forms of interaction with cryptographic schemes, for example when they are implemented on quantum devices and accessible via hybrid quantum-classical networks. This scenario calls for a further extension of the adversary model in a quantum era, as one may now be able to *quantumly* interact with cryptographic schemes via messages in superposition. This type of cryptanalysis is often called “quantum chosen message/plaintext/ciphertext”, “quantum superposition”, or simply “quantum” attacks. This setting lies beyond the post-quantum security model, as the adversary

is no longer restricted to (offline) access to a CRQC, but may also interact with the cryptosystem using quantum resources. Importantly, this attack setting does not require the underlying hardness assumption to be efficiently solvable with quantum computing resources; rather, it permits the manipulation of executions of the protocol on inputs in superposition, creating interference patterns that may be exploited for the recovery of the cryptographic secret.

In this work, we assess the security of the most prominent post-quantum signature scheme, the NIST standard *ML-DSA* [Nat24], under this extended quantum adversarial model, known as the *Q2* model [KLLN16b]. We begin by analysing the identification schemes in [Lyu08, Lyu09] based on Short Integer Solution (SIS)/Learning With Errors (LWE) (and their corresponding module versions, Module-SIS (MSIS), Module-LWE (MLWE)) under quantum superposition attacks. We show that they are vulnerable to full key-recovery attacks when the adversary is allowed to control and query the challenge in superposition.

Interestingly, these lattice-based ID schemes, which feature a Schnorr-like structure and can therefore be turned into a signature scheme via the Fiat-Shamir transform, may be viewed as the precursor of the *Dilithium/ML-DSA* signature schemes [BDK⁺21, DKL⁺18, Nat24]. We then extend our analysis to show that, under reasonable choices for their implementation in a quantum computer, the original *Dilithium* scheme [DKL⁺18], as well as the proposals for the NIST PQC standardisation Round 1 [DKL⁺17] and Round 2 [DKL⁺19], and the template version of Round 3 [BDK⁺21], are vulnerable to full key-recovery attacks in the *Q2* model. However, the attack does not directly apply to *ML-DSA* due to changes introduced during the NIST standardisation process. We discuss how the choices for their implementation and execution impact the effectiveness of our superposition attacks.

Our results contribute to the ongoing effort to establish the full quantum security of the *Dilithium/ML-DSA* signature scheme. In particular, it provides a different perspective on the importance of a full security proof in the Quantum Random Oracle Model (QROM), the generalisation of the Random Oracle Model (ROM) to the “quantum superposition model”, which has been the subject of ongoing research [BBD⁺23, DFP⁺23, JMW24].

1.1 Related work on superposition attacks

Superposition attacks have been studied for over a decade and remain an active area of research in quantum cryptanalysis. The first results in this direction were presented in [KM10, KM12], targetting symmetric-key cryptosystems. The authors proposed a distinguisher for Feistel ciphers and a key-recovery attack against the Even-Mansour cipher, respectively, when the schemes are queried in superposition. The formalisation of the concept of superposition attacks was introduced in [DFNS13], where it is shown that the security of multiple secret-sharing, zero-knowledge and multi-party computation protocols does not hold under this model. Using the different terminology of “quantum chosen-message attacks”, the work [BZ13] focuses on message authentication codes and shows the conditions under which the security of traditional constructions does not hold.

In [KLLN16a], several attacks against symmetric cryptographic primitives are presented, including on the LRW construction (which is provably secure in the classical setting), on the standard modes of operation of block ciphers and CAESAR candidates (an attack against another CAESAR candidate is given in [Bon18]), as well as an exponential speed-up for the slide attack technique. The quantum slide-attack technique is later extended in [BNS20]. In [LM17], it is shown that using whitening keys in the FX-construction to extend the key length of block ciphers does not improve the security when encrypting quantum messages. The analysis of Feistel constructions is further developed in [IHM⁺19], where access to the decryption oracle leads to a distinguisher for a higher number of rounds. In [CHLS20], the authors target the contracting Feistel structure and related-key attacks.

In [HS18], a truncation technique is proposed where the output bits of a Boolean function evaluated coherently can be isolated from each other, by writing them into an output register initialised to an eigenvector whose eigenvalue is 1. The work [BLNS21] introduces quantum linearization attacks, targeting multiple message authentication code schemes, which leverage this output truncation technique. Similarly, the results in [BBC⁺21] apply the truncation technique to Boolean linear functions to assess the security of block cipher operation modes. In [BS24], this technique is exploited to break authenticated encryption schemes with a Boolean hidden shift, with linear post-processing functions.

More recently, the work [CEG26] introduces quantum superposition attacks in the analysis of identification schemes featuring a linear relation between the secret and the adversary-controlled challenge. The work shows that this relation enables full key-recovery attacks against members of the Learning Parity with Noise (LPN)-based HB family of identification schemes, some of which admit proofs of security in the classical setting.

This line of work is further extended in the present work, where we broaden the scope to study the security of post-quantum signature schemes against superposition attacks.

1.2 Contributions

In this work, we study the security of lattice-based identification schemes and their corresponding Fiat-Shamir signatures against superposition attacks. We focus on a series of works [BDK⁺21, DKL⁺18, Lyu08, Lyu09], which led to the development of the new post-quantum cryptography standard *ML-DSA* [Nat24]. In our analysis, we adopt the Q2 attack model, where an adversary is allowed to interact with the schemes via quantum superposition queries. Working under this model, the main technical contributions of our work are:

- Two new superposition attack techniques that construct a relative phase oracle for the Least Significant Bits (LSBs) of an arbitrary function over any (including non-binary) domain (Sections 5.2 and 5.3).
- A new superposition attack technique that proceeds bit-by-bit (or dit-by-dit) and constructs a relative-phase oracle for any bit (or dit) of an arbitrary function (Section 5.4).
- Building on these attack techniques, we show how a quantum adversary can recover secret information in schemes based on linear functions of an adversary-controlled input. In particular, we apply these ideas to attack several lattice-based identification schemes (Section 6).
- Under reasonable implementation assumptions, we extend these techniques to the corresponding Fiat-Shamir signature schemes, resulting in key-recovery attacks (Section 7).
- We present full key-recovery attacks in the Q2 model against the original *CRYSTALS-Dilithium* scheme, the NIST PQC standardisation Round 1 and Round 2 proposals, as well as the template version of Dilithium Round 3 (version 3.1) proposal (Section 8).
- Our attack does not extend to the final NIST standard version *ML-DSA*. In Section 8.2 we identify and analyse specific aspects of the algorithm’s execution that impact the applicability of our techniques.

Moreover, our work provides conceptual contributions to the study of Q2 security. It shows that superposition attacks can be mounted against public-key cryptosystems with a Schnorr-like structure combined with the Fiat-Shamir transform, a paradigm widely used in post-quantum cryptography. This suggests that care is required in the design of such schemes and that existing proof techniques for quantum security (e.g., in the QROM) may require further refinement.

2 Preliminaries

Notation. We represent the natural numbers by $\mathbb{N}$, the integers by $\mathbb{Z}$, and the complex numbers by $\mathbb{C}$. If $m \in \mathbb{Z}$, we denote the ring $\mathbb{Z}/\langle m \rangle$ by $\mathbb{Z}_m$. Vectors and matrices are represented by bold lower- and upper-case letters, respectively. Let p be a prime and $\mathbf{x} \in \mathbb{Z}_p^n$ be an n -dimensional vector over $\mathbb{Z}_p$; we represent the i -th entry of the vector $\mathbf{x}$ as $\mathbf{x}^{[i]}$. If $q(x) = q^{(0)} + q^{(1)}x + \dots + q^{(n)}x^n = \sum_{i=0}^n q^{(i)}x^i$ is a univariate polynomial of degree n , the coefficient corresponding to degree i is written as $q^{(i)}$ or $q(x)^{(i)}$. If $y \in \mathbb{Z}_p$, the binary representation of y considered as an element of $\mathbb{Z}_{2^\ell}$, with $\ell = \lceil \log_2 p \rceil$, is expanded as $y_0 y_1 \dots y_{\ell-1}$, such that $y = \sum_{i=0}^{\ell-1} y_i 2^i$. We adopt the convention that the binary representation is written with the LSB on the left and the Most Significant Bit (MSB) on the right. Sum in $\mathbb{Z}_2$ is denoted by the usual $\oplus$ operator.

Quantum Computation. We use the standard Dirac notation for pure quantum states. In this work we only deal with pure states $|\psi\rangle$, $\langle\psi|\psi\rangle = 1$, and our basic unit of computation are *qubits*, i.e., two-level quantum systems, which are represented by vectors in $\mathbb{C}^2$, $|\psi\rangle = (a, b) \in \mathbb{C}^2$. It is customary to denote the canonical basis of this space by $|0\rangle = (1, 0)$ and $|1\rangle = (0, 1)$, which is usually referred to as the “computational basis”. Any qubit may thus be represented as $|\psi\rangle = a|0\rangle + b|1\rangle = (a, b) \in \mathbb{C}^2$. We denote the “Hadamard

basis” vectors by $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. An n -qubit system is represented by a vector in $\mathbb{C}^2 \otimes \mathbb{C}^2 \otimes \dots \otimes \mathbb{C}^2 \cong \mathbb{C}^{2^n}$, with the elements of the computational/canonical basis denoted as $|s\rangle \in \mathbb{C}^{2^n}$, where $s \in \mathbb{Z}_2^n$. If a state is not an element of the computational basis, i.e., $|\psi\rangle \neq |s\rangle$ for all $s \in \mathbb{Z}_2^n$, then $|\psi\rangle$ is said to be in *superposition*.

A system of dimension larger than one that can be expressed as the tensor product of individual systems is said to be *separable*—e.g., $|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \in \mathbb{C}^2 \otimes \mathbb{C}^2$ for some states $|\psi_1\rangle, |\psi_2\rangle$. Conversely, a system that cannot be expressed as the tensor product of individual systems is said to be *entangled*. Entangled systems may have correlations that are inherently quantum and cannot be described by classical probabilities.

The evolution of a quantum system is described by unitary transformations, and at any time the quantum system fully describes the statistical behaviour of a “measurement”, as specified by the Born rule. In this work, we will represent this evolution using quantum circuits, with unitary operations described as quantum gates. Two gates of particular relevance are the Hadamard gate, which maps between the computational and Hadamard bases, and the CNOT gate, which flips a target qubit conditioned on the value of a control qubit. A comprehensive reference on quantum computation and information can be found in [NC10].

3 Quantum security

The security of a cryptosystem establishes certain guarantees for participating honest parties even in the presence of malicious actors. Traditionally, when considering the (computational) security of a cryptographic scheme, both honest parties and adversaries are modelled as Probabilistic Polynomial Time (PPT) machines, leading to the definition of *classical security* in cryptography. However, ongoing advances in quantum computing and quantum technologies have made the potential development of a CRQC a significant concern. In this scenario, an adversary must be modelled as a Quantum Polynomial Time (QPT) machine, with capabilities beyond those of classical adversaries, including the ability to solve problems previously considered intractable, such as factoring or discrete logarithm. In the simplest setting, such an adversary is assumed to have access to a CRQC, which can be used (offline) to solve an instance of a computational problem and thereby break the security of a cryptosystem, while all communicating and honest parties continue to be modelled as classical entities. Security against such an adversary is the basis of *post-quantum security*. A more encompassing definition considers that a cryptographic scheme or protocol may be executed on a quantum computer (e.g., an authentication protocol for a hybrid classical-quantum network), and accounts for not only adversaries that may have access to a quantum computer, but also for the communication to use quantum information resources. This more general security model ought to be considered when defining *quantum security*.

In [KLLN16b], two models of quantum cryptanalysis were introduced in the context of quantum security of block ciphers. However, these models generalise naturally to the cryptanalysis of other primitives, including those considered in this work.

- *Q1 model:* a cryptosystem is secure against quantum adversaries if no QPT algorithm can break its security by making only classical queries.
- *Q2 model:* a cryptosystem is secure against quantum adversaries if no QPT algorithm can break its security by making quantum queries.

In this work, we will always work under the Q2 model. We formally define what means to make quantum queries to a classical function in Section 3.1.

3.1 Quantum function evaluation

We consider the standard evaluation of a classical function $f : \mathbb{Z}_M \rightarrow \mathbb{Z}_N$ by a quantum operator $\mathcal{F}$ as

$$\mathcal{F} : |a\rangle |b\rangle \rightarrow |a\rangle |b + f(a) \mod N\rangle, \quad (3.1)$$

with $a \in \mathbb{Z}_M, b \in \mathbb{Z}_N$. We remark that this is the same oracle that is often written with $|a\rangle |b\rangle \rightarrow |a\rangle |b \oplus f(a)\rangle$ when directly dealing with binary functions, as is prevalent in the literature. Here, since the codomain of the classical function f is $\mathbb{Z}_N$, its image must first be represented in binary, and we write $b + f(a) \mod N$ to

simplify this underlying procedure. Our results are independent of the choice of oracle, and analogous results could also have been derived with other quantum oracles evaluating classical functions.

We assume that a quantum computer is implemented solely using qubits. Thus one requires $m = \lceil \log_2(M) \rceil$ and $n = \lceil \log_2(N) \rceil$ qubits to implement a function f with $|a\rangle \in \mathbb{C}^{2^m}$, $|b\rangle \in \mathbb{C}^{2^n}$. By letting the input $|b\rangle = \text{QFT}_N |N-1\rangle$ (corresponding to the output register), one can then create a phase oracle

$$\mathcal{F} : |a\rangle |b\rangle \rightarrow \omega_N^{f(a)} |a\rangle |b\rangle, \quad (3.2)$$

with ω_N a primitive N -th root-of-unity. It is often the case that the separable register $|b\rangle$ is omitted, and the phase oracle for f is written as

$$\mathcal{F} : |a\rangle \rightarrow \omega_N^{f(a)} |a\rangle. \quad (3.3)$$

4 Identification schemes and Fiat-Shamir signatures

Identification (ID) schemes are cryptographic mechanisms that allow one party holding a secret key, the Prover, to prove its identity (i.e. possession of the key) to another party, the Verifier. ID schemes are often constructed from Sigma protocols (commit-challenge-response protocols), as depicted in Figure 1.

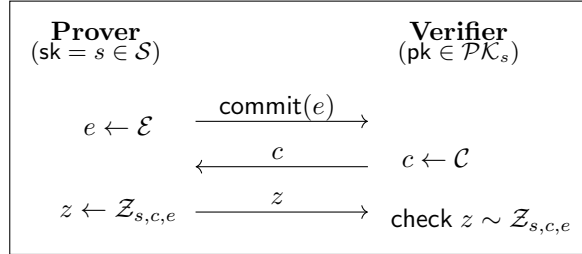


Figure 1: Generic structure of an ID scheme. **commit** is a commitment scheme, and the Verifier’s **check** tests if the Prover’s response z follows the correct distribution that depends on s, c, e .

Schnorr ID scheme [Sch89]: In the original proposal, the Prover holds a secret key $\text{sk} = s \in \mathbb{Z}_q$ with a corresponding verification key $\text{pk} = (g, g^s)$. It first commits to g^e for a uniformly random $e \in \mathbb{Z}_q$. Then, given a challenge $c \in \mathbb{Z}_q$, it responds as $z = cs + e$, such that g^z must be equal to $g^e(g^s)^c$. The security of the Schnorr ID scheme is based on the discrete logarithm problem and is therefore not considered secure in the Q1 model. Nevertheless, the basic construction based on setting $z = cs + e$ has been widely adopted in new proposals for ID schemes relying on different computational hardness assumptions. In particular, it is relevant for this work that the relation between the secret s and the challenge c is given by an affine function with a randomly sampled error e .

