Received: 22 December 2021 | Revised: 23 April 2022 | Accepted: 13 June 2022

DOI: 10.1049/ise2.12081

REVIEW

Quantum algorithms for attacking hardness assumptions in classical and post-quantum cryptography

J.-F. Biasse[1] | X. Bonnetain[2] | E. Kirshanova[3] | A. Schrottenloher[4] | F. Song[5]

[1]Center for Cryptographic Research, University of South Florida, Tampa, Florida, USA

[2]CARAMBA Team, Inria Nancy - Grand Est, Villers-lès-Nancy, France

[3]Mathematical Methods in Information Security Lab, I.Kant Baltic Federal University, Kaliningrad, Kaliningrad Oblast, Russia

[4]Cryptology Group, CWI, Amsterdam, The Netherlands

[5]Department of Computer Science, Portland State University, Portland, Oregon, USA

Correspondence

J.-F. Biasse, Center for Cryptographic Research, University of South Florida, 4202 E Fowler Ave, Tampa, FL 33620, USA.
Email: jf.biasse@gmail.com

Funding information

National Science Foundation, Grant/Award Numbers: 1846166, 2127742; Russian Science Foundation, Grant/Award Number: 21-71-00114

Abstract

In this survey, the authors review the main quantum algorithms for solving the computational problems that serve as hardness assumptions for cryptosystem. To this end, the authors consider both the currently most widely used classically secure cryptosystems, and the most promising candidates for post-quantum secure cryptosystems. The authors provide details on the cost of the quantum algorithms presented in this survey. The authors furthermore discuss ongoing research directions that can impact quantum cryptanalysis in the future.

1 | INTRODUCTION

Quantum computers are a form of computers that leverage quantum-mechanical phenomena to perform computations—unlike today's classical computers that leverage classical physical phenomena.

Sufficiently capable large-scale quantum computers—that are either not prone to errors or error-corrected—would pose a threat to most currently widely deployed asymmetric cryptosystems. This is because Shor [1] has introduced polynomial time quantum algorithms for solving the Integer Factoring Problem (IFP) and the Discrete Logarithm Problem (DLP) in cyclic groups.

A quantum computer capable of executing Shor's algorithms for sufficiently large problem instances would, for example, be able to break RSA [2], which is based on the IFP, and DSA [3] and Diffie–Hellman (DH) [4], which are based on the DLP—mainly in multiplicative groups of finite fields, or groups of points of elliptic curves, in the case of elliptic curve cryptography (ECC) [5, 6].

The aforementioned cryptosystems are currently used to secure most transactions that take place over the Internet. They are used not only to provide confidentiality but also for authentication and to issue binding signatures for non-repudiation purposes. Examples of use cases for these cryptosystems include, but are not limited to, a plethora of services dealing with sensitive personal, corporate or government data.

Needless to say, the aforementioned cryptosystems will need to be replaced with post-quantum secure alternatives prior to the advent of large-scale quantum computers. They will then become susceptible to attacks by adversaries with access to such computers. At first, the set of adversaries with this capability may admittedly be fairly limited, but over time it will likely grow larger.

Whenever a cryptosystem is used to protect confidentiality, it is critical to have idea of the length of time for which the information protected has to remain confidential: Say that the information needs to remain confidential for Δ years. Then the cryptosystem used to protect it must be replaced at the very latest Δ years before it can be broken—by any relevant adversary, quantum or classical. This is because we must assume the adversary to be able to record encrypted traffic sent today for decryption in the future.

For cryptosystems that are used for authentication, or for issuing binding signatures for non-repudiation, the situation is different: Such cryptosystems may be replaced much closer in time to the point when they are broken, even if mitigating actions may then have to be taken to extend the lifetime or signatures that need to remain binding in the long term. For instance, mitigation may be accomplished by a trusted party attesting to having seen the signature at a given point in time prior to when the cryptosystem originally used to issue the signature is broken. Similar mitigation is not possible in the context of providing confidentiality in the long term.

The threat posed by the possible future advent of large-scale quantum computers is sufficiently concerning to already have prompted standardisation processes for post-quantum secure cryptography to be initiated. For example, the National Institute for Standards and Technology (NIST) in the United States (US) has started a standardisation process for post-quantum secure cryptosystems [7]. This process was started after the National Security Agency (NSA) in the US announced back in 2015 that it would transition to post-quantum primitives in its Suite B of cryptosystems [8].

Some standards are already available [9], and over the coming years more are projected to follow [10]. Once available, these standards will need to be integrated into other standards—for example, for protocols—prior to being adopted in products and finally being widely deployed on the Internet. This whole process will take time.

At the same time, it is prudent to begin the process of transitioning to post-quantum secure cryptosystems as soon as possible. In particular, use cases where confidentiality needs to be provided in the long term should be prioritised, for the reasons explained earlier.

Early adopters are wise to deploy post-quantum secure cryptosystems alongside existing classically secure cryptosystems, in such a way that both systems have to be broken simultaneously for the combined hybrid system to be broken. Furthermore, early adopters are wise to beware of the risk of introducing vulnerabilities—for instance in the form of side channels—when implementing new post-quantum secure cryptosystems for which no well-established industry best practices do as of yet exist.

In order to inform decisions on what post-quantum cryptography to standardise and the pace at which the transition must be executed, research into quantum cryptanalysis is needed. Such research allows the community to better understand the costs of attacking classically and post-quantum secure cryptosystems quantumly.

It is against this backdrop that we decided to jointly write this survey paper: In it, we—a subset of the participants at the 2021 Schloss Dagstuhl seminar on quantum cryptanalysis—come together to jointly review the current state of quantum cryptanalysis, with each expert focussing of her or his own area of expertise.

In particular, we survey quantum algorithms for solving the hard problems that underpin the currently most widely deployed classically secure cryptosystems, alongside the post-quantum secure cryptosystems currently primarily considered for standardisation.

The scope of this survey does not include discussing the impact of quantum computing on security notions. In particular, security proofs are impacted in a way that goes beyond the mere ability of a quantum adversary to solve some hard problems more efficiently than a classical adversary, yet such aspects of quantum cryptanalysis are not covered in this survey.

We provide asymptotic cost estimates for some of the quantum algorithms that we discuss in this survey. It is worth noting that such estimates can be used to derive security parameters for cryptosystems that are based on the hard problems that these algorithms solve. This is however not something that we would recommend, without first carrying out additional analyses, including a more fine-grained analysis of the quantum circuit for the algorithm in question and a careful review of the specific design of the cryptosystem in question—all of which is beyond the scope of this survey.

1.1 | Overview

This paper is organised as follows:

In Section 2, we first review some background information on quantum computing and quantum algorithms. We furthermore discuss different methods and models for costing quantum algorithms.

Then, in Sections 3 and 4, we review the two main overarching families of quantum algorithms: search algorithms, which derive from the seminal work of Grover [11], and algorithms for finding a hidden subgroup inside of a control group, which derive from the seminal work of Shor [1].

Next, we survey concrete quantum algorithms for breaking currently widely deployed symmetric and asymmetric cryptosystems:

In Section 5, we first survey quantum algorithms for breaking currently widely deployed asymmetric cryptosystems that are based on the IFP or DLP in some form. In particular, we survey Shor's algorithms for the IFP and DLP and their various derivatives. In Section 6, we briefly digress by discussing generalisations of Shor's algorithms, before surveying algorithms for breaking symmetric cryptosystems, such as hash functions and block ciphers, in Section 7.

Finally, in Sections 8–10, we survey the main quantum algorithms for attacking the hardness assumptions that underpin future lattice-based, code-based and isogeny-based cryptosystems considered for standardisation. For hardness assumptions that underpin hash-based cryptosystems, see instead Section 7.

Note that this survey does not cover all hardness assumptions that have been proposed as foundations for post-quantum cryptography. Instead, we chose to focus on the computational problems that underpin some of the most promising candidates for future post-quantum secure cryptosystems. Additional problems relevant to the cryptanalysis of post-quantum cryptography that are not covered in this survey include the resolution of systems of quadratic equations, the conjugacy problem over certain algebraic groups, and the search for fixed-weight linear coefficients of a relation modulo a Mersenne prime.

2 | BACKGROUND

In this section, we briefly recall basic notions pertaining to quantum computing and quantum algorithms. We furthermore briefly discuss different methods and models for costing quantum algorithms.

For a thorough introduction to the subject, we recommend introductory books such as Nielsen and Chuang [12] or the more concise Kaye, Laflamme and Mosca [13].

2.1 | Qubits

The smallest unit of quantum information is the *qubit*—the name 'qubit' being a contraction of 'quantum bit'. The qubit may be perceived as the quantum analogue of the classical bit; the smallest unit of classical information. But whereas the classical bit is in either one of two states, denoted 0 or a 1, and a qubit is in one of two *computational basis states*, denoted $|0\rangle$ and $|1\rangle$, or in some *superposition* thereof:

Definition 1 (Qubit). *A qubit is a two-level quantum-mechanical system. It is in a state $|\psi\rangle$ given by a normalised sum*

$$|\psi\rangle = c_0\,|0\rangle + c_1\,|1\rangle, \quad c_0, c_1 \in \mathbb{C}, \quad |c_0|^2 + |c_1|^2 = 1,$$

where the two computational basis states $|0\rangle$ and $|1\rangle$ form an orthonormal basis for $\mathbb{C}^2$ given by

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

A classical bit may be read without affecting its state. Such a read operation yields either 0 or 1, depending on the state of the bit.

A qubit may be *observed* in a *measurement*. The probability of observing $j \in \{0, 1\}$ in such a measurement is $|c_j|^2$. If the qubit is in a superposition when it is measured, the superposition *collapses* to the basis state $|j\rangle$ conditioned on the measurement. The state of a qubit is defined up to a *global phase* $\exp^{i\phi}$ which is not observable. Thus, in the definition above, c_0 can be made a real number without loss of generality.

The notion of the qubit is abstract. As is the case for the classical bit, there are possible physical realisations of the qubit. For example, the linear polarisation of a photon could be used to form a qubit (the two levels being the so-called 'up-down polarisation' $\updownarrow$ and 'left-right polarisation' $\leftrightarrow$). Another example are the spins of a spin-1/2 particle (the two levels are being $\uparrow$ (spin up) and $\downarrow$ (spin down)).

2.2 | Systems of qubits

A set of n qubits may be combined to form an n-qubit *system*. Such a system can be in one of 2^n computational basis states, denoted $|j\rangle$ for $j \in [0, 2^n)$, or in some superposition thereof:

Definition 2 (Quantum system). *An n-qubit system is in a state $|\psi\rangle$ given by a normalised sum*

$$|\psi\rangle = \sum_{j=0}^{2^n-1} c_j\,|j\rangle, \quad c_j \in \mathbb{C}, \quad \sum_{j=0}^{2^n-1} |c_j|^2 = 1,$$

where the computational basis states $|j\rangle$ for $j \in [0, 2^n)$ form an orthonormal basis for $\mathbb{C}^{2^n}$ given by

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix}, \quad \cdots, \quad |2^n - 1\rangle = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix}.$$

Two independent quantum systems, of n_1 and n_2 qubits, that are in the states $|\psi_1\rangle$ and $|\psi_2\rangle$, respectively, may be perceived as a single system. The resulting $n_1 + n_2$-qubit system is then in the *product state* given by $|\psi_1\rangle \otimes |\psi_2\rangle$, where $\otimes$ denotes the tensor product. For compactness, we write $|\psi_1\rangle \otimes |\psi_2\rangle = |\psi_1\rangle|\psi_2\rangle = |\psi_1, \psi_2\rangle$. If a state is not a product state, then it said to be *entangled*. Like for an individual qubit, a global phase is irrelevant.

An n-qubit system may be split into one or more n_i-qubit *sub-systems* or *registers*. The computational basis states of such an n_i-qubit sub-system may be indexed using any set, for as long as there is an injective map from the set to the 2^{n_i} basis states.

2.3 | Measurements

An arbitrary n-qubit subsystem of an $n + m$-qubit system may be measured, in any orthonormal basis, by applying the below measurement postulate (up to re-ordering the qubits):

Definition 3 (Measurement postulate). *Let $(|\psi_i\rangle)_{i \le 2^n}$ be an orthonormal basis corresponding to an observable of an n-qubit system. Assume that the state $|\psi\rangle \in \mathbb{C}^{2^{n+m}}$ of an $n + m$-qubit system decomposes as*

$$|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |\psi_i\rangle|\gamma_i\rangle \quad \textit{for some} \quad |\gamma_i\rangle \in \mathbb{C}^{2^m}, \alpha_i \in \mathbb{C},$$

where $|\gamma_i\rangle$ is a state and hence of norm one. Then the measurement of the first n qubits yields the outcome i with probability $|\alpha_i|^2$ and collapses the system to the state $|\psi_i\rangle|\gamma_i\rangle$.

2.4 | Quantum computation

In essence, the goal of a *quantum algorithm* is to *evolve* the state of a quantum system to a point where a measurement yields classical information on the solution to a computational problem of interest.

Quantum theory postulates that the evolution over time of the state of a closed quantum system is described by a unitary *operator*:

Definition 4 (Evolution postulate). *The time evolution of the state of a closed quantum system is described by a unitary operator. For any evolution $|\psi_1\rangle \rightarrow |\psi_2\rangle$ of the closed system, there is a unitary operator U such that*

$$|\psi_2\rangle = U\,|\psi_1\rangle.$$

A consequence of this postulate is that quantum computing is always *reversible*, in contrast to classical computing. The reverse of a computation U is also called its *uncomputation* and denoted $U^\dagger$.

Quantum algorithms are compiled to *quantum circuits*. A circuit consists of a sequence of *gates* and measurements. A gate applies a unitary operator to one or more qubits in a quantum system.

Some common single-qubit operators are the Pauli operators

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix},$$

the Hadamard (H) operator, and the T and S phase-shift operators

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}, S = T^2.$$

The X operator is also known as the 'NOT' operator since it maps $|0\rangle$ onto $|1\rangle$, and vice versa.

Operators may be *controlled*, that is, applied to some qubit conditioned on other qubits being in either the $|1\rangle$ or $|0\rangle$ state. A common such operator is the two-qubit controlled-NOT or CNOT operator

$$\text{CNOT} = \begin{pmatrix} I & 0 \\ 0 & X \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \qquad (1)$$

that maps $|a, b\rangle \mapsto |a, a \oplus b\rangle$. It may be seen to perform a reversible XOR-operation (denoted $\oplus$). Similarly, there is a doubly controlled-NOT or Toffoli operator that maps $|a, b, c\rangle \mapsto |a, b, c \oplus (a \wedge b)\rangle$. It may be seen to perform a reversible AND operation (denoted $\wedge$).

Any single-qubit operator U may be controlled by replacing X for U in the expression (1) for the CNOT operator. There are methods for constructing more advanced controlled operators.

For U_1 an n_1-qubit operator, and U_2, and n_2 qubit operator, the $n_1 + n_2$-qubit operator $U = U_1 \otimes U_2$ applies U_1 to the first n_1 qubits of an $n_1 + n_2$-qubit system and U_2 to the trailing n_2 qubits. Similarly, for U_1 and U_2, two n-qubit operators, the product operator $U = U_2U_1$ applies U_1 and then U_2 to an n-qubit system.

This provides the necessary theoretical framework for building operators corresponding to large circuits from few-qubit operators.

2.5 | Universal quantum computation

Any n-qubit operator U can be reduced to an n-qubit circuit C containing only single-qubit gates and CNOT gates—although this reduction is not necessarily efficient—giving rise to one notion of universal quantum computation.

Furthermore, C can be efficiently approximated to any given degree of precision by a circuit C' that consists exclusively of H and S gates—that generate the *Clifford group* for single qubits—along with the CNOT operator and the non-Clifford T operator.

This is a consequence of Solovay–Kitaev's theorem—if one glosses over some details, such as adjustments to the global phase and to determinants, and the need to include inverses in the gate set. The interested reader is referred to [Ref. 12, Appendix 3]. Note that there are many *universal gate sets*, but Clifford + CNOT + T (for Clifford = $\{H, S\}$) is arguably one of the most popular.

For $f(x)$ any efficient classical function, we can use operators from a universal gate set to construct an efficient quantum circuit that takes $|a, b\rangle \rightarrow |a, b \oplus f(a)\rangle$. Indeed, generic techniques due to Bennett [14] convert any classical algorithm taking time T and space S into a reversible algorithm taking time $T^{1+\epsilon}$ and space $O(S \log T)$. Note that this implies that an *ancillary register*, or *ancilla*, may be required for this to work, depending on f and the size of the two registers. If $f(x)$ has an efficient classical inverse, then the above implies that we can construct a circuit that takes $|a\rangle \rightarrow |f(a)\rangle$.

2.6 | Architectural constraints

Up to this point, we have essentially assumed that single- and two-qubit quantum gates may be freely applied to any of the qubits that make up our system. The same goes for measurements.

In practice, there are, however, often architectural constraints. For instance, the connectivity between qubits may be limited to some form of nearest-neighbour connectivity. There may, furthermore, be restrictions on which gates can be applied to which qubits, or to which pairs of qubits, and so forth. Such constraints can be overcome, for instance by routing, but routing comes at a cost.

When costing circuits for quantum algorithms in this survey, we do not account for architectural constraints, since we do not specify a specific architecture.

2.7 | Quantum error correction

Most quantum computers as currently envisaged are prone to errors arising—for instance when gates are applied or when measurements are performed—necessitating *quantum error correction*.

The basic idea in quantum error correction is to use multiple *physical* qubits to construct a *logical* qubit. The redundancy thus induced allows for errors to be detected and corrected.

The number of physical qubits required to construct a logical qubit depends on the number of operations that the logical qubit is required to be able to undergo in sequence without the risk of uncorrectable errors arising growing too large. It furthermore depends on the physical characteristics of the quantum computer, such as the physical error rates of operators and measurements, and of course on the choice of quantum error-correcting code to employ.

There are a number of proposals for quantum error-correcting codes. One of the key contenders is stabiliser codes, such as the surface code. For a good introduction to the surface code, see Ref. [15].

It is important to understand that the overhead induced by quantum error correction may be substantial. A distinction must therefore be made between physical and logical qubit counts and between physical and logical cost estimates for circuits in general.

When costing circuits for quantum algorithms in this survey, we do so without accounting for the need for error correction since we do not specify a specific architecture.

2.8 | Logical cost estimates

To estimate the post-quantum security level of a cryptosystem, we need to determine not only which quantum algorithm is currently the most efficient for breaking the system but also which metric to use to best cost the algorithm depending on the context at hand.

Metrics commonly used to coarsely quantify the cost of quantum algorithms at a logical level of abstraction include

- the number of gates in some logical circuit for the algorithm,
- the (maximum) depth in gates of said circuit,
- the (maximum) width in logical qubits of said circuit,
- the product of the (maximum) depth and width of said circuit, and
- similar metrics considering only non-Clifford gates, since they are typically assumed to be harder to implement.

When using metrics such as the above, the gate set employed to lay out the circuit, and in particular the level to which the circuit is optimised when it is laid out, may impact the cost estimate, as may any architectural limitations that are taken into account.

The choice of a certain metric may be influenced by a hypothesis on the behaviour of quantum hardware. For example, one might focus on the execution time without considering costs relative to memory. This is done by using only the depth of the circuit. On the other hand, the quantification of the impact of memory requirement is influenced by assumptions on error correction. Indeed, as pointed out in Ref. [16], if we assume active error correction, then the depth-width product (hereinafter DW-cost) is a better measure of the cost than the gate count. To see why, note that the latter metric, for example, captures the cost of an idling qubit while the former completely ignores it.

To complicate matters further, qubits may be measured, left idle and later re-initialised when a circuit is executed. This is the case in particular for ancillary registers. This implies that there is not necessarily a completely unambiguous definition of the width and depth of the circuit. One option is to use the maximum depth and the maximum width at any one point during execution.

Alternative high-level metrics include, but are not limited to, counting the number of high-level operations that must be performed or the maximum depth of the circuit in such operations.

For instance, the number of group operations or oracle invocations may be counted. Such metrics are useful for comparing algorithms that use essentially the same basic building blocks, or for proving lower bounds on the cost of algorithms. They are fairly simple to understand and use in a meaningful way.

2.9 | Full-stack physical cost estimates

The derivation of full-stack physical cost estimates is beyond the scope of this survey, as it requires assumptions to be made on the future architecture and performance characteristics of large-scale quantum computers. It furthermore requires careful analysis and optimisation of all layers in the quantum stack, including of the algorithmic layer, the logical circuit layer, the error correction layer, and of any required classical pre- and post-processing.

This being said, we do reference recent full-stack cost estimates for Shor's algorithms and derivatives thereof where available. Such estimates may prove useful when seeking to quantify the feasibility of breaking currently widely deployed asymmetric cryptosystems quantumly in, for example, 10, 15, 20, 30, … years' time.

2.10 | Memory cost estimates

The number of computational qubits required is often taken as the amount of quantum memory required to execute a circuit. This being said, there are cost estimates such as Ref. [17] where quantum information is swapped out to non-computational quantum memory.

In general, a *quantum memory access* can be defined as the following unitary, which takes an *index register* i, an output register x, M memory registers $y_0, \ldots, y_{M-1}$, and writes in the output register the contents of register i:

$$|i\rangle|x\rangle|y_0,\ldots,y_{M-1}\rangle \mapsto |i\rangle|x \oplus y_i\rangle|y_0,\ldots,y_{M-1}\rangle \ .$$

Such an operation can be implemented using $\tilde{O}(M)$ gates of a universal gate set, width $O(M)$ and depth $O(\log M)$. It should be noted that using only bounded-arity gates, one cannot do asymptotically better. The *qRAM model* assumes that this operation can be implemented at low cost, typically $O(\log M)$ or $O(1)$. Following the naming conventions in Ref. [18]:

- quantumly addressable classical memory (QRACM) concerns the case of a classical memory $(y_0, \ldots, y_{M-1})$
- quantumly addressable quantum memory (QRAQM) is the generic case where the memory can be in a superposition state as well

It should be noted that if the index register is in a classical state, then it can be measured and the operation can be performed efficiently in the standard quantum circuit model (because we can apply quantum gates between any qubits at the same cost). Likewise, if all the registers are classical, then this model becomes classical random access memory (CRACM), which is a standard assumption in classical cryptanalysis.

Note that the QRACM and QRAQM assumptions are debatable. It is entirely possible that quantum memory in a physical realisation of a quantum computer could come at a significant cost. When costing quantum algorithms, it is therefore important to state which memory model is used and what the memory cost is in this model.

3 | SEARCH ALGORITHMS

3.1 | Grover's algorithm

The most versatile tool from quantum computing in the scope of quantum cryptanalysis is probably Grover's search algorithm. In a nutshell, it allows to search for preimages of an unstructured Boolean function $f: [1, N] \to \{0, 1\}$, assuming that there exists an efficient quantum algorithm that implements f. When a single preimage of 1 exists (a *marked* element), Grover's algorithm finds it at the cost of $O(\sqrt{N})$ applications of the inspection function f, whereas a classical circuit would need to inspect at least $O(N)$ elements to succeed with constant probability. Therefore, many techniques use Grover either as the main routine or as a sub-routine. In particular, Grover's search in theory enhances all brute-force search procedures.

More generally, let $S \subseteq [1, N]$ be the set of preimages of 1, and assume $|S| = M$. To start the search, we create the state $|\psi\rangle := \frac{1}{\sqrt{N}}\sum_{x\in[1,N]}|x\rangle$. In the case where $N = 2^n$, this is done by applying $H^{\otimes n}$ to the input state $|0\rangle^{\otimes n}$. At this point, the measurement of the state of the system would yield a marked element with probability M/N. Grover's algorithm will repeatedly act on the state to increase those odds.

Note that Grover's search is often described as searching a *database*, which fits in the definition above if we think of f as performing a memory access at a given index. However, in order to implement this efficiently, one may need the *quantum random-access* model presented in Section 2.10.