Lattice-based ID schemes: Using Schnorr identification scheme as a template, lattice-based proposals were introduced [BDK⁺21, Lyu08, Lyu09], a line of work that led to the current post-quantum digital signature standard *ML-DSA* [Nat24]. In these constructions, the keys comprise $\text{sk} = \mathbf{s} \in D_s^m$ and $\text{pk} = (\mathbf{A}, \mathbf{y}) \in D \times D$, $\mathbf{y} = \mathbf{A}\mathbf{s}$; commit is $\mathbf{b} = \mathbf{A}\mathbf{e}$ for an $\mathbf{e} \in D_e^m$ of appropriate norm; challenge c is sampled from D_c ; and the response is $\mathbf{z} = \mathbf{c}\mathbf{s} + \mathbf{e}$, which may be rejected so that its norm does not leak information about sk . To verify, one checks that $\mathbf{z}$ has small norm and $\mathbf{A}\mathbf{z} = \mathbf{c}\mathbf{y} + \mathbf{b}$. The domains D, D_s, D_e, D_c vary depending on the construction, with [Lyu08] working in $D = \mathbb{Z}_p^m$, and [BDK⁺21, Lyu09, Nat24] working in $D = R^m$, where $R = \mathbb{Z}_p[x]/\langle x^n + 1 \rangle$. The works [BDK⁺21, Nat24] propose an optimisation where $\text{sk} = (\mathbf{s}_1, \mathbf{s}_2)$ and $\text{pk} = (\mathbf{A}, \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2)$. The security of these schemes is based on the SIS and LWE problems or their extended structured versions (MSIS and MLWE).

Fiat-Shamir transform/heuristic [FS86]: From an interactive ID scheme, one can construct a non-interactive signature scheme in the ROM. In this construction, the message is queried to the random oracle, and its output used as the challenge c . In practice, a Cryptographic Hash Function (CHF) is often used to instantiate the random oracle. Although proof techniques in the ROM are well established in classical cryptography,

allowing quantum access to the oracle, i.e., when working in the QROM, introduces well-known complications. In particular, it was shown in [ARU14, DFG13] that the Fiat-Shamir transform is insecure in the QROM, and black-box proofs do not directly carry over to the quantum setting.

In the context of *Dilithium*, several studies have attempted to bypass these results. In [KLS18], it is shown that, in specific security models, the Fiat-Shamir transform in the QROM may still be secure; hence, their reduction has been used to set the parameters of *Dilithium*. Critically, their proposal for *Dilithium-QROM* only targets a deterministic version (see Section 8.2 for a discussion). In [DFMS19, LZ19], new notions of security are introduced in order to circumvent the negative results of [ARU14, DFG13] and achieve security of the Fiat-Shamir transform in the QROM. However, these notions do not directly apply to the specification of *Dilithium*. In [BBD⁺23, DFP⁺23], multiple issues are raised with previous proofs of the Fiat-Shamir with Aborts in the ROM and QROM, concerning infinite-loop iterations and reprogramming of the random oracle. The work [JMW24] extends the analysis of the *Dilithium* construction via a reduction from MSIS to SelfTargetMSIS in the QROM, and suggests new parameters.

In the next sections, we introduce our superposition attack techniques and show how they can be applied to recover the secret keys of several lattice-based ID schemes and corresponding signature schemes, including versions of *Dilithium*.

5 LSB superposition attacks

We now present our new attack techniques for quantum cryptanalysis in the Q2 model, which we call “LSB superposition attacks”. They form the core technical tools of our superposition attacks against *Dilithium*.

In Section 5.1 we show how to change the modulus of a classical function evaluation (as described in Section 3.1) while maintaining its unitary characteristic. Then, in Section 5.2, we introduce our first technique, which enables the creation of a phase for the LSB of a classical function. In Section 5.3, we present a generalised version of this technique to recover more than one LSB at a time by extending the domain and replacing the Hadamard transform with an arbitrary Quantum Fourier Transform (QFT). Finally, in Section 5.4 we introduce our main technique: a method for constructing a phase bit-by-bit (from LSB to MSB) for an arbitrary function. In the special case of an affine function of a secret, this procedure enables its full recovery under appropriate conditions.

5.1 Modulus rescaling

We assume that all circuits operate over qubits, and thus a general (implementation independent) adder function takes two ℓ -qubit registers and computes $a + b$ with $a, b \in \mathbb{Z}_{2^\ell}$, with bit decompositions $a = a_0 a_1 \dots a_{\ell-1}$ and $b = b_0 b_1 \dots b_{\ell-1}$:

$$\begin{aligned} \text{ADD}_{2^\ell} : |a\rangle |b\rangle &\rightarrow |a\rangle |a + b \bmod 2^\ell\rangle \\ \text{ADD}_{2^\ell} : |a_0\rangle \dots |a_{\ell-1}\rangle |b_0\rangle \dots |b_{\ell-1}\rangle &\rightarrow \\ &|a_0\rangle \dots |a_{\ell-1}\rangle |a_0 \oplus b_0\rangle |a_1 \oplus b_1 \oplus \text{carry}_0\rangle \dots |a_{\ell-1} \oplus b_{\ell-1} \oplus \text{carry}_{\ell-2}\rangle \end{aligned} \quad (5.1)$$

where $\text{carry}_0 = 1$ iff $a_0 = 1$ and $b_0 = 1$, and $\text{carry}_i = 1$ if $a_i + b_i + \text{carry}_{i-1} > 1$. Although this is the same addition operation performed in a classical computer, the situation changes when one exploits the fact that $|a\rangle, |b\rangle$ are provided to the adder ADD_{2^ℓ} in superposition. In this case, the registers become entangled, since the carries depend recursively on $\text{carry}_{\ell-2}$ and $a_{\ell-1}, b_{\ell-1}$, extending to a_0, b_0 .

An important observation is that we may *rescale the modulus* of the sum, from the addition $\bmod 2^\ell$ to $\bmod 2^m$, while preserving the operation unitary, so that the resulting sum becomes $a + b \bmod 2^m$ for an arbitrary $m < \ell$, while requiring no manipulation of the internal structure of the adder, only of its inputs and outputs. Operationally, this is similar to the truncation technique presented in [HS18], even though we are working over arbitrary domains while their technique is developed for Boolean functions (which do not have to deal with the entanglement resulting from the carries between the qubits).

For this, let the input $|b_i\rangle = |+\rangle$ for $m \leq i \leq \ell$, which works since $|+\rangle$ is an eigenvector with eigenvalue 1 of the ADD operator, as $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|+\rangle \oplus |j\rangle = \frac{1}{\sqrt{2}}(|0 \oplus j\rangle + |1 \oplus j\rangle) = |+\rangle$, for both $j = 0$

and 1. We can then avoid any entanglement from higher order (than m) qubits, effectively computing in the output register $|a + b \bmod 2^m\rangle |+\rangle^{\otimes \ell-m}$. For $a \in \mathbb{Z}_{2^\ell}, b \in \mathbb{Z}_{2^m}$,

$$\begin{aligned} \text{ADD}_{2^m} : |a\rangle |b\rangle &\rightarrow |a\rangle |a + b \bmod 2^m\rangle \\ \text{ADD}_{2^m} : |a_0\rangle \dots |a_{\ell-1}\rangle |b_0\rangle \dots |b_{m-1}\rangle |+\rangle^{\otimes \ell-m} &\rightarrow \\ |a_0\rangle \dots |a_{\ell-1}\rangle |a_0 \oplus b_0\rangle |a_1 \oplus b_1 \oplus \text{carry}_0\rangle \dots &|a_{m-1} \oplus b_{m-1} \oplus \text{carry}_{m-2}\rangle |+\rangle^{\otimes \ell-m}. \end{aligned} \quad (5.2)$$

Thus, one can implement the operator ADD_{2^m} using the operator ADD_{2^ℓ} as

$$\text{ADD}_{2^m} |a\rangle |b\rangle \equiv \text{ADD}_{2^\ell} |a\rangle |b_0 \dots b_{m-1}\rangle |+\rangle^{\otimes \ell-m}. \quad (5.3)$$

5.2 LSB phase oracle

From this construction, one can directly obtain a phase oracle for the LSB of an arbitrary function $f : \mathbb{Z}_M \rightarrow \mathbb{Z}_N$ (with $\ell = \lceil \log_2 N \rceil$). First, we consider the description of the quantum function evaluation of f as $\mathcal{F}$

$$\begin{aligned} \mathcal{F} : |a\rangle |b\rangle &\rightarrow |a\rangle |b + f(a) \bmod 2^\ell\rangle \\ \Leftrightarrow \mathcal{F} : |a\rangle |b_0\rangle \dots |b_{\ell-1}\rangle &\rightarrow |a\rangle |b_0 \oplus f(a)_0\rangle \dots |b_{\ell-1} \oplus f(a)_{\ell-1} \oplus \text{carry}_{\ell-2}\rangle. \end{aligned} \quad (5.4)$$

Then we rescale the modulus from 2^ℓ to 2, by evaluating $\mathcal{F}$ as

$$\mathcal{F} |a\rangle |b_0\rangle |+\rangle^{\otimes \ell-1} = |a\rangle |b_0 \oplus f(a)_0\rangle |+\rangle^{\otimes \ell-1}. \quad (5.5)$$

Finally, we construct a phase oracle in the standard way by evaluating $\mathcal{F}$ as

$$\mathcal{F} |a\rangle |-\rangle |+\rangle^{\otimes \ell-1} = (-1)^{f(a)_0} |a\rangle |-\rangle |+\rangle^{\otimes \ell-1}. \quad (5.6)$$

5.3 Multiple LSBs phase oracle

Building on the last construction, one may apply the same technique to create a phase for multiple consecutive LSBs. In this setting, one may rescale the modulus from 2^ℓ to 2^m for $1 \leq m < \ell$, and evaluate a function $f : \mathbb{Z}_M \rightarrow \mathbb{Z}_N$ ($\ell = \lceil \log_2 N \rceil$) as

$$\mathcal{F} |a\rangle |b_0\rangle \dots |b_{m-1}\rangle |+\rangle^{\otimes \ell-m} = |a\rangle |b + f(a) \bmod 2^m\rangle |+\rangle^{\otimes \ell-m}. \quad (5.7)$$

Moreover, it is again possible to construct a corresponding phase oracle by setting $|b\rangle = \text{QFT}_{2^m} |2^m - 1\rangle$, such that

$$\mathcal{F} |a\rangle |b\rangle |+\rangle^{\otimes \ell-m} = \omega_{2^m}^{f(a)} |a\rangle |b\rangle |+\rangle^{\otimes \ell-m}, \quad (5.8)$$

where ω_{2^m} is a primitive 2^m -th root-of-unity.

Here we are working with qubits, and therefore employ a power-of-2 QFT. However, the same procedure applies to any other base for which the QFT can be implemented.

5.4 Iterative LSBs phase oracle

An alternative oracle construction considers multiple evaluations of the function to generate a phase for one LSB at a time, rather than rescaling the modulus to a smaller power-of-2. To avoid having carries in superposition, which would otherwise interfere in the phase, each previous register must be initialised to $|0\rangle$. Note that this entangles the previous output qubits with the input $|a\rangle$, but the construction remains useful in the case where one is able to uncompute these registers. The technique applies to any basis (as in Section 5.3), but we restrict ourselves to the basis-2 case as it will be sufficient for our cryptanalytic use cases, while also keeping the exposition simpler. The procedure is as follows:

1. Define the oracle for the LSB as the same as in Equation (5.6),

$$\mathcal{F} |a\rangle |-\rangle |+\rangle^{\otimes \ell-1} = (-1)^{f(a)_0} |a\rangle |-\rangle |+\rangle^{\otimes \ell-1}. \quad (5.9)$$

2. For the second LSB, construct a phase oracle as

$$\mathcal{F} |a\rangle |0\rangle |- \rangle |+\rangle^{\otimes \ell-2} = (-1)^{f(a)_1} |a\rangle |f(a)_0\rangle |- \rangle |+\rangle^{\otimes \ell-2}. \quad (5.10)$$

and uncompute $|f(a)_0\rangle$ (previously computed in Step 1) to obtain

$$\mathcal{F} |a\rangle |0\rangle |- \rangle |+\rangle^{\otimes \ell-2} = (-1)^{f(a)_1} |a\rangle |0\rangle |- \rangle |+\rangle^{\otimes \ell-2}. \quad (5.11)$$

3. Iterate the process for the i -th LSB as

$$\mathcal{F} |a\rangle |0\rangle^{\otimes i} |- \rangle |+\rangle^{\otimes \ell-i-1} = (-1)^{f(a)_i} |a\rangle |f(a)_0\rangle \dots |f(a)_{i-1}\rangle |- \rangle |+\rangle^{\otimes \ell-i-1}. \quad (5.12)$$

and uncompute $|f(a)_{i-1}\rangle$ to $|f(a)_0\rangle$ (previously computed) and obtain

$$\mathcal{F} |a\rangle |0\rangle^{\otimes i} |- \rangle |+\rangle^{\otimes \ell-i-1} = (-1)^{f(a)_i} |a\rangle |0\rangle \dots |0\rangle |- \rangle |+\rangle^{\otimes \ell-i-1}. \quad (5.13)$$

6 Superposition attacks against lattice-based ID schemes

As described in Section 4, Schnorr ID schemes work by setting the response of the Prover as an affine function that directly multiplies (inner-product) the challenge of the Verifier with the secret of the Prover. Multiple lattice-based ID schemes follow this paradigm, and we start by analysing two proposals of ID schemes provably secure under active attacks [Lyu08, Lyu09].

Modelling error sampling by a quantum oracle: When evaluating a classical function $f : \mathbb{Z}_M \rightarrow \mathbb{Z}_N$ as a quantum procedure, it is standard to consider its unitary evaluation as $\mathcal{F} : |a\rangle |b\rangle \rightarrow |a\rangle |b + f(a)\rangle$. Nevertheless, there might be the case that f is not an actual function but a random variable, e.g., resulting from the evaluation of a probabilistic algorithm that requires random sampling. In the analysed protocols, it will be recurrent that $f(x)$ outputs $ax + e$, where e is sampled from some probability distribution. In this case, to evaluate f by a quantum functionality $\mathcal{F}$, unitarity is not as direct, as the random sampling of e induces a mixed state. Nevertheless, if one considers that the sampling of e is a procedure executed before the evaluation of the quantum functionality $\mathcal{F}$, this operation becomes unitary, as $\mathcal{F}$ depends on an e that has already been sampled (is fixed during the computation). An alternative interpretation may be considering the error e as an internal register of $\mathcal{F}$, which is added (control sum gate) to the output register, but is not output by $\mathcal{F}$. The operation is still unitary, but in the description given only the output of f considers a random sampling of e . This is an analogous setting to the modelling of the initialization vector in the superposition attacks on message authentication codes in [BLNS21].

About the error model: As we are dealing with a *classical* function computing $f(x) = ax + e$, which will be quantumly evaluated for (possibly) an $|x\rangle$ in superposition, the value of e will be the same for all elements of the superposition, and independent of a . Indeed, the function f is a classical procedure and cannot change each element of the superposition—that would require an exponential-time quantum computation for a uniform superposition of $|x\rangle$ [GKZ19]. Thus, we emphasise that this is not the same as introducing an independent error per element of the superposition, as in the purely quantum-sample version of LWE, which has been studied in [GKZ19].

Modelling aborts in quantum function evaluation: In the analysed lattice-based ID schemes, the Prover is allowed to abort depending on the result of some function of its response $\text{abort}(z) \in \{\perp, \top\}$. When implementing a protocol as a function $f : \mathcal{A} \rightarrow \mathcal{B} \cup \{\perp\}$ in the usual quantum evaluation $\mathcal{F} : |a\rangle |b\rangle \rightarrow |a\rangle |b + f(a)\rangle$, one must account for the meaning of $|b + \perp\rangle$. One possibility is that when aborting, the function returns a random value, as an implicit rejection. Another possibility more fitting for the analysed protocols would extend the binary operation “+” with $|\perp\rangle$ being described by some element of the canonical basis orthogonal to every other element of the co-domain of f (i.e., $\mathcal{B}$), as an explicit rejection. A simple example of such an implementation could be to have an indicator qubit, such that (e.g., for $|\top\rangle := |0\rangle$ and $|\perp\rangle := |1\rangle$)

$$\mathcal{F} |a\rangle |b\rangle |0\rangle = \begin{cases} |a\rangle |b + f(a)\rangle |\top\rangle & \text{if } \text{abort}(f(a)) = \top, \\ |a\rangle |b\rangle |\perp\rangle & \text{if } \text{abort}(f(a)) = \perp. \end{cases} \quad (6.1)$$

Moreover, due to the unitary nature of the function evaluation, and the fact that aborting depends on result of the function evaluation, it is possible that the result could be in the superposition of aborting for some inputs, and not aborting for the remaining inputs. Then, if one would measure the indicator register, the result would be a mixed state, except if for all $f(a)$, $\text{abort}(f(a)) = \top$, where the qubit would be separable and always equal to $|\top\rangle$. Thus, to extract information about $f(a)$, we will need to exploit evaluations such that the probability of measuring $|\perp\rangle$ is zero, i.e., there is not $\perp$ in the superposition. Even if some executions have $|\perp\rangle$ in the superposition, we will show that with a non-negligible probability no $|\perp\rangle$ exists for all inputs a . This allows us to get a correct result in a polynomial number of repetitions in the case of $\perp$ meaning explicit rejection. Or, in the case of $\perp$ meaning implicit rejection, enable us to do majority voting as a non-negligible amount of runs will return the correct answer.