The first building block of Grover's algorithm is an *oracle* that implements the inspection function f (See Figure 1). Intuitively, this state acts on a superposition of basis states by multiplying the phase of all the component $|x\rangle$ corresponding to the index x of a marked element by -1. The iteration of the Grover algorithm uses $\mathcal{O}_f$ alongside a similar operator $\mathcal{O}_\phi$ which satisfies $\mathcal{O}_\phi|0\rangle = |0\rangle$ and $\mathcal{O}_\phi|x\rangle = -|x\rangle$ for $x \neq 0$. Note that $\mathcal{O}_f$ can be efficiently implemented from a circuit that implement f, while $\mathcal{O}_\phi$ is efficiently implementable with Clifford + T + CNOT gates. An iteration of Grover's algorithm is shown in Figure 2.

The initial state $|\psi\rangle$ is in the span of the vectors $|\alpha\rangle, |\beta\rangle$ defined as

$$|\alpha\rangle = \frac{1}{\sqrt{N-M}}\sum_{x\notin S}|x\rangle \quad \text{and} \quad |\beta\rangle = \frac{1}{\sqrt{M}}\sum_{x\in S}|x\rangle$$

The vectors $|\alpha\rangle, |\beta\rangle$ are an orthonormal basis, and we can see that $|\psi\rangle = \sqrt{\frac{N-M}{N}}|\alpha\rangle + \sqrt{\frac{M}{N}}|\beta\rangle$. Let us define θ by $\cos(\theta/2) = \sqrt{\frac{N-M}{N}}$ and $\sin(\theta/2) = \sqrt{\frac{M}{N}}$. This yields

$$|\alpha\rangle = \cos(\theta/2)|\alpha\rangle + \sin(\theta/2)|\beta\rangle.$$

Moreover, we can view the action of the Grover iterate as a rotation in the span of $|\alpha\rangle, |\beta\rangle$.

Proposition 5 *The Grover iterate G acts on* $\mathrm{Span}\{|\alpha\rangle, |\beta\rangle\}$ *as*

$$G = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}.$$

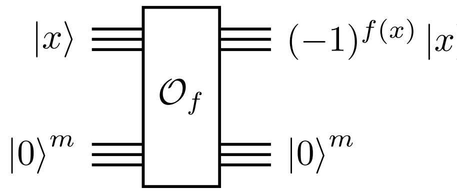

FIGURE 1 Oracle function

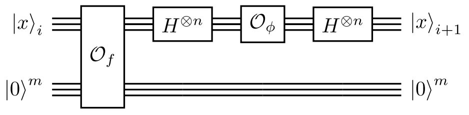

FIGURE 2 Grover iterate

Proof: We first show that the Grover iterate operator is equal to $G = (2|\psi\rangle\langle\psi| - I)\mathcal{O}_f$. Then we apply this operator on the two basis vectors $|\alpha\rangle$ and $|\beta\rangle$, and we observe that the matrix of G has the claimed shape. □

Starting with $|\psi\rangle$, the state we reach after k iterations of G is

$$G^k|\psi\rangle = \cos\left(\frac{2k+1}{2}\theta\right)|\alpha\rangle + \sin\left(\frac{2k+1}{2}\theta\right)|\beta\rangle.$$

We need to select k so that this state is as close as possible to $|\beta\rangle$ (whose measurement would yield a marked element with probability 1). This means aiming for $\frac{2k+1}{2}\theta \approx \frac{\pi}{2}$.

Proposition 6 *Assuming that $M \le N/2$, a measurement of the state after $k \le \left\lceil \frac{\pi}{4}\sqrt{\frac{N}{M}} \right\rceil$ Grover iterations yields $x \in S$ with probability at least $\frac{1}{2}$.*

This means that we measure a marked element with constant probability after $O(\sqrt{N/M})$ oracle calls. The total cost is $O\left(\sqrt{N/M}\,\mathrm{Cost}(\mathcal{O}_f)\right)$ in terms of time, gate count and DW-cost.

3.2 | Amplitude amplification

Grover's search algorithm can be generalised to include a subroutine to replace the creation of the uniform superposition over all elements in $[1, N]$ with another algorithm with better odds of leading to the measurement of a marked element. This could typically be used to implement nested searches. For example, the inner search might identify marked elements in a set S' such that $S \subset S'$. In the *Oracles with costs* framework of Kimmel et al. [19], this strategy is used to take advantage of the situation where the oracle $\mathcal{O}_f$ used to mark elements in S' is significantly less expensive to implement than the oracle $\mathcal{O}_g$ that marks elements in S.

Formally, in amplitude amplification, we assume the knowledge of an algorithm A that produces a superposition over all possible outcomes with certain weights

$$A|0\rangle^{\otimes n} = \sum_{x<N} \alpha_x |x\rangle|\mathrm{junk}(x)\rangle = |\psi\rangle.$$

Here junk(x) is a state resulting from the computation of A (i.e. a collection of intermediate values that are kept due to reversibility). In the case of Grover's algorithm, $A = H^{\otimes n}$, but in general, the measurement of $|\psi\rangle$ yields $x \in S$ with probability

$$1 > p = \sum_{x\in S} |\alpha_x|^2 > 0$$

that is not necessarily M/N. The amplitude amplification circuit is almost identical to that of Grover's search except that

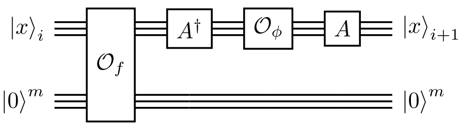

FIGURE 3 Amplitude amplification iterate

calls to $H^{\otimes n}$ are replaced by A. It consists in the repetition of the iterate shown in Figure 3. Similar to the analysis of Grover's algorithm, we define the states

$$|\alpha\rangle = \frac{1}{\sqrt{1-p}} \sum_{x\notin S} \alpha_x |x\rangle\ |\mathrm{junk}(x)\rangle$$
$$|\beta\rangle = \frac{1}{\sqrt{p}} \sum_{x\in S} \alpha_x |x\rangle\ |\mathrm{junk}(x)\rangle$$

Proposition 7 *The amplitude amplification iterate Q acts on $\mathrm{Span}\{|\alpha\rangle, |\beta\rangle\}$ as $G = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$, where θ is defined by $\cos(\theta/2) = \sqrt{1-p}$ and $\sin(\theta/2) = \sqrt{p}$. This means that Q acts as a rotation of angle θ.*

We start the procedure with the state $|\psi\rangle = \cos(\theta/2)|\alpha\rangle + \sin(\theta/2)|\beta\rangle$. The state we reach after k iterations is

$$Q^k|\psi\rangle = \cos\left(\frac{2k+1}{2}\theta\right)|\alpha\rangle + \sin\left(\frac{2k+1}{2}\theta\right)|\beta\rangle.$$

Proposition 8 *Assuming that $p \le 1/2$, a measurement of the state after $k \le \lceil \frac{\pi}{4\sqrt{p}} \rceil$ steps yields $x \in S$ with probability at least $\frac{1}{2}$.*

Therefore, the cost of amplitude amplification is $O\left(\frac{\mathrm{Cost}(A)}{\sqrt{p}}\right)$ where the cost can be viewed as time, gates, or DW.

As with classical algorithms, time (i.e. circuit depth) can be reduced by performing computations in parallel. Indeed, a classical exhaustive search of the key is expected to be performed massively in parallel by segmenting the search space. Unfortunately, the performance of Grover search deteriorates badly in a parallel setting. If we use R processors to search in parallel for the secret key, we can make each of them search in a subspace of size $2^\kappa/R$ and aggregate their results: this reduces the depth by a factor $\sqrt{R}$ but *increases the total workload* by a factor $\sqrt{R}$. Indeed, R independent searches that perform $O\left(\frac{2^{\kappa/2}}{\sqrt{R}}\right)$ oracle calls each. This yields a total of $O\left(\sqrt{R}\cdot 2^{\kappa/2}\right)$ oracle calls. This naive parallelisation is actually optimal.

3.3 | Random walks

Grover's search algorithm is generalised by the notion of random walk on a graph. We assume that a graph G is given by a set of vertices V and edges E, and we assume that we are looking for a marked element in $M = f^{-1}(\{1\})$ for some $f: V \to \{0, 1\}$. The general strategy of a random walk is to start from a vertex $x \in V$, check if $f(x) = 1$, and if not, then walk in the graph by sampling neighbours uniformly at random long enough to ensure the new vertex x' attained is distributed almost uniformly at random in V and then test if $f(x') = 1$. This is repeated until a marked element is found. In addition to running $\mathcal{O}_f$, there are two main steps in a quantum walk that contribute to the overall cost:

- Setup: sampling the first vector and initialising the data structure.
- Update: sampling a neighbour and updating the data structure (we need to update the current node and its neighbours).

Each of the aforementioned steps have a cost that depends on the data structure that is chosen to navigate the graph (note that depending on the model of computation chosen, memory-intensive data structures can penalise the cost). Moreover, the cost is impacted by the shape of the transition matrix M. In the case of a d-regular graph (which is relevant to many computational problems), $M = \frac{1}{d}A$ where A is the adjacency matrix of the graph. The number of update steps required to reach a node almost uniformly distributed is $\tilde{O}\left(\frac{1}{\delta}\right)$ where δ is the *spectral gap* of M, that is, $\delta := 1 - \max_{i>1}|\lambda_i|$ where $(\lambda_i)_{i>1}$ are the eigenvalues of M not equal to 1. A similar approach can be used quantumly [20]. The cost becomes

$$\text{Cost(Setup)} + \frac{1}{\sqrt{\varepsilon}}\left(\text{Cost}\left(\mathcal{O}_f\right) + \frac{1}{\sqrt{\delta}}\,\text{Cost(Update)}\right)$$

Many computational problems relevant to cryptanalysis reduce to a walk in the Johnson graph of a set. In general, a Johnson graph $J(n, r)$ is an undirected graph whose vertices and the subsets of size r of a given set U of size n. There is an edge between vertices $S \subseteq U$ and $S' \subseteq U$ if and only if $|S \cap S'| = r - 1$ (i.e. they differ by only 1 element). The Johnson graph $J(n, r)$ has $|V| = \binom{n}{r}$ vertices, is $r(n - r)$-regular and its spectral gap is

$$\delta = \frac{n}{r(n-r)}.$$

The product $J^m(n, r)$ of m copies of $J(n, r)$ is the graph whose vertices are of the form $(v_1, \ldots, v_m)$ where each v_i is a vertex of $J(n, r)$, and there is an edge between $(v_1, \ldots, v_m)$ and $\left(v'_1, \ldots, v'_m\right)$ if and only if there is an edge between v_i and v'_i for some i, and $v_j = v'_j$ for all $j \neq i$. The product $J^m(n, r)$ has $\binom{n}{r}^m$ elements, is $mr(n - r)$-regular, and its spectral gap satisfies

$$\delta(J^m(n,r)) \geq \frac{1}{m}\delta(J(n,r)).$$

3.4 | Quantum backtracking

Relevant to cryptanalysis of algorithms for lattices is an extension of Grover's search where instead of searching for a marked element in a list, the task is to find a marked leaf in a (large) tree. Let us describe the setup. We assume the following query access to a tree T: for a given node v, we have an oracle that returns the number of its children; also we have an oracle that for a node v and index i, returns the ith child of v. The maximal degree of T is the largest number of children among all nodes.

Backtracking algorithms is a classical method to solve problems where we can partially enumerate solutions and check whether the current sub-solution can be extended to the actual solution. Hence, a backtracking algorithm constructs a tree in depth-first manner where leaves represent solutions. An example for such problem is SAT. Backtracking requires an oracle $\mathcal{P}_T$ that operates on nodes s.t. given a leaf v it tells whether v is a solution ('marked') and given any other node the oracle returns 'intermediate'. Classically, finding a 'marked' leaf can be done on time $\mathcal{O}$(nodes). Montanaro in Ref. [21] gives a Grover-like speed-up for this task:

Theorem 9 ([21]). *There is a quantum algorithm that given a query access to a tree T as described above with maximal degree $\mathcal{O}(1)$, an oracle $\mathcal{P}_T$, n–an upper bound on the depth of T and $\varepsilon > 0$, outputs either a marked leaf x or $\perp$ if no marked x exists, by making $\mathcal{O}\left(\sqrt{\#\text{nodes}} \cdot \mathsf{poly}(n)\log(1/\varepsilon)\right)$ queries to T and to $\mathcal{P}_T$. The algorithm uses $\mathsf{poly}(n)$ qubits and is correct with probability larger than $1 - \varepsilon$.*

4 | HIDDEN SUBGROUP PROBLEMS

Hidden Subgroup Problems (HSP) consist in the search for a secret subgroup H inside of a control group G, given access to an oracle function $f: G \to X$ for some set X of quantum states that satisfies $f(x) = f(y)$ if and only if $x \in y + H$. Many interesting computational problems reduce to an HSP instance, including factoring and the discrete logarithm problem, as shown in Section 5.

4.1 | The abelian quantum Fourier transform

Solving the HSP in an abelian group in quantum polynomial time is usually done by using the Quantum Fourier Transform (QFT). The QFT depends on the control group G we are working with. To simplify this introduction, we start with the QFT used in Shor's original work [1] which applies to $G = \mathbb{Z}_{2^n}$.

Definition 10 (QFT over $G = \mathbb{Z}_{2^n}$). *The QFT circuit over $G = \mathbb{Z}_{2^n}$ performs the following operation on the basis states:*

$$|x\rangle \longmapsto \frac{1}{\sqrt{2^n}}\sum_{y=0}^{2^n-1} e^{\frac{2i\pi xy}{2^n}}|y\rangle.$$

Such a circuit solves the *phase estimation problem* which, given an input state $|\psi\rangle = \frac{1}{\sqrt{2^n}}\sum_{y=0}^{2^n-1} e^{2i\pi\omega y}|y\rangle$, asks to find an approximation of $\omega \in \mathbb{R}$. One can show that the measurement of the state $\mathrm{QFT}_{2^n}|\psi\rangle$ yields y such that $\left|\frac{y}{2^n} - \omega\right| \leq \frac{1}{2^n}$ with constant probability.

A circuit realising the exact QFT over $\mathbb{Z}_{2^n}$ cannot be implemented with Clifford + T + CNOT gates. Instead, it should be approximated. The essential component for this QFT that does not belong in our gate set are the controlled rotations. Given $k \leq n$, these gates realise the operation $|0\rangle|a\rangle \mapsto |0\rangle|a\rangle$ and $|1\rangle|a\rangle \mapsto |1\rangle R_k|a\rangle$ where

$$R_k := \begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{2\pi i}{2^k}} \end{pmatrix}.$$

The QFT_{2^n} circuit is realised with a combination of Hadamard and controlled rotations shown in Figure 4. We use the notation $0.\, x_1 \ldots x_k$ to denote $\frac{x}{2^k}$ where $x_1 \ldots x_k$ is the binary expansion of x.

The QFT can be generalised to any finite abelian group G. This uses the notion of the *character group* $\widehat{G}$ of G of group morphisms $f : G \to \mathbb{C}^*$. Elements in $y \in G$ are in correspondence with characters $\chi_y \in \widehat{G}$. In a cyclic group $\mathbb{Z}_d$, $\chi_1 : x \mapsto \mathrm{e}^{2i\pi x/d}$, while $\chi_i : x \mapsto \chi_1(x)^i$ (i.e. $\chi : y \in G \mapsto \chi_y \in \widehat{G}$ is a group morphism). If $G = \mathbb{Z}_{d_1} \oplus \mathbb{Z}_{d_2}$, and $g = (g_1, g_2) \in G$, then $\chi_g = \chi_{g_1}\chi_{g_2}$, which allows us to define χ_g by induction for any finite abelian group. For example, when $G = \mathbb{Z}_{2^n}$, we have $\chi_y : x \mapsto e^{\frac{2i\pi xy}{2^n}}$.

Definition 11 (QFT over a finite abelian group G). *The QFT circuit over an arbitrary finite abelian G performs the following operation on the basis states:*

$$|x\rangle \longmapsto \frac{1}{\sqrt{|G|}}\sum_{y \in G}\chi_y(x)|y\rangle.$$

Similar to the case $G = \mathbb{Z}_{2^n}$, the QFT over an arbitrary finite abelian group can be efficiently approximated by a polynomial size circuit of Clifford + T + CNOT gates.

4.2 | Solving the HSP in a finite abelian group

In this section, we assume that G is finite and abelian and that we have an implementation of a function f that satisfies $f(x) = f(y)$ if and only if $x - y \in H \subseteq G$ where H is a secret subgroup. The goal of the HSP algorithm we present is to recover H by measuring elements of $\widehat{H}$. When a generating set for $\widehat{H}$ is known, we use classical methods to recover H.

We initiate this process by creating a uniform superposition over the elements of G: $|\psi_0\rangle := \frac{1}{\sqrt{|G|}}\sum_{x\in G}|x\rangle$. When $G = \mathbb{Z}_{2^n}$, this is done by using $H^{\otimes n}$ while efficient methods exist for other groups as well. We then use the function implementing f to obtain the state

$$|\psi_1\rangle := \frac{1}{\sqrt{|G|}}\sum_{x\in G}|x, f(x)\rangle.$$

Now, we measure the second register, which yields $y = f(x)$ in the range of f and collapses the state into a superposition of all preimages of y. By assumption on the periodicity of f, we know that $f(x') = y = f(x)$ if and only if $x' - x \in H$, that is, $x' \in x + H$. This means that the system is left in the state

$$|\psi_2\rangle = |x + H\rangle := \frac{1}{\sqrt{|H|}}\sum_{h\in H}|x + h\rangle.$$

The next stage consists in applying the QFT to the state $|\psi_2\rangle$. This yields the following state:

$$|\psi_3\rangle := \mathrm{QFT}|\psi_2\rangle = \frac{1}{\sqrt{|H|\cdot|G|}}\sum_{y\in G}\sum_{h\in H}\chi_y(x+h)|y\rangle$$

$$= \sqrt{\frac{|H|}{|G|}}\sum_{y\in G}\chi_y(x)\cdot\underbrace{\left(\frac{1}{|H|}\sum_{h\in H}\chi_h(h)\right)}_{\chi_y(H)}|y\rangle$$

Proposition 12 *The measurement of the system in the state $|\psi_3\rangle$ yields y such that $H \subseteq \ker(\chi_y)$.*

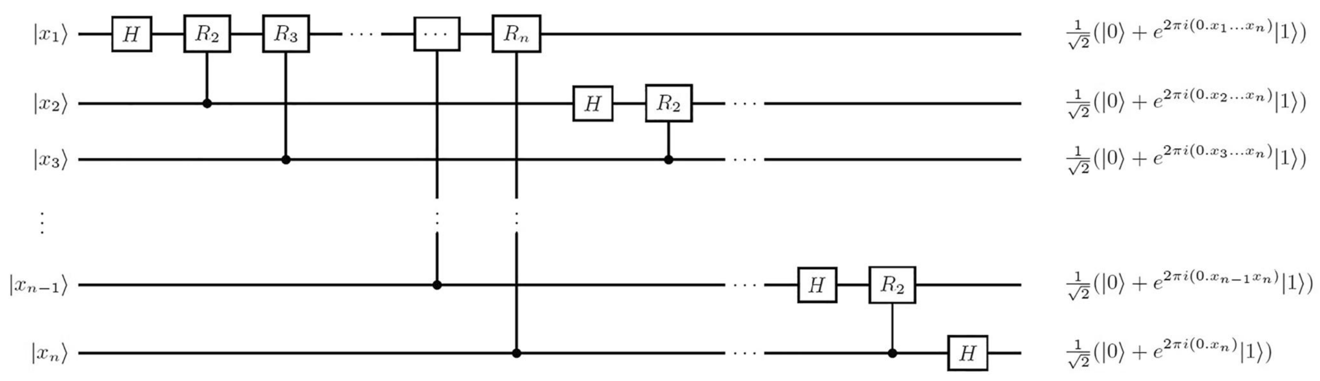
$|x_1\rangle$
$|x_2\rangle$
$|x_3\rangle$
$|x_{n-1}\rangle$
$|x_n\rangle$
H
R_2
R_3
R_n
$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i(0.x_1\ldots x_n)}|1\rangle)$
$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i(0.x_2\ldots x_n)}|1\rangle)$
$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i(0.x_3\ldots x_n)}|1\rangle)$
$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i(0.x_{n-1}x_n)}|1\rangle)$
$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i(0.x_n)}|1\rangle)$

FIGURE 4 Quantum circuit for QFT_{2^n}

Proof: If $\chi_y(h) = 1$ for all $h \in H$, then clearly $\chi_y(H) = 1$. On the other hand, if there is an $h \in H$ such that $\chi_y(h) \neq 1$, then we have $\chi_y(H) = \frac{1}{|H|}\sum_{h' \in H}\chi_y(h + h') = \chi_y(h)\chi_y(H)$, which means that $\chi_y(H) = 0$. Hence, we have

$$|\psi_3\rangle = \sqrt{\frac{|H|}{|G|}}\sum_{y:\chi_y(H)=1}\chi_y(x)|y\rangle.$$

Since $|\chi_y(x)|$, we see that the only possible measurements are y such that $\chi_y(H) = 1$, that is, $H \subseteq \ker(\chi_y)$. □

The strategy to compute H consists in running the above procedure several times and to compute $\cap_{y \text{ measured}} \ker(\chi_y)$. With high probability, this yields H in $O(\log(|G|))$ steps.

4.3 | HSP in D_N

In general, the HSP in non-abelian groups can be a hard problem. Still, subexponential algorithms for the dihedral group (D_N) have been proposed [18, 22, 23]. These algorithms have a time complexity in $2^{(\log(N))}$, and Ref. [23] has polynomial memory. In this section, we introduce Kuperberg's method [22] to solve the HSP in D_N, which is a non-abelian group. Given a positive integer N, we can define the dihedral group D_N by $D_N = \mathbb{Z}/N\mathbb{Z} \rtimes_\phi \mathbb{Z}/2\mathbb{Z}$, where $\phi(0)$ is the identity and $\phi(1)$ is the inversion. Concretely, this means that the elements of D_N are those of $(\mathbb{Z}/N\mathbb{Z}) \times (\mathbb{Z}/2\mathbb{Z})$ and that the group law is given by

$$(a, b) \cdot (a', b') = \left(a + (-1)^b a', b + b'\right).$$

Assume H is a subgroup of D_N. Let us show how the HSP instance defined by H can be reduced to a simpler instance of the HSP where the secret subgroup has the shape $H' = \{(0, 0)\ (k, 1)\}$. We can define $H_1 := H \cap \mathbb{Z}/N\mathbb{Z} \times \{0\} = \{(a, b) \in H \text{ with } b = 0\}$. This subgroup H_1 is isomorphic to a subgroup of $\mathbb{Z}/N\mathbb{Z}$, which is of the form $M(\mathbb{Z}/N\mathbb{Z})$ for $M \mid N$. The subgroup H_1 is normal in D_N, and we have the following identity.

Proposition 13 *Let H_1 be a subgroup of D_N that is isomorphic to $M(\mathbb{Z}/N\mathbb{Z})$ for $M \mid N$, then the map*

$$\begin{aligned} \phi : D_N &\to D_M \\ (a, b) &\mapsto (a \bmod M, b). \end{aligned}$$

is surjective with kernel H_1, which means that $DN/H_1 \simeq DM$.

Proof: The map ϕ is clearly surjective since $M \mid N$. In addition, if $\phi(a, b) = (0, 0)$, then $b = 0$ and $M \mid a$, which means that $a \in M(\mathbb{Z}/N\mathbb{Z})$ and thus $(a, b) \in H_1$. Hence, $\ker(\phi) \subseteq H_1$. On the other hand, it is immediate that $H_1 \subseteq \ker(\phi)$, so $\ker(\phi) = H_1$, and by the fundamental theorem of algebra $D_N/\ker(\phi) = D_N/H_1 \simeq D_M$. □

So, given a function $f: D_N \to X$ that hides H, our strategy is to first find H_1 that is hidden by the function $f' : D_N \cap \mathbb{Z}/N\mathbb{Z} \times \{0\} \to X$, which is the restriction of f. This is done by re-casting this as an instance of the HSP in the abelian group $G = \mathbb{Z}_N$. Once this is done, we learn the integer M and we turn our attention to the resolution of the HSP instance defined by the hidden subgroup H_2 of D_M.