6.1 Superposition attack against the ID scheme of [Lyu08]

In this section, we describe a superposition attack against the lattice-based ID scheme of [Lyu08], which is parametrised by the dimensions $m, n \in \mathbb{N}$ and modulus $p \in \mathbb{N}$. The scheme works in iterations, with a single iteration depicted in Appendix A, Figure A.1a. The full parameters used in this analysis are given in Table B.1 in Appendix B. The Prover has a secret-key $\mathbf{s} \in \mathbb{Z}_2^m$ with a corresponding public-key $(\mathbf{A}, \mathbf{y}) = (\mathbf{A}, \mathbf{A}\mathbf{s}) \in \mathbb{Z}_p^{n \times m} \times \mathbb{Z}_p^n$. At each iteration:

1. the Prover samples a random vector $\mathbf{e} \xleftarrow{\$} \{0, 1, \dots, 5m-1\}^m$ and sends a commitment $\mathbf{b} = \mathbf{A}\mathbf{e} \in \mathbb{Z}_p^n$ to the Verifier;
2. the Verifier sends a random challenge $c \in \{0, 1\}$ to the Prover;
3. the Prover sends the response $\mathbf{z} = c\mathbf{s} + \mathbf{e}$ to the Verifier, conditioned on $\mathbf{z} \in \{1, \dots, 5m-1\}^m$ if $c = 1$.

The Verifier accepts a Prover if it is able to pass a round (based on the norm of $\mathbf{z}$ and the commitment $\mathbf{b}$) with probability at least $p = 0.65$ for t iterations played, as depicted in Figure A.1b.

The first important remark is that, conditioned on not aborting ($\mathbf{z} \neq \perp$), the response $\mathbf{z} = c\mathbf{s} + \mathbf{e} \in \mathbb{Z}_p$ is affine in c . Moreover, while the operations are performed in $\mathbb{Z}_p$, each $\mathbf{e}^{[i]} \in \{0, 1, \dots, 5m-1\}$, and c and $\mathbf{s}^{[i]}$ are both in $\{0, 1\}$. Given the parameter sizes, $m \ll p$, so the computation of $\mathbf{z}$ is the same whether performed over $\mathbb{Z}$ or $\mathbb{Z}_p$ (never wraps around p), as $c\mathbf{s}^{[i]} \leq 1$ for all i . Consequently, if $\mathbf{z}$ were to be computed over $\mathbb{Z}_2$, the computation would still be affine (over $\mathbb{Z}_2$), with only reduction happening to the vector $\mathbf{e}$, i.e.,

$$\mathbf{z} \bmod 2 = c\mathbf{s} + (\mathbf{e} \bmod 2) \bmod 2 = c\mathbf{s} \oplus (\mathbf{e} \bmod 2). \quad (6.2)$$

Given this fact, the following attack construction will leverage the LSB phase oracle from Section 5.2, which relies on the ability of doing this modulus rescaling operation (Section 5.1) applied to an affine function.

Suppose that a malicious Verifier is allowed to communicate with the Prover using quantum resources, meaning that it is able to choose the challenge $c \in \mathbb{Z}_2$ in superposition. We show that in one round of interaction with an honest Prover, a malicious Verifier can leak one bit of the key $\mathbf{s} \in \mathbb{Z}_2^m$ using the LSB phase oracle from Section 5.2. It is important to note that some executions may fail, as the Prover aborts with $\mathbf{z} = \perp$. Thus, we will first show the attack conditioned on the Prover not aborting, i.e., $\mathbf{z} = c\mathbf{s} + \mathbf{e}$, and then show that an abort occurs with low probability and independently of the challenge $c \in \mathbb{Z}_2$, such that the attack can be repeated until it successfully recovers the full key $\mathbf{s} \in \mathbb{Z}_2^m$.

The attack, executed by a malicious Verifier, is described in Algorithm 1 and works as follows. Let $\mathcal{P}$ be an oracle that implements the Prover, which receives a challenge c as input and outputs the response $\mathbf{z} = p(c) = c\mathbf{s} + \mathbf{e} \in \mathbb{Z}_p$ (or $p(c) = \perp$, which we will not consider for now):

$$\mathcal{P} : |c\rangle |\mathbf{u}\rangle \rightarrow |c\rangle |\mathbf{u} + p(c)\rangle. \quad (6.3)$$

First, note that the input register $|c\rangle$ is a single qubit, while the output register $|\mathbf{u}\rangle$ is composed by ℓm qubits ($\ell = \lceil \log_2 p \rceil$). Then, let $\mathcal{P}^{[i]}$ be the LSB phase oracle for the i -th entry of the secret $\mathbf{s} \in \mathbb{Z}_2^m$, constructed from $\mathcal{P}$ as in Equation (5.6), by decomposing the output registry $|\mathbf{u}\rangle$ as $|\mathbf{u}^{[i]}\rangle$ for each entry of the image of $p(c) \in \mathbb{Z}_p^m$, as

$$|\mathbf{u}\rangle \equiv \left| \mathbf{u}^{[0]} \right\rangle \left| \mathbf{u}^{[1]} \right\rangle \dots \left| \mathbf{u}^{[m-1]} \right\rangle, \quad (6.4)$$

and then decomposing each entry $\mathbf{u}^{[i]} \in \mathbb{Z}_p$ into its binary encoding, as

$$|\mathbf{u}^{[i]}\rangle \equiv |\mathbf{u}_0^{[i]}\rangle |\mathbf{u}_1^{[i]}\rangle \dots |\mathbf{u}_{\ell-1}^{[i]}\rangle. \quad (6.5)$$

Then, the LSB phase oracle $\mathcal{P}^{[i]}$ sets the qubit corresponding to the LSB of the i -th entry of $p(c)$ as $|-\rangle$ and the others to $|+\rangle$,

$$\mathcal{P}^{[i]} |c\rangle |+\rangle^{\otimes i\ell} |-\rangle |+\rangle^{\otimes m\ell-i\ell-1} = (-1)^{p(c)^{(i)}} |c\rangle |+\rangle^{\otimes i\ell} |-\rangle |+\rangle^{\otimes m\ell-i\ell-1}. \quad (6.6)$$

We will omit the separable output register $|+\rangle^{\otimes i\ell} |-\rangle |+\rangle^{\otimes m\ell-i\ell-1}$ when evaluating $\mathcal{P}$, and consider the Prover to be described by a family of oracles $\mathcal{P} = \{\mathcal{P}^{[i]}\}_i$. This way, we build an LSB phase oracle for each entry of $p(c) \in \mathbb{Z}_p^m$, $p(c)^{[i]} \in \mathbb{Z}_p$, that yields a phase $(-1)^{\mathbf{cs}^{[i]} \oplus \mathbf{e}^{[i]}}$, where $\mathbf{cs}^{[i]} \oplus \mathbf{e}^{[i]}$ is an affine function of $\mathbf{s}^{[i]} \in \mathbb{Z}_2$ and $c \in \mathbb{Z}_2$ and $\mathbf{e}^{[i]} \bmod 2$.

$$\mathcal{P}^{[i]} |c\rangle = (-1)^{\mathbf{cs}^{[i]} \oplus \mathbf{e}^{[i]}} |c\rangle. \quad (6.7)$$

Then, the main step of the attack (Step 5 of Algorithm 1) works by setting $|c\rangle = |+\rangle$, such that

$$\begin{aligned} |\varphi^{(5)}\rangle &= \mathcal{P}^{[i]} \frac{1}{\sqrt{2}} \sum_{j=0}^1 |j\rangle \\ &= \frac{1}{\sqrt{2}} \sum_{j=0}^1 (-1)^{p(j)^{[i]}} |j\rangle \\ &= \frac{1}{\sqrt{2}} \sum_{j=0}^1 (-1)^{j\mathbf{s}^{[i]} \oplus \mathbf{e}^{[i]}} |j\rangle \\ &= (-1)^{\mathbf{e}^{[i]}} \frac{1}{\sqrt{2}} \sum_{j=0}^1 (-1)^{j\mathbf{s}^{[i]}} |j\rangle. \end{aligned} \quad (6.8)$$

Applying the Hadamard transform to $|\varphi^{(5)}\rangle$ yields $H|\varphi^{(5)}\rangle = |\mathbf{s}^{[i]}\rangle$, as $(-1)^{\mathbf{e}^{[i]}}$ factors out as a global phase. Finally, the attack recovers $\mathbf{s} \in \mathbb{Z}_2^m$ bit-by-bit, and is fully described in Algorithm 1.

Algorithm 1 Key-recovery (quantum) attack to the [Lyu08] ID scheme.

Require: Oracle $\mathcal{P} = \{\mathcal{P}^{[i]}\}_i$; quantum reg. $|\varphi\rangle \in \mathbb{C}^2$; classical reg. $\mathbf{r} \in \mathbb{Z}_2^m$

Ensure: $\mathbf{r} = \mathbf{s}$

- 1: $\mathbf{r} \leftarrow \vec{0}$
 - 2: **for** $i = 0$ to $m - 1$ **do** ▷ For each entry (bit) of vector $\mathbf{s}$
 - 3: $|\varphi\rangle \leftarrow |0\rangle$
 - 4: $|\varphi\rangle \leftarrow H|\varphi\rangle = \frac{1}{\sqrt{2}} \sum_{c \in \{0,1\}} |c\rangle$
 - 5: $|\varphi\rangle \leftarrow \mathcal{P}^{[i]}|\varphi\rangle = \frac{1}{\sqrt{2}} \sum_{c \in \{0,1\}} (-1)^{\mathbf{cs}^{[i]} \oplus \mathbf{e}^{[i]}} |c\rangle$
 - 6: $|\varphi\rangle \leftarrow H|\varphi\rangle = (-1)^{\mathbf{e}^{[i]}} |\mathbf{s}^{[i]}\rangle$
 - 7: $\mathbf{r}^{[i]} \leftarrow \text{measure}(|\varphi\rangle)$
 - 8: **return** $\mathbf{r}$
-

Failure probability due to abort: The possibility of the Prover aborting is directly related to the sampling of the error vector $\mathbf{e}$. In fact, the protocol only aborts if $\mathbf{e}^{[i]} = 0 \wedge \mathbf{s}^{[i]} = 0$ or $\mathbf{e}^{[i]} = 5m-1 \wedge \mathbf{s}^{[i]} = 1$. In the superposition attack, for a single iteration, we run the Prover with a uniform superposition of $c = 0$ and $c = 1$, $|c\rangle = |+\rangle$, and thus the Prover will pass the check with the same probability as it would for an honest run of the protocol. Still, there is the possibility that the output includes $\perp$ in the superposition,

$$\frac{1}{\sqrt{2}} \sum_{c=0}^1 |c\rangle |b + p(c)\rangle = \begin{cases} \frac{1}{\sqrt{2}}(|0\rangle |b + p(0)\rangle + |1\rangle |b + p(1)\rangle) & \text{if } p(1) \neq \perp, \\ \frac{1}{\sqrt{2}}(|0\rangle |b + p(0)\rangle + |1\rangle |\perp\rangle) & \text{if } p(1) = \perp. \end{cases} \quad (6.9)$$

The probability (over the choice of $\mathbf{e}$, for a fixed $\mathbf{s}$) of not having any abort in the superposition is

$$\Pr_{\mathbf{e}}[p(1) \neq \perp] = \left(1 - \frac{1}{5m}\right)^m, \quad (6.10)$$

which is lower bounded by $4/5$ for $m \geq 1$. Then, the adversary may repeat the attack to recover the same bit $\mathbf{s}^{[i]}$ if it is able to detect an abort (the protocol was implemented with explicit rejection). Even if unable to detect the abort (the protocol was implemented with implicit rejection), the adversary may recover the secret bit $\mathbf{s}^{[i]}$ by repeating the attack multiple times for the same bit and doing majority voting (at least $4/5$ of the time it will recover the correct $\mathbf{s}^{[i]}$).

Oracle complexity analysis: The full execution of the protocol considers $t < m$ parallel executions of the single round. However, all t executions are independent, allowing for t bits of the secret $\mathbf{s} \in \mathbb{Z}_2^m$ to be leaked in a single interaction. Still, some runs will result in aborts, which requires repeating the experiment multiple times for the same bit. This translates to making $O(m/t)$ calls to the oracle.

6.2 Superposition attack against the ID scheme of [Lyu09]

We now analyse the protocol of [Lyu09], which is schematically represented in Appendix A, Figure A.2a. While similar to the protocol analysed in Section 6.1, this ID scheme bases its security in ideal-lattice problems. It is parametrised by $m, n, \sigma, \kappa, p \in \mathbb{N}$, defining the ring $R = \mathbb{Z}_p[x]/\langle x^n + 1 \rangle$, domain $D = \{g \in R : \|g\|_\infty \leq mn\sigma\kappa\}$, and a collision-resistant hash (linear) function family $\mathcal{H}$. The parameters used in this analysis are given in Table B.2 in Appendix B.

The protocol considers a secret-key $\mathbf{s} \in D_s^m = \{g \in R : \|g\|_\infty \leq \sigma\}^m$ with a corresponding public-key $(h, h(\mathbf{s}))$ (where $h \xleftarrow{\$} \mathcal{H}$), and works in three rounds:

1. the Prover samples a random error vector $\mathbf{e} \xleftarrow{\$} D_e^m = \{g \in R : \|g\|_\infty \leq mn\sigma\kappa\}^m$, and computes and sends $\mathbf{b} = h(\mathbf{e})$ to the Verifier;
2. the Verifier samples and sends a random challenge polynomial $c \xleftarrow{\$} D_c = \{g \in R : \|g\|_1 \leq \kappa\}$;
3. the Prover sends the response $\mathbf{z} = c\mathbf{s} + \mathbf{e}$ (over R^m), conditioned on $\mathbf{z} \in G^m = \{g \in R : \|g\|_\infty \leq mn\sigma\kappa - \sigma\kappa\}^m$.

The Verifier accepts a Prover if $h(\mathbf{z}) = ch(\mathbf{s}) + \mathbf{b}$ and the domain of $\mathbf{z}$ is valid.

Again, notice that, conditioned on not aborting ($\mathbf{z} \neq \perp$), the response $p(c) = \mathbf{z} = c\mathbf{s} + \mathbf{e} \in R^m$ is affine in c , and given the domains of $c, \mathbf{s}, \mathbf{e}$, it never wraps around p , since each entry of the vector it is at most $\sigma\kappa + mn\sigma\kappa$ which for all proposed parameter sets is smaller than p by at least two orders of magnitude (see Table B.2). So, $\mathbf{z}$ is effectively computed over $\mathbb{Z}[x]/\langle x^n + 1 \rangle$ (analogously to the computation of $\mathbf{z}$ in $\mathbb{Z}$ instead of $\mathbb{Z}_p$ in Section 6.1).