Proposition 14 *With the notations above, the hidden subgroup of D_M is either $H_2 = \{(0, 0)\}$, or of the form*

$$H_2 = \{(0,0),(k,1)\} \text{ for some } 0 \leq k < M.$$

Therefore, we look for a secret $s \in \mathbb{Z}_N$, which defines a hidden subgroup $H = \{(0, 0)\ (s, 1)\} \subseteq D_N$. We describe Kuperberg's sieve algorithm from a high level perspective by restricting ourselves to the case of $N = 2^n$. There are two main ingredients to this method: first, the creation of so-called coset states and then the sieve itself that recombines the *coset states* together to learn information about the secret s. To create a coset state, we compute a uniform superposition of all elements of D_N:

$$|\phi_0\rangle = \frac{1}{\sqrt{2N}}\sum_{(a,b)\in D_N}|a, b\rangle = \frac{1}{\sqrt{2N}}\sum_{a=0}^{N-1}\sum_{b=0}^{1}|a\rangle|b\rangle.$$

Then, we use a circuit $U_f : |a, b\rangle|c\rangle \mapsto |a, b\rangle|c + f(a, b)\rangle$ for the function f that hides H on the input state $|\phi_0\rangle|0\rangle$, thus creating the state

$$\frac{1}{\sqrt{2N}}\sum_{a=0}^{N-1}\sum_{b=0}^{1}|a\rangle|b\rangle|f(a, b)\rangle.$$

We measure the second register and learn $f(a, b)$ for some a, b. This leaves the state in the superposition of all elements (x, y) such that $f(x, y) = f(a, b)$. As f hides the subgroup $H = \{(0, 0)\ (s, 1)\}$, the sum should have two elements, one with $b = 0$ and one with $b = 1$. Let $t = a$ for the pair (a, b) with $b = 0$, then the other pair is $(t, 0) \cdot (s, 1) = (t + s, 1)$ so that the resulting state is

$$|\phi_1\rangle := \frac{1}{\sqrt{2}}(|t\rangle|0\rangle + |t + s\rangle|1\rangle)$$

We apply $\text{QFT}_N \otimes I_2$ to $|\phi_1\rangle$ (i.e. we apply the QFT to the first register and leave the second one alone), thus producing the state

$$\frac{1}{\sqrt{2N}}\sum_{k=0}^{N-1}\left(\omega^{tk}|k, 0\rangle + \omega^{(t+s)k}|k, 1\rangle\right).$$

We measure the first register, thus obtaining a value k drawn uniformly at random in $[0, N-1]$ and leaving the second (single-qubit) register in the state

$$\frac{1}{\sqrt{2}}\left(\omega^{tk}|0\rangle + \omega^{(t+s)k}|1\rangle\right) \propto |\psi_k\rangle := \frac{1}{\sqrt{2}}\left(|0\rangle + \omega^{sk}|1\rangle\right),$$

which is one of the N possible *coset states*.

We now assume that we have a circuit that can produce coset states $|\psi_k\rangle$ for k distributed uniformly at random in $[0, N-1]$. Note that the classical information of the label k is known as well. Our goal is to create the coset state

$$|\psi_{N/2}\rangle = \frac{1}{\sqrt{2}}(|0\rangle + (-1)^s|1\rangle)$$

If this state is known then $(-1)^s = (-1)^{\mathrm{b}}$ where $b = 0$ if s is even and $b = 1$ if s is odd. As $H|\psi_{N/2}\rangle = |b\rangle$, we readily obtain the parity of s from $|\psi_{N/2}\rangle$, that is, its binary digit of lowest order. Unfortunately, the odds of drawing $|\psi_{N/2}\rangle$ are small. So instead, we collect many coset states $|\psi_k\rangle$ for a random k, and we recombine them with the hope of creating $|\psi_{N/2}\rangle$. Given two coset states $|\psi_k\rangle, |\psi_l\rangle$, we perform the circuit described in Figure 5. The input state is the product state $|\psi_k, \psi_l\rangle$, and after the CNOT gate, the system is left in the state

$$\frac{1}{\sqrt{2}}\left(|\psi_{k+l}\rangle|0\rangle + \omega^{sl}|\psi_{k-l}\rangle|1\rangle\right).$$

Hence, with probability $1/2$, the measurement of the second qubit is 1 and the system is left in a state that is proportional (up to a global phase) to $|\psi_{k-l}\rangle$.

We look for indices k, l whose first $m := \lceil\sqrt{n-1}\rceil$ binary digits are the same. Then, the first m binary digits of $k-l$ will be 0. We need to create a large enough initial list L_0 of coset states so that enough of the corresponding indices have their first m binary digits in common. The resulting coset states (which have indices with m initial zeros in their binary decomposition) will be in a list L_1. This process is then repeated with the next m bits in the binary decomposition of the indices. The last coset state should be $|\psi_{2^n-1}\rangle$. We begin with $|L_0| = 2^{n_0}$ random coset states, and at each stage $|L_{i+1}| \geq \frac{|L_i|}{8}$, which means that $|L_m| \geq 2^{n_0-3m}$. Starting with $n_0 \geq 4m$ guarantees enough coset states after the mth stage of the sieve to obtain $|\psi_{2^n-1}\rangle$.

Once we know $b \in \{0, 1\}$ such that $s = 2s' + b$, we need to repeat the sieve in $D_{N/2}$ to learn the bit of s' of lowest order. We can immediately see that the function

$$\begin{aligned} f' : D_{N/2} &\to X \\ (x, y) &\mapsto f(2x + b, y). \end{aligned}$$

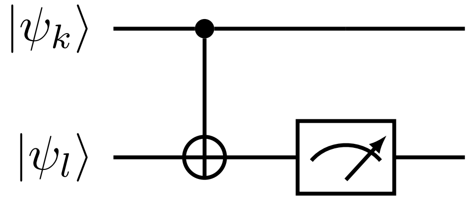

FIGURE 5 Iterate of Kuperberg's sieve

hides the subgroup $H' := \{(0, 0), (1, s')\} \subseteq D_{N/2}$. So, the above procedure can be repeated to learn the first bit b' of s'. Eventually, after $\log(s)/\log(2) \leq \log(N)/\log(2)$ steps, the process allows us to learn all the bits of the binary decomposition of secret s. This can be generalised to arbitrary N, and even to the HSP in $G \rtimes_\phi \mathbb{Z}/2\mathbb{Z}$ for an arbitrary finite abelian group G.

5 | FACTORING AND DISCRETE LOGARITHM PROBLEMS

Shor's algorithm [1, 24] for the IFP splits any integer N—that is odd and not a perfect prime power—into two non-trivial factors. To completely factor N, the algorithm may be applied recursively.

There exist efficient classical algorithms for testing primality [25, 26] and for reducing perfect powers. The restrictions imposed by Shor, therefore, do not imply a loss of generality.

5.1 | Shor's original factoring algorithm

Shor's algorithm works by first classically reducing the IFP to an order-finding problem (OFP) in a cyclic subgroup of $\mathbb{Z}_N^*$. This OFP is then solved quantumly using an order-finding algorithm.

It should be noted that there exist other potential applications for efficient order-finding algorithms, besides factoring integers: Shor's order-finding algorithm computes orders in any cyclic group for which the group arithmetic may be efficiently implemented.

5.2 | Reducing the IFP to an OFP

First, an element g is selected uniformly at random from $\mathbb{Z}_N^*$.

In practice, this may be accomplished by selecting an integer g uniformly at random from the interval $(1, N)$ and testing if g is coprime to N. If it is not, we will have found a non-trivial factor of N. Otherwise, we will have successfully selected g.

The order r of g is then computed quantumly. The order is the least positive integer such that $g^r \equiv 1 \pmod N$. If r is even,

$$g^r - 1 = \left(g^{r/2} + 1\right)\left(g^{r/2} - 1\right) \equiv 0 \pmod{\mathrm{N}},$$

so $N \mid (g^r - 1)$, whilst $N \nmid (g^{r/2} - 1)$ by the definition of r.

If, furthermore, $N \nmid g^{r/2} + 1$, it must be that $\gcd(N, g^{r/2} \pm 1)$ are non-trivial factors of N. Specifically, it must be that $N = ab$, for some $a, b \neq 1$ where $a \mid (g^{r/2} + 1)$ and $b \mid (g^{r/2} - 1)$. Shor cites Miller [25] for this randomised reduction. See also Long [27].

Shor proves [Ref. [1], p. 1498] that the probability of r being even, and of the above condition being met, is at least $1/2$. If either condition is not met, the whole algorithm may simply be re-run. After n runs, the failure probability is then at most 2^{-n}.

5.2.1 | Improved reductions

Ekerå [28] has recently shown how the complete factorisation of N may be computed efficiently classically, with very high probability, given the order r of g, and in a follow-up work [29], he gave probability estimates that account for the possibility of failing to recover r.

In the worst case, for N an m-bit integer with n distinct prime factors, the failure probability when using this approach is at most

$$2^{-k}\binom{n}{2} + \frac{1}{2c^2 \log_2^2 cm}$$

by [Ref. [28], Th. 1], where $c, k \geq 1$ are parameters that may be freely selected. Increasing c, k increases the success probability at the expense of having to perform more classical post-processing work.

In practice, c, k may be selected so as to guarantee a very low failure probability without compromising efficiency. Furthermore, for nearly all integers N, the failure probability is much smaller than the bound indicates: In general, it is expected to be insignificant.

Note that the failure probability tends to zero asymptotically as $N \to \infty$, if $c = 1$ and we, for example, let k depend on m such as $k \leq 3 \log_2 m$.

5.3 | Quantum order finding

Shor's order-finding algorithm—if slightly generalised, for reasons that will soon become clear, and with notation from Ref. [30]—uses two registers: a *control* register of $\nu : = m + \ell$ qubits—for m the bit length of the order and $\ell \sim m$—and a *work* register of some t qubits—for t sufficiently large to allow group elements to be represented and group operations to be performed.

The work register may be initialised to any value. For simplicity, we initialise it to $|1\rangle$, to have it represent the identity in $\mathbb{Z}_N^*$. The control register is initialised to $|0\rangle$, after which H gates are independently applied to the ν qubits in the register, yielding

$$\frac{1}{\sqrt{2^\nu}} \sum_{a=0}^{2^\nu - 1} |a, 1\rangle.$$

As may be seen above, the effect of applying the H gates is to induce a uniform superposition over all possible states in the control register. Next, we compute g^a to the work register, yielding

$$\frac{1}{\sqrt{2^\nu}} \sum_{a=0}^{2^\nu - 1} |a, g^a\rangle.$$

Above, and in what follows, we perceive g as an element of $\mathbb{Z}_N^*$ and forego writing out mod N. In practice, the exponentiation would typically be performed by classically pre-computing

$$\left\{g, g^2, g^{2^2}, \ldots, g^{2^{\nu-1}}\right\},$$

and multiplying g^{2^i} into the work register if the ith control qubit $a_i = 1$ in what amounts to the square-and-multiply algorithm:

$$a = \sum_{i=0}^{\nu-1} 2^i a_i \Rightarrow g^a = \prod_{i=0}^{\nu-1} g^{2^i a_i} = g^{a_0 + 2a_1 + 2^2 a_2 + \ldots}.$$

Note that multiplication by powers of g mod N is an invertible operation. This is what allows us to multiply pre-computed powers of g directly into the work register. Note, furthermore, that the set of powers of g may be efficiently computed by repeated squaring.

We now have

$$\frac{1}{\sqrt{2^\nu}} \sum_{a=0}^{2^\nu - 1} |a, g^a\rangle = \frac{1}{\sqrt{2^\nu}} \sum_{e=0}^{r-1} \left(\sum_{b=0}^{m_e - 1} |rb + e\rangle \right) |g^e\rangle$$

where $m_e := \lfloor (2^\nu - e - 1)/r \rfloor$, and r is the order of g.

If, at this point, we measure the work register and obtain g^e for some $e \in [0, r)$, this leaves the system in the state

$$\frac{1}{\sqrt{m_e}} \sum_{b=0}^{m_e - 1} |rb + e, g^e\rangle.$$

As may be seen, the control register is now periodic in r, with an unknown offset e. To eliminate e, we apply QFT_{2^ν} to the control register, leaving the system in the state

$$\frac{1}{\sqrt{2^\nu m_e}} \sum_{j=0}^{2^\nu - 1} \left(\sum_{b=0}^{m_e - 1} \exp\left[\frac{2\pi i\, rjb}{2^\nu}\right] \right) |j, g^e\rangle =$$

$$\frac{1}{\sqrt{2^\nu m_e}} \sum_{j=0}^{2^\nu - 1} \left(\sum_{b=0}^{m_e - 1} \mathrm{e}^{i\theta_r b} \right) |j, g^e\rangle$$

where we define

$$\theta_r := \frac{2\pi i\, \alpha_r}{2^\nu} \quad \text{where} \quad \alpha_r := \{rj\}_{2^\nu}$$

and where $\{u\}_n$ is u reduced mod n constrained to $[n/2, n/2)$. The probability of measuring a given j (that yields $\theta \neq 0$) is then

$$\frac{1}{2^\nu m_e}\left|\sum_{z=0}^{m_e-1} \mathrm{e}^{i\theta_r b}\right|^2 = \frac{1}{2^\nu m_e}\frac{\sin^2(m_e\theta_r/2)}{\sin^2(\theta_r/2)}.$$

The measured value j implicitly defines α_r which is unknown to the user. If $\alpha_r \in [-r/2, r/2]$, Shor uses that for some unknown

$$z := \frac{rj - \{rj\}_{2^\nu}}{2^\nu} = \left\lceil \frac{rj}{2^{m+\ell}} \right\rfloor \in [0, r],$$

the convergent z/r will be found in the continued fraction expansion of $j/2^\nu$ (see Hardy and Wright [[31], Th. 184] for a proof) if

$$\left|\frac{j}{2^\nu} - \frac{z}{r}\right| = \frac{|\alpha_r|}{2^{m+\ell} r} < \frac{1}{2r^2},$$

thus enabling us to find the convergent z/r when $\ell \sim m$. We may increase the success probability by increasing ℓ.

It may be shown that we expect to observe j that produces α_r such that $|\alpha_r| \sim 2^m$, efficiently yielding ℓ bits of information on r.

For instance, Shor [1] asymptotically lower-bounds by $4/\pi^2$ the probability of observing j that produces $\alpha_r \in [-r/2, r/2]$.

In [Ref. [32], App. A], the probability of observing α_r such that $|\alpha_r|$ is of length τ bits is shown to be exponentially suppressed in τ as τ grows larger than m. Furthermore, the probability distribution in $\log(|\alpha_r|)$ is plotted in [Ref. [33], Fig. A1], for maximal $r = 2^m - 1$ and for specific choices of m and ℓ. This figure is, however, representative also for other choices of parameters, for as long as r is not divisible by a very large power of two: Picking a smaller $r \in [2^{m-1}, 2^m)$ shifts the distribution slightly. Varying m and ℓ has no visible effect for as long as m and ℓ remain sufficiently large.

In practice, it is more efficient—see [Ref. [1], p. 1501]—to try to solve not only j but also $j \pm 1$, $j \pm 2$, … for the convergent z/r instead of increasing ℓ (This is when assuming classical computation to be cheap compared to quantum computation).

Since we expect to observe j yielding α_r such that $|\alpha_r| \sim 2^m$ as stated above, we expect $\alpha_r = \{rj\}_{2^\nu}$ to increase or decrease by $r \sim 2^m$ if we increase or decrease j. Solving for small offsets in j, therefore, yields the convergent z/r with high probability.

A further concern when performing order finding is that factors may cancel between r and z. Shor [1] points out that the probability of r and z being coprime is $\phi(r)/r = O(1/\log\log r)$. Hence, the expected number of runs is $O(\log\log r)$. Odlyzko furthermore states in private communication to Shor [1] that this may be reduced to $O(1)$ runs by searching for the missing factor $d = \gcd(z, r)$.

In fact, assuming we also solve for offsets in j, it is shown in Ref. [32] that z mod r is selected essentially uniformly at random from $[0, r)$ via j. The probability of a d not being cm-smooth, for c some small constant, may then be upper-bounded and shown to be small—see the main result in Ref. [32]. And if d is cm-smooth, d may be found efficiently classically as is explained in for example, Ref. [32] and [Ref. [33], Section 6.2.4].

5.4 | Practical implementation and control qubit recycling

In practice, the circuit is typically not implemented as described in the previous section: It is not necessary to actually measure the work register. If one foregoes this measurement and interleaves the QFT with the modular multiplications, then the QFT may be performed semi-classically [34]. A single qubit, that is recycled [35, 36], then suffices to implement the control register in practice. In total, only $t + 1$ qubits are then required to implement the circuit.

5.5 | Tradeoffs and lattice-based post-processing

Seifert [37] has proposed to pick $\ell \approx m/s$ for some $s \geq 1$. The quantum circuit then still leaks $\sim\ell$ bits of information on r, so a total of $n \geq s$ runs are then required to solve for r.

Seifert states that his aim is to save control qubits, but in a more modern interpretation in which control qubits are recycled, Seifert reduces the circuit depth from $\sim 2m$ multiplications in Shor's algorithm to $m + m/s$ multiplications. The reduction comes at the expense of having to perform $n \geq s$ runs, so Seifert effectively makes a tradeoff between the circuit depth and the number of runs.

As for post-processing, Seifert generalises continued fractions to higher dimensions using simultaneous Diophantine approximation techniques. Ekerå [33, App. A and Section 6.2] describes a similar lattice-based post-processing technique:

Specifically, let L be the integer lattice spanned by the rows of

$$\begin{bmatrix} j_1 & \dots & j_n & 1 \\ 2^\nu & \dots & 0 & 0 \\ 0 & \ddots & 0 & 0 \\ 0 & 0 & 2^\nu & 0 \end{bmatrix}. \qquad (2)$$

where $\{j_1, \dots, j_n\}$ are the measurement results in the n runs.

Then, the short vector $\mathbf{u} = \left(\{rj_1\}_{2^\nu}, \dots, \{rj_n\}_{2^\nu}, r\right) \in L$ may be found by enumerating short vectors in L, as is shown in Ref. [33]. For $s \geq 1$, we require about s runs to solve; see, for example, [Ref. [33]., Table A2] for estimates of the number of runs n required to solve for $r = 2^m - 1$ as a function of m and s. (Solving for smaller r, that is not partially very smooth, is in general easier.)

5.6 | Specialised algorithms for RSA and DH

If the goal is to factor RSA integers $N = pq$, for p, q, two random primes of equal bit length, the algorithm of Ekerå and Håstad [30, 38] outperforms factoring via Shor's original order-finding algorithm and via Seifert's algorithm when making tradeoffs [30].

Ekerå and Håstad use that for $g \in \mathbb{Z}_N^*$, it holds that

$$x = g^{N+1} = g^{p+q}.$$

If g is selected uniformly at random from $\mathbb{Z}_N^*$, it holds with overwhelming probability that $p + q < r$ for r the order of g. The idea is, hence, to compute the logarithm $d = \log_g x = p + q$ quantumly. Given $N = pq$ and $d = p + q$, it is then trivial to solve for p, q.

To compute d, Ekerå and Håstad introduce a quantum algorithm that is derived from Shor's algorithms [1] and that efficiently computes *short* discrete logarithms in groups of unknown order. A logarithm d is said to be short if it is smaller than the order r of g by some order of magnitude.

For m the bit length of d and $\ell \sim m/s$ for $s \geq 1$ the tradeoff factor, the algorithm first induces the state

$$\frac{1}{\sqrt{2^{\nu+\ell}}} \sum_{a=0}^{2^\nu} \sum_{b=0}^{2^\ell} \left| a, b, g^a x^{-b} \right\rangle$$

and then applies QFT_{2^ν} and QFT_{2^ℓ} to the first and second control registers, respectively, to obtain

$$\frac{1}{2^{\nu+\ell}} \sum_{a,j=0}^{2^\nu} \sum_{b,k=0}^{2^\ell} \exp\left[\frac{2\pi i\,(aj + 2^m bk)}{2^\nu}\right] \left| j, k, g^{a-bd} \right\rangle.$$

The probability of observing (j, k) and g^e for $e = a - bd$ is then

$$\frac{1}{2^{2(\nu+\ell)}} \left| \sum_{b=b_0(e)}^{b_1(e)} \mathrm{e}^{i\theta_d b} \right|^2$$

by [Ref. [30], Section 3], for $b_0(e)$ and $b_1(e)$ as given in [Ref. [30], Section 3.1], and

$$\alpha_d := \{dj + 2^m k\}_{2^\nu} \quad \text{and} \quad \theta_d := \frac{2\pi\alpha_d}{2^\nu},$$

if it is assumed that $r \geq 2^{m+\ell} + (2^\ell - 1)d$, so as to simplify the analysis: It then holds that $e = a - bd$ (Otherwise, it only holds that $e = (a - bd) \bmod r$). The requirement that $r \geq 2^{m+\ell} + (2^\ell - 1)d$ defines what it means for d to be short. It may in some cases be possible to relax the requirement, at the expense of complicating the analysis. For one generalisation, see Section 5.7.

Further analysis [[30], Section 3.2] yields an expression for

$$P(\theta_d) := \frac{1}{2^{2(\nu+\ell)}} \sum_{e=-(2^\ell-1)d}^{2^\nu-1} \left| \sum_{b=b_0(e)}^{b_1(e)} \mathrm{e}^{i\theta_d b} \right|^2$$

that is exact and on closed form. It may be shown [Ref. [30], Lemma 2] that we expect to observe (j, k) yielding $|\alpha_d| \sim 2^m$, leaking $\sim\ell$ bits of information on d. To recover d, we use that the known vector

$$\mathbf{v} = \left(\{-2^m k_1\}_{2^\nu}, \ldots, \{-2^m k_n\}_{2^\nu}, 0\right) \in \mathbb{Z}^{n+1}$$

is close to the unknown vector $\mathbf{u} = (dj_1, \ldots, dj_n, d) \in L$ for L the same lattice as in Section 5.5 spanned by the rows of (2). By the analysis in Ref. [30], we expect to recover $\mathbf{u}$, and hence d, by enumerating all vectors in L within a ball centred on $\mathbf{v}$.

The algorithm of Ekerå and Håstad is useful not only for breaking RSA [2] but also for breaking Diffie–Hellman (DH) [4] in safe-prime groups with short exponents, as standardised in Refs. [39–41].

For RSA, Ekerå–Håstad performs between 3/4 and 1/4 as many modular multiplications as Shor's order-finding algorithm, as a function of s. For DH, the constant factor advantage is greater when comparing to Shor's algorithm for the DLP. For details, see [Ref. [30], Tables 2–4].

5.7 | Generalisations and extensions

If, in the algorithm as presented in the previous section, we take m equal to the bit length of r and ignore the requirement that d must be short, we obtain an algorithm [33] that computes both the order r and the logarithm d. It has potential cryptanalytic applications in the computation of discrete logarithms in random Schnorr groups of unknown order, since Shor's DLP algorithm requires r to be known.

The analysis in Ref. [33] shows that this generalised algorithm induces a two-dimensional probability distribution in (θ_r, θ_d): The probability distribution induced by Shor's order-finding algorithm [33, Fig. A], and by Ekerå–Håstad's algorithm for short discrete logarithms [[33], Figure 5], re-appear as marginal distributions of this two-dimensional distribution [[33], Figure 4], indicating that the requirement that d is short may be relaxed without impacting the distribution.

5.8 | General discrete logarithms

Shor's algorithm for the general DLP [1] in groups of known order is a good option for solving the elliptic curve DLP (EC-DLP), to, for example, break EC-DSA and EC-DH. Proos and Zalka [42] provide an early implementation of the group arithmetic. Shor's DLP algorithm is also a good option for solving the DLP in safe-prime groups with full-length exponents and in Schnorr groups of known order.