Now consider that a malicious Verifier is able to use quantum resources in the protocol and send the challenge $c(x) = c^{(0)} + c^{(1)}x + \dots + c^{(n-1)}x^{n-1} \in D_c$ in superposition, (arbitrarily) represented as quantum register^a

$$|c\rangle = |c^{(0)}\rangle |c^{(1)}\rangle \dots |c^{(n-1)}\rangle, \quad (6.11)$$

where each $c^{(i)} \in \mathbb{Z}_{2^{\lceil \log_2 \kappa \rceil}}$ and has a binary representation in qubits as

$$|c^{(i)}\rangle = |c_0^{(i)}\rangle |c_1^{(i)}\rangle \dots |c_{\lceil \log_2 \kappa \rceil - 1}^{(i)}\rangle. \quad (6.12)$$

We now show that by leveraging the possibility of sending $|c\rangle$ in superposition, one may fully recover the secret-key $\mathbf{s}$ using the oracle in Section 5.4. As before, in Section 6.1, it is possible that $\mathbf{z} = \perp$. We will again show the attack assuming that the Prover does not abort, and then show that the abort can be circumvented

^aWe will consider the trivial encoding of writing the entire polynomial, but given the restriction on their 1-norm, more efficient encodings would also be possible; our attack is independent of any encoding of c .

by repeating the execution multiple times, since the abort depends only on the vector of polynomials $\mathbf{e}$ and is independent of the polynomial c .

The full description of the attack procedure is described in Algorithm 2, and works by playing a malicious Verifier against an honest Prover to recover its secret-key $\mathbf{s} \in D_s^m$. Again, let $\mathcal{P}$ be an oracle that implements the Prover, receiving a challenge polynomial c , and returns the response $\mathbf{z} = p(c) = c\mathbf{s} + \mathbf{e}$,

$$\mathcal{P} : |c\rangle |\mathbf{u}\rangle \rightarrow |c\rangle |\mathbf{u} + p(c)\rangle. \quad (6.13)$$

Since the challenge polynomial $c \in D_c$ is given as the input to the Prover oracle $\mathcal{P}$, we may choose the LSB of the constant term of c to be the state $c_0^{(0)} = |+\rangle$, while all remaining qubits of this term, as well as all the qubits corresponding to the non-constant coefficients are set to $|0\rangle$. This effectively makes the polynomial c to be a single bit in superposition, i.e. $c = |+\rangle \otimes |0\rangle^{\otimes (n \lceil \log_2 \kappa \rceil - 1)}$. This will avoid unwanted entanglement resulting from the multiplication of $c\mathbf{s}$ in two ways: the reduction over $\langle x^n + 1 \rangle$ due to the degree increase making the monomials entangled with each other, and the higher order bits becoming entangled with the lower order ones inside each coefficient. (See the *Remark* below.)

Remark. Although the attack may be performed using any bit of any other coefficient, we will only make use of the LSB of the constant term of the polynomial $c(x)$, $c_0^{(0)}$, as this will simplify the (already very convoluted) computations, making the challenge to be sent to the Prover either 0 or 1. This is mainly due to two reasons that regard the multiplication $c\mathbf{s}$: First, in polynomial multiplication, multiplying by a constant does not increase the degree of the polynomial, which would make the monomials of degree larger than n to wrap around $\langle x^n + 1 \rangle$ —this avoids that the reduction by $x^n = -1$ depends on the superposition and hence entanglement between coefficients. Second, the multiplier c multiplies each bit of the multiplicand $\mathbf{s}^{[i]}$ independently (for each entry of the vector, i , separately), avoiding carries and multiplying the entire polynomial by either 0 or 1 (or putting these in a superposition)—this avoids carries in the bits of the coefficients and the resulting entanglement. Notwithstanding, it may be possible to extend this setting to recover more than a single bit at a time (e.g., recover $\lceil \log_2 \kappa \rceil$ using the $\text{QFT}_{2^{\lceil \log_2 \kappa \rceil}}$ as in the Multiple LSBs procedure of Section 5.3). For sake of simplicity and clear presentation, we will leak the secret-key $\mathbf{s}$ bit-by-bit using the Iterative LSBs procedure of Section 5.4, rather than focus on optimising the minimum number of operations/qubits used or oracle calls (leaking d bits at a time solely provides a constant speed-up).

Let $\ell = \lceil \log_2 p \rceil$ and $v = \lceil \log_2 \sigma \rceil$ and $w = \lceil \log_2 \kappa \rceil$. The challenge register $(|c\rangle)$ encodes a polynomial of degree $n - 1$, with coefficients in $\{0, \dots, 2^w - 1\}$. The output register $|\mathbf{u}\rangle$, where the output of $p(c) = \mathbf{z}$ will be written, can be decomposed as $|\mathbf{u}^{[i]}\rangle$ for $i \in \{0, \dots, m - 1\}$ (i.e., the elements of a vector of polynomials). Then, each polynomial $|\mathbf{u}^{[i]}\rangle$ can be decomposed in coefficients $|\mathbf{u}^{[i](j)}\rangle$ for $j \in \{0, \dots, n - 1\}$. In addition, each coefficient $|\mathbf{u}^{[i](j)}\rangle$ can be decomposed into bits $|\mathbf{u}_k^{[i](j)}\rangle$ for $k \in \{0, \dots, \ell - 1\}$. Expanding $|\mathbf{u}\rangle$ into ℓmn qubits yields:

$$\begin{aligned} & |\mathbf{u}_0^{[0](0)}\rangle \dots |\mathbf{u}_{\ell-1}^{[0](0)}\rangle \quad |\mathbf{u}_0^{[0](1)}\rangle \dots |\mathbf{u}_{\ell-1}^{[0](1)}\rangle \quad \dots \quad |\mathbf{u}_0^{[0](n-1)}\rangle \dots |\mathbf{u}_{\ell-1}^{[0](n-1)}\rangle & (\text{polynomial } \mathbf{u}^{[0]}) \\ & |\mathbf{u}_0^{[1](0)}\rangle \dots |\mathbf{u}_{\ell-1}^{[1](0)}\rangle \quad |\mathbf{u}_0^{[1](1)}\rangle \dots |\mathbf{u}_{\ell-1}^{[1](1)}\rangle \quad \dots \quad |\mathbf{u}_0^{[1](n-1)}\rangle \dots |\mathbf{u}_{\ell-1}^{[1](n-1)}\rangle & (\text{polynomial } \mathbf{u}^{[1]}) \\ & \dots \\ & |\mathbf{u}_0^{[m-1](0)}\rangle \dots |\mathbf{u}_{\ell-1}^{[m-1](0)}\rangle \quad |\mathbf{u}_0^{[m-1](1)}\rangle \dots |\mathbf{u}_{\ell-1}^{[m-1](1)}\rangle \quad \dots \quad |\mathbf{u}_0^{[m-1](n-1)}\rangle \dots |\mathbf{u}_{\ell-1}^{[m-1](n-1)}\rangle & (\text{polynomial } \mathbf{u}^{[m-1]}) \end{aligned} \quad (6.14)$$

The attack works again by using the Iterative LSBs procedure of Section 5.4, applying it to each coefficient of each polynomial individually to split the Prover oracle $\mathcal{P}$ into an oracle for each bit of the secret-key $\mathbf{s}$, $\mathcal{P} = \{\mathcal{P}_k^{[i](j)}\}$. However, it is important to remark that since we are now writing an element of $\mathbb{Z}_p$ as bits (i.e., the output of $p(c)$), there will be *carries* between the qubits that are dependent on the superposition of $|c\rangle$, avoiding the constructive interference necessary for the attack. Effectively, this introduces a key difference, as it makes all qubits entangled, and requires an extra uncomputation step. We remark that each call to the Prover oracle corresponds to a different instance of the Iterative LSB oracle, as the error term is resampled. Still, since the function computed by the Prover is affine over the same fixed secret ($\mathbf{s}$), at each call we are able to leak a bit despite the oracle technically not computing the same exact function.

To perform the Iterative LSBs (Section 5.4) we decompose the Prover oracle $\mathcal{P}$ into an oracle for each bit of the secret-key $\mathbf{s}$, $\mathcal{P} = \{\mathcal{P}_k^{[i](j)}\}$. We start by showing how to construct these oracles for leaking a single coefficient of only one polynomial, say (wlog) $s = \mathbf{s}^{[0](0)}$.^b

- 1.1 Start with output (polynomial) register $|u\rangle = |\mathbf{u}^{[0](0)}\rangle$, so set $|\mathbf{u}_k^{[i](j)}\rangle = |+\rangle$ for all $i, j \neq 0$ and all k . This is an eigenvector of the ADD operation, meaning that no interference will result from these elements in the computation.
- 1.2 Start with the LSB u_0 , so set $|u_k\rangle = |+\rangle$ for all $k > 0$. Set $|u_0\rangle = |-\rangle$. This creates an LSB phase oracle as described in Section 5.2, since

$$\begin{aligned} \mathcal{P} \left(|c\rangle |0\rangle^{\otimes wn-1} \right) \left(|-\rangle |+\rangle^{\otimes \ell mn-1} \right) \\ = (-1)^{p(c)_0^{[0](0)}} \left(|c\rangle |0\rangle^{\otimes wn-1} \right) \left(|-\rangle |+\rangle^{\otimes \ell mn-1} \right) \\ = (-1)^{cs_0 \oplus e_0} \left(|c\rangle |0\rangle^{\otimes wn-1} \right) \left(|-\rangle |+\rangle^{\otimes \ell mn-1} \right). \end{aligned} \quad (6.15)$$

Let this operation be denoted (omitting the separable $|0\rangle, |+\rangle, |-\rangle$) by

$$\mathcal{P}_0^{[0](0)} |c\rangle = (-1)^{cs_0 \oplus e_0} |c\rangle. \quad (6.16)$$

- 1.3 Recover s_0 by letting $|c\rangle = |+\rangle$, evaluating $\mathcal{P}_0^{[0](0)}$, and measuring the first register in the Hadamard basis (the error e_0 factors out as a global phase since it is independent of c , and thus is not observed in the measurement),

$$\begin{aligned} H \mathcal{P}_0^{[0](0)} |+\rangle &= H \mathcal{P}_0^{[0](0)} \sum_{c \in \{0,1\}} |c\rangle \\ &= H \sum_{c \in \{0,1\}} (-1)^{p(c)_0^{[0](0)}} |c\rangle \\ &= H \sum_{c \in \{0,1\}} (-1)^{cs_0 \oplus e_0} |c\rangle \\ &= (-1)^{e_0} H \sum_{c \in \{0,1\}} (-1)^{cs_0} |c\rangle \\ &= (-1)^{e_0} |s_0\rangle. \end{aligned} \quad (6.17)$$

- 2.1 Then, proceed to the second LSB of u , so set $|u_k\rangle = |+\rangle$ for all $k > 1$,

$$\begin{aligned} |\psi\rangle &= \left(|c\rangle |0\rangle^{\otimes wn-1} \right) \left(|u_0\rangle |u_1\rangle |+\rangle^{\otimes \ell mn-2} \right) \\ &= \left(|c\rangle |0\rangle^{\otimes wn-1} \right) \left(|0\rangle |-\rangle |+\rangle^{\otimes \ell mn-2} \right). \end{aligned} \quad (6.18)$$

- 2.2 Create a phase oracle for the second LSB by evaluating the oracle $\mathcal{P}$,

$$\begin{aligned} \mathcal{P} |\psi\rangle &= (-1)^{p(c)_1^{[0](0)}} \left(|c\rangle |0\rangle^{\otimes wn-1} \right) \left(|cs_0 \oplus e_0\rangle |-\rangle |+\rangle^{\otimes \ell mn-2} \right) \\ &= (-1)^{cs_1 \oplus e_1 \oplus \text{carry}_0} \left(|c\rangle |0\rangle^{\otimes wn-1} \right) \left(|cs_0 \oplus e_0\rangle |-\rangle |+\rangle^{\otimes \ell mn-2} \right). \end{aligned} \quad (6.19)$$

Note that there is no modular reduction at this step, since the image of $p(c)$ is strictly contained in $\mathbb{Z}_p$, as $\sigma\kappa + mn\sigma\kappa \ll p$ for all proposed security parameters. Thus, the computation $cs + e$ is done over $\mathbb{Z}$.

^bWe enumerate with $i.1, i.2, \dots$ for the steps relative to bit i . We use standard items for other coefficients or polynomials.

2.3 Since carry_0 depends on $|c\rangle$, it has to be uncomputed from the phase. For this, redo the computation $cs_0 + e_0 \in \mathbb{Z}_4$, creating a phase of -1 for the MSB/carry (by setting the register to $|-\rangle$), ignoring the LSB that is not needed (by setting the register to the eigenvector $|+\rangle$). This can be done by recomputing $|cs_0\rangle$ from $|c\rangle$ and the previously computed $|s_0\rangle$ (iteration 1.) with a **Toffoli** gate, using it to obtain $|e_0\rangle$ from $|cs_0 \oplus e_0\rangle$ with a **CNOT** gate, and recomputing the sum with an adder gate,

$$|c\rangle \text{ ADD } |cs_0\rangle |e_0\rangle |+\rangle_{\text{sum}} |-\rangle_{\text{carry}} = (-1)^{\text{carry}_0} |c\rangle |cs_0\rangle |e_0\rangle |+\rangle_{\text{sum}} |-\rangle_{\text{carry}}. \quad (6.20)$$

A schematic circuit for the procedure—including the uncomputation of intermediate values to avoid interference—is presented in Figure 2. Let the full operation (with the cancelled $(-1)^{\text{carry}_0}$ phase) be denoted by $\mathcal{P}_1^{[0](0)}$,

$$\mathcal{P}_1^{[0](0)} |c\rangle = (-1)^{cs_1 \oplus e_1} |c\rangle. \quad (6.21)$$

2.4 Finally, recover s_1 by preparing $|c\rangle = |+\rangle$, evaluating $\mathcal{P}_1^{[0](0)}$, and measuring the first register in the Hadamard basis, as

$$\begin{aligned} H \mathcal{P}_1^{[0](0)} |+\rangle &= H \mathcal{P}_1^{[0](0)} \sum_{c \in \{0,1\}} |c\rangle \\ &= H \sum_{c \in \{0,1\}} (-1)^{p(c)} \mathcal{P}_1^{[0](0)} |c\rangle \\ &= H \sum_{c \in \{0,1\}} (-1)^{cs_1 \oplus e_1} |c\rangle \\ &= (-1)^{e_1} H \sum_{c \in \{0,1\}} (-1)^{cs_1} |c\rangle \\ &= (-1)^{e_1} |s_1\rangle. \end{aligned} \quad (6.22)$$

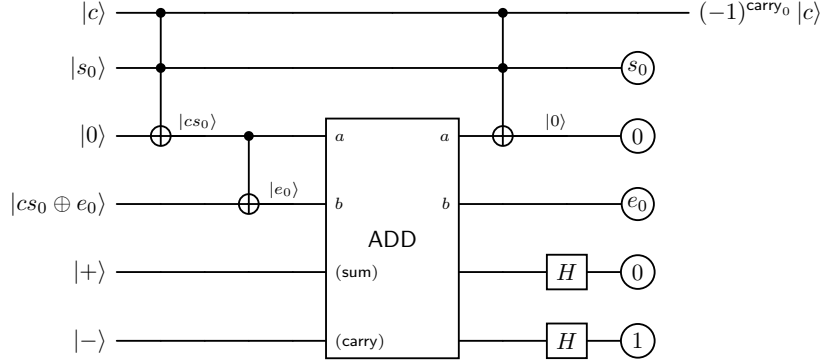


Figure 2: Quantum circuit to create a $(-1)^{\text{carry}_0}$ phase. The adder gate adds the a, b operands and writes the LSB of the sum in the **sum** wire and the MSB in the **carry** wire. Circles in the right-end of the ancilla wires correspond to their value in the computational basis.

3.1 The next bit (third) works similarly. Prepare the register

$$|\psi\rangle = \left(|c\rangle |0\rangle^{\otimes wn-1}\right) \left(|0\rangle |0\rangle |-\rangle |+\rangle^{\otimes \ell mn-3}\right). \quad (6.23)$$

3.2 Evaluate the oracle $\mathcal{P}$

$$\begin{aligned} \mathcal{P} |\psi\rangle &= (-1)^{p(c)} \mathcal{P}_1^{[0](0)} \left(|c\rangle |0\rangle^{\otimes wn-1}\right) \left(|0\rangle |0\rangle |-\rangle |+\rangle^{\otimes \ell mn-3}\right) \\ &= (-1)^{cs_2 \oplus e_2 \oplus \text{carry}_1} \left(|c\rangle |0\rangle^{\otimes wn-1}\right) \left(|cs_0 \oplus e_0\rangle |cs_1 \oplus e_1 \oplus \text{carry}_0\rangle |-\rangle |+\rangle^{\otimes \ell mn-3}\right). \end{aligned} \quad (6.24)$$

3.3 Uncompute the carry_1 from the phase, which can again be computed with an adder gate with appropriately constructed inputs that recompute $cs_1 + e_1 + \text{carry}_0 \in \mathbb{Z}_4$ (note that now there are three parcels in the sum, but the sum is still in $\mathbb{Z}_4$),

$$|c\rangle \text{ ADD } |cs_1\rangle |e_1\rangle |\text{carry}_0\rangle |+\rangle_{\text{sum}} |-\rangle_{\text{carry}} = (-1)^{\text{carry}_1} |c\rangle |cs_1\rangle |e_1\rangle |\text{carry}_0\rangle |+\rangle_{\text{sum}} |-\rangle_{\text{carry}}. \quad (6.25)$$

A schematic circuit for the procedure is given in Figure 3. Denote the full operation with the cancelled $(-1)^{\text{carry}_1}$ phase by $\mathcal{P}_2^{[0](0)}$,

$$\mathcal{P}_2^{[0](0)} |c\rangle = (-1)^{cs_2 \oplus e_2} |c\rangle. \quad (6.26)$$

3.4 As before, compute s_2 by setting $|c\rangle = |+\rangle$, evaluating $\mathcal{P}_2^{[0](0)}$, and measuring the first register in the Hadamard basis.

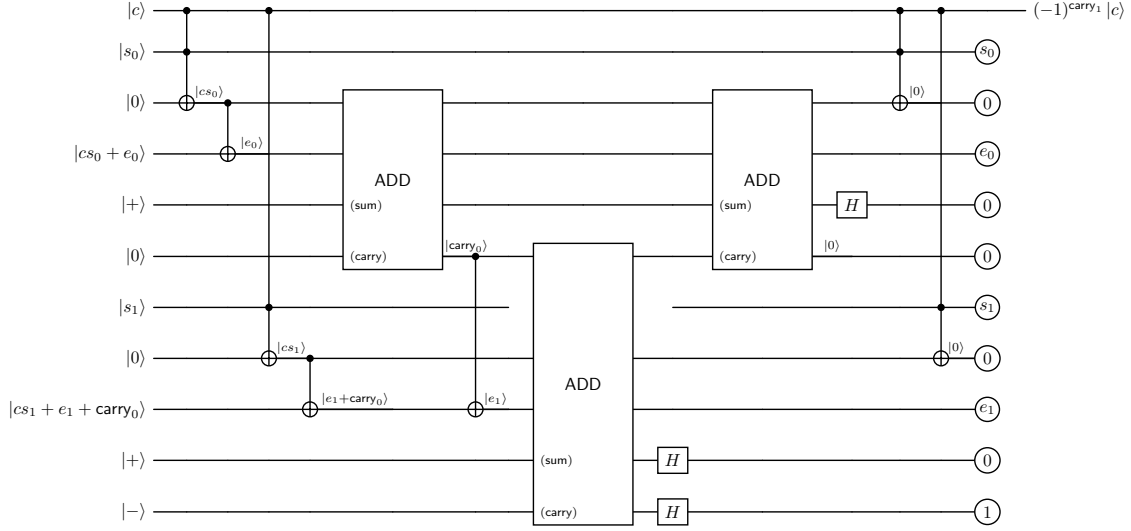


Figure 3: Quantum circuit to create a $(-1)^{\text{carry}_1}$ phase. The 4-input adders are as in Figure 2, and the 5-input adder is analogous but adds the first 3 inputs, writing the LSB in the sum wire and the MSB in the carry wire.

4...v The remaining bits follow analogously, up until $v = \lceil \log_2 \sigma \rceil$, yielding $\mathcal{P}_i^{[0](0)}$ for each bit i of $\mathbf{s}^{[0](0)}$. The circuit to compute an arbitrary phase $(-1)^{\text{carry}_i}$ can be seen in Figure C.1 in Appendix C.

The procedure generalises for any coefficient of any polynomial of $\mathbf{s}$:

- Since we are only multiplying a constant c with the entire polynomial, the degree of each monomial does not increase and therefore there is no reduction $x^n = -1$, and thus no wrapping around of the coefficients. Hence, they are independent, and we iterate and recover $\mathbf{s}^{(j)}$ coefficient-by-coefficient ($j \in \{0, \dots, n-1\}$) (setting all other qubits corresponding to the other coefficients to the eigenvector $|+\rangle$).
- Also, multiplication of the constant c by the vector of polynomials $\mathbf{s}$ maps the operation inside the vector, and is independent for each polynomial. Consequently, we iterate and recover the m polynomials of the secret-key $\mathbf{s}$ one at a time (setting all other qubits corresponding to the other polynomials to the eigenvector $|+\rangle$).

Algorithm 2 Key-recovery (quantum) attack to the [Lyu09] ID scheme.

Require: $\mathcal{P} = \{\mathcal{P}_k^{[i](j)}\}_{i,j,k}$; quantum reg. $|\varphi\rangle \in \mathbb{C}^{2^{wn}}$; classical reg. $\mathbf{r} \in D^m$

Ensure: $\mathbf{r} = \mathbf{s}$

```

1:  $\mathbf{r} \leftarrow \vec{0}$ 
2: for  $i = 0$  to  $m - 1$  do                                ▷ For each polynomial of vector  $\mathbf{s}$ 
3:   for  $j = 0$  to  $n - 1$  do                                ▷ For each coefficient of  $\mathbf{s}^{[i]}$ 
4:     for  $k = 0$  to  $v - 1$  do                                ▷ For bit of coefficient  $\mathbf{s}^{[i](j)}$ 
5:        $|\varphi\rangle \leftarrow |0\rangle^{\otimes wn}$ 
6:        $|\varphi\rangle \leftarrow (H \otimes \mathbb{I}^{\otimes wn-1}) |\varphi\rangle = \frac{1}{\sqrt{2}} \sum_{c \in \{0,1\}} |c\rangle |0\rangle^{\otimes wn-1}$       ▷ Superposition LSB of  $c(x)^{(0)}$ 
7:        $|\varphi\rangle \leftarrow \mathcal{P}_k^{[i](j)} |\varphi\rangle = \frac{1}{\sqrt{2}} \sum_{c \in \{0,1\}} (-1)^{c \mathbf{s}_k^{[i](j)} \oplus \mathbf{e}^{[i](j)}} |c\rangle |0\rangle^{\otimes wn-1}$ 
8:        $|\varphi\rangle \leftarrow (H \otimes \mathbb{I}^{\otimes wn-1}) |\varphi\rangle = (-1)^{\mathbf{e}^{[i](j)}} \left| \mathbf{s}_k^{[i](j)} \right\rangle |0\rangle^{\otimes wn-1}$ 
9:        $\mathbf{r}_k^{[i](j)} \leftarrow \text{measure}(|\varphi\rangle)$ 
10: return  $\mathbf{r}$ 

```

Failure probability due to abort: There is now the need to analyse the probability of the Prover aborting the protocol as $p(c) = \perp$. The protocol aborts when $\mathbf{z} = c\mathbf{s} + \mathbf{e} \notin G^m$. Thus, the value of $\mathbf{z}$ must fall in the range $\{g : \|g\|_\infty \leq mn\sigma\kappa - \sigma\kappa\}^m$. In turn, this means that for each coefficient of each polynomial of $\mathbf{z}$, the cardinality of the domain of an arbitrary $\mathbf{z}$ is $2\sigma\kappa$ larger than the valid range for not aborting defined in G . Indeed, by sampling an error vector (of polynomials) with mn coefficients in $\{g : \|g\|_\infty \leq mn\sigma\kappa - 2\sigma\kappa\}$, the protocol never aborts. Hence, the probability of not having any abort value $\perp$ in the superposition, over the sampling of $\mathbf{e}$, can be computed as

$$\Pr_{\mathbf{e}}[p(c) \neq \perp] = \left(\frac{mn\sigma\kappa - 2\sigma\kappa}{mn\sigma\kappa} \right)^{mn} = \left(1 - \frac{2}{mn} \right)^{mn}, \quad (6.27)$$

which is lower bounded by $1/10$ for all proposed security parameters. Notice that the probability of aborting when $c = 1$ is substantially larger than the probability when $c = 0$, as the range of each coefficient (of each polynomial) of $\mathbf{z}$ is $\{\leq mn\sigma\kappa + \sigma\kappa\}$ for $c = 1$, but only $\{\leq mn\sigma\kappa\}$ for $c = 0$, translating to twice as many bad error coefficients. Moreover, note that this is a worst-case scenario that is conservative in relation to our implementation of the attack, as in these computations we considered the entire domain of c , while in the attack we restrict c to take only values in $\{0, 1\}$, reducing the bad domain elements from $2\sigma\kappa$ to $\sigma(\kappa + 1)$. Again, the attack works either in the case of explicit rejection where the adversary repeats the attack an average of 10 times until recovering each bit, or in the case of implicit rejection where it must be repeated enough (but constant) times such that 10% is significant to do majority voting.

Oracle complexity analysis: While optimising the attack for the minimum number of oracle queries was not the focus of the construction, it is important to note that our attack is in fact efficient, i.e., the number of queries is at most polynomial. Indeed, to recover the full secret-key, the oracle has to be called once for each bit of the key, meaning $mn \lceil \log_2 \sigma \rceil$ calls to the oracle, and repeated a constant number of times to circumvent the abort iterations, so $O(mn \log_2 \sigma)$ iterations are required. Optimisations such as leaking $\lceil \log_2 \kappa \rceil$ bits at a time using an Iterative LSB phase oracle (Section 5.4) but with a power-of-2 QFT over $2^{\lceil \log_2 \kappa \rceil}$ instead (as in Section 5.3) would reduce this complexity to $O\left(\frac{mn \log_2 \sigma}{\log_2 \kappa}\right)$.

7 Superposition attacks against lattice-based Fiat-Shamir signatures

We now describe how to extend our techniques to apply to signature schemes resulting from the Fiat-Shamir transform. To recall, the Fiat-Shamir transform converts an interactive ID scheme into a signature scheme

using a hash function H , by considering the challenge to be the hash of the input μ together with the Prover's commitment com , and relying on the properties of the CHF H for security. In this section, we extend the analysis from Section 6.2 to apply it to the proposed signature scheme of [Lyu09]. We continue our analysis in Section 8, by considering superposition attacks against the *Dilithium* signature algorithm [BDK⁺21] and its standardised version *ML-DSA* [Nat24].

Modelling the “Signer” Oracle $\mathcal{S}$: When considering the signing process for the signatures schemes, both the challenge (c) and the response ($\mathbf{u}$) registers are included as part of the output. We note that in the quantum function evaluation of the classical **Sign** procedure, multiple unitary constructions satisfy the reversibility requirement. The following implementation portrays one of the simplest and most straightforward ways to implement **Sign**, chaining the two classical procedures of the challenge creation ($H(\mu, \text{com})$) and response ($c\mathbf{s} + \mathbf{e}$). We consider this is a reasonable assumption for a possible implementation of the signing procedure in a quantum computer, and develop our attacks under this model. A schematic circuit for this oracle is shown in Figure 4.

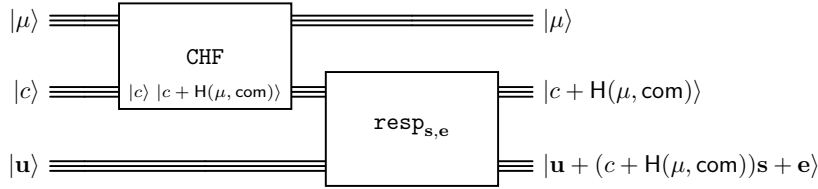


Figure 4: Example implementation for the oracle for the **Sign** procedure. The classical **hash** procedure takes a message and outputs the corresponding challenge (with commitment com), $a \mapsto H(a, \text{com})$. The classical $\text{resp}_{s,e}$ procedure takes a challenge and outputs the corresponding response, $b \mapsto bs + e$.

Fundamentally, we are assuming a specific behaviour for the oracle $\mathcal{S}$, which uses the register of the challenge $|c + H(\mu, \text{com})\rangle$ directly to compute the response register $|\mathbf{u} + (c + H(\mu, \text{com}))\mathbf{s} + \mathbf{e}\rangle$. Effectively, this allows us to bypass the need to set the message μ in superposition, which would otherwise be given as input to the hash function H which we do not control.

$$\mathcal{S} : |\mu\rangle |c\rangle |\mathbf{u}\rangle \rightarrow |\mu\rangle |c + H(\mu, \text{com})\rangle |\mathbf{u} + (c + H(\mu, \text{com}))\mathbf{s} + \mathbf{e}\rangle. \quad (7.1)$$

Schematically, the oracle behaviour is represented in Figure 5.

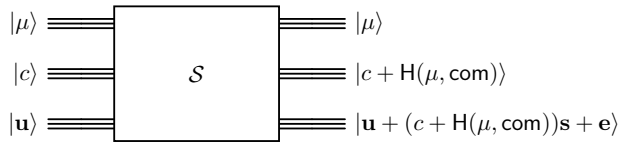


Figure 5: Assumption of the behaviour of the signing oracle $\mathcal{S}$.

In the remaining of this section, and in Section 8, all algorithms we develop will be assuming this implementation. In other words, the superposition attacks against the signature schemes that follow are all based on the assumption:

Assumption. The quantum oracle $\mathcal{S}$ that models the **Sign** procedure is implemented as in Figure 5. This makes our attacks implementation-dependent.

Rejection in the signing procedure: In the analysed signature schemes, there is a step where a generated signature may be rejected and a new one is computed, in order to avoid leaking information about the secret-key (corresponding to the abort step in the ID schemes presented). In the quantum setting, this can be implemented in multiple ways, but due to the unitarity requirement of the quantum function evaluation, it implies that entire previous computation performed for the invalid signature needs to be *uncomputed* (since

the signature is rejected), and a new candidate is computed until a valid signature is found. This prevents any interference with the challenge register $|c\rangle$, which is a crucial object for our attack to work. Still, as we change the value of c , we must then show that the new value, which is used to compute the signature, does not invalidate the checks and does not prevent the signature generation from terminating.

7.1 Superposition attack against the signature scheme of [Lyu09]

In [Lyu09], an ID scheme and the corresponding Fiat-Shamir signature scheme are presented. We have considered ID scheme in Section 6.2). In this section, working under the signature implementation assumptions discussed above, we describe our superposition attack against the signature scheme, which is schematically represented in Figure A.2b in Appendix A.

The scheme follows from the interactive ID scheme (Figure A.2a), where Fiat-Shamir transform takes the message μ and the commitment $h(\mathbf{e})$, and derives the challenge c using a hash function H mapping to the challenge space D_c , as $c = H(\mu, h(\mathbf{e}))$. The parameters used in our analysis are in Table B.2 (Appendix B).

To recover the secret-key from the signing oracle $\mathcal{S}$, we leverage the fact that we control the challenge register $|c\rangle$, which we can place in superposition and keeping the message μ constant (e.g., $\mu = 0$) so as not to create interference. This will make the output signature,

$$\sigma = (c, (c + H(\mu, h(\mathbf{e}))) \mathbf{s} + \mathbf{e}), \quad (7.2)$$

be an affine function in c with $\mathbf{e}' = H(\mu, h(\mathbf{e})) \mathbf{s} + \mathbf{e}$ constant in c . Indeed, the output of the Signer, given the challenge c as the input, for a fixed message μ (say $\mu = 0$), is given by

$$\begin{aligned} p(c) &= (c + H(\mu, h(\mathbf{e})) \mathbf{s} + \mathbf{e}) \\ &= c\mathbf{s} + H(\mu, h(\mathbf{e})) \mathbf{s} + \mathbf{e} \\ &= c\mathbf{s} + \mathbf{e}'. \end{aligned} \quad (7.3)$$

Note that the error distribution of $\mathbf{e}'$ corresponds to the sum of two independent uniform distributions, and thus is no longer uniform (it is the convolution of the distributions). The resulting distribution is independent of the challenge c and its absolute value is still bounded by $mn\sigma\kappa + \sigma\kappa$ (there is a possible increase of up to $\sigma\kappa$). This influences the check as now

$$\Pr_{\mathbf{e}}[p(c) \neq \perp] = \left(\frac{mn\sigma\kappa - 3\sigma\kappa}{mn\sigma\kappa} \right)^{mn} = \left(1 - \frac{3}{mn} \right)^{mn}, \quad (7.4)$$

which is approximately 5% for all proposed security parameters (as can be seen in the values in Table B.2). This means that the check for the signature generation will pass with a few enough attempts.