If Shor's DLP algorithm is modified as in Ref. [43] to induce the state

$$\frac{1}{2^{2(m+\varsigma)}} \sum_{a,j=0}^{2^{m+\varsigma}} \sum_{b,k=0}^{2^{m+\varsigma}} \exp\left[\frac{2\pi i\,(aj+bk)}{2^{m+\varsigma}}\right] \left|j, k, g^a x^{-b}\right\rangle$$

for m the bit length of the order r and $\varsigma \geq 0$, some small constant, when solving g and $x = g^d$ for $d \in [0, r)$, then the qubit recycling optimisations described in Section 5.4 are directly applicable.

In Ref. [43], it is shown heuristically that this modified algorithm achieves a very high success probability [43, Table 1], if ς is selected to slightly increase the lengths of the control registers compared to what Shor originally proposed and if a small search is performed in Shor's classical post-processing when solving (j, k) for d given r.

It is furthermore explained in [Ref. 43, Section 5.2] how tradeoffs may be achieved by instead using lattice-based post-processing. The idea is very similar to that in Section 5.6, but it requires r to be known.

Kaliski [44] has proposed a different kind of tradeoffs, the idea being to compute the logarithm d, one half-bit at the time via the Blum–Micali [45] reduction. This achieves a maximal tradeoff, at the expense of performing very many runs of the quantum algorithm.

5.9 | Simulations

As previously stated, the probability distributions induced by Shor's algorithms, Seifert's algorithm, and the various algorithms by Ekerå and Håstad may be captured exactly, as error bounded approximation or—for Shor's algorithm for the DLP—heuristically.

This in turn implies that the algorithms may be simulated classically [30, 33, 43], when the solution to the problem instance is known, for example, enabling the characteristics of various post-processing strategies to be evaluated in practice.

5.10 | Cost estimates

The hard part when implementing the aforementioned algorithms is to implement the group arithmetic in a fault-tolerant manner. There are a number of studies in the literature that propose efficient circuits for the group arithmetic and that cost them in various models.

For recent full-stack physical cost estimates for the aforementioned algorithms when working in $\mathbb{Z}_N^*$, see Refs. [17, 46]. For recent optimised logical circuits and cost estimates for the EC-DLP, see Refs. [47, 48]. For a recent physical cost estimate, see Ref. [49].

For physical cost estimates that cover both RSA and the EC-DLP, alongside some widely used symmetric cryptosystems, see Ref. [50].

For a recent survey that compiles expert opinions on the impact of some of these algorithms and cost estimates on the migration timeline, see Ref. [51]. For broader studies, see for example, Refs. [52, 53].

6 | GENERALISING SHOR's ALGORITHM

Shor's factoring algorithm can be viewed as the resolution of the HSP problem on $\mathbb{Z}$. Namely given $f : \mathbb{Z} \to \mathbb{Z}_N$ defined by $f(x) = a^x \pmod{N}$, which has period r and is injective on $\mathbb{Z}/r\mathbb{Z}$, find r. Hallgren [54] generalised Shor's algorithm by considering HSP on $\mathbb{R}$. Since quantum computers are digital, additional conditions are introduced on the oracle function to facilitate finding the hidden subgroup once the function is discretised. Efficient quantum algorithms for several basic number-theoretical problems were then devised, thereby reducing them to the HSP on $\mathbb{R}$, including solving Pell's equation, whereas the best classical algorithms need superpolynomial time. This was later extended to $\mathbb{R}^c$ with constant dimension, which allowed the design of efficient quantum algorithms for computing the unit group and the ideal class group of a number field and finding a generator of a principal ideal (a.k.a. the principal ideal problem PIP), in number fields [55, 56]. These algorithms were proven to run in quantum polynomial time in infinite classes of number fields of fixed degree.

However, this approach is inefficient in families of number fields of arbitrary degree. Indeed, it is not clear how to solve an analogous HSP on $\mathbb{R}^n$ efficiently, given the exponentially growing error with the dimension due to discretisation. This is a consequence of the fact that a *unique* representation of the HSP oracle function's output is essential. However, when reducing the number-theoretical problems to HSP, the output of the oracle is typically a real-valued lattice, lacking a canonical representation. The methods of Refs. [55, 56] designed for families of fields of constant degree do not satisfy these requirements, and computing a suitable oracle function in polynomial time poses a roadblock.

These difficulties were overcome by Eisenträger, Hallgren, Kitaev and Song [57]. They introduced a restricted HSP in $\mathbb{R}^n$, which imposes a Lipschitz continuity condition on the oracle function. The methods for solving this HSP problem introduced in Ref. [57] successfully take advantage of this property of the oracle and achieve a polynomial run time. In Ref. [57], the authors also gave an efficient reduction from computing the unit group in high-degree number fields to the continuous HSP they introduced. A key ingredient is a quantum encoding of a real-valued lattice, addressing the issue of unique representation. Later, Biasse and Song [58] followed this framework and constructed reductions that enable the computation of any S-unit group in arbitrary-degree number fields, which as a consequence gave efficient quantum algorithms for PIP and computing the class group as well. Their quantum algorithm for the PIP turned out to be useful to break several cryptosystems based on a specialised lattice problem [59, 60]. This also inspired more recent quantum cryptanalysis on lattice-based cryptosystems [61, 62].

Here, we give a brief overview of the continuous HSP and the quantum algorithm for solving it. The exposition is adapted from Ref. [57].

Problem 1 (The continuous HSP over $\mathbb{R}^m$). *Assume that the unknown subgroup $L \subseteq \mathbb{R}^m$ is a full-rank lattice satisfying the following promise: there are positive parameters (a, r, ε) and a function $f : \mathbb{R}^m \to S$, where S is the set of unit vectors in some Hilbert space, such that*

1. *f is periodic on L,* that is, *$f(x) = f(x + v)$ for all $x \in \mathbb{R}^m$ and $v \in L$;*
2. *f is Lipschitz with constant a, that is, $\| |f(x)\rangle - |f(y)\rangle \| \le a\|x - y\|$ for all $x, y \in \mathbb{R}^m$;*
3. *If the distance between the cosets $(x \bmod L)$ and $(y \bmod L)$ is greater or equal to r, that is, if $\min_{v \in L}\|x - y - v\| \ge r$, then $|\langle f(x)|f(y)\rangle| \le \varepsilon$.*

   Under these conditions, the continuous HSP problem is to compute a basis of L.

The resolution of Problem 1 relies on the ability to make (efficient) oracle calls $|\mathbf{x}, 0\rangle \mapsto |\mathbf{x}\rangle \otimes |f(\mathbf{x})\rangle$. The quantum algorithm presented in Ref. [57] first creates a superposition of points in $\mathbb{R}^m$ of the form $\sum_{\mathbf{x}\in\mathbb{R}^m} w(\mathbf{x})|\mathbf{x}, 0\rangle$ (normalisation factor omitted) with a sufficiently broad wave function w. This is similar to the first stage of Shor's algorithm. Here, w effectively truncates the function to a finite domain. Then, the oracle f is applied in superposition. In fact, since quantum computers are digital, we can only prepare the superposition over a fine *discrete* grid that approximates $w(\mathbb{R}^m)$. Then, we would like to measure the state in the Fourier basis. Shor's algorithm would perform the Quantum Fourier transform over a finite group $\mathbb{Z}_N$ for a large enough N and measure in the standard basis. However, the resulting approximation errors would be difficult to analyse. In Ref. [57], the authors develop a variation on the phase estimation technique, which may be viewed as approximately performing Fourier sampling over $\mathbb{Z}$. To see how this helps reveal the hidden lattice, note that loosely speaking, in the Fourier domain, the HSP function would be peaked at points in the dual lattice $L^* := \{\mathbf{x} \in \mathbb{R}^m | \, \forall\, \mathbf{y} \in L, \langle \mathbf{x}, \mathbf{y}\rangle \in \mathbb{Z}\}$. In the course of the algorithm, the truncation only disturbs the Fourier domain lightly for the window function w we choose. Then the disturbance due to discretisation can also be controlled due to the Lipschitz condition of the function.

In slightly more technical terms, the subsequent measurement yields $\mathbf{u} \in L^*$ with the probability distribution

$$q_{\mathbf{u}} = \frac{1}{d(L)^2}\int_{(\mathbb{R}^m/L)^2} \langle f(\mathbf{x}')|f(\mathbf{x})\rangle \, e^{2\pi i\langle \mathbf{x}-\mathbf{x}', \mathbf{u}\rangle}\, d\mathbf{x}\, d\mathbf{x}'.$$

Repeating the procedure sufficiently many times, we obtain an approximate set of generators for L^*, from which we can compute an approximate basis for L efficiently.

7 | SYMMETRIC CRYPTOGRAPHY

While not entirely immune to the quantum computing threat, symmetric cryptography is expected to retain a significant level of security in face of the threat. Indeed, the computational assumptions used in symmetric cryptography usually do not exhibit a structure that could be exploited by a quantum adversary, as is the case for the IFP and DLP hardness assumptions and Shor's algorithm.

7.1 | Block ciphers

A block cipher E_k with *block size* n and *key size* κ is a family of permutations of $\{0,1\}^n$ indexed by $k \in \{0,1\}^\kappa$. The first and most important security feature expected from a block cipher design is security in the secret-key setting:

Problem 2 (Secret key recovery). *Given access to encryption and decryption oracles for E_k, with a key k chosen uniformly at random, recover k.*

In practice, an adversary can also try to recover one key among multiple targets, classically as well as quantumly. We focus here on the simplest case of a single key. It is expected that the best algorithm for recovering k is generic exhaustive search. This attack is applicable to any block cipher. It performs $\lfloor n/\kappa \rfloor + 1$ queries with arbitrary plaintexts p_i, stores the obtained plaintext–ciphertext pairs (p_i, c_i), and searches for a candidate key k such that $E_k(p_i) = c_i$ for all the pairs. Note that false positives (wrong keys satisfying this condition) may occur. But if we assume that all permutations E_i are drawn uniformly at random, the choice of $\lfloor n/\kappa \rfloor + 1$ pairs ensures that the probability of false positives is exponentially small in n. Thus, we can safely assume that there is a single solution to the exhaustive search problem. However, this cannot be proven from the specification of E.

Usually, $\lfloor n/\kappa \rfloor$ is constant, and this algorithm requires $O(2^\kappa)$ evaluations of the cipher. Any algorithm performing better is considered as a valid *break*, even if it is computationally infeasible to mount the attack in practice.

Since being *broken* in this definition is a one-bit information, cryptanalysts usually consider reduced versions as attack targets. Let us take as an example the well-studied, 20-year old block cipher standard AES [63]. AES comes in three versions, AES-128, AES-192, and AES-256, with a block size $n = 128$ and a key size $\kappa \in \{128, 192, 256\}$. It is a Substitution-Permutation Network (SPN) with, respectively, $r = 10, 12, 14$ rounds for the three versions. To date, the full AES has withstood cryptanalysis in the secret-key model: Only 7, 8 and 9 rounds have been successfully attacked [64].

Key search with Grover's algorithm: The problem of recovering the key, given a few plaintext–ciphertext pairs (p_i, c_i), is an instance of unstructured search, to which Grover's algorithm can be applied.

The search space is the set of all keys $\{0,1\}^\kappa$, and as noticed above, we can ensure that there is only one marked element. The oracle $\mathcal{O}_f$ evaluates the function

$$f(k) = 1 \text{ iff } \forall\, i, E_k(p_i) = c_i, \quad f(k) = 0 \text{ otherwise.}$$

In general, the cost is reduced from $O(2^\kappa)$ to $O(2^{\kappa/2})$ evaluations. This is why it is often recommended to double the bit length of keys when aiming for long-term security, *when the design permits it*, which is the case for AES-128. For other block ciphers, doubling the key length would require proposing a new design, whose security would need to be studied. Note that as mentioned in Section 3.2, Grover's search parallelises inefficiently (contrary to classical search). This means that doubling the key size is actually a very conservative measure, especially if one considers a realistic adversary limited in time.

Several authors have studied the exact cost of the quantum circuits involved in Grover search, starting with Ref. [65]. As with the classical exhaustive search, one encryption costs more than one gate; instead of 2^{64} basic operations, the Grover search on AES-128 has been optimised so far to about 2^{80} quantum gates [66].

Quantum break of block ciphers: The secret-key cryptanalysis of block ciphers can be naturally extended to the quantum setting. Instead of a key-recovery algorithm faster than the classical exhaustive search, the goal becomes to find a quantum key-recovery algorithm faster than Grover's search. These procedures usually combine classical cryptanalysis techniques with amplitude amplification and quantum walks [67]. For example, the best (but only) results obtained with AES so far [68] attack, respectively, 6, 7 and 8 rounds instead of 7, 8 and 9.

Since attacks are always studied with respect to a generic algorithm, the availability of Grover search, which is a powerful generic algorithm, makes many of them relatively less powerful. However, it is still necessary to study them, as classical attacks do not give meaningful information for the expected quantum security (see the example of SPHINCS-Simpira below).

Finally, note that block ciphers do not exist in a vacuum: They are used as building blocks in operation modes, whose security is discussed in Section 7.3.

7.2 | Hash functions

A hash function H is a one-way function that transforms a message of any size into a *digest* of fixed bit size n. The following problems are expected to be difficult for a secure hash function:

1. preimage search: given a target t, find a *preimage* x such that $H(x) = t$ (requires $O(2^n)$ evaluations of H classically)
2. second-preimage search: given a message y, find a *second preimage* x such that $H(x) = H(y)$ (requires $O(2^n)$ evaluations of H classically)
3. collision search: find a pair (x, y) such that $H(x) = H(y)$ (requires $O(2^{n/2})$ evaluations of H classically)

Preimage search: Grover's algorithm can also quadratically accelerate preimage search, and to this respect, the situation is similar to block cipher key-recovery: The non-generic attacks known so far are quadratic accelerations of classical preimage attacks.

Collision search: Finding a collision of n-bit digests with constant probability can be done classically in time $O(2^{n/2})$, thanks to the *birthday paradox* and polynomial memory, thanks to *Pollard's rho method*. In the quantum setting, this reduces to $O(2^{n/3})$ with Brassard, Høyer and Tapp's algorithm [69]:

1. Make $2^{n/3}$ queries to H on arbitrary inputs $\{x_0, \ldots, x_{2^{n/3}-1}\}$. These queries are classical and stored in classical memory.
2. Using a Grover search, find $y \notin \{x_0, \ldots, x_{2^{n/3}-1}\}$ such that $H(y) \in \{H(x_0), \ldots, H(x_{2^{n/3}-1})\}$.

In the second step, the probability that a random y collides on one of the precomputed values is $2^{n/3} \times \frac{1}{2^n} = 2^{-2n/3}$, which leads to a time complexity $O(2^{n/3})$. This algorithm is optimal for a random function [70]. However, not only does it reach a less than quadratic speedup but also consumes a considerable amount of memory: the table built at step 1. Also, during Step 2, we need to check if y belongs to the table *in superposition over* y. Thus, the time complexity of $O(2^{n/3})$ is only obtained in the QRACM model (see the definitions in Section 2).

By reducing the number of elements below $2^{n/3}$, one obtains the time-memory trade-off curve $T^2 \times M = 2^n$, which contains all quantum collision search algorithms known to date. When $M \leq 2^{n/5}$, the QRACM requirement can be dropped [71]. The memory storage becomes *classical memory with sequential access*, which is easy to implement.

For comparison, classical collision search can be parallelised with the time-space trade-off $T \times S = 2^{n/2}$ [72]. Here 'space' includes both memory and parallel processors. No quantum collision algorithm known to date reaches below this curve, so they remain only applicable if a large memory is considered cheaper than a comparable amount of processing units.

Quantum break of hash functions: Although generic quantum collision search suffers from less speedup and memory usage, this actually makes non-generic quantum collision search algorithms more appealing. Hosoyamada and Sasaki [73] demonstrated that some collision attacks on hash functions could benefit from quadratic speedups, yielding new and *improved attacks* in the quantum setting. Indeed, suppose that a classical algorithm of time complexity $2^{n/2} < T < 2^{2n/3}$ finds a collision of H. Its time complexity is too large to qualify as a classical attack. However, if it benefits from a quadratic acceleration, then we have $\sqrt{T} < 2^{n/3}$, meaning that it yields a *quantum* attack.

There are two important conclusions to draw from this: First, quantum collision attacks on hash functions can be stronger (in terms of rounds attacked) than the classical ones. Second, one should not assume a post-quantum security level for collision search only based on the generic BHT algorithm.

For example, in the proposal Gravity-SPHINCS [74], a modified version of the post-quantum hash-based signature scheme SPHINCS, the hash functions used in the scheme need collision resistance (instead of only preimage resistance for SPHINCS). The authors considered a generic level of security $2^{n/2}$ for quantum collision search, equal to the quantum preimage security, due to the time-memory tradeoff of collision algorithms detailed above. But there could exist hash functions which, although considered classically secure, would invalidate these security claims.

7.3 | Structured constructions and superposition attacks

Contrary to what is often perceived, symmetric cryptographic designs are not devoid of structure. Most symmetric cryptography algorithms exhibit, in fact, a strong algebraic structure, as they need to build high-level functionalities (encryption, authenticated encryption etc.) from very small components (e.g. block ciphers of fixed block size). However, this structure is not necessarily exploitable by a quantum adversary. In this section, we look at *structural* attacks without classical equivalent (contrary to the examples presented above).

The first structural attacks on (classically secure) symmetric designs were published by Kuwakado and Morii in Refs. [75, 76]. Notably in Ref. [76], they remarked that the key-recovery in the Even–Mansour block cipher could be solved as an instance of Boolean period-finding. The Even–Mansour cipher is a generic construction of a block cipher E_{k_1,k_2} from an n-bit public permutation Π and two n-bit keys k_1, k_2 (Figure 6):

$$E_{k_1,k_2}(x) = k_2 \oplus \Pi(k_1 \oplus x) \quad .$$

Kuwakado and Morii's attack defines the following function:

$$f \; : x \mapsto E_{k_1,k_2}(x) \oplus \Pi(x) = k_2 \oplus \Pi(k_1 \oplus x) \oplus \Pi(x),$$

which is such that $f(x \oplus k_1) = f(x)$. Finding the secret k_1 becomes an instance of the following problem:

Problem 3 (Boolean period-finding). *Given access to a two-to-one function f such that $\forall x, y, f(x) = f(y) \Leftrightarrow y \in \{x, x \oplus s\}$ for some value s (the period), then find s.*

This problem is a special case of the Hidden Subgroup Problem in the group $G = (\mathbb{Z}_2)^n$ with the subgroup $H = \{0, s\}$. It is solved by Simon's algorithm [77], which was an inspiration for Shor's, in about $O(n)$ queries to f (thus, to E_{k_1,k_2}), breaking the cipher. However, it requires *superposition queries* to the cipher, that is, the ability to create a state of the form $\frac{1}{2^{n/2}} \sum_{x \in \{0,1\}^n} |x, E_{k_1,k_2}(x)\rangle$. Therefore, the implementation of the attack must use a quantum embedding of E_{k_1,k_2}, which means that a black-box that contains the secret keys must be available.

Later on, it was shown that such *superposition attacks* can target many constructions that are known to be classically secure [67], using Simon's algorithm, but also Kuperberg's algorithm [78] and even Shor's algorithm itself [79]. The practical implications of these attacks remain debated, since without superposition queries, they are inapplicable.

Structured attacks with classical queries: It is now known that the structure exploited by some superposition attacks can also be exploited by attacks making only classical queries, that is, by a standard quantum attacker listening to today's classical communications. Though these attacks do not lead to polynomial-time breaks, they allow one to obtain significantly better time-memory tradeoffs [80] and a more-than-quadratic quantum time speedup on a key-recovery attack [81]. We will now review the principle of the *offline-Simon* algorithm, on which they are based.

A typical target example is the FX construction (Figure 7), which increases the key length of a cipher E_k by XORing two additional n-bit keys k_1 and k_2. If k is known, the FX cipher becomes an Even–Mansour cipher. Using Kuwakado and Morii's attack on Even–Mansour, k is found by looking for a value z such that $f(z) = 1$, where

$$f(z) = \begin{cases} 1 \text{ if } x \mapsto FX_{k,k_1,k_2}(x) \oplus E_z(x) \text{ is periodic} \\ 0 \text{ otherwise} \end{cases} \quad .$$

The oracle function $\mathcal{O}_f$ is computed in superposition over z by calling Simon's algorithm. By using a Grover search on the space $\{0,1\}^{2n}$, if k is of $2n$ bits, this requires $O(2^n)$ iterations. This is the *Grover-meet-Simon* approach of Ref. [82]; a superposition attack so far, since each iteration needs to call FX in superposition.

In Ref. [80], the authors introduced the *offline-Simon* algorithm, which performs the same attack, but makes only $O(n)$ superposition queries to the *FX* quantum oracle at the beginning of the algorithm. Intuitively, the queries done at each iteration in Grover-meet-Simon are actually redundant and can be removed, leaving us only with a single layer of *offline* queries. These queries are stored using $\mathsf{poly}(n)$ qubits only, kept in a *database* and reused at each iteration.

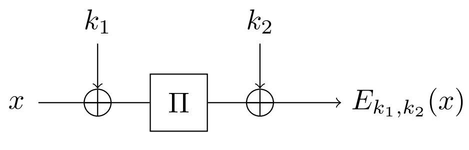

FIGURE 6 The Even–Mansour cipher [85]

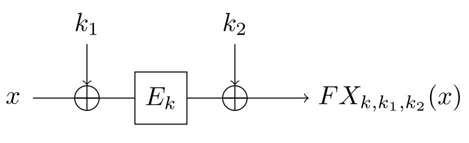

FIGURE 7 The FX cipher [123]

Since only $O(n)$ superposition queries are now made, it becomes possible to replace them entirely with classical queries. Indeed, one can construct a superposition $\frac{1}{2^{n/2}} \sum_{x\in\{0,1\}^n} |x, FX(x)\rangle$ using 2^n classical queries to the black-box FX (the whole codebook) and $\tilde{O}(2^n)$ quantum time. One starts with a state $\frac{1}{2^{n/2}}\sum_{x\in\{0,1\}^n}|x, 0\rangle$, then for each i, XORs $FX(i)$ to the second register if the first one is equal to i. Note that the memory used is still $\mathsf{poly}(n)$ qubits, since the classical queries to FX are consumed online and do not need to be stored. There is also no memory access here, since the database is not accessed in a classical sense; it is only reused to perform instances of Simon's algorithm.

Then, the resulting algorithm runs in quantum time $\tilde{O}(2^n)$, with a first step of $\tilde{O}(2^n)$ computations to prepare the database, and a Grover search of time $\tilde{O}(2^n)$. It does not have a classical equivalent, though a classical attack in time $O(2^{2n})$ does exist.

In Ref. [81], the *offline-Simon* algorithm was extended to target more constructions and in particular, constructions of the form:

$$x \mapsto E'_k(k_2 \oplus E_k(k_1 \oplus x)),$$

with two independent block cipher calls keyed by k. Any classical key-recovery attack requires time $\Omega(2^{5n/2})$, and the best attack requires also $\Omega(2^{n/2})$ memory. But the *offline-Simon* attack uses a quantum time $O(2^n)$, still with $\mathsf{poly}(n)$ qubits. It shows that in this case, *doubling* the key length would not restore the initial level of security.

These recent results show that there exist *inherently quantum* key-recovery attacks, even in the classical query scenario. Although only polynomial gains are expected in this case, the scope of application of these attacks is not fully understood yet, as they have been discovered recently.

7.4 | Open problems

The most important open questions here are related to the quantum attacks without classical equivalents, which so far happened on structured constructions.

Question 1 Can we extend the scope of attacks on structured constructions, in order to target dedicated designs such as AES?

Question 2 Concerning superposition queries, could there exist security arguments of ciphers against superposition attacks, the same way there exist security arguments against standard classical cryptanalysis techniques (differential, linear, algebraic etc.)?

Question 3 Finally, concerning classical queries, what is the limit of quantum speedups? For example, is there a fixed exponent α such that if there is no classical attack of complexity T, then there will be no quantum attack of complexity T^α?