The attack then is analogous to the one presented in Section 6.2, using the Iterative LSB phase oracle of Section 5.4. For this, consider challenge register prepared in the state $|c\rangle = |+\rangle |0\rangle^{\otimes wn-1}$, where again $w = \lceil \log_2 \kappa \rceil$, and iterate over the qubits of the output register $|\mathbf{u}\rangle$ as before, setting $|\mathbf{u}_k^{[i](j)}\rangle = |-\rangle$ to generate the desired relative phase following the procedure of Section 5.4. Therefore, we will use Algorithm 2, but consider the signing oracle $\mathcal{S}$ (as defined in Equation 7.1) instead of the Prover oracle $\mathcal{P}$, where the signing oracle for the bit k of the monomial j of polynomial i of the secret-key $\mathbf{s}$ is also decomposed bit-by-bit into the family $\mathcal{S} = \{\mathcal{S}_k^{[i](j)}\}_{i,j,k}$, which we show how to construct below. We again start by considering a single coefficient of a single polynomial, (wlog) $s = \mathbf{s}^{[0](0)}$.

1. To recover the LSB of s , use the LSB phase oracle of Section 5.2 by setting $|\mathbf{u}_0^{[0](0)}\rangle = |-\rangle$ and $|\mathbf{u}_k^{[i](j)}\rangle = |+\rangle$ for $(i, j, k) \neq (0, 0, 0)$. Set the challenge register $|c\rangle = |+\rangle |0\rangle^{\otimes wn-1}$, evaluate the oracle

$\mathcal{S}$, and measure the first qubit of the register $|c\rangle$ in the Hadamard basis to recover s_0 , since

$$\begin{aligned}
& \left(I^{\otimes |\mu|} \otimes H \otimes I^{\otimes wn + \ell mn - 1} \right) \mathcal{S} \underbrace{|\mu\rangle}_{\text{message reg.}} \underbrace{\sum_{c \in \{0,1\}} |c\rangle |0\rangle^{\otimes wn-1}}_{\text{challenge reg.}} \underbrace{|-\rangle |+\rangle^{\otimes \ell mn-1}}_{\text{response reg.}} \\
&= |\mu\rangle \left(H \sum_{c \in \{0,1\}} (-1)^{p(c)_0^{[0](0)}} |c\rangle \right) |\mathbf{H}(\mu, h(\mathbf{e}))_{1...wn-1}\rangle |-\rangle |+\rangle^{\otimes \ell mn-1} \\
&= |\mu\rangle \left(H \sum_{c \in \{0,1\}} (-1)^{(c \oplus \mathbf{H}(\mu, h(\mathbf{e}))_0) s_0 \oplus e_0} |c\rangle \right) |\mathbf{H}(\mu, h(\mathbf{e}))_{1...wn-1}\rangle |-\rangle |+\rangle^{\otimes \ell mn-1} \\
&= (-1)^{\mathbf{H}(\mu, h(\mathbf{e}))_0 s_0 \oplus e_0} |\mu\rangle \left(H \sum_{c \in \{0,1\}} (-1)^{cs_0} |c\rangle \right) |\mathbf{H}(\mu, h(\mathbf{e}))_{1...wn-1}\rangle |-\rangle |+\rangle^{\otimes \ell mn-1} \\
&= (-1)^{e'_0} |\mu\rangle |s_0\rangle |\mathbf{H}(\mu, h(\mathbf{e}))_{1...wn-1}\rangle |-\rangle |+\rangle^{\otimes \ell mn-1}.
\end{aligned} \tag{7.5}$$

2. To recover the second LSB of s , use again the Iterative LSB phase oracle of Section 5.4. Let $|\mathbf{u}_0^{[0](0)}\rangle |\mathbf{u}_1^{[0](0)}\rangle = |0\rangle |-\rangle$ and $|\mathbf{u}_k^{[i](j)}\rangle = |+\rangle$ for all other i, j, k . Set the register $|c\rangle = |+\rangle |0\rangle^{\otimes wn-1}$, evaluate the oracle $\mathcal{S}$, to get

$$\begin{aligned}
& \mathcal{S} |\mu\rangle \left(\sum_{c \in \{0,1\}} |c\rangle |0\rangle^{\otimes wn-1} \right) (|0\rangle |-\rangle |+\rangle^{\otimes \ell mn-2}) \\
&= |\mu\rangle \left(\sum_{c \in \{0,1\}} (-1)^{p(c)_1^{[0](0)}} |c\rangle |\mathbf{H}(\mu, h(\mathbf{e}))_{1...wn-1}\rangle |cs_0 \oplus e'_0\rangle \right) |-\rangle |+\rangle^{\otimes \ell mn-2} \\
&= |\mu\rangle \left(\sum_{c \in \{0,1\}} (-1)^{cs_1 \oplus e'_1 \oplus \text{carry}_0} |c\rangle |\mathbf{H}(\mu, h(\mathbf{e}))_{1...wn-1}\rangle |cs_0 \oplus e'_0\rangle \right) |-\rangle |+\rangle^{\otimes \ell mn-1}.
\end{aligned} \tag{7.6}$$

Now, as in the corresponding ID scheme (Section 6.2), there is the need to (i) uncompute the carry_0 that depends on c from the phase, and (ii) uncompute c from the register $|cs_0 \oplus e'_0\rangle$, where s_0 is known from Step 1. Finally, we measure the qubit of the register $|c\rangle$ in the Hadamard basis to recover s_1 .

$$\begin{aligned}
& (-1)^{e'_1} |\mu\rangle \left(H \sum_{c \in \{0,1\}} (-1)^{cs_1} |c\rangle \right) |\mathbf{H}(\mu, h(\mathbf{e}))_{1...wn-1}\rangle |-\rangle |+\rangle^{\otimes \ell mn-1} \\
&= (-1)^{e'_1} |\mu\rangle |s_1\rangle |\mathbf{H}(\mu, h(\mathbf{e}))_{1...wn-1}\rangle |-\rangle |+\rangle^{\otimes \ell mn-1}.
\end{aligned} \tag{7.7}$$

3...v The remaining bits follow analogously^c to recover the entire monomial s .

Since each polynomial of $\mathbf{s}$ multiplies independently by c , and each monomial multiplies independently by the constant term of c (that is the only element in superposition), the other terms $\mathbf{s}^{[i](j)}$ can be recovered analogously, setting the qubits of the terms not in the monomial $\mathbf{u}^{[i](j)}$ to the eigenvector $|+\rangle$, as in Algorithm 2. This completes the full key-recovery superposition attack.

8 Key-recovery attack against *Dilithium* in the Q2 model

The *Dilithium* signature scheme [BDK⁺21, DKL⁺18] and its standardised version *ML-DSA* [Nat24] (FIPS 204) are also based on the Fiat-Shamir with Aborts construction proposed in [Lyu09]. In this section we assess

^cFor a schematic view of the entire process, refer to Section 6.2.

how the signature schemes in the *Dilithium* family hold with respect to the attack techniques introduced in this work. As a result, we present key-recovery superposition attacks against the original *Dilithium* signature scheme [DKL⁺18], the NIST PQC standardisation Round 1 [DKL⁺17] and Round 2 [DKL⁺19] versions, as well as the “Template Scheme” of the final pre-standard version 3.1 of *Dilithium* [BDK⁺21, Figure 1].

However, we are not able to apply the attack against the final, standardised version of the cipher, the NIST standard *ML-DSA*. Changes introduced in the Round 3 and FIPS 204 versions mitigate the attack by introducing a hash function between the output challenge value ($\tilde{c}$) and the challenge used to compute the response (c). We note that this step appears to have been introduced for efficiency purposes, and while it thwarts our specific attack, further investigation of the resilience of this single design choice against quantum superposition attacks is important and remains an interesting direction for further work. We discuss this in more detail in Section 8.2.

Before proceeding, we note one substantial difference between the constructions analysed in this section and the previous one in Section 7.1: the secret-key is now composed of two secret vectors, $\mathbf{s} = (\mathbf{s}_1, \mathbf{s}_2)$. Moreover, the public-key depends on both secret vectors, $\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$, while the signature still has the same form as before and depends on a single secret, $\mathbf{z} = c\mathbf{s}_1 + \mathbf{e}$. Hence, we will describe the attack for recovering $\mathbf{s}_1$ using the Iterative LSB phase oracle of Section 5.4; then, with $\mathbf{s}_1$ and $\mathbf{A}$ it is trivial to recover $\mathbf{s}_2 = \mathbf{t} - \mathbf{A}\mathbf{s}_1$. Therefore it suffices to the recovery of $\mathbf{s}_1$ for the full key-recovery of the signature scheme. The parameters used in our analysis are given in Table B.3 in Appendix B.

8.1 Key-recovery attacks on original *Dilithium*, NIST PQC Round 1–2 versions, and the *Dilithium* 3.1 template version

We start our analysis by considering this final scheme, the “Template version” of *Dilithium* 3.1. In the technical specification of the final pre-standard version of *Dilithium* (version 3.1), a “Template version” is introduced ([BDK⁺21, Figure 1], also presented in Figure A.3 in Appendix A). This description presents the main operations of the full digital signature, without going into detail on the implementation specifics. It serves as a good ground to present our analysis. While the specification and notation in this section focus on this “Template version” the same procedures apply to the full scheme of the original *Dilithium* proposal [DKL⁺18] and NIST PQC Round 1 [DKL⁺17] and Round 2 [DKL⁺19] candidates, as presented in Figure A.4 in Appendix A. It will also help understanding the operational details in the full signature implementation that will introduce challenges for applying the attack techniques to the final standardised version *ML-DSA*, as discussed in Section 8.2.

The “Template version” (Figure A.3) is very similar to the signature scheme of [Lyu09] (Figure A.2b). Hence, the same attack procedure will work, and we will again use the LSB of the register $|c\rangle$ in superposition to recover $\mathbf{s}_1$ iteratively bit-by-bit using the Iterative LSB phase oracle of Section 5.4. The behaviour of the oracle given a challenge c may again be described as

$$p(c) = (c + \mathbf{H}(\mu, \mathbf{w}_1))\mathbf{s}_1 + \mathbf{e} = c\mathbf{s}_1 + \mathbf{e}'. \quad (8.1)$$

From here, the same procedure described in Section 7.1 may be used; we refer to that section to avoid repetition. Yet, a significant difference lies in the checks performed, and we use the same assumptions as the heuristic analysis performed in [BDK⁺21] to verify that they still pass even under the proposed attack. The parameters used in this analysis are given in Table B.3.

The first check tests whether $\|\mathbf{z}\|_\infty < \gamma_1 - \beta$. Here, the only difference with the original signature is that the constant coefficient of the challenge polynomial c may have increased its absolute value by 1. Let $c' = c + \mathbf{H}(\mu, \mathbf{w}_1)$, where c is the value of the challenge register input to the oracle (in our case, 0 or 1). The norm of the response $\mathbf{z}$ is bounded by

$$\begin{aligned} \|\mathbf{z}\|_\infty &= \|c'\mathbf{s}_1 + \mathbf{e}\|_\infty \\ &= \|(c + \mathbf{H}(\mu, \mathbf{w}_1))\mathbf{s}_1 + \mathbf{e}\|_\infty \\ &= \|c\mathbf{s}_1 + \mathbf{H}(\mu, \mathbf{w}_1)\mathbf{s}_1 + \mathbf{e}\|_\infty \\ &\leq \|c\mathbf{s}_1\|_\infty + \|\mathbf{H}(\mu, \mathbf{w}_1)\mathbf{s}_1\|_\infty + \|\mathbf{e}\|_\infty \\ &\leq (1 + \|\mathbf{H}(\mu, \mathbf{w}_1)\|_1)\|\mathbf{s}_1\|_\infty + \|\mathbf{e}\|_\infty \\ &= (1 + \tau)\eta + \gamma_1. \end{aligned} \quad (8.2)$$

Thus, given the η increase in the norm of $\mathbf{z}$, the probability that the check passes is given by

$$\Pr_{\mathbf{e}}[\|\mathbf{z}\|_{\infty} < \gamma_1 - \beta] = \left(\frac{2(\gamma_1 - 2\tau\eta - \eta) - 1}{2\gamma_1 - 1} \right)^{\ell n}, \quad (8.3)$$

which for all proposed security level parameters is larger than 29%. The second check, $\|\text{LowBits}(\mathbf{A}\mathbf{e} - c'\mathbf{s}_2, 2\gamma_2)\|_{\infty} < \gamma_2 - \beta$ is required to ensure the correctness of the signature during verification. Here, $\text{LowBits}(r, \alpha) = r \bmod \pm\alpha$, and as in [BDK⁺21, Equation 5], by heuristically assuming that the low order bits are uniformly distributed modulo $2\gamma_2$, one gets

$$\Pr_{\mathbf{e}}[\|\text{LowBits}(\mathbf{A}\mathbf{e} - c'\mathbf{s}_2, 2\gamma_2)\|_{\infty} < \gamma_2 - \beta] = \left(\frac{2(\gamma_2 - 2\tau\eta - \eta) - 1}{2\gamma_2} \right)^{kn}, \quad (8.4)$$

which for all proposed security level parameters is larger than 9.5%. Since both checks need to pass simultaneously, the expected value for the number of repetitions to get a valid signature is approximately 36.

These same check equations are present in the full scheme of the original *Dilithium* proposal [DKL⁺18] and NIST PQC Round 1 [DKL⁺17] and Round 2 [DKL⁺19] candidates, presented in Figure A.4 in Appendix A. Indeed, the dependence of the signature $\mathbf{z}$ on the challenge c is exactly the same as in the “Template version”. However, these include additional checks, such as the ones concerning the hint vector $\mathbf{h}$ and the norm of the $c\mathbf{t}_0$ vector. These checks are considerably more complex to analyse and only an empirical estimate on their rejection probability is known [BDK⁺21, DKL⁺17, DKL⁺18, DKL⁺19], stating a probability of rejection between 1–2%. Still, this probability of failure is substantially smaller than the previously analysed checks in Equations (8.3) and (8.4), hence it does not affect the feasibility of the proposed attacks, beyond a slight increase in the number of repetitions.

8.2 Analysis of *Dilithium 3.1* and *ML-DSA*

In the actual final proposal of *Dilithium 3.1* [BDK⁺21] and the *ML-DSA* standard [Nat24], there are some crucial differences that prevent the direct application of our superposition attacks. Consequently, the attacks proposed in this paper *do not apply to Dilithium 3.1* and *ML-DSA*. However, we believe our work highlights the need for further analysis of their security in the Q2 model, should this setting be incorporated into the threat model.

Concretely, the following change, introduced between the Round 2 and Round 3 versions, is relevant. We quote from the “Changelog” [BDK⁺21]:

Changed the way the “challenge” c is sampled and transmitted. It is now sampled in two stages. In the first, a hash outputting a 32-byte challenge seed $\tilde{c}$ is created by the Prover, and this is what is included in the signature. The polynomial challenge used in the signature scheme is then created using SHAKE with $\tilde{c}$ as the seed. This saves 8 bytes in the signature size.