8 | EUCLIDEAN LATTICES

8.1 | Definitions

For $m \geq 1$, a lattice L is a discrete finitely generated additive subgroup of $\mathbb{R}^m$. Equivalently, for a linearly independent set of vectors (called a basis) $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\} \subset \mathbb{R}^m$, a lattice L is the set of all integer linear combinations of $\mathbf{b}_i$'s, that is, $L(B) := \{\sum_i x_i \mathbf{b}_i \mid x_i \in \mathbb{Z}\} = B\mathbb{Z}^m$, where the matrix $B \in \mathbb{R}^{m\times n}$ has $\mathbf{b}_i$'s as columns. The number of basis vectors, n, is the rank of L. In the rest, we shall be concerned with the case $m = n$ (such lattices are called full-rank), as all the algorithms we described here can be easily adapted to the general case.

Associated to a lattice are its invariants: the determinant and the first successive minimum.

Definition 15 (Determinant of a lattice). *The determinant of a lattice is defined as* $\det(L) = \sqrt{\det(B^t B)}$. *Geometrically, it is the volume of* $\mathcal{P}(B)$*, the fundamental parallelepiped of L, defined as* $\mathcal{P}(B) = \{\sum_i c_i \mathbf{b}_i, c_i \in [-1/2, 1/2)\}$.

This volume is independent of the basis B. All lattice bases are related by unimodular transformations U (i.e. det $U = \pm 1$), that is, $B' = BU$ is another basis of $L(B)$.

Definition 16 (Minimum distance of a lattice). T*he first successive minimum (or the minimum distance)* $\lambda_1(L)$ *of lattice L is the Euclidean length of its non-zero vector:* $\lambda_1(L) := \min_{\mathbf{v} \in L \setminus \{\mathbf{0}\}} \|\mathbf{v}\|$.

In general, the ith successive minimum of a lattice, $\lambda_i(L)$, is the smallest r, s.t. L contains i linearly independent vectors of norms at most r. Minkowski's inequality states that for n-rank lattice L, $\lambda_1(L) \leq \sqrt{n}\det(L)^{1/n}$. It is tight up to a constant and is usually treated as equality to approximate the length of the shortest vector.

The *Gram–Schmidt orthogonalisation* (GSO) $B^\star = (\mathbf{b}_1^\star, \ldots, \mathbf{b}_n^\star)$ is obtained iteratively by setting $\mathbf{b}_1^\star = \mathbf{b}_1$, and $\mathbf{b}_i^\star$ as the orthogonal projection of $\mathbf{b}_i$ on $(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})^\perp$ for $i = 2, \ldots, n$. This orthogonalisation process can be described via matrix-decomposition $B = B^\star \mu^t$, where μ is a lower-triangular matrix with $\mu_{i,j} = \langle \mathbf{b}_i, \mathbf{b}_j^\star \rangle / \|\mathbf{b}_j^\star\|^2$ for $i \geq j$.

Hard problems on lattices: There are several fundamental problems related to lattices. Some of them are used in the security proofs of some of the most promising proposals for quantum-safe cryptography.

Problem 4 (Closest Vector Problem (CVP)). *Given a (target) point* $\mathbf{t} \in \mathbb{R}^n$ *and a basis for a lattice L, find* $\mathbf{v} \in L$ *closest to* $\mathbf{t}$.

In the promise variant of CVP, the *Bounded Distance Decoding* (BDD) problem, we know in addition that $\|\mathbf{t} - \mathbf{v}\| < R$ where $R \ll \lambda_1(L)$. In this case, the solution $\mathbf{v}$ is unique.

Problem 5 (Shortest Vector Problem (SVP)). *Given a basis of a lattice L, find $\mathbf{v} \in L$ s.t. $\|\mathbf{v}\| = \lambda_1(L)$.*

Note that a solution of the SVP is not unique: $\|\mathbf{v}\| = \|-\mathbf{v}\|$. We can relax the above and ask for a vector $\mathbf{v}$ s.t. $\|\mathbf{v}\| \leq \gamma\lambda_1(L)$. This problem is called the approximate Shortest Vector Problem (γ-appSVP). The approximation factor γ can be a function of n, in particular, $\gamma = \mathsf{poly}(n)$ is relevant for security of lattice-based cryptographic constructions. We should also emphasise that many of known SVP algorithms actually solve a variant of SVP, called Hermite SVP (or Hermite appSVP)—the problem that asks to find $\mathbf{v}$ s.t. $\|\mathbf{v}\| \leq \gamma \det(L)^{1/n}$. This variant of SVP is stated relative to the determinant of the lattice, not to $\lambda_1(L)$. Again, this version of SVP is more relevant in cryptanalysis. Also, finding the exact value of $\lambda_1(L)$ is by itself a hard problem (as opposed to computing $\det(L)$), so we are interested in algorithms that solve Hermite SVP.

Solving appSVP and BDD: lattice basis reduction and embedding techniques: Lattice basis reduction aims at improving the quality of the input basis, where the 'quality' of a basis is measured by the orthogonality of its vectors (it also translates into the slow decay of $\|\mathbf{b}_i^\star\|$'s). There are several notions of reducedness of a basis ranging from fast but weak (in terms of quality of the output) LLL reduction due to A. Lenstra, H. Lenstra, and L. Lovász [83] to strong but computationally inefficient Hermite–Korkine–Zolotarev reduction [84]. The trade-off between the output quality and the runtime is achieved by a so-called BKZ reduction (short for Block Korkine–Zolotarev [85]). Together with a lattice basis, it receives as input integer parameter β and produces a basis with the first (i.e. the shortest) vector satisfying

$$\|\mathbf{b}_1\| \leq 2\beta^{\frac{n}{2\beta}}(\det L)^{1/n}. \tag{3}$$

In other words, β-BKZ solves $\widetilde{\mathcal{O}}\left(2^{\frac{\beta}{2n}\log\beta}\right)$-appSVP. BKZ works by calling an SVP-solver on certain (projected) sublattices of L of dimension β. In Ref. [86] it was shown that after $\mathsf{poly}(n)$ number of SVP-calls, the guarantee defined in Equation (3) is achieved. Hence, if the running time of an SVP solver for dimension n is $\mathsf{T}_{\mathsf{SVP}}(n)$, the running time of BKZ is $\mathsf{T}_{\mathsf{BKZ}}(\beta) = \mathsf{poly}(n)\cdot\mathsf{T}_{\mathsf{SVP}}(\beta)$.

An SVP solver can also be used to solve CVP (and hence, BDD). In Ref. [87], Kannan shows that given a BDD instance for a lattice L in dimension n, one can construct an $(n+1)$-dimensional lattice L' such that a solution to (a slightly modified) SVP problem in L' gives a solution to the original BDD problem in L. Therefore, the most important algorithmic task is SVP, and it will be our focus in the rest of this section.

8.2 | Algorithms for SVP

Assume that we are given as input a lattice represented by a basis $B \in \mathbb{Q}^{n\times n}$ (we chose to work with rational bases in order not to deal with approximation issues, see Ref. [88] for real-valued input bases) with entries of bit-sizes $\mathsf{poly}(n)$. Our task is to find a $\mathsf{poly}(n)$ approximation to the shortest vector in $L(B)$, that is, solve $\mathsf{poly}(n)$-appSVP. This is the most relevant setting in lattice-based constructions of signatures and encryption schemes [89, 90]. More exotic constructions such as FHE schemes [91] reside on the hardness of $\Omega\left(2^{\log^c n}\right)$-appSVP, which is even easier for known algorithms. Furthermore, we shall be focussing mostly on heuristic algorithms, which may not be able to find exactly the shortest vector, but a $\mathsf{poly}(n)$ approximation to it with a small-degree polynomial. In practice [92], only heuristic versions of SVP algorithms are currently competitive for solving SVP in high dimensions.

State-of-the-art SVP solvers are presented in Table 1. Let us take a closer look at it. Runtimes T are given on the lg-scale relative to the dimension n with smaller order terms omitted, that is, the best known runtime for provable sieving is $2^{2.465n+o(n)}$. The same is for the memory complexities M. Quantum algorithms may require classical random access memory (CRACM), quantumly addressable classical memory (QRACM), or quantumly addressable quantum memory (QRAQM). Most of the quantum algorithms mentioned in this section require QRACM.

Based on asymptotic time and memory complexities (in terms of the lattice dimension n), SVP solvers can be classified into two groups: (1) algorithms requiring super-exponential time $2^{\omega(n)}$ and $\mathsf{poly}(n)$ space and (2) algorithms requiring both exponential time and space $2^{\Theta(n)}$. The former includes the family of so-called enumeration algorithms [93, 94] that run in time $2^{\Theta(n\lg n)}$. Recent result of Albrecht et al. [95] shows the runtime exponent of $\frac{1}{8}n\lg n + o(n\lg n)$ beating the long-standing exponent $\frac{1}{2e}n\lg n + o(n\lg n)$ from Ref. [86]. The latter has been quantumly sped up in Ref. [96] using the quantum backtracking techniques (see Theorem 9) achieving the square-root improvement. It is not obvious that this backtracking technique immediately applies to the result of Albrecht et al.; hence, we do not put the potential exponent of $\frac{1}{16}n\lg n$ in Table 1. We give details on enumeration algorithms later in this section.

Single-exponential time and space algorithms vary in their underlying ideas. *Sieving* algorithms have received arguably most of the attention. Since their introduction in 2001 [97], a series of studies [98–102] have proposed various improvements culminating in the currently best known heuristic runtime and memory exponents of $2^{0.292n+o(n)}$ and $2^{0.2075n+o(n)}$, respectively [103]. There is a significant gap between provable and heuristic sieving algorithms, and closing this gap is an interesting open problem. A very good introduction to provable sieving algorithms can be found in Ref. [104]. In this survey, we shall concentrate on heuristic sieving and make the assumption explicit when we describe the algorithms.

Another approach to solve SVP are algorithms that either use specific properties of discrete Gaussian distribution over a lattice (such as algorithms from Ref. [105] (called BDD-based in Table 1) and Discrete Gaussian Sampling (DGS) algorithms from Ref. [106]), or rely on the properties of the Voronoi cell of a lattice [107]. We are not aware of any quantum speed-ups reported on DGS or Voronoi cell SVP algorithms (hence, we put '—' in Table 1).

TABLE 1 SVP algorithms and their complexities

	Classical		Quantum		
Algorithm	$\frac{\lg T}{n}$	$\frac{\lg M}{n}$	$\frac{\lg T}{n}$	$\frac{\lg M}{n}$	Technique
Enumeration	$\frac{1}{8}\lg n$	$\frac{\lg \mathsf{poly}(n)}{n}$	$\frac{1}{4e}\lg n$	$\frac{\lg \mathsf{poly}(n)}{n}$	Quantum
	[95]		[96]		Backtracking
Sieving (provable)	2.465	1.325	1.799	1.286 (QRACM)	AA
	[114]		[116]		
Sieving (heuristic)	0.292	0.2075	0.257	0.257 (CRAM)	AA
				0.0767 (QRACM)	
				0.0496 (QRAQM)	
	[103]		[118, Theorem 1]		
			0.953	0.5 (CRAM)	AA
BDD-based	1.741	0.5	0.873	$\mathsf{poly}(n)$ qubits	
				0.5 (CRAM)	
				0.1604 (QRACM)	
				$\mathsf{poly}(n)$ qubits	
	[105]		[105]		
DGS-based	$\frac{n}{4c}$	$\frac{n}{4c}$	—	—	
for n^c-appSVP	[107]		—	—	
Voronoi cell	1	1	—	—	
	[66]		—	—	

Classical enumeration: We now give a high-level description of enumeration-based SVP. The algorithms rely on the process that given a radius R searches for all points in $L \cap \mathcal{B}_n(R)$, where $\mathcal{B}_n(R)$ denotes the n-dimensional ball of radius R centred at $\mathbf{0}$. Given a lattice basis B together with its GSO $B^\star$ (recall that $B = B^\star \mu^{\mathrm{t}}$), some $\mathbf{t} \in \mathrm{Span}(L(B))$ and $R > 0$, the algorithm will enumerate candidates $\mathbf{x} = \sum_i x_i \mathbf{b}_i = \sum_i x_i \left(\mathbf{b}_i^\star + \sum_{j=1}^{i-1} \mu_{i,j} \mathbf{b}_j^\star\right)$ by enumerating their coefficients $x_n, \ldots, x_1$ as described below:

- Take all $x_n \in \mathbb{Z}$ s.t. $|x_n| \le R/\|\mathbf{b}_n^\star\|$. These x_n's will be the candidates for the last coefficients of $\mathbf{x}$'s from the ball. Note that the projection of $\mathbf{x}$ orthogonally to Span($\mathbf{b}_1, \ldots, \mathbf{b}_{n-1}$) is $x_n \mathbf{b}_n^\star$, and it is contained in the ball $\mathcal{B}_1(R)$.
- For every x_n from the previous step, take all $x_{n-1} \in \mathbb{Z}$ s.t.

$$|x_{n-1}| \le -x_n \mu_{n,n-1} + \frac{\sqrt{R^2 - x_n^2 \|\mathbf{b}_n^\star\|^2}}{\|\mathbf{b}_{n-1}^\star\|}. \quad (4)$$

- For such choices of x_{n-1}, the projection of $\mathbf{x}$ orthogonally to Span($\mathbf{b}_1, \ldots, \mathbf{b}_{n-2}$) is $x_n \mathbf{b}_n^\star + \left(x_n \mu_{n,n-1} + x_{n-1}\right) \mathbf{b}_{n-1}^\star$ and it lives inside the ball $\mathcal{B}_2(R)$.
- Continue analogously choosing x_i's for each previously found x_{i+1} s.t. the projection of $\mathbf{x}$ with fixed coordinates $x_n, \ldots, x_{i+1}$ orthogonally to Span($\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$) is contained in $\mathcal{B}_{(n-i+1)}(R)$.

Enumeration thus constructs a tree whose nodes of level $i = n, \ldots, 1$ are labelled by the possible values of x_i, and where an edge between a node of level $i + 1$ and a node of level i exists if the corresponding value for x_i is a possible choice given the values $x_n, \ldots, x_{i+1}$. A path from level n down to the level 1 gives a lattice vector $\mathbf{x}$ in $\mathcal{B}_n(R)$. Certain paths do not lead to a leaf of level 1, because Ineq. (4) might not be satisfied for certain integers. The runtime of the enumeration is determined by the size of the tree. Traversing the tree can be done in the depth-first manner requiring only $\mathsf{poly}(n)$ space: we do not keep all the leaves and paths to them, but only the leaf that gives the current shortest solution.

Let us estimate the size of the tree. Let N_k be the expected number of nodes on some level k of the tree that maximises N_k. By construction, N_k is the number of points in the ball $\mathcal{B}_{(n-k+1)}(R)$ that belong to L projected to the orthogonal complement of Span($\mathbf{b}_1, \ldots, \mathbf{b}_k$). According to Gaussian heuristic, $N_k \approx \frac{\mathrm{Vol}\mathcal{B}_{(n-k+1)}(R)}{\|\mathbf{b}_n^\star\| \cdots \|\mathbf{b}_{n-k}^\star\|}$, where the denominator is the determinant of the projected lattice. Computations show [108] that if the input basis B is BKZ-preprocessed (i.e. the shape of $\|\mathbf{b}_i^\star\|$'s can be controlled), then $N_k = 2^{n/2e \lg n + o(n \lg n)}$. Using a more involved preprocessing and tricks [95], one can further improve the runtime exponent to the currently best known

$$T^{\mathrm{C}}_{\mathsf{Enum}}(n) = 2^{\frac{1}{8}n\ \lg\ n + o(n\ \lg\ n)}.$$

In order to set up the radius R, one can, for instance, use Minkowski's upper bound on λ_1. Such enumeration is rather costly, and one can hope that the projection of the shortest vector might be shorter than λ_1. This idea lies behind the *pruned* enumeration [93] strategy, where instead of a ball, the enumeration chooses a different shape (e.g. a cylinder), thus pruning some branches of the tree. There is a trade-off between the success probability and the size of the tree for pruned enumeration that offer practical improvements. Asymptotically, pruning strategies affect the $o(n\ \lg\ n)$ term.

Quantum enumeration: Aono et al. in Ref. [96] showed how to perform SVP enumeration with the quantum backtracking technique of Montanaro (see Theorem 9), leading to roughly the square-root speed-up. Interestingly, their result extends to pruned enumeration as well. Their proof applies to the enumeration that classically runs in $2^{\frac{1}{2e}n\ \lg\ n + o(n\ \lg\ n)}$ time, thus leading to

$$T^{\mathrm{Q}}_{\mathsf{Enum}}(n) = 2^{\frac{1}{4e}n\ \lg\ n + o(n\ \lg\ n)}.$$

Whether the result can be extended to the recent classical enumeration strategy from Ref. [95] potentially leading to $T^{\mathrm{Q}}_{\mathsf{Enum}}(n) = 2^{\frac{1}{16}n\ \lg\ n + o(n\ \lg\ n)}$ is an open question.

Classical sieving: Sieving algorithms receive on input a lattice basis B and start by sampling an exponentially large list $\mathcal{A}$ of (long) lattice vectors from $L(B)$. Sampling relatively long lattice vectors can be done in $\mathsf{poly}(n)$ time [109]. The elements of $\mathcal{A}$ are then iteratively combined to form shorter lattice vectors, $\mathbf{x}' = \mathbf{x}_1 \pm \mathbf{x}_2 \pm \ldots \pm \mathbf{x}_k$ such that $\|\mathbf{x}'\| < \max_{i \le k}\|\mathbf{x}_i\|$, for some $k \ge 2$. Newly obtained vectors $\mathbf{x}'$ are stored in a new list, and the process is repeated with this new list of shorter vectors. It can be shown [101] that after $\mathsf{poly}(n)$ such iterations we obtain a list that (heuristically) contains a shortest vector. Here we shall focus on the case $k = 2$, also known as 2-Sieve, and refer the reader to Ref. [110, 111] for $k \ge 2$ classical sieving and to Ref. [112] for $k \ge 2$ quantum sieving. We only note here that varying k gives time-memory trade-offs: For larger k's, the runtime increases but the algorithm requires less memory.

Following the analysis of 2-Sieve from Ref. [101], we can state the central *heuristic assumption*: all the lists of lattice points appearing during the sieve can be thought of as independently chosen uniform vectors on the unit sphere $\mathcal{S}^{n-1}$. In reality, however, we deal with lattice vectors of Euclidean norm larger than 1 and, furthermore, a lattice may not even (and it likely does not) have exponentially many vectors of the same norm. Yet, we imagine that if we normalised all the vectors in the list $\mathcal{A}$, then they would behave like independently chosen uniform vectors on $\mathcal{S}^{n-1}$. The heuristic analysis proceeds as if we dealt with such normalised identically distributed uniform vectors. The main purpose of introducing this assumption is to aid the *complexity analysis*, although we cannot show the *correctness* of heuristic sieve. In particular, we cannot prove that it does not output $\mathbf{0}$. This is in contrast to provable algorithms [113, 114] that make a lot of analysis effort to show that the output is not $\mathbf{0}$.

Under this heuristic, we estimate the size of the list $\mathcal{A}$. Two vectors $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{A} \subset \mathcal{S}^{n-1}$ satisfy $\|\mathbf{x}_1 \pm \mathbf{x}_2\| < 1 - \varepsilon$ for some small $\varepsilon = 1/\mathsf{poly}(n)$, if the angle between them is less than $\pi/3$. This means that if we place a vector $\mathbf{x}_1$ on $\mathcal{S}^{n-1}$, any other vector $\mathbf{x}_2$ with angular distance from $\mathbf{x}_1$ being less than $\pi/3$ will produce the pair $(\mathbf{x}_1, \mathbf{x}_2)$ whose sum/difference is a shorter vector. In other words, $\mathbf{x}_2$ belongs to the surface of the spherical cap 'centred' at $\mathbf{x}_1$—the area then contains all unit vectors s.t. the angle between them and $\mathbf{x}_1$ is $\le \pi/3$, see Figure 8. The area of this surface relative to the area of the unit sphere is $\sin^n(\pi/3)$.

Thus, in order to cover the unit sphere with such spherical caps, we need on expectation $|\mathcal{A}| \approx 1/\sin^n(\pi/3) = (\sqrt{3}/2)^n = 2^{0.2075n + o(n)}$ vectors (as we assume that the list vectors are independently uniform on $\mathcal{S}^{n-1}$). By increasing this value by a $\mathsf{poly}(n)$ factor, we can heuristically guarantee that almost all spherical caps will contain at least a constant number of list elements, outputting new shorter vectors. We expect the output to be of size $2^{0.2075n\ +\ o(n)}$. This allows us to repeat the whole process with these new shorter vectors being the new list $\mathcal{A}$. After $\mathsf{poly}(n)$ repetitions of sieving, we end up with a non-empty list of short(est) lattice vectors.

We have just established the memory requirement for heuristic 2-Sieve. If we perform the search for all pairs $\mathbf{x}_i$, $\mathbf{x}_j$ that sum to a shorter vector, the runtime will be $|\mathcal{A}|^2 = 2^{0.415n + o(n)}$.

A faster approach is built on the observation that the problem of searching for $\mathbf{x}_i$, $\mathbf{x}_j$ is an instance of the near neighbour problem on the unit sphere. The near neighbour search technique of Becker–Ducas–Gama–Laarhoven [103] leads to the fastest 2-Sieve known today with $2^{0.292n\ +\ o(n)}$ runtime and $2^{0.2075n\ +\ o(n)}$ memory complexities. The algorithm starts by creating a set of 'buckets' indexed by vectors $\mathbf{v} \in \mathcal{S}^{n-1}$ *not* from the given lattice. In the sequel, we make use of the following definition.

Definition 17 (Angular distance). *We say that two vectors* $\mathbf{v}$, $\mathbf{w} \in \mathcal{S}^{n-1}$ *are* α*-close for some* $\alpha \in [0, 1]$*, when* $\langle \mathbf{v}, \mathbf{w} \rangle \ge \alpha$.

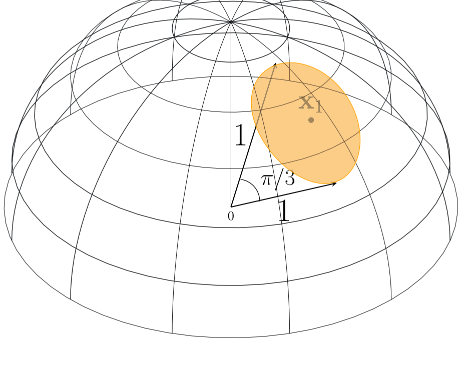

FIGURE 8 Surface (in orange) of the spherical cap on the unit semi-sphere. Subtracting any vector from the orange with $\mathbf{x}_1$ yields a shorter vector

The definition is driven by the distance measure on the unit sphere: The closer the angular distance between **v** and **w** to 0, the more orthogonal the vectors are. For the 2-Sieve, 1/2-close vectors produce a short sum/difference. Identical vectors are at distance 1. We shall drop the word 'angular' in the rest of the section.

For a given set of buckets, a list $\mathcal{A}$, and closeness parameters $0 \le \alpha, \beta \le 1$, the search for close pairs proceeds in two steps:

1. for each $\mathbf{x} \in \mathcal{A}$, search for all buckets **v**'s that are α-close to it and put **x** into the **v**-bucket;
2. for each $\mathbf{x} \in \mathcal{A}$, search for all buckets **v**'s that are β-close to it and then check if these buckets contain a vector close to **x**.

The idea behind this bucketing is that vectors that end up in the same bucket are more likely to give a shorter sum than just two random unit vectors, so we do not have to search for a close pair through the whole list $\mathcal{A}$ but rather through a much shorter bucket. The relation between the parameters α and β is as follows: The closer α to 1 is, the more restrictive the condition on being in the bucket becomes; hence, the buckets contain fewer vectors, so the first step of the above procedure is fast. However, since we intend to find almost all close pairs, we are required to search through more buckets, or equivalently, β should be smaller.

To solve the task of finding relevant buckets for a given **x**, vectors **v** are chosen with some structure. For example, Ref. [103] proposes to choose **v** to be a concatenation of codewords from a specially crafted spherical code. The advantage of choosing such **v** is that it enables us to find relevant buckets for a given **x** in time proportional to the number of such buckets, which is (up to lower order terms) optimal. For further details on this technique, we refer the reader to Ref. [103]; here we simply assume that we can find all relevant buckets for a given vector in time equal to the size of the output.

Let us now analyse the algorithm. We use the following theorem proven in Ref. [115]. We give here its simplified version, where the '≈' sign hides $\mathsf{poly}(n)$ factors.

Theorem 18 (adapted from Th. 1 of Ref. [115]). *If, for some constant $k < n$, $\mathbf{x}_1, \ldots, \mathbf{x}_k$ are independently uniformly distributed on S^{n-1}, then the probability that their pairwise inner-products satisfy $\langle \mathbf{x}_i, \mathbf{x}_j \rangle = C_{i,j}$ is*

$$P\left[\langle \mathbf{x}_i, \mathbf{x}_j \rangle = C_{i,j}\right] \approx \det\,(C)^{n/2},$$

where $C \in \mathbb{R}^{k\times k}$ is a symmetric positive-semidefinite matrix that stores the pairwise inner-products of $\mathbf{x}_i$'s.

- From Theorem 18, the probability that $\mathbf{x} \in \mathcal{A}$ belongs to a fixed bucket is $\det\begin{pmatrix} 1 & \alpha \\ \alpha & 1 \end{pmatrix}^{n/2} = (1-\alpha^2)^{n/2}$. Hence, the expected size of buckets is $|\mathcal{A}|(1-\alpha^2)^{n/2}$.
- If we let $|V|$ be the number of buckets, then for each $\mathbf{x} \in \mathcal{A}$, finding all its relevant α-close buckets takes time $|V|(1-\alpha^2)^{n/2}$, which is the expected number of buckets a point will be put into.
- Analogously, for each $\mathbf{x} \in \mathcal{A}$, inspecting all its β-close buckets takes time $|V|(1-\alpha^2)^{n/2} \cdot |\mathcal{A}|(1-\alpha^2)^{n/2}$.
- Finally, the number of needed buckets $|V|$ can be computed from the probability of the event that a triple $(\mathbf{x}, \mathbf{x}', \mathbf{v})$ satisfies $\langle \mathbf{x}, \mathbf{v}\rangle \ge \alpha$, $\langle \mathbf{x}', \mathbf{v}\rangle \ge \beta$ *provided* that $\langle \mathbf{x}, \mathbf{x}'\rangle \ge \cos(\pi/3) = 1/2$. The inequalities can be treated as equalities [111, Theorem 3], leading to (up to $\mathsf{poly}(n)$ factors)

$$|V|^{-1} = \det\begin{pmatrix} 1 & \alpha & \beta \\ \alpha & 1 & 1/2 \\ \beta & 1/2 & 1 \end{pmatrix}^{n/2} \Big/ \det\begin{pmatrix} 1 & \alpha \\ \alpha & 1 \end{pmatrix}^{n/2}.$$

The optimal values that balance the costs are $\alpha = \beta = 1/2$, resulting in $|V| = \left(\frac{3}{2}\right)^{n/2} \approx 2^{0.292n+o(n)}$. This value determines the runtime. There are techniques [103] that allow not to store all the buckets at once at the price of a slight increase in runtime, which does not affect the leading order term. We conclude on the time and space complexities of classical 2-Sieve:

$$T^{C}_{\mathsf{2Sieve}}(n) = 2^{0.292n+o(n)} \qquad S^{C}_{\mathsf{2Sieve}}(n) = 2^{0.2075n+o(n)}.$$

Quantum sieving: One can apply amplitude amplification techniques to the naive 2-sieve to speed up the $2^{0.415n\ +\ o(n)}$ algorithm to $2^{0.312n\ +\ o(n)}$, [116]. Assuming that we can store the (classical) list $\mathcal{A}$ in QRACM enables us, for each $\mathbf{x} \in \mathcal{A}$, to find a $\mathbf{y} \in \mathcal{A}$ s.t. $\|\mathbf{x} \pm \mathbf{y}\| < \max\{\|\mathbf{x}\|, \|\mathbf{y}\|\}$ in time roughly $\sqrt{|\mathcal{A}|}$ (here we also assumed that we have 1 yper **x** on expectation). Following the notations from Section 3.2, we can efficiently implement an algorithm A that produces a superposition over the pairs from $\mathcal{A}$, that is, (up to normalisation)

$$A|0\rangle^{\otimes n} = |\psi\rangle \coloneqq \frac{1}{|\mathcal{A}|}\sum_{\mathbf{x}_1,\mathbf{x}_2 \in \mathcal{A}} |\mathbf{x}_1\rangle|\mathbf{x}_2\rangle$$

We then apply the amplitude amplification procedure to $|\psi\rangle$ with the checking function $f(\mathbf{x}_1, \mathbf{x}_2)$ that evaluates to 1 if $\|\mathbf{x}_1 \pm \mathbf{x}_2\| < \max\{\|\mathbf{x}_1\|, \|\mathbf{x}_2\|\}$. If, as in the classical 2-Sieve, $|\mathcal{A}| = 2^{0.2075n+o(n)}$, heuristically we expect $|\mathcal{A}|$ such pairs. Hence, after $\mathcal{O}\left(\sqrt{\frac{|\mathcal{A}|^2}{|\mathcal{A}|}}\right) = \mathcal{O}(\sqrt{|\mathcal{A}|})$ applications of f, we measure a pair $(\mathbf{x}_1, \mathbf{x}_2)$ that leads to a shorter vector. Repeating the process $\mathcal{O}(|\mathcal{A}|)$ times leads to a new list of shorter vectors. We repeat the whole process $\mathsf{poly}(n)$ times with the new list of shorter vectors. We only need $\mathsf{poly}(n)$-size registers to perform the algorithm in addition to a QRACM of size $O(|\mathcal{A}|)$.

A quantum version of the near neighbour-based sieve is proposed by Laarhoven in Ref. [117]. The idea is to modify the second step of the 2-sieve algorithm with near neighbour search, namely

1. *classically* store $\mathbf{x}$'s in their α-close buckets. As in Section 7.2, the memory model used is QRACM: classical memory with quantum random access.
2. for each $\mathbf{x} \in \mathcal{A}$, find *classically* all its β-close buckets, create a superposition over all these relevant buckets, and apply amplitude amplification to only those buckets that contain vectors close to $\mathbf{x}$.

As in the classical 2-Sieve with near neighbour search, Laarhoven's algorithm profits from the fact that the buckets we run Grover on are much smaller than the whole list $\mathcal{A}$, and yet, due to the way we construct these buckets, we do not miss the solutions (a rigorous proof of this statement can be found in [Ref. 111, Theorem 3]).

In more details, for each classically known $\mathbf{x} \in \mathcal{A}$, we can design an algorithm $A_{\mathbf{x}}$ that uses the buckets stored in QRACM and satisfies $A_{\mathbf{x}}|0\rangle^{\otimes n} = |\psi\rangle$ with

$$|\psi\rangle = \frac{1}{\sqrt{(\beta\text{-close buckets})}} \cdot \frac{1}{\sqrt{|\text{bucket}|}} \sum_{\substack{\mathbf{v}: \\ \langle \mathbf{x}, \mathbf{v}\rangle < \beta}} \sum_{\mathbf{y} \in \text{Bucket}_{\mathbf{v}}} |\mathbf{y}\rangle,$$

where the outer sum ranges over all buckets (indexed by $\mathbf{v}$) that are β-close to $\mathbf{x}$, and the inner sum ranges over all elements $\mathbf{y}$ from each bucket $\text{Bucket}_{\mathbf{v}}$. We apply amplitude amplification with $A_{\mathbf{x}}$ and the checking function $f_{\mathbf{x}}(\mathbf{y})$ that outputs 1 if $\|\mathbf{x} \pm \mathbf{y}\| < \max\{\|\mathbf{x}\|, \|\mathbf{y}\|\}$. Using the above analysis of the classical 2-Sieve, it is not difficult to see that Laarhoven's quantum 2-Sieve algorithm is optimised for $\alpha = \beta = \sqrt{3}/4$, leading to

$$T^{\mathrm{Q}}_{\text{2Sieve}}(n) = 2^{0.265n + o(n)} \qquad S^{\mathrm{Q}}_{\text{2Sieve}}(n) = 2^{0.265n + o(n)}.$$

Here again, we only need $\mathsf{poly}(n)$-sized quantum registers. Note that we *decreased* α and β in comparison to the classical near neighbour 2-Sieve, since now we can allow for larger buckets as the search inside the buckets is improved. In turn, it means that in total we need fewer buckets. Recall that the total number of buckets determined the runtime. However, contrary to classical sieve, we must store all the buckets because we run Grover search over them. Hence, the space complexity $S^{\mathrm{Q}}_{\text{2Sieve}}(n)$ is asymptotically the same as the time complexity.

Recent results of Refs. [118, 119] further improve quantum 2-Sieve by either exploiting quantum random walks such as in Ref. [118] or in Ref. [119] by running amplification not only over the β-close buckets found classically but over all buckets using as the 'checking' oracle a circuit that samples a β-close bucket with a potential close vector from its bucket. This algorithm sets $\alpha = 1/2$ and $\beta \approx 0.44$, which results in

$$T^{\mathrm{Q}}_{\text{2SieveAA}}(n) = 2^{0.2571n + o(n)} \; S^{\mathrm{Q}}_{\text{2SieveAA}}(n) = 2^{0.2571n + o(n)}.$$

It is more subtle to describe memory requirements of the result from Ref. [118]. As it is based on quantum walks, it requires QRAQ, QRAC, and classical memories, see Table 1.

8.3 | Average-case problems: LWE and SIS

So far, we have been considering the so-called *worst-case* lattice problems, that is, the problems where the input lattice may be arbitrary. Cryptographically relevant problems are *average-case* problems where an instance is generated using some random (known) process. The purpose of this subsection is to describe two main average-case problem on lattices: the Learning with Errors (LWE) problem and the Short Integer Solution (SIS) problem. We refer the reader to the comprehensive survey [120] about the hardness of these two problems and their use in cryptography. We do not consider here the NTRU problem; instead, we refer the reader to the recent survey on NTRU by Albrecht and Ducas [121].

The Learning with Errors problem: introduced by Regev in Ref. [122]; LWE is an average-case instance of BDD on the lattice

$$L_q(A) = A\mathbb{Z}_q^n + q\mathbb{Z}^m,$$

where $A \in \mathbb{Z}_q^{m \times n}$ is a uniform random matrix, $q > 1$ is a modulus, and $m \geq n \geq 1$ are integers. With overwhelming probability, $L_q(A)$ has rank m, and $\det(L_q(A)) = q^{m-n}$. A BDD instance requires a target vector, and in case of LWE, it is

$$\mathbf{b} = A\mathbf{s} + \mathbf{e} \bmod q,$$

for the secret vector $\mathbf{s} \in \mathbb{Z}_q^n$ and the 'noise' vector $\mathbf{e} \in \mathbb{Z}_q$, where $\|\mathbf{e}\| \leq \sqrt{m}\alpha q$ for some $\alpha \in (0, 1)$. We can think of n as of the main security parameter, and the other parameters m, q, α are the functions of n, e.g., $n = \Theta(\text{bit security})$, $m = \Theta(n \log q)$, $q = n^{\Theta(1)}$, and $\alpha = \sqrt{n}/q$.

According to Minkowski's bound, $\lambda_1(L_q(A)) \leq \sqrt{m}q^{1-n/m}$, which is much larger than $\|\mathbf{e}\|$ in the LWE setting; thus, we have a valid BDD instance $(L_q(A), \mathbf{b})$.

The beauty of LWE lies in its versatile use in cryptography: the constructions based on the hardness of LWE range from 'standard' cryptographic primitives such as encryption schemes [122, 123] to fully homomorphic encryption [124] and attribute-based encryption [125]. From the complexity perspective, LWE is particularly attractive due to what is called a worst-case guarantee: Regev [122] shows a quantum reduction from the worst-case problem called the Short Independent Vector Problem (SIVP) to LWE.

Problem 6 (Approximate SIVP). *Given a lattice L and $\gamma \geq 1$, the approximate shortest independent vector problem asks to find n linearly independent vectors from L s.t. their norms are at most $\gamma\lambda_n(L)$.*

Regev gives a polynomial time quantum reduction from SIVP with parameter γ on an arbitrary n-dimensional lattice to LWE with $m = \mathsf{poly}(n), q < 2^{\mathsf{poly}(n)}$ and $\alpha q > 2\sqrt{n}$ for the LWE error $\mathbf{e}$ following the 'discrete Gaussian' distribution. We shall not define this distribution formally here; it is enough to think about this distribution as of discrete analogue of the

Gaussian distribution over $\mathbb{R}$ restricted to a discrete set, say $\mathbb{Z}$, and extended to $\mathbb{Z}^n$ by sampling each coefficient independently.

Regev's reduction tells us that any algorithm for LWE, be it classical or quantum, efficiently transfers to a quantum algorithm for SIVP, which we believe to be a hard problem. Therefore, a reasonable question is, *how hard* is LWE?

Since LWE is a BDD instance, a natural approach is to use Kannan's embedding and apply BKZ reduction. Under Kannan's embedding, an LWE instance $(A, \mathbf{b})$ translates into a $\mathcal{O}\left(\frac{q^{1-n/m}}{\alpha q}\right)$-appSVP instance. In order to solve this appSVP, we run BKZ reduction with parameter β set to be of order

$$\beta_{\mathsf{LWE}} = \frac{\log q}{2\log^2 \alpha} \log\left(\frac{n \log q}{\log^2 \alpha}\right) \cdot n.$$

This value for β_{LWE} is obtained by solving for β the equation $2^{\frac{m}{2\beta}\log\beta} = \frac{q^{1-n/m}}{\alpha q}$ and minimising it for m. We can plug-in β_{LWE} into any complexity of SVP solver (classical or quantum) for a β_{LWE}-dimensional lattice, thus obtaining an estimate on the asymptotic hardness of LWE.

More elaborate analyses [126] and LWE estimators [127, 128] refine the behaviour of BKZ reduction and consider the evolution of the norms of Gram–Schmidt vectors during BKZ, leading to a more accurate estimate aiding to set the security levels of concrete LWE parameters. We do not detail this technique here, but refer the reader to the above references.

Do there exist specific non-lattice-based attacks on LWE? First, there are *classical* combinatorial approaches to solve LWE [129–131]. Second, there is a method (also classical) due to Arora-Ge [132], see also Ref. [133], that uses Gröbner-basis solvers to attack LWE. These methods are currently inferior to lattice-based attacks for all practically relevant parameter sets, partially due to fact that they perform poorly (or even do not work at all) when m is as small as $\Theta(n)$, or even $\mathsf{poly}(n)$. We are not aware of any reported quantum speed ups specific to these attacks.

There are also interesting results, when 'LWE is given as quantum states'. One can define at least two versions of what it means. First, following Ref. [134], one can ask whether for a fixed $\boldsymbol{s}$ and $\boldsymbol{e}$, given a superposition over all possible rows of A as $\frac{1}{q^{n/2}} \sum_{\mathbf{a}\in\mathbb{F}_q^n} |\mathbf{a}\rangle|\langle \mathbf{a}, \mathbf{s}\rangle + \mathbf{e}\rangle$, one can efficiently find $\boldsymbol{s}$ and $\boldsymbol{e}$. The authors in Ref. [134] give affirmative answer, noticing that QFT, applied to the above state, reveals the solution.

In another formulation of "quantum LWE", stated in Ref. [135], one is given a uniform random $A \in \mathbb{Z}_q^{m\times n}$ with $\boldsymbol{a}_i$'s being the rows of A, and $\frac{1}{\sqrt{\sum_{i\in\mathbb{Z}}|f(i)|^2}} \sum_{e_i\in\mathbb{Z}} f(e_i)|\langle \mathbf{a}_i, \mathbf{s}\rangle + e_i \bmod q\rangle$, where $f()$ can be any function over $\mathbb{Z}$ s.t. $\sum_{i\in\mathbb{Z}} i\cdot |f(i)|^2 < +\infty$ (for example, f can be a pdf of discrete Gaussian distribution or an indicator function of a finite set). Now finding $\mathbf{s}$, $\mathbf{e}$ in this variant of quantum LWE appears to be hard, as we do not know any efficient algorithm for it [135], apart for some specific cases of f and of LWE parameters [136].

The Shortest Integer Solution problem: for $n > 0$, m $\geq n$, $q > 1$, and $b > 0$, asks, given a uniform matrix $A \in \mathbb{Z}_q^{m\times n}$, to find $\mathbf{x} \in \mathbb{Z}^n$, s.t. 1) $\mathbf{x}^{\mathrm{t}}A = 0 \bmod q$, and 2) $0 < \|\mathbf{x}\| \leq b$.

As in LWE, one can think of n as of the main security parameter, $m = \Theta(n \log n)$, $q = \mathsf{poly}(n)$.

The problem has been introduced by Ajtai in Ref. [137], where he showed a classical worst-case reduction from SIVP with approximation factor $\gamma = b\cdot \mathsf{poly}(n)$ on an arbitrary n-dimensional lattice to SIS with $m = \mathsf{poly}(n)$, $q > b\cdot\mathsf{poly}(n)$.

The relation to lattices is immediate once we look at the so-called orthogonal lattice

$$L_q^{\perp}(A) = \{\mathbf{x}\in\mathbb{Z}^m \ : \ \mathbf{x}^{\mathrm{t}}A = 0 \bmod q\}.$$

This lattice is of dimension m, and with overwhelming probability its determinant is $\det\left(L_q^{\perp}(A)\right) = q^n$. Therefore, according to Minkowski, $\lambda_1\left(L_q^{\perp}(A)\right) \leq \sqrt{m}q^{n/m}$, which asymptotically belongs to $\Theta(\sqrt{n\log n})$ when $m = \Theta(n \log n)$, $q = \mathsf{poly}(n)$. Analogous to LWE, SIS is the appSVP problem, but now on the lattice $L_q^{\perp}(A)$ with approximation factor $\frac{\beta}{\Theta(\sqrt{n\log n})}$. Exactly the same arguments as for LWE lead to the estimate on the BKZ parameter β needed to solve SIS:

$$\beta_{\mathsf{SIS}} = \frac{\log q}{2\log^2 \beta} \log\left(\frac{n \log q}{\log^2 \beta}\right) \cdot n.$$

Again, as in LWE, one can run BKZ with either classical SVP oracles, thus having $2^{0.292\beta_{\mathsf{SIS}}+o(\beta_{\mathsf{SIS}})}$ as the currently best achievable runtime, or with quantum SVP oracle improving the 0.292 constant to 0.2571.

There is much less non-lattice-based approaches to solve SIS than for LWE. A recent result from Ref. [136] gives a quantum algorithm for SIS in the infinity norm, that is, when the SIS solution $\mathbf{x}$ is required to be bounded in ℓ_∞-norm. The algorithm achieves polynomial time when $= (q-c)/2$, and $m = \Omega(q^4 n^{c+1} \log q)$ for some constant c. Note that SIS (in ℓ_∞-norm) is easy when $b = q/2$; thus, this result gives an efficient algorithm for a non-trivial range, yet it is far from what is used in cryptographic applications. In particular, certain signature schemes, such as Ref. [90], rely on the hardness of SIS in ℓ_∞ for $b \approx q/8$. Apart from this quantum algorithm, Ref. [136] gives a classical algorithm that solves SIS in time $\mathcal{O}\left(n^{\log q}\right)$ when $m = \mathcal{O}\left(n^{\log q}\right)$.

8.4 | Open problems

The results presented in this section are purely asymptotic and it is far from an easy task to estimate the small order terms. Currently to get a 'feeling' on how the algorithms perform we actually implement and run them in practice. While this is possible for classical algorithms [138], at this stage it seems very unlikely that we shall be able to run quantum sieve (or even low-memory enumeration) in the near future. The result from Ref. [139] suggests that speed-ups for memory-intense SVP approaches are rather dubious under our today's understanding of quantum gate complexity. Hence,

Question 1 are known quantum speed-ups for SVP algorithms plausible, or do the hidden terms and (quantum) memory overheads diminish them?

Switching to a more optimistic side and assuming that we shall be able to run SVP algorithms quantumly, the question of designing efficient quantum circuits for *polynomial*-time algorithm like LLL is open. Therefore,

Question 2 what are potential quantum speed-ups for 'easy' lattice-reductions like LLL or BKZ with small block-size?

Turning to the average-case problem, like LWE and SIS, the currently most efficient approach to solve these problems for cryptographically relevant parameters is to run the appSVP solvers, that is, BKZ reduction. So,

Question 3 are there any non-lattice-based approaches (classical or quantum) to solve LWE/SIS that are faster, may be even for some limited parameter ranges, than the approach via lattice-reduction?

The last question is not limited to the ℓ_2-norm: LWE is also interesting when the error vector comes form a uniform distribution. Respectively, SIS is of importance as well [90] when the solution is required to be bounded in ℓ_∞-norm.

9 | CODE-BASED ASSUMPTIONS

9.1 | Definitions

Throughout this section $\mathbb{F}$ denotes a finite field.

Definition 19 (Hamming weight). *For $\mathbf{x} \in \mathbb{F}^n$, the Hamming weight, denoted by $\omega(\mathbf{x})$, is the number of non-zero coordinates of* $\mathbf{x}$.

Definition 20 (Linear code). *For integers $1 \le k \le n$ and $d < n$, a linear $[n, k, d]$-code $\mathcal{C}$ is a subspace of $\mathbb{F}^n$ of dimension k, where $d := \min_{\mathbf{c}\neq\mathbf{0},\mathbf{c}\in\mathcal{C}} \omega(\mathbf{c})$ is called the minimal distance of the code.*

Associated to a linear code $\mathcal{C}$ are its generator matrix $G \in \mathbb{F}^{k\times n}$ and its parity-check matrix $H \in \mathbb{F}^{n-k\times n}$. A code $\mathcal{C}$ can be equivalently defined as the row space of G or as a kernel of H.

Definition 21 (Information Set Decoding (ISD)). *The Information Set Decoding problem asks to find the error-vector $\mathbf{e} \in \mathbb{F}^n$ given a parity-check matrix $H \in \mathbb{F}^{(n-k)\times n}$ of a linear $[n, k, d]$-code $\mathcal{C}$, a vector $\mathbf{s} = H\mathbf{e}^{\mathrm{t}}$ called the syndrome, and w—the Hamming weight of* $\mathbf{e}$.

If $w \le \lfloor\frac{d-1}{2}\rfloor$, then the solution of ISD is unique. The weight w can be unknown, in which case one can simply guess it as it lies in the known small range. Let us make some specific instantiations of the ISD parameters relevant to (post-quantum) cryptography:

- *Classic McEliece KEM.* The security of Classic McEliece Key Encapsulation Mechanism (KEM) [140] is based on the ISD problem over $\mathbb{F}_2$, where H is systematic form of the parity-check matrix of a Goppa code, and $w = (n-k)/\lceil \lg\ n\rceil$. Among the existing post-quantum cryptographic schemes, Classic McEliece is considered to be the most conservative from the security perspective.
- *BIKE* [141] is another code-based key encapsulation mechanism over $\mathbb{F}_2$, where $n = 2$k, the parity-check matrix H is of the form $[\mathsf{rot}(h)|I_k]$, where $\mathsf{rot}(h)$ is a circulant matrix consisting of the coefficients of cyclic rotations of public polynomial h, and $w \approx \sqrt{n}$. In essence, BIKE is a binary version of the NTRU cryptosystem [142].
- *WAVE signature* [143], contrary to the above examples, is based on a version of the ISD problem that asks for a *dense* error solution, not a *sparse* one. This is partially driven by the fact that the problem is formulated over $\mathbb{F}_3$. Relevant to WAVE is the setting $w \approx 0.95n$ and with multiple solutions (intuitively, there usually exists many signatures for a message, hence the non-uniqueness of ISD solutions).