Crucially, the introduction of SHAKE, modelled by a random oracle H' , removes the linear dependence of $\mathbf{z}$ on the input challenge register $|c\rangle$, as now

$$\mathbf{z} = H'(c + H(\mu, \mathbf{w}_1))\mathbf{s}_1 + \mathbf{e}. \quad (8.5)$$

While this change appears to have been introduced for efficiency reasons, it makes our attack as applied to the “Template version” of the scheme (Section 8.1) infeasible. Nevertheless, the fact that the attack is only mitigated through an efficiency optimisation motivates in our opinion further security analysis of *Dilithium/ML-DSA* in the Q2 model.

Moreover, in *Dilithium 3.1* there are two modes of operation, where the error $\mathbf{e}$ may be either random or deterministic (dependent on the message and pre-computed randomness in the secret-key). While for our attack model this mode of operation is not relevant, since we do not use the message in superposition, it gives a strong indication that the deterministic version is more robust against superposition attacks than the randomised version. This is because in the former, the error depends on the message and will therefore

be also in superposition when the message is queried in superposition. In fact, this may be related to the observation that some security proofs for *Dilithium* in the QROM have only been established for its deterministic variant [KLS18]. Moreover, in *ML-DSA*, the randomised mode of operation is converted to a “hedged” variant, where the error takes fresh randomness, together with the message and pre-computed randomness in the secret-key (to mitigate faulty random number generation). This seems to give some security against superposition attacks when taking the message in superposition.

Still, the fact that all underlying ID schemes are vulnerable to the proposed LSB attacks warrants further investigation, independently of implementation assumptions on the Fiat-Shamir signatures. Finally, while these attacks to the ID schemes require a dishonest verifier, the use of the ROM to prove security (both in the Fiat-Shamir transform and the MSIS to SelfTargetMSIS reduction) instead of the QROM, introduces issues in the security guarantees. Our work suggests that, in order to claim full quantum security, a more in-depth analysis of the security of *Dilithium/ML-DSA* in the Q2 model is required.

9 Discussion

In this work, we demonstrate the first superposition attacks against public-key cryptography schemes, with a focus on the post-quantum *Dilithium* signature scheme that underlies the *ML-DSA* standard. We present multiple attacks on the line-of-work—which includes several ID schemes and digital signature constructions—that led to the realisation of Dilithium, demonstrating key-recovery attacks in the Q2 model. This is the same security model pursued to achieve the full quantum security of the *Dilithium* signature scheme [BBD⁺23, DFP⁺23, JMW24] in the QROM. To achieve this, we introduce three novel techniques: an *LSB phase oracle*, a *multiple LSBs phase oracle*, and an *iterative LSBs phase oracle*, which beyond their application to our results, should themselves be of independent interest. These techniques are the first of their kind to allow the extraction of private information even against schemes that operate over non-Boolean functions, prevalent in public-key cryptography.

Conceptually, the conversion of the attacks from ID schemes to signature schemes using the Fiat-Shamir transform introduces several challenges on how to model the output of the signature scheme. In fact, since in a quantum computer implementation of a Fiat-Shamir-based signature the challenge register needs to be output, it must also be an input register. Hence, no single black-box, implementation independent, transformation may exist, as for instance the input challenge register may or may not be used to directly perform the multiplication with the secret, impacting the applicability of the attacks. We show that in a proposed implementation (of Figure 4) the attack is often possible. We believe this to be the most natural implementation that simply chains the two classical operations without the need for an ancillary system. However, using ancillary systems it is possible to avoid this direct multiplication, and therefore we consider our work an important demonstration of the added care required for a quantum implementation, or even for a security model to be considered in a full quantum proof of a signature scheme (e.g., the proof of *Dilithium* in the QROM).

Moreover, our results demonstrate important facts regarding the security of the Fiat-Shamir transform in the Q2 model, showing that, under certain conditions, the honest-verifier zero-knowledge security property of an identification scheme is not sufficient to guarantee security. In our setting, this stems from the fact that the input register which accounts for the output of the challenge may itself be used in the computation. This again raises concerns that must be considered when evaluating the quantum security of digital signature algorithms based on similar constructions.

This work leaves several open questions, most notably whether the attacks can be adapted and extended to apply to the final standardised version of *ML-DSA*, since the mitigation to our attacks in relation to the previous versions of *Dilithium* appears to rely primarily on an efficiency improvement introduced in its third version. Still, further important directions would include studying the attack against different implementations of the signature oracle of a signature scheme, connecting these attacks to the current proofs of security in the QROM of *Dilithium*, and extending our attacks to more post-quantum cryptosystems.

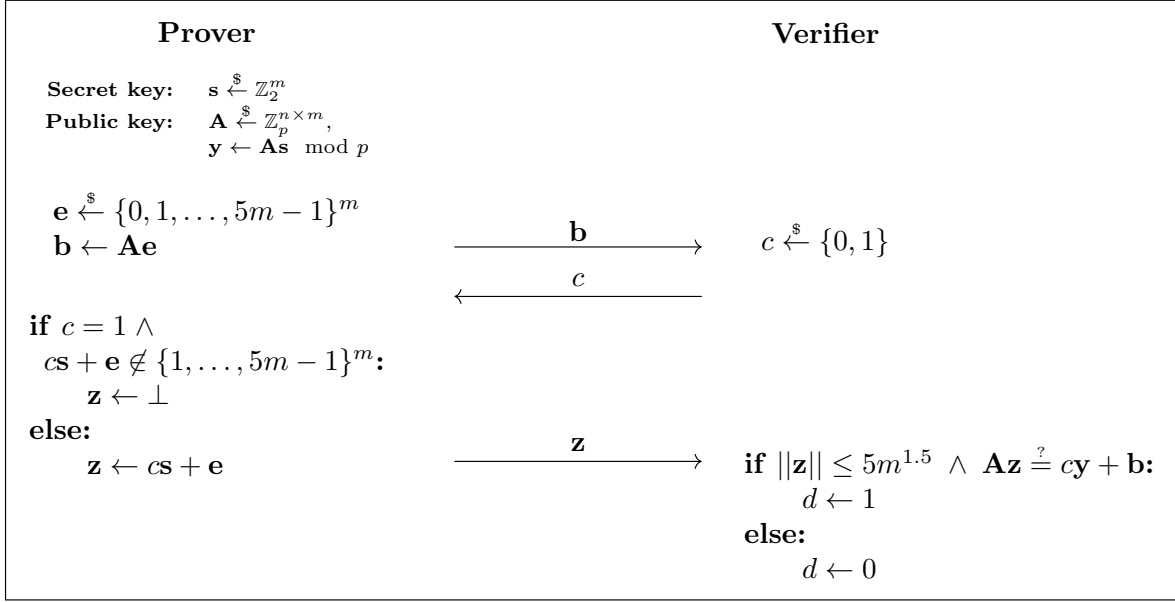
References

- [ARU14] Andris Ambainis, Ansis Rosmanis, and Dominique Unruh. Quantum Attacks on Classical Proof Systems: The Hardness of Quantum Rewinding. In *Proceedings of the 2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, page 474–483, USA, 2014. IEEE Computer Society. doi:10.1109/FOCS.2014.57.
- [BBC⁺21] Ritam Bhaumik, Xavier Bonnetain, André Chailloux, Gaëtan Leurent, María Naya-Plasencia, André Schrottenloher, and Yannick Seurin. QCB: Efficient Quantum-Secure Authenticated Encryption. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021*, pages 668–698, Cham, 2021. Springer International Publishing. doi:10.1007/978-3-030-92062-3_23.
- [BBD⁺23] Manuel Barbosa, Gilles Barthe, Christian Doczkal, Jelle Don, Serge Fehr, Benjamin Grégoire, Yu-Hsuan Huang, Andreas Hülsing, Yi Lee, and Xiaodi Wu. Fixing and Mechanizing the Security Proof of Fiat-Shamir with Aborts and Dilithium. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023*, pages 358–389, Cham, 2023. Springer Nature Switzerland. doi:10.1007/978-3-031-38554-4_12.
- [BDK⁺21] Shi Bai, Léo Lucas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium: Algorithm Specifications and Supporting Documentation (Version 3.1). Technical report, Submission to the NIST Post-Quantum Cryptography Standardization, 2021. <https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf>.
- [BLNS21] Xavier Bonnetain, Gaëtan Leurent, María Naya-Plasencia, and André Schrottenloher. Quantum Linearization Attacks. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021*, pages 422–452, Cham, 2021. Springer International Publishing. doi:10.1007/978-3-030-92062-3_15.
- [BNS20] Xavier Bonnetain, María Naya-Plasencia, and André Schrottenloher. On Quantum Slide Attacks. In Kenneth G. Paterson and Douglas Stebila, editors, *Selected Areas in Cryptography – SAC 2019*, pages 492–519, Cham, 2020. Springer International Publishing. doi:10.1007/978-3-030-38471-5_20.
- [Bon18] Xavier Bonnetain. Quantum Key-Recovery on Full AEZ. In Carlisle Adams and Jan Camenisch, editors, *Selected Areas in Cryptography – SAC 2017*, pages 394–406, Cham, 2018. Springer International Publishing. doi:10.1007/978-3-319-72565-9_20.
- [BS24] Xavier Bonnetain and André Schrottenloher. Single-Query Quantum Hidden Shift Attacks. *IACR Transactions on Symmetric Cryptology*, 2024(3):266–297, Sep. 2024. doi:10.46586/tosc.v2024.i3.266-297.
- [BZ13] Dan Boneh and Mark Zhandry. Quantum-Secure Message Authentication Codes. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, pages 592–608, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. doi:10.1007/978-3-642-38348-9_35.
- [CEG26] Carlos Cid, David Elkouss, and Manuel Goulão. Superposition Attacks Against LPN-Based Authentication Protocols. *IACR Communications in Cryptology*, 3(1), 2026. doi:10.62056/abhey7n4e.
- [CHLS20] Carlos Cid, Akinori Hosoyamada, Yunwen Liu, and Siang Meng Sim. Quantum Cryptanalysis on Contracting Feistel Structures and Observation on Related-Key Settings. In Karthikeyan Bhargavan, Elisabeth Oswald, and Manoj Prabhakaran, editors, *Progress in Cryptology – INDOCRYPT 2020*, pages 373–394, Cham, 2020. Springer International Publishing. doi:10.1007/978-3-030-65277-7_17.

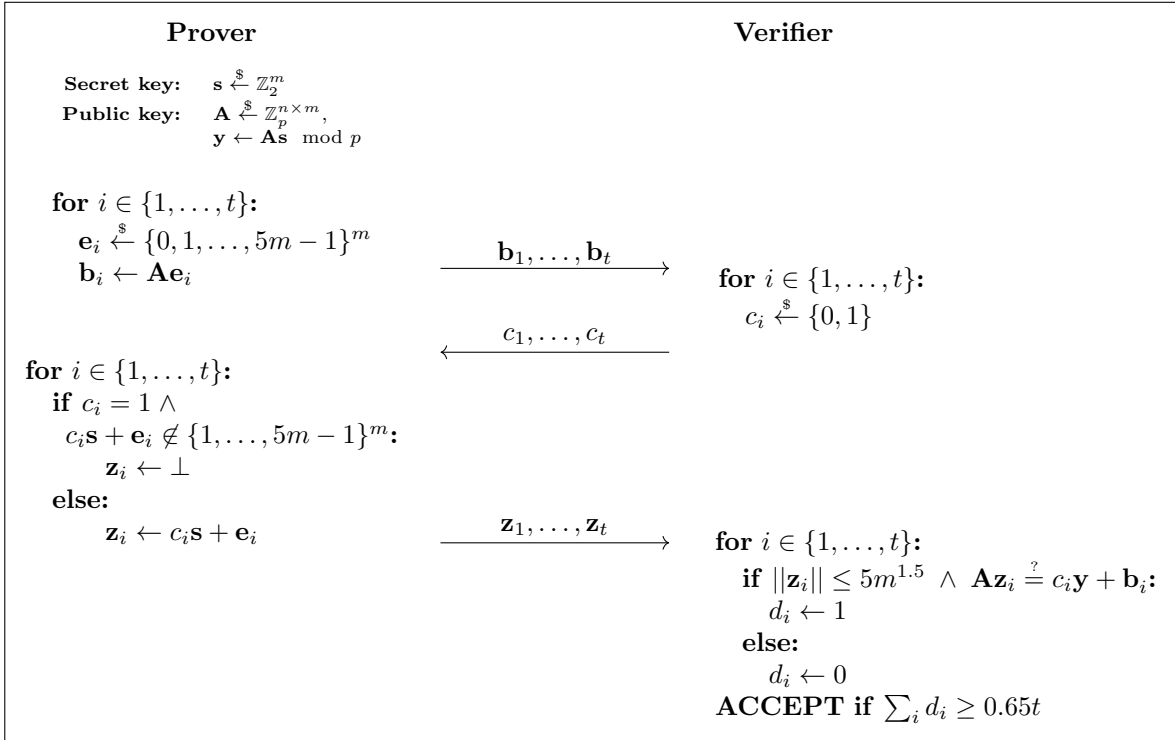
- [DFG13] Özgür Dagdelen, Marc Fischlin, and Tommaso Gagliardoni. The Fiat-Shamir Transformation in a Quantum World. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology - ASIACRYPT 2013*, pages 62–81, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. doi:10.1007/978-3-642-42045-0_4.
- [DFMS19] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Security of the Fiat-Shamir Transformation in the Quantum Random-Oracle Model. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019*, pages 356–383, Cham, 2019. Springer International Publishing. doi:10.1007/978-3-030-26951-7_13.
- [DFNS13] Ivan Damgård, Jakob Funder, Jesper Buus Nielsen, and Louis Salvail. Superposition Attacks on Cryptographic Protocols. In Carles Padró, editor, *Information Theoretic Security*, pages 142–161, Cham, 2013. Springer International Publishing. doi:10.1007/978-3-319-04268-8_9.
- [DFP⁺23] Julien Devevey, Pouria Fallahpour, Alain Passelègue, Damien Stehlé, and Keita Xagawa. A Detailed Analysis of Fiat-Shamir with Aborts. Cryptology ePrint Archive, Paper 2023/245, previous version in CRYPTO 2023, 2023. URL: <https://eprint.iacr.org/2023/245>.
- [DKL⁺17] Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium: Algorithm Specifications and Supporting Documentation. Technical report, Submission to the NIST Post-Quantum Cryptography Standardization (Round 1), 2017. <https://pq-crystals.org/dilithium/data/dilithium-specification.pdf>.
- [DKL⁺18] Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(1):238–268, Feb. 2018. doi:10.13154/tches.v2018.i1.238-268.
- [DKL⁺19] Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium: Algorithm Specifications and Supporting Documentation. Technical report, Submission to the NIST Post-Quantum Cryptography Standardization (Round 2), 2019. <https://pq-crystals.org/dilithium/data/dilithium-specification-round2.pdf>.
- [FS86] Amos Fiat and Adi Shamir. How To Prove Yourself: Practical Solutions to Identification and Signature Problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology — CRYPTO’ 86*, pages 186–194, Berlin, Heidelberg, 1986. Springer Berlin Heidelberg. doi:10.1007/3-540-47721-7_12.
- [GKZ19] Alex B. Grilo, Iordanis Kerenidis, and Timo Zijlstra. Learning-with-errors problem is easy with quantum samples. *Phys. Rev. A*, 99:032314, Mar 2019. doi:10.1103/PhysRevA.99.032314.
- [HS18] Akinori Hosoyamada and Yu Sasaki. Quantum Demirci-Selçuk Meet-in-the-Middle Attacks: Applications to 6-Round Generic Feistel Constructions. In Dario Catalano and Roberto De Prisco, editors, *Security and Cryptography for Networks*, pages 386–403, Cham, 2018. Springer International Publishing. doi:10.1007/978-3-319-98113-0_21.
- [IHM⁺19] Gembu Ito, Akinori Hosoyamada, Ryutaroh Matsumoto, Yu Sasaki, and Tetsu Iwata. Quantum Chosen-Ciphertext Attacks Against Feistel Ciphers. In Mitsuru Matsui, editor, *Topics in Cryptology - CT-RSA 2019*, pages 391–411, Cham, 2019. Springer International Publishing. doi:10.1007/978-3-030-12612-4_20.
- [JMW24] Kelsey A. Jackson, Carl A. Miller, and Daochen Wang. Evaluating the Security of CRYSTALS-Dilithium in the Quantum Random Oracle Model. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology - EUROCRYPT 2024*, pages 418–446, Cham, 2024. Springer Nature Switzerland. doi:10.1007/978-3-031-58751-1_15.