From the above examples, the hardness of ISD directly impacts the security of various cryptographic scheme. In the rest of the section, we specialise to the case $\mathbb{F} = \mathbb{F}_2$. The methods we describe generalise to the $\mathbb{F}_q$ setting, see Ref. [144] and, for cryptanalysis of the WAVE setting see Ref. [145].

In order to simplify the analysis of ISD algorithms, it is usually assumed that the matrix H is drawn uniformly at random from $\mathbb{F}_2^{n-k\times n}$. Despite the fact that some cryptographic schemes use structured H, such as quasi-cyclic H in BIKE, we are not aware of significant speed-ups that use this structure (the speed-ups we know are at most linear in n, see [Ref. 141, Sec. B.2.1]). With this assumption on H, the hardness of ISD resides on two parameters: n and $\omega(\boldsymbol{e})$. The dimension k is usually related to n as $k = \Theta(n)$, often $k \approx n/2$.

All known classical ISD algorithms [146–149] try to find $\mathbf{e}$ by enumerating its search space, which is of size $\binom{n}{w}$. The difference between various ISD algorithms lies in the way this enumeration is performed. All known *quantum* algorithms [150–152] for ISD are quantum versions of the existing classical algorithms, in which some routines are sped up by the quantum methods such as amplitude amplification or quantum random walks. Below, we describe known ISD algorithms and their quantum speed-ups.

We should warn the reader that for simplicity of the exposition we omit $\mathcal{O}$ or $\widetilde{\mathcal{O}}$ terms. We do not present the complexities of the form $2^{cn+o(n)}$ for some constant c, as it is often done in the cited studies. Instead, we let the reader be able to optimise the complexity formulas by themselves for the concrete parameters they are interested in.

9.2 | ISD algorithms

Prange's algorithm: In the 60s, Prange showed [148] how a simple linear algebra trick can improve the enumeration of $\mathbf{e}$. Notice first that permuting the columns of H is equivalent to permuting the positions of 1's in $\mathbf{e}$. Prange's algorithm consists in finding a permutation π such that $\pi(\boldsymbol{e})$ has exactly zero 1's on the first k coordinates and all the weight w is distributed over the last $n - k$ coordinates.

To check whether a candidate π is good, we transform $\pi(H)$ into systematic form $[Q|I_{n-k}]$ (provided the last $n - k$ columns of $\pi(H)$ form an invertible matrix, which happens with constant probability) for $Q \in \mathbb{F}_2^{n-k\times k}$. The same transformation is applied to the syndrome $\boldsymbol{s}$ giving a new syndrome $\bar{\mathbf{s}}$ and a new decoding equation $Q\mathbf{e}_1 + \mathbf{e}_2 = \bar{\mathbf{s}}$, where $\boldsymbol{e} = [\boldsymbol{e}_1 | \boldsymbol{e}_2]$ and $\omega(\boldsymbol{e}_1) = 0$, $\omega(\boldsymbol{e})_2 = w$. It follows that for such π, $\mathbf{e}_2 = \bar{\mathbf{s}}$, and one checks whether $\omega(\bar{\mathbf{s}}) = w$.

We expect to find a 'good' permutation π (i.e. a π that gives the correct weight distribution and makes the last $n - k$ columns of $\pi(H)$ invertible) after

$$T^{C}_{\texttt{Prange}} = \frac{\binom{n}{w}}{\binom{n-k}{w}}$$

trials, which is the inverse of the probability of finding a 'good' π.

To speed-up Prange's algorithm quantumly, Bernstein in Ref. [150] uses Grover's search over the space of permutations, where Grover's function f evaluates to 1 on a 'good' permutation π, that is, $\mathrm{Cost}(\mathcal{O}_f) = \mathsf{poly}(n)$ and implements computation of the systematic form of H, matrix-vector multiplication and evaluation of the Hamming weight. It follows from Proposition 6 that the number T^Q of calls to $\mathcal{O}_f$ is

$$T^{Q}_{\texttt{Prange}} = O\left(\sqrt{\frac{\binom{n}{w}}{\binom{n-k}{w}}}\right).$$

Classical and quantum memory complexities of Prange's algorithm is $\mathsf{poly}(n)$ as we only store one matrix H and one vector $\boldsymbol{s}$. Such a low memory requirement is a significant advantage of this algorithm over the other memory challenging ISD solvers we discuss next.

Stern's algorithm: It is convenient to work with the ISD equation where H is in systematic form: $[Q|I_{n-k}] \cdot \mathbf{e}^{t} = \bar{\mathbf{s}}$ for $Q \in \mathbb{F}_2^{n-k\times k}$. Rather than restricting the weight of $\boldsymbol{e}$ to be 0 on the first k coordinates, Stern in Ref. [149] proposed to allow $p > 0$ non-zero coordinates in the first k indices at the price of a more expensive check for π. It was later improved in Ref. [153] and also in Ref. [154].

Stern's idea can be viewed as a meet-in-the-middle technique: Assume a good permutation π gives us $\boldsymbol{e}$ with $(w - p)$ 1's on $n - k$ coordinates that correspond to I_{n-k}, and with p 1's on k coordinates that correspond to Q. Then represent $\mathbf{e}$ as $\mathbf{e} = (\mathbf{e}_1\|0) + (\mathbf{e}_2\|0) + (\mathbf{0}\|\mathbf{e}_3)$, where $\mathbf{e}_1 \in \mathbb{F}_2^{k/2} \times 0^{k/2}$, $\mathbf{e}_2 \in 0^{k/2} \times \mathbb{F}_2^{k/2}$, $\mathbf{e}_3 \in \mathbb{F}_2^{n-k}$. Hence, the ISD equation rewrites as

$$Q\mathbf{e}_1 \approx \mathbf{s} + Q\mathbf{e}_2, \tag{5}$$

where the $\approx$ sign indicates that the right-hand side of Equation (5) differs from the left-hand side only on $\omega(\mathbf{e}_3) = w - p$ coordinates. Stern's algorithm enumerates two lists, $L_1 = \{Q\mathbf{e}_1\}$ and $L_2 = \{\mathbf{s} + Q\mathbf{e}_2\}$, and searches for two elements $\mathbf{v}_1$, $\mathbf{v}_2$ from each list s.t. $\omega(\mathbf{v}_1 - \mathbf{v}_2) = w - p$. The corresponding error vectors $\mathbf{e}_1$, $\mathbf{e}_2$ are implicitly stored alongside in the lists.

This is an instance of the so-called near neighbour problem for the Hamming distance: Given two lists L_1, L_2, the problem asks to find (almost) all pairs $(\mathbf{v_1}, \mathbf{v_2}) \in L_1 \times L_2$ that are close under the Hamming distance. For instance, it can be solved by testing whether on certain fixed ℓ coordinates, the vectors from L_1 and L_2 match, that is, equal to each other, exactly. If so, they are then tested for approximate match on the remaining coordinates. In the near neighbour literature, this method was put forward by Indyk–Motwani [155].

Finiasz and Sendrier in Ref. [154] proposed an improvement to Stern's algorithm: They introduced the ℓ-length 0-window into the systematic form of H, so now it is of the form $\left[Q' \middle| \frac{I_{n-k-\ell}}{0}\right]$ for $Q' \in \mathbb{F}_2^{n-k\times k+\ell}$. This shape can be reached by applying a partial Gaussian elimination on the right-hand upper square $(\mathrm{n} - k - \ell) \times (\mathrm{n} - k - \ell)$ submatrix of H, giving us $I_{n-k-\ell}$. Then we force the bottom ℓ rows to be 0 by adding the appropriate rows of $I_{n-\mathrm{k}-\ell}$.

From now on, we will be working with the ISD equation of the form

$$\left[Q' \middle| \begin{matrix} I_{n-k-\ell} \\ 0 \end{matrix}\right] \mathbf{e}^{\mathrm{t}} = \mathbf{s}.$$

Then the same strategy as in the original Stern's algorithm applies. Let $\mathbf{e}_1 \in \mathbb{F}_2^{\frac{k+\ell}{2}} \times 0^{\frac{k+\ell}{2}}$, $\mathbf{e}_2 \in 0^{\frac{k+\ell}{2}} \times \mathbb{F}_2^{\frac{k+\ell}{2}}$ and $\mathbf{e}_3 \in \mathbb{F}_2^{n-k-l}$. If $\mathbf{e} = (\mathbf{e}_1\|0) + (\mathbf{e}_2\|0) + (\mathbf{0}\|\mathbf{e}_3) \in \mathbb{F}^n$ is solution, then

$$Q'\mathbf{e}_1 + \mathbf{s} + Q'\mathbf{e}_2 = \left[\begin{matrix} I_{n-k-\ell} \\ 0 \end{matrix}\right] \mathbf{e}_3 \in \mathbb{F}_2^{n-k-\ell} \times 0^{\ell}.$$

This means that $Q'\mathbf{e}_1$ and $\mathbf{s} + Q'\mathbf{e}_2$ are necessarily equal on their last ℓ coordinates. We search for two vectors, $\mathbf{v}_1 \in L_1 = \{Q'\mathbf{e}_1\}$ and $\mathbf{v}_2 \in L_2 = \{\mathbf{s} + Q'\mathbf{e}_2\}$, that are equal on this ℓ-window. We expect to find $\frac{|L_1|\cdot|L_2|}{2^{\ell}}$ such pairs, one of them being our solution. It turns out that the additional parameter ℓ gives a better handle for balancing the time for finding a good permutation and enumerating the lists. In particular, this approach leads to classical time complexity

$$T^{C}_{\texttt{Stern}} = \frac{\binom{n}{w}}{\binom{\frac{k+\ell}{2}}{p}\binom{n-k-\ell}{w-p}} \cdot \max\left\{\left|L_2\right|, \frac{|L_1|\cdot|L_2|}{2^{\ell}}\right\},$$

where the first multiple is the expected number of permutations we need to choose before we hit the needed weight distribution, the arguments inside max{} are the complexity of creating L_1, L_2 and sorting L_2, and the expected number of pairs from $L_1 \times L_2$ that are equal on the ℓ-window, which we check for a solution. Note that by construction $|L_1| = |L_2| = \binom{k+\ell}{p}$. The space complexity of this algorithm is $S^C_{\mathtt{Stern}} = |L_1|$. The parameters p, ℓ are subject to optimisations for concrete n, k, w. It might be instructive to view Prange's algorithm as a one for-loop procedure searching for a good permutation. Then, Stern's algorithm is a two for-loop procedure with the outer loop searching for a permutation and the inner loop checking whether the permutation is correct. The purpose of the parameters p, ℓ is to remove the workload from the outer loop to the inner loop, thus rebalancing the overall cost. It turns out that in the ISD setting when $w = \Theta(n)$, we have $p, \ell = \Theta(n)$, hence improving the decoding asymptotically.

We describe quantum speed-ups for Stern's algorithm (and its modifications) after we explain another useful technique to solve ISD: the representation-based algorithms.

Representation-based techniques: The ideas from Refs. [146, 156] enable us to obtain the lists L_1, L_2 faster by first enlarging the enumeration space for **e**, thus creating many solutions, and then only looking for specific ones. Concretely, let $L_1 = \{Q'\mathbf{e}_1\}$, with $\mathbf{e}_1 \in \mathbb{F}_2^{k+\ell}$ (as opposed to $\mathbf{e}_1 \in \mathbb{F}_2^{\frac{k+\ell}{2}}$ in Stern) with $\omega(\mathbf{e}_1) = p/2$. Analogously, $L_2 = \left\{\mathbf{s} + Q'\mathbf{e}_2 \mid \mathbf{e}_2 \in \mathbb{F}_2^{k+\ell}, \omega(\mathbf{e}_2) = p/2\right\}$. The key observation is that now there are $\mathcal{R}_{\mathrm{MMT}} := \binom{p}{p/2}$ ways to represent the target **e** as $\mathbf{e} = \mathbf{e}_1 + \mathbf{e}_2$. Hence, on average, it is enough to construct only an $\mathcal{R}_{\mathrm{MMT}}$−fraction of L_1, L_2.

We do so by restricting the elements in L_1, L_2 to be 0 on $\lfloor \log_2(\mathcal{R}_{\mathrm{MMT}}) \rfloor$ coordinates (these coordinates are subsets of the ℓ coordinates we aim to match on at the end). We construct such L_1 by merging in the meet-in-the-middle way yet another two smaller lists $L_{1,1}$ and $L_{1,2}$. Henceforth, we refer to the term 'merging' as a process of creating one list out of another two given lists (vectors) by summing only those elements from the given lists that satisfy a certain relation. In ISD algorithms, this relation is equal to 0 on some coordinates. In the MMT algorithm, due to May–Meurer–Thomae [146], $L_{1,1}$, $L_{1,2}$ are of the form

$$L_{1,1} = \left\{ Q'\left[\mathbf{e}_{1,1} \| \mathbf{0}^{\frac{k+\ell}{2}}\right] \mid \mathbf{e}_{1,1} \in \mathbb{F}_2^{\frac{k+\ell}{2}}, \omega(\mathbf{e}_{1,1}) = p/4 \right\},$$

$$L_{1,2} = \left\{ Q'\left[\mathbf{0}^{\frac{k+\ell}{2}} \| \mathbf{e}_{1,2}\right] \mid \mathbf{e}_{1,2} \in \mathbb{F}_2^{\frac{k+\ell}{2}}, \omega(\mathbf{e}_{1,2}) = p/4 \right\},$$

We require that during the merge the sum $Q'[\mathbf{e}_{1,1} \| \mathbf{0}^{(k+\ell)/2}] + Q'[\mathbf{0}^{(k+\ell)/2} \| \mathbf{e}_{1,2}]$ is zero on $\lfloor \log_2(\mathcal{R}_{\mathrm{MMT}}) \rfloor$ last coordinates.

The list L_2 is constructed analogously except that, similar to Stern, we include the syndrome **s** to $L_{2,2}$. Pictorially the algorithm has a tree-structure with each node being a list, and such a view is provided in Figure 9.

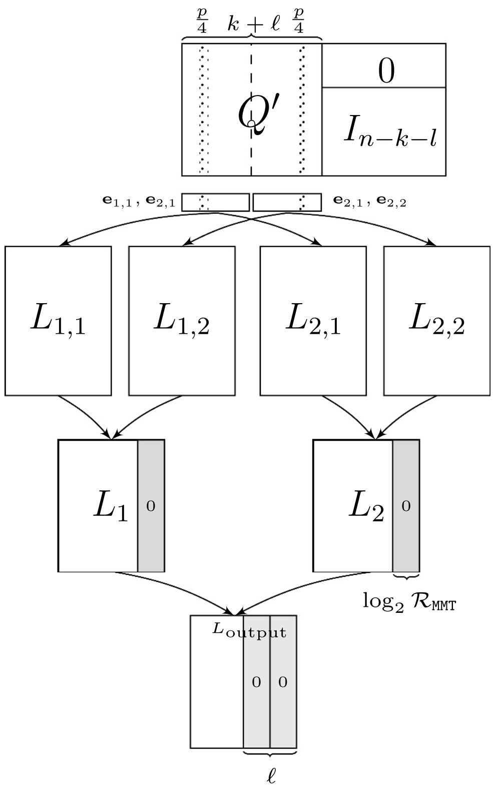

FIGURE 9 Representation-based ISD algorithm due to Ref. [146]

The correctness of this algorithm can be found in Ref. [146]. Let us analyse its complexity. The number of necessary permutations we need to try before we have the correct weight distribution on **e** is

$$P_{\mathtt{MMT}} := \frac{\binom{n}{w}}{\binom{(k+\ell)/2}{p/2}^2 \binom{n-k-\ell}{w-p}}. \tag{6}$$

To check whether a permutation π is good, we attempt to find **e** by constructing the lists from Figure 9. Provided the lists on the same level are of the same excepted size, the complexity of this routine is given by

$$\max\left\{ |L_{1,1}|, \frac{|L_{1,1}|^2}{2^{\lfloor \log_2 \mathcal{R}_{\mathrm{MMT}} \rfloor}}, \frac{|L_{1,1}|^4}{2^{\ell + 2\lfloor \log_2 \mathcal{R}_{\mathrm{MMT}} \rfloor}} \right\}.$$

This quantity is the maximum between (I) the size of top 4 lists (II) the size of the output after the first merge on $\lfloor \log_2 \mathcal{R}_{\mathrm{MMT}} \rfloor$ coordinates, and (III) the size of the output after merging on the remaining $\ell - \lfloor \log_2 \mathcal{R}_{\mathrm{MMT}} \rfloor$ coordinates. Since $|L_{1,1}| = \binom{(k+\ell)/2}{p/2}$, we obtain

$$T^C_{\mathtt{MMT}} = P_{\mathtt{MMT}} \cdot \max\left\{ |L_{1,1}|, \frac{|L_{1,1}|^2}{2^{\lfloor \log_2 \mathcal{R}_{\mathrm{MMT}} \rfloor}}, \frac{|L_{1,1}|^4}{2^{\ell + 2\lfloor \log_2 \mathcal{R}_{\mathrm{MMT}} \rfloor}} \right\}.$$

The memory complexity of this algorithm is

$$S^C_{\mathrm{MMT}} = \max\left\{\left|L_{1,1}\right|, \frac{\left|L_{1,1}\right|^2}{2^{\lfloor \log_2 \mathcal{R}_{\mathrm{MMT}} \rfloor}}\right\}$$

(we omit the size of the output list since we do not have to store it).

Becker–Jeux–May–Meurer in Ref. [156] notice that we can increase the number of representations by splitting zero-coordinates of $\mathbf{e}$ not only as $0+0$ but also as $1+1$. It turns out that constructing longer top-level lists $L_{i,j}$ with $\mathbf{e}_{i,j}$ of weights $\omega(\mathbf{e}_{i,j}) = p/2 + \varepsilon$ for some integer $\varepsilon > 0$ improves the algorithm as it significantly increases the number of representations from $\binom{p}{p/2}$ to $\mathcal{R}_{\mathrm{BJMM}} := \binom{p}{p/2}\binom{k+\ell-p}{\varepsilon}$, where the second multiple is the number of ways we can choose ε 1's out of $k+\ell-p$ coordinates. Intuitively, the strategy allows for a better balance between the two merges: the first merge on $\lfloor \log_2 \mathcal{R}_{\mathrm{BJMM}} \rfloor$ coordinates and the second on $\ell - \lfloor \log_2 \mathcal{R}_{\mathrm{BJMM}} \rfloor$ coordinates. The expected running time of the BJMM algorithm is given by

$$T^C_{\mathrm{BJMM}} = P_{\mathrm{MMT}} \cdot \max\left\{\left|L_{1,1}\right|, \frac{\left|L_{1,1}\right|^2}{2^{\lfloor \log_2 \mathcal{R}_{\mathrm{BJMM}} \rfloor}}, \frac{\left|L_{1,1}\right|^4}{2^{\ell + 2\lfloor \log_2 \mathcal{R}_{\mathrm{BJMM}} \rfloor}}\right\},$$

where $|L_{1,1}| = \binom{(k+\ell)/2}{p/4+\varepsilon}$.

For $\varepsilon = 0$, we recover the MMT algorithm. In fact, the authors in Ref. [156] propose to construct trees of depth higher than 2, merging on each level to 0 on the appropriate number of coordinates, thus removing the representations. We shall not describe this extension here, but note that this depth is yet another parameter to be optimised and the optimal value differs for concrete ISD parameters.

Quantum walks for the list matching problem: At the heart of the above ISD algorithms (except Prange's) is the search for tuples of vectors from given lists, where a good tuple should satisfy a certain relation. This task can be generalised to the list matching problem (also known as the k-list problem [157], but we decided to remove k not to be confused with the code dimension).

Definition 22 (List matching problem). *Let m be fixed. Given m equal sized lists $L_1, \ldots, L_m$ of binary vectors and a function g that decides whether an m-tuple $(\mathbf{v}_1, \ldots, \mathbf{v}_m) \in L_1 \times \cdots \times L_m$ forms a 'match' or not (outputs 1 in case of a 'match'), finds (almost) all m-tuples $(\mathbf{v}_1, \ldots, \mathbf{v}_m) \in L_1 \times \cdots \times L_m$ s.t. $g(\mathbf{v}_1, \ldots, \mathbf{v}_m) = 1$.*

We have already met some instances of this problem: Stern's algorithm is an example for $m = 2$, where g decides for a 'match' whenever a pair $(\mathbf{v}_1, \mathbf{v}_2) \in L_1 \times L_2$ is equal on certain fixed ℓ coordinates. In representation-based algorithms such as Refs. [146, 156] there are (at least) 4 lists $L_1, \ldots, L_4$, and function g decides for the match if $\mathbf{v}_1 + \mathbf{v}_2$, $\mathbf{v}_3 + \mathbf{v}_4$ are equal on certain coordinates (first merge) and, in addition, if $\mathbf{v}_1 + \mathbf{v}_2 + \mathbf{v}_3 + \mathbf{v}_4$ is 0 on the designated ℓ coordinates (second merge). In decoding, g additionally checks if the weight of the sum is correct.

A special version of the list matching problem, where $g = 1 \Leftrightarrow \mathbf{v}_1 = ,\ldots, = \mathbf{v}_m$, called the Element distinctness problem, has its history in quantum computing [158] since it serves as an illustrative application of quantum random walk techniques. Let us specialise this technique to the ISD setting.

The setup phase of the walk consists in preparing a uniform superposition over all r-size subsets (optimal value for r will be discussed later) $S_i \subset L_i$ together with an auxiliary register $|Aux\rangle$ (normalisation omitted): $\sum_{S_i \subset L_i,\ |S_i| = r} |S_1\rangle \otimes \ldots \otimes |S_m\rangle \otimes |Aux\rangle$.

The auxiliary register $|Aux\rangle$ contains information needed to decide whether $S_1, \ldots, S_m$ contains a match. In the ISD setting, $|Aux\rangle$ stores intermediate and output lists of the matching process. For example, in Stern's algorithm $(m = 2)$ $|Aux\rangle$ contains all pairs $(\mathbf{v}_1, \mathbf{v}_2) \in S_1 \times S_2$ that match on ℓ coordinates. In case the merge is done in several steps such as in MMT $(m = 4)$, the intermediate lists are also stored in $|Aux\rangle$. Hence, when we talk about quantum memory of an ISD algorithm in this section, we mean the size of the $|Aux\rangle$ register.

The running time and the space complexities of the Setup phase are essentially the running time and the space complexities of the corresponding classical ISD algorithm with the input lists of size $|S_i| = r$ instead of $|L_i|$.

The **Setup** phase finishes with a superposition over all r-sublists $S_1, \ldots, S_m$ of $L_1, \ldots, L_k$, where each $(S_1, \ldots, S_m)$ is entangled with the register $|Aux\rangle$ that contains the result of merging $(S_1, \ldots, S_m)$ into the final output list S_{output}.

The **Update** phase consists in choosing a sublist S_i and replacing one of its element $\mathbf{v}_i \in S_i$ by $\mathbf{v}'_i \notin S_i$. This is one step of a walk on the Johnson graph, see Section 3.3 for the relevant definitions. We update the data stored in $|Aux\rangle$: Remove all the pairs in the merged lists that involve $\mathbf{v}_i$ and create possibly new matches with $\mathbf{v}'_i$. We assume that throughout the walk the sublists S_i's are kept sorted and stored in a data-structure that allows fast insertions/removals (e.g. radix trees as proposed in Ref. [159]). We also assume that elements in $S_1, \ldots, S_m$ that result in a match store pointers to their match to be able to quickly update the output of the checking function.

After we have performed $\Theta(1/\sqrt{\delta})$ updates, where δ is the eigenvalue gap of the Johnson graph $J(N, r)$, we check if the updated register $|S_1\rangle \otimes \ldots \otimes |S_m\rangle \otimes |Aux\rangle$ gives a match. We give a lower bound on δ below. This is the **Checking** phase of the walk.