- [KLLN16a] Marc Kaplan, Gaëtan Leurent, Anthony Leverrier, and María Naya-Plasencia. Breaking Symmetric Cryptosystems Using Quantum Period Finding. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016*, pages 207–237, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg. doi:10.1007/978-3-662-53008-5_8.
- [KLLN16b] Marc Kaplan, Gaëtan Leurent, Anthony Leverrier, and María Naya-Plasencia. Quantum Differential and Linear Cryptanalysis. *IACR Transactions on Symmetric Cryptology*, 2016(1):71–94, Dec. 2016. doi:10.13154/tosc.v2016.i1.71-94.
- [KLS18] Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. A Concrete Treatment of Fiat-Shamir Signatures in the Quantum Random-Oracle Model. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018*, pages 552–586, Cham, 2018. Springer International Publishing. doi:10.1007/978-3-319-78372-7_18.
- [KM10] Hidenori Kuwakado and Masakatu Morii. Quantum distinguisher between the 3-round Feistel cipher and the random permutation. In *2010 IEEE International Symposium on Information Theory*, pages 2682–2685, 2010. doi:10.1109/ISIT.2010.5513654.
- [KM12] Hidenori Kuwakado and Masakatu Morii. Security on the quantum-type Even-Mansour cipher. In *2012 International Symposium on Information Theory and its Applications*, pages 312–316, 2012. <https://ieeexplore.ieee.org/document/6400943>.
- [LM17] Gregor Leander and Alexander May. Grover Meets Simon – Quantumly Attacking the FX-construction. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 161–178, Cham, 2017. Springer International Publishing. doi:10.1007/978-3-319-70697-9_6.
- [Lyu08] Vadim Lyubashevsky. Lattice-Based Identification Schemes Secure Under Active Attacks. In Ronald Cramer, editor, *Public Key Cryptography – PKC 2008*, pages 162–179, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. doi:10.1007/978-3-540-78440-1_10.
- [Lyu09] Vadim Lyubashevsky. Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, pages 598–616, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. doi:10.1007/978-3-642-10366-7_35.
- [LZ19] Qipeng Liu and Mark Zhandry. Revisiting Post-quantum Fiat-Shamir. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019*, pages 326–355, Cham, 2019. Springer International Publishing. doi:10.1007/978-3-030-26951-7_12.
- [Nat17] National Institute of Standards and Technology. Post-Quantum Cryptography Standardization webpage, 2017. URL: <https://csrc.nist.gov/pqc-standardization>. <https://csrc.nist.gov/pqc-standardization>.
- [Nat24] National Institute of Standards and Technology. FIPS PUB 204: Module-Lattice-Based Digital Signature Standard. Federal Information Processing Standards Publication 204, National Institute of Standards and Technology, 2024. <https://csrc.nist.gov/pubs/fips/204/final>.
- [NC10] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.
- [Sch89] C. P. Schnorr. Efficient Identification and Signatures for Smart Cards. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO’ 89 Proceedings*, pages 239–252, New York, NY, 1989. Springer New York. doi:10.1007/0-387-34805-0_22.
- [Sho94] P.W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994. doi:10.1109/SFCS.1994.365700.

A Protocol schemes

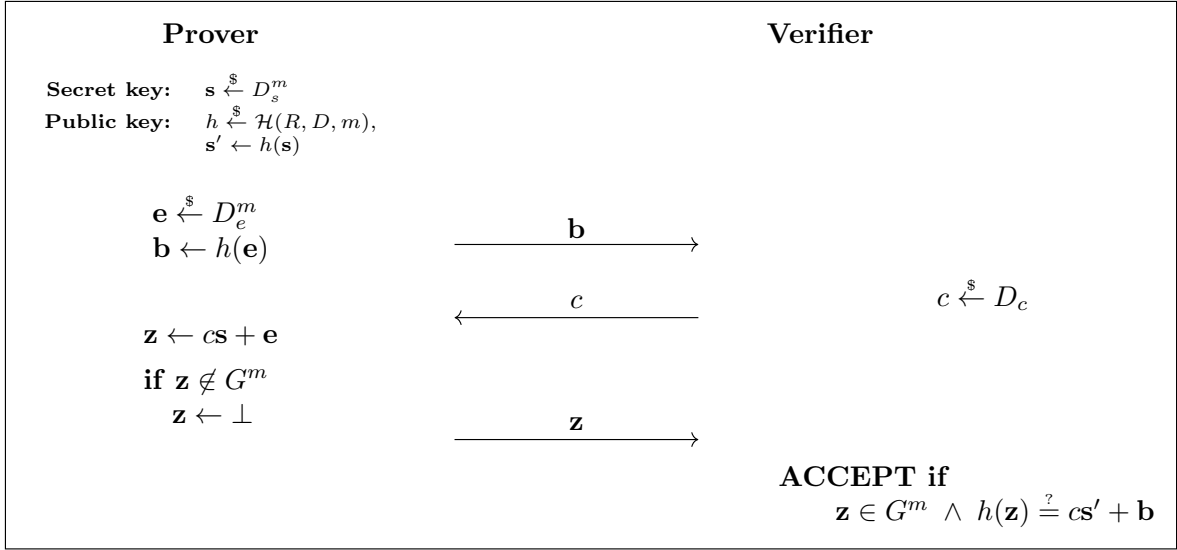


(a) One iteration.

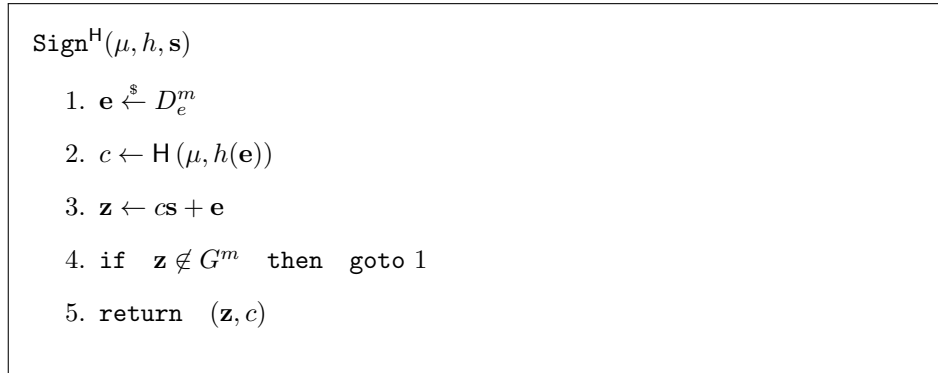


(b) Batch (full) execution.

Figure A.1: ID protocol [Lyu08]



(a) ID scheme [Lyu09]



(b) Signature algorithm [Lyu09]

Figure A.2: Lattice based ID-scheme and signature scheme [Lyu09]

```

SignH( $\mu, \mathbf{A}, \mathbf{s}_1, \mathbf{s}_2$ )
1.  $\mathbf{e} \xleftarrow{\$} D_{\mathbf{e}}$ 
2.  $\mathbf{w}_1 \leftarrow \text{HighBits}(\mathbf{A}\mathbf{e}, 2\gamma_2)$ 
3.  $c \in B_r \leftarrow \text{H}(\mu, \mathbf{w}_1)$ 
4.  $\mathbf{z} \leftarrow c\mathbf{s}_1 + \mathbf{e}$ 
5. if  $\|\mathbf{z}\|_{\infty} \geq \gamma_1 - \beta$  or  $\|\text{LowBits}(\mathbf{A}\mathbf{e} - c\mathbf{s}_2, 2\gamma_2)\|_{\infty} \geq \gamma_2 - \beta$  then
    goto 1
6. return  $(\mathbf{z}, c)$ 

```

Figure A.3: Dilithium “Template for signature scheme” [BDK⁺21].

```

SignH( $\mu, \mathbf{A}, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0$ )
1.  $\kappa \leftarrow 0$ 
2.  $\mathbf{e} \xleftarrow{\$} D_{\mathbf{e}}$  (or  $\mathbf{e} \leftarrow \text{H}(K, \mu, \kappa)$ )
3.  $\mathbf{w} \leftarrow \mathbf{A}\mathbf{e}$ 
4.  $\mathbf{w}_1 \leftarrow \text{HighBits}(\mathbf{w}, 2\gamma_2)$ 
5.  $c \in B_{60} \leftarrow \text{H}(\mu, \mathbf{w}_1)$ 
6.  $\mathbf{z} \leftarrow c\mathbf{s}_1 + \mathbf{e}$ 
7.  $(\mathbf{r}_1, \mathbf{r}_0) \leftarrow \text{Decompose}(\mathbf{w} - c\mathbf{s}_2, 2\gamma_2)$ 
8. if  $\|\mathbf{z}\|_{\infty} \geq \gamma_1 - \beta$  or  $\|\text{LowBits}(\mathbf{A}\mathbf{e} - c\mathbf{s}_2, 2\gamma_2)\|_{\infty} \geq \gamma_2 - \beta$  then
     $\kappa \leftarrow \kappa + 1$ 
    goto 2
9. else
     $\mathbf{h} \leftarrow \text{MakeHint}(-c\mathbf{t}_0, \mathbf{w} - c\mathbf{s}_2 + c\mathbf{t}_0, 2\gamma_2)$ 
    if  $\|c\mathbf{t}_0\|_{\infty} \geq \gamma_2$  or  $\text{HW}(\mathbf{h}) \geq \omega$  then
         $\kappa \leftarrow \kappa + 1$ 
        goto 2
10. return  $(\mathbf{z}, \mathbf{h}, c)$ 

```

Figure A.4: Original Dilithium proposal [DKL⁺18] and NIST PQC Round 1 [DKL⁺17] and Round 2 [DKL⁺19] candidates. Line 2 represents the two versions, randomized and deterministic.

B Parameter tables

Table B.1: Parameters for the analysis of [Lyu08].

Parameter	Description
p	prime order of base finite field ($\tilde{\mathcal{O}}(n^3)$)
n	dimension of public-key (security parameter)
m	dimension of private vectors ($m = \lceil 4n \log_2 n \rceil$)
$5m$	upper bound on the elements of the error vector
t	number of repetitions ($\omega(\log_2 n)$)
ℓ	number of bits to write elements in $\mathbb{Z}_p$ ($\ell = \lceil \log_2 p \rceil$)

Table B.2: Parameters for the analysis of [Lyu09, adapted from Figure 2].

Parameter	Definition	Sample Instantiations			
n	integer that is a power of 2	512	512	512	1024
m	any integer	4	5	8	8
σ	any integer	127	2047	2047	2047
κ	integer s.t. $2^\kappa \binom{n}{\kappa} \geq 2^{160}$	24	24	24	21
p	integer $\approx (2\sigma + 1)^m \cdot 2^{-128/n}$	$2^{31.7}$	$2^{59.8}$	$2^{95.8}$	$2^{95.9}$
R	ring $\mathbb{Z}_p[x]/\langle x^n + 1 \rangle$				
D	$\{g \in R : \ g\ _\infty \leq mn\sigma\kappa\}$				
D_s	$\{g \in R : \ g\ _\infty \leq \sigma\}$				
D_c	$\{g \in R : \ g\ _1 \leq \kappa\}$				
D_e	$\{g \in R : \ g\ _\infty \leq mn\sigma\kappa\}$				
G	$\{g \in R : \ g\ _\infty \leq mn\sigma\kappa - \sigma\kappa\}$				
ℓ	$\lceil \log_2 p \rceil$				
v	$\lceil \log_2 \sigma \rceil$				
w	$\lceil \log_2 \kappa \rceil$				

Table B.3: Parameters for the analysis of [BDK⁺21, adapted from Table 2].

NIST Security Level		2	3	5
Parameters				
q	modulus	8380417	8380417	8380417
d	dropped bits from $\mathbf{t}$	13	13	13
τ	# of ± 1 's in c	39	49	60
γ_1	$\mathbf{e}$ coefficient range	2^{17}	2^{19}	2^{19}
γ_2	low-order rounding range	$(q-1)/88$	$(q-1)/32$	$(q-1)/32$
(k, ℓ)	dimensions of $\mathbf{A}$	(4, 4)	(6, 5)	(8, 7)
η	secret-key range	2	4	2
β	$\tau \cdot \eta$	78	196	120

C General circuit to compute a phase with the i -th carry

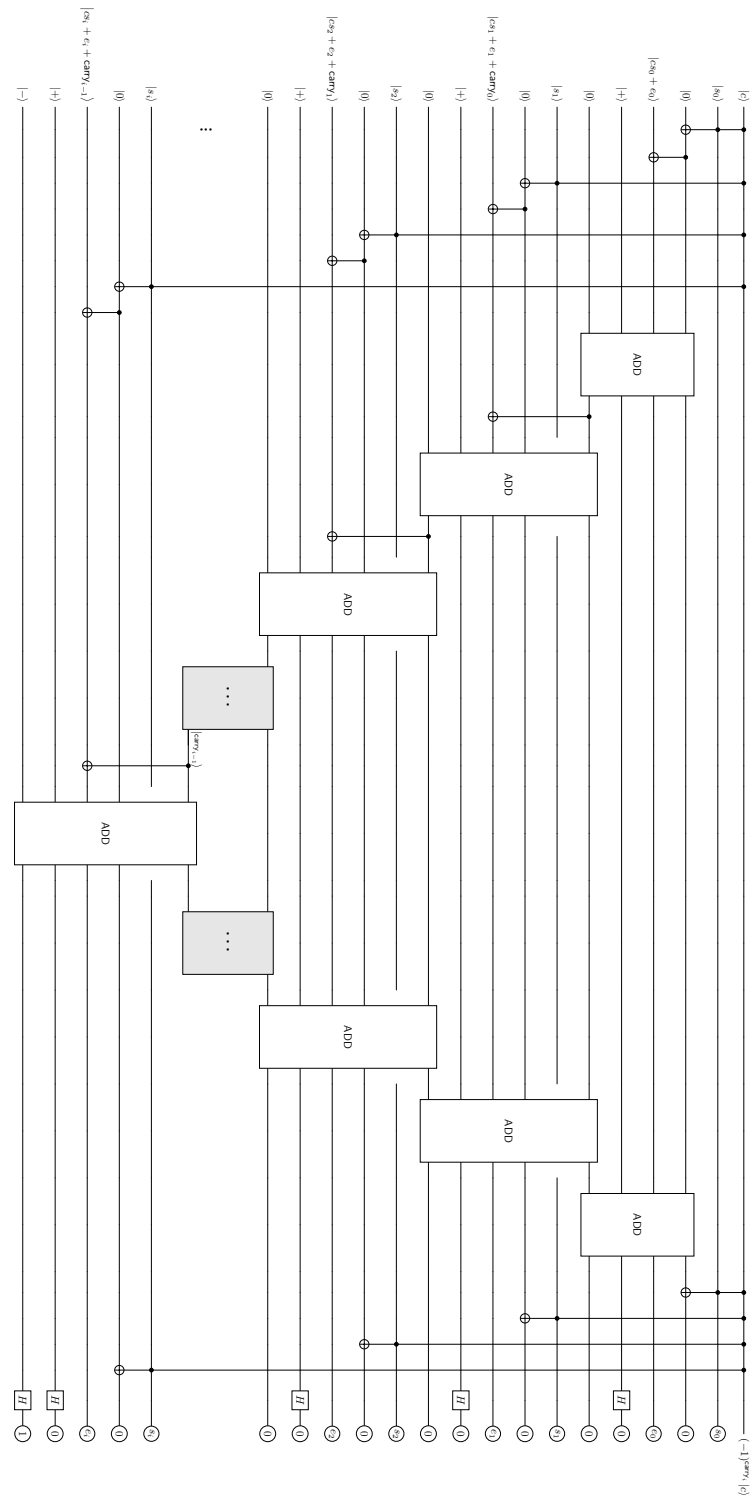


Figure C.1: Quantum circuit to create a $(-1)^{\text{carry}_i}$ phase.