Thanks to the quantum walk framework described in Section 3.3, once we know the costs of the Setup phase T_{S}, the Update phase T_{U}, and the Checking phase T_{C}, we know that after T_{QW} many steps, we measure a register $|S_1\rangle \otimes \ldots \otimes |S_k\rangle \otimes |Aux\rangle$ that contains the correct error-vector with overwhelming probability, where

$$T_{\mathrm{QW}} = T_{\mathrm{S}} + \frac{1}{\sqrt{\varepsilon}}\left(\frac{1}{\sqrt{\delta}} \cdot T_{\mathrm{U}} + T_{\mathrm{C}}\right), \tag{7}$$

where ε is a fraction of vertices in $J(N, r)$ that contain the correct error vector. For a fixed m, we have $\varepsilon \approx r^m/N^m$ where $N = |L_1|$.

Strictly speaking, the walk we have just described is a walk on an m-Cartesian product of Johnson graphs—one for each sublist S_i, so the value δ in Equation (7) must be the eigenvalue gap for such a Cartesian product graph. As shown in [Ref. 151, Theorem 2], for fixed constant m, it is bounded from below by $\frac{N}{m \cdot r(N-r)}$.

Quantum walks speed-ups for high-memory ISD algorithms: The quantum walk search algorithm described above solves the ISD problem provided we have found a permutation π that gives the desired distribution of 1's in the error-vector. Kachigar and Tillich in Ref. [151] suggest to run Grover's algorithm for π with the 'checking' function for Grover's search being the quantum walk routine for the List matching problem. In particular, we operate on the quantum state (normalisation omitted):

$$\sum_i |\pi_i\rangle|\pi_i(H)\rangle$$

$$\underbrace{\otimes\left[\sum_{S_i \subset L_i,\ |S_i|=r} |S_1\rangle \otimes \ldots \otimes |S_k\rangle \otimes |Aux\rangle\right]}_{\text{Quantum Walk = Check for the outer Grover}}$$

$$\otimes|\text{ Is } \pi \text{ good?}\rangle.$$

The outer summation is performed over a set of all permutations from $\pi_i \in S_n$. Let N denote the total number of permutations and M—the number of marked permutations, that is, those give the desired weight distribution of the error. If we can check that π is a marked permutation, then after $O\left(\sqrt{\frac{N}{M}}\right)$ Grover steps, the state is (almost) equal to the superposition over the marked π's. For example, for the MMT algorithm, we have $O\left(\sqrt{\frac{N}{M}}\right) = O\left(\frac{1}{\sqrt{P_{\text{MMT}}}}\right)$, where P_{MMT} is as in Equation (6). The check if a permutation π is good is realised via quantum walk search for m vectors $\mathbf{v}_1, \ldots, \mathbf{v}_m \in S_1 \times \cdots \times S_m$ that match on certain coordinates and lead to the correct error vector. Note an important difference between classical and quantum settings: during the quantum walk we search over size-r sublists $S_i \subset L_i$, which are exponentially shorter than L_i. After T_{QW} steps, the register $|Aux\rangle$ contains an m-tuple $(\mathbf{v}_1, \ldots, \mathbf{v}_m)$ that leads to the correct error vector provided a permutation π is good. Hence, after $\sqrt{\#\text{permutations}} \cdot T_{\text{QW}}$ steps, the measurement of the first register gives a good π with constant success probability. The resulting state will be entangled with registers that store $S_1, \ldots, S_m$ together with the pointers to the matching elements. Once we measure $S_1, \ldots, S_m$, we retrieve these pointers and, finally, reconstruct the error vector as in the classical case.

With all the above, we are ready to analyse quantum versions of ISD algorithms that use the list matching problem as a subroutine. For Stern's algorithm, the quantum random walk will have the following complexities:

- $T_{\text{S}} = \max\{r, r^2/2^\ell\}$, where the max{} is taken between time to construct r-size sublists S_1, S_2 and the expected output size of pairs from these sublists that match on ℓ coordinates;
- $T_{\text{U}} = r/2^\ell$, which is the expected number of elements we will need to update if we change one element in S_1;
- $T_{\text{C}} = \log r$.

Assuming $r^2/2^\ell \geq r$, the value for T_{QW} from Eq. (7) is minimised when $r = |L_1|^{2/3}$, where $|L_1| = \binom{k+\ell}{p}$ as in the classical Stern. Including the outer Grover search for a good permutation, the overall complexity of quantum Stern becomes

$$T^{\text{Q}}_{\texttt{Stern}} = \sqrt{\frac{\binom{n}{w}}{\binom{k+\ell}{p}\binom{n-k-\ell}{w-p}}} \cdot |L_1|^{2/3}.$$

During the quantum walk we store the r-size sublists on quantum registers, so the quantum space complexity of Stern's algorithm is $S^{\text{Q}}_{\texttt{Stern}} = |L_1|^{2/3}$.

Similar arguments apply to the ISD algorithms that use the representation technique. Let us consider the quantum walk complexities of the BJMM algorithm, since MMT can be viewed as a special case of BJMM. Similar to Stern, we have

- $T_{\text{S}} = \max\{r, r^2/2^{\lfloor \log_2 \mathcal{R}_{\text{BJMM}} \rfloor}, r^4/2^{\ell - 2\lfloor \log_2 \mathcal{R}_{\text{BJMM}} \rfloor}\}$,
- $T_{\text{U}} = \max\{r/2^{\lfloor \log_2 \mathcal{R}_{\text{BJMM}} \rfloor}, r^2/2^{\lfloor \ell - 2\log_2 \mathcal{R}_{\text{BJMM}} \rfloor}\}$,
- $T_{\text{C}} = \log r$.

Assuming the second level dominates the list construction, that is, the max in T_{S} is achieved by $r^2/2^{\lfloor \log_2 \mathcal{R}_{\text{BJMM}} \rfloor}$ and by $r/2^{\lfloor \log_2 \mathcal{R}_{\text{BJMM}} \rfloor}$ in T_{U}, T_{QW} from Eq. (7) is minimised when $r = |L_{1,1}|^{4/5}$, where $|L_{1,1}| = \binom{(k+\ell)/2}{p/4+\varepsilon}$ as in the classical BJMM. Taking into account the square-root speed-up for the number of permutations P_{MMT} given in Eq. (6), we obtain

$$T^{\text{Q}}_{\text{BJMM}} = \sqrt{\frac{\binom{n}{w}}{\binom{(k+\ell)/2}{p/2}^2\binom{n-k-\ell}{w-p}}} \cdot |L_{1,1}|^{4/5}.$$

The quantum space complexity of BJMM is $S^{\text{Q}}_{\text{BJMM}} = |L_{1,1}|^{4/5}$.

9.3 | Open problems

Despite a lot of effort that has been put into lowering the complexity of ISD over the last 50 years, the picture is far from satisfactory. Essentially all the speed-ups we know and have described here are asymptotically applicable to the dense error regime $w = \Theta(n)$, which is less cryptographically relevant. It was shown in Ref. [160] that the improvements starting from Stern until the very recent ones [147, 161] vanish when $w = o$

(n) for n going to infinity. So, for sparse errors, we are in the strange situation when asymptotically the best known attack remains Prange's algorithm. Hence,

Question 1 can we improve over-Prange's algorithm for sparse errors both classically and quantumly?

One might argue that perhaps asymptotic complexity is not the only metric one should care about when evaluating security levels for specific ISD settings. And this is certainly true: Hidden low-order terms may have serious impact on concrete hardness. The fact that the fastest known algorithms are memory intensive makes the actual costs even less predictable from the asymptotics. Hence,

Question 2 what are the precise (classical and quantum) costs of solving ISD for concrete parameters?

First steps in this direction in the classical setting have been done in Ref. [162], where the authors provide an estimator for concrete ISD parameters. In the quantum setting it seems to be a much harder task to give any meaningful statement about the concrete costs as one would need to analyse in details the complexity of implementing quantum walks.

Instead of analysing the general ISD problem, one can turn their attention to actual cryptographic schemes. The assumption that all known code-based constructions use is that the structure hidden in the parity-check matrix H does not impact security. While we are not aware of any attack that exploits either the structure of the Goppa code in McEliece KEM or the cyclic structure in BIKE KEM, it is still natural to ask

Question 3 can we speed up ISD routines using the knowledge that the parity-check matrix H is not chosen uniformly at random?

Continuing analysing concrete schemes, the hardness of ISD over $\mathbb{F}_3$ has only recently started drawing attention. To gain our confidence in the security of the code-based signature WAVE, the decoding problem over $\mathbb{F}_3$ for dense error requires more investigations.

10 | ISOGENIES

An elliptic curve E defined over a finite field $\mathbb{F}_q$ of characteristic $p \neq 2, 3$ is a projective algebraic curve with an affine plane model given by an equation of the form $y^2 = x^3 + ax + b$, where $a, b \in \mathbb{F}_q$ and $4a^3 + 27b^2 \neq 0$. The set of points of an elliptic curve is equipped with an additive group law. Details about the arithmetic of elliptic curves can be found in many references, such as [Ref. [163], Chap. 3]. Isogenies between elliptic curves are non-zero maps that are given by rational functions, and that are group homomorphisms. The main problem we are interested in is:

Problem 7 (Main isogeny problem). *Given elliptic curves E, E' defined over a finite field $\mathbb{F}_q$, find an isogeny between E and E'.*

Problem 7 is deliberately phrased in an ambiguous way. First, we do not specify the representation of the solution $\varphi: E \to E'$ required. Indeed, the natural representation of an isogeny is via polynomials $f, g, h \in \mathbb{F}_q[x]$ such that

$$\varphi((x,y)) = \left(\frac{f}{h^2}(x), y \cdot \frac{g}{h^3}(x)\right).$$

The *degree* of φ is its degree as a rational function. Therefore, such a representation cannot be efficient when the degree is large. On the other hand, we can decide to accept a representation as a composition of small degree isogenies. Indeed, the degree is multiplicative: $\deg(\varphi \circ \psi) = \deg(\varphi)\deg(\psi)$; therefore, it might be possible to represent a degree-2^n as a composition of n degree-2 isogenies, thus trading an exponential representation for a polynomial-sized one. Another aspect of Problem 7 that could be further specified is whether the existence of an isogeny of small degree between E and E' is known. Indeed, depending on the properties of the curves, generic bounds are known on the existence of an isogeny of bounded degree between the input curves. However, certain popular cryptosystems deliberately choose curves with an unusually low degree isogeny between them. This naturally influences the performances of the computational methods to find the secret isogeny.

In this section, we avoid presenting extensive background on isogenies in order to focus on the properties that are relevant to the state-of-the-art quantum algorithms. We need however to introduce the following important fact. The search for an isogeny between two curves can be reduced to the search for a possible kernel of a map. Indeed, an isogeny $\varphi : E \to E'$ is always the composition $\varphi = \alpha \circ \varphi'$ of a purely inseparable isogeny α and an inseparable isogeny φ'. Purely inseparable isogenies are the composition of the Frobenius endomorphism π: $(x, y) \mapsto (x^p, y^p)$ where $p = \mathrm{char}(q)$ and an isomorphism. In particular, $\ker(\alpha) = \{\infty\}$ is trivial, and the kernel of α' determines φ' uniquely up to isomorphisms. There is an explicit way to construct a separable isogeny from the points of its kernel using Vélu's formulas [164]. All the algorithms we present in this section focus on computing separable isogenies.

10.1 | Isogenies arising as group actions

We begin our overview of quantum algorithms for solving the isogeny problem with the most structured one, namely when we know the action of a finite abelian group G on isomorphism classes of elliptic curves. In this context, we assume that given $g \in G$, such that $g * \overline{E} = \overline{E'}$ (where $\overline{E}$ denotes the isomorphism class of E, i.e. all curves isomorphic to E), and we can efficiently derive a (separable) isogeny $\varphi : E \to E'$. This is the framework of *hard homogeneous spaces* described by Couveignes [165] and of *cryptographic group actions* [166]. This motivates the formulation of the problem of inverting a group action on isomorphism classes of elliptic curves:

Problem 8 (Group action on curves). *Let G be an abelian group acting faithfully and transitively on a set X of isomorphism classes of elliptic curves defined over $\mathbb{F}_q$, and $\overline{E}, \overline{E'} \in X$. Find $g \in G$ such that $g * \overline{E} = \overline{E'}$.*

The hardness of this problem is the security assumption of multiple isogeny-based schemes [165, 167, 168], including the key exchange mechanism CSIDH [169].

To understand the relevance of Problem 8 to the computation of isogenies, we need to introduce the distinction between *ordinary* and *supersingular* curves. An elliptic curve defined over $\mathbb{F}_q$ of characteristic p is said to be supersingular if $p | E(\mathbb{F}_q) - q - 1$ where $E(\mathbb{F})$ denotes the group of points of E defined over the field $\mathbb{F}$. If an elliptic curve is not supersingular, then it is ordinary. The ring of endomorphisms End(E) of an ordinary elliptic curve is an order $\mathcal{O}$ within the quadratic field $\mathbb{Q}(\pi)$ where π is the Frobenius endomorphism. The embedding of π into a quadratic number field is done by noticing that it satisfies the equation $\pi^2 - t\pi + q = 0$ where $t = q + 1 - E(\mathbb{F}_q)$ is the *trace* of E. Then the *ideal class group* $\mathrm{Cl}(\mathcal{O})$ of $\mathcal{O}$ acts on isomorphism classes of elliptic curves E such that $\mathrm{End}(E) \simeq \mathcal{O}$. More specifically, the class $[\mathfrak{a}]$ of an ideal $\mathfrak{a} \subseteq \mathcal{O}$ acts on E up to isomorphisms via a separable isogeny of kernel $E[\mathfrak{a}] := \{P \in E | \forall \alpha \in \mathfrak{a} : \alpha(P) = 0\}$ and degree $\mathcal{N}(\mathfrak{a})$, the algebraic norm of $\mathfrak{a}$. An introduction to number fields, orders, ideal and ideal class groups can be found in standard textbooks of algebraic number theory, including [Ref. [170], Chap. 1]. From a high level standpoint, the takeaway is that the group $G = \mathrm{Cl}(\mathcal{O})$ with size $|G| \in O(\sqrt{q})$ acts faithfully and transitively on isomorphism classes of elliptic curves, and that there is a natural correspondence between an element $g \in G$ such that $g * \overline{E} = \overline{E'}$ and the kernel of an isogeny $\varphi : E \to E'$. Therefore, in ordinary elliptic curves, Problem 7 reduces to Problem 8.

The group action framework also applies to certain supersingular elliptic curves. Typically, by $\overline{E}$, we mean the isomorphism class of E for isomorphisms defined over $\overline{\mathbb{F}_q}$. In this case, there is always a curve $E' \in \overline{E}$ defined over $\mathbb{F}_{p^2}$. It might also be possible that $\overline{E}$ admits a representative E'' that is defined over $\mathbb{F}_p$. In any case, End(E) is isomorphic to an order in a quaternion algebra, which is a 4-dimensional non-commutative ring. When E is defined over $\mathbb{F}_p$, the ring $\mathrm{End}(E)_{\mathbb{F}_p}$ of endomorphisms of E that are defined over $\mathbb{F}_p$ is isomorphic to an order $\mathcal{O}$ in the quadratic number field $K(\sqrt{-p})$. In this case, $\mathrm{Cl}(\mathcal{O})$ acts faithfully and transitively on classes of $\mathbb{F}_p$-isomorphisms of curves defined over $\mathbb{F}_p$ (we denote the class of $\mathbb{F}_p$-isomorphisms of E by $\overline{E}^p$). As above, the action of (the class of) an ideal $\mathfrak{a} \subseteq \mathcal{O}$ is through a separable isogeny of degree $\mathcal{N}(\mathfrak{a})$. Therefore, when E, E' are defined over $\mathbb{F}_p$, Problem 7 also reduces to Problem 8. In this case, $G = \mathrm{Cl}(\mathcal{O})$ has size $|G| \in O(\sqrt{p})$, and as before, the knowledge of g such that $g * \overline{E}^p = \overline{E'}^p$ yields an isogeny $\varphi : E \to \mathrm{E}'$. This framework can be partially extended to supersingular curves defined over $\mathbb{F}_{p^2}$ when an orientation is known, that is, an injective morphism $\iota : \mathcal{O} \to \mathrm{End}(E)$ where $\mathcal{O}$ is an imaginary quadratic order [[171], Def. 2].

Problem 8 subsequently reduces to the dihedral hidden subgroup problem [172] for which we presented an algorithm in Section 4.3. Assume we are looking for $\mathfrak{a}$ such that $[\mathfrak{a}] * \overline{E}_1 = \overline{E}_2$. Let $A = \mathbb{Z}/d_1\mathbb{Z} \times \cdots \times \mathbb{Z}/d_k\mathbb{Z} \simeq \mathrm{Cl}(\mathcal{O})$ be the elementary decomposition of $\mathrm{Cl}(\mathcal{O})$. Then we define a quantum oracle $f : \mathbb{Z}/2\mathbb{Z} \ltimes A \to \{\text{quantum states}\}$ by

$$f(x, \mathbf{y}) := \begin{cases} \left|[\mathfrak{a}_{\mathbf{y}}] * \overline{E}_1\right\rangle & \text{if } x = 0, \\ \left|[\mathfrak{a}_{-\mathbf{y}}] * \overline{E}_2\right\rangle & \text{if } x = 1, \end{cases} \tag{8}$$

where $[\mathfrak{a}_{\mathbf{y}}]$ is the element of $\mathrm{Cl}(\mathcal{O})$ corresponding to $\mathbf{y} \in A$ via the isomorphism $\mathrm{Cl}(\mathcal{O}) \simeq A$. Let H be the subgroup of $\mathbb{Z}/2\mathbb{Z} \ltimes A$ of the periods of f. This means that $f(x, \mathbf{y}) = f(x', \mathbf{y}')$ if and only if $(x, \mathbf{y}) - (x', \mathbf{y}') \in H$. Then since the action is transitive, we have $H = \{(0, \mathbf{0}), (1, \mathbf{s})\}$ where $\mathbf{s} \in$ A such that $[\mathfrak{a}_{\mathbf{s}}] * \overline{E}_1 = \overline{E}_2$. The computation of $\mathbf{s}$ can thus be done through the resolution of the Hidden Subgroup Problem in $\mathbb{Z}/2\mathbb{Z} \ltimes A$.

Proposition 23 *There is a quantum algorithm that solves Problem 8 in quantum time $2^{\tilde{O}\left(\sqrt{\log(|G|)}\right)}$ using a polynomial amount of memory.*

This yields solutions to the isogeny problem in time $2^{\tilde{O}(\sqrt{q})}$ between ordinary elliptic curves with the same endomorphism ring and in time $2^{\tilde{O}(\sqrt{p})}$ between two supersingular curves defined over $\mathbb{F}_p$. A non-asymptotic analysis of these algorithms and of the cost of the attack against the initial parameters of the scheme CSIDH can be found in Refs. [173, 174]. In another recent work [175], safer parameter sizes were proposed.

10.2 | Memoryless algorithms for small-degree isogenies

We now assume that no group action is known. Otherwise, the best known algorithms for solving the isogeny problem are always those of the previous section. This means that we are considering the case of supersingular curves defined over $\mathbb{F}_{p^2}$. In this section, we focus on the sub-case where we know the existence of a bounded-degree isogeny between the two input curves. This is the case for the prominent isogeny-based cryptographic scheme SIDH [176] which resulted in the SIKE submission [177] of NIST standardisation of post-quantum KEM protocols. This leads us to the formulation of the following problem:

Problem 9 (Small-degree isogeny). *Set E, E' be two elliptic curves over $\mathbb{F}_q$ and ℓ be a prime. Suppose that there is a degree ℓ^k-separable isogeny φ: $E \to E'$ for some k. Find φ.*

In the SIKE system, $\ell = 2$ or 3, and $k = \frac{1}{2}\log(p)$. These are the typical parameters of interest. The secret isogeny could be viewed as a walk in the ℓ-isogeny graph, which is an $\ell + 1$-regular graph where nodes are isomorphism classes of elliptic curves, and there is an edge between $\overline{E_1}$ and $\overline{E_2}$ if there is a degree-ℓ isogeny $\varphi : E_1' \to E_2'$ for some $E_1' \in \overline{E_1}$ and $E_2' \in \overline{E_2}$. The choice of a non-backtracking walk of length k originating from $\overline{E}$ in the ℓ-isogeny graph can be mapped to a bit string in $\{0,1\}^{\lceil k \log_2 \ell \rceil}$. Such a choice represents the choice of one of the possible kernels. To efficiently compute the map corresponding to a given bit string, we follow the approach of Ref. [176, Sec. 4.2.2]. In a nutshell, it consists in identifying a cyclic kernel $\langle R \rangle \subseteq E[\ell^k] = \{P \in E | [\ell^k]P = 0\}$ and computing the corresponding isogeny with Vélu's formulas. Instead of directly applying the formulas using all the points of $\langle R \rangle$, we rather only compute ℓ-isogenies. By interleaving ℓ-isogeny computation/evaluations and multiplications of the points defining the kernel by ℓ, we can compute an ℓ^k isogeneous curve E'' in time O($k \log k \log q$) operations. Then we need to decide whether $E'' \in \overline{E'}$. If that is the case, then we have solved our problem. To decide if two curves are isomorphic, we compute their so-called *j-invariant*, which is a value of $\mathbb{F}_q$ for which we have a closed formula from the coefficient of the curve. If these match, the curves are isomorphic (i.e. the j invariants identify isomorphism classes of curves).

The natural strategy is therefore to perform a quantum search on all possible strings in $\{0,1\}^{\lceil k \log_2 \ell \rceil}$ that encode a length-k walk in the ℓ-isogeny graph until one of them yields a curve whose j-invariant matches that of E'. This can be done by using Grover's search with oracle

$$\mathcal{O}_f : \begin{array}{ccc} \{0,1\}^{\lceil k \log_2 \ell \rceil} & \to & \{0,1\} \\ s & \mapsto & 1 \text{ if } \varphi_s : E \to E_s \\ & & \text{with } j(E_s) = j(E'), \end{array}$$

where φ_s is the degree-ℓ^k isogeny obtained from s by the procedure described above.

Proposition 24 *Using Grover's search algorithm there is an algorithm to solve Problem 9 in quantum time O($\ell^{k/2} k \log k \log q$) with a polynomial amount of memory.*

Interestingly, the above approach is the best quantum attack against SIDH/SIKE with low memory. Marginal improvements are known (to improve on the polynomial factors). For example, it is possible to reduce the length of the walk by pre-computing all k'-length paths from E' and storing all the corresponding j-invariants in the circuit of the oracle which makes a comparison between the j-invariant obtained after a walk of length $k - k'$ originating from E and all the pre-computed ones [178].

10.3 | Large-memory algorithms for small-degree isogenies

The search for a path in the ℓ-isogeny graph between $\overline{E}$ and $\overline{E'}$ can be done with a *meet-in-the-middle* strategy. Indeed, finding a path of length k from $\overline{E}$ to $\overline{E'}$ can be done by fixing $1 < k' < k$ and finding two paths: the one of length k' originating from $\overline{E}$ and the other of length $k - k'$ originating from $\overline{E'}$ that land in the same isomorphism class, that is, on curves that share the same j-invariant. Typically, $k' = k - k' = k/2$ (i.e. the meeting point is really in the middle), but it is not fundamentally required. Once the two paths are found, we have a path from $\overline{E}$ to $\overline{E'}$ (note that we know how to backtrack the length $k - k'$ path all the way to $\overline{E'}$ using dual isogenies). This process fits the framework of *claw finding* which, given $f : X - \{0,1\}^{k_1} \to Z$ and $g : X = \{0,1\}^{k_2} \to Z$ for some set Z, consists in looking for x, y such that $f(x) = g(y)$. Such a pair is called a claw. In the context of isogenies of small degree (parameterised by k), we assume that the claw is unique.

To solve the claw finding problem with optimal circuit depth (at the cost of increased quantum memory), we can use Tani's algorithm [179]. It consists in a quantum random walk in the product of the two Johnson graphs, $J_f := J(|X|, (|X||Y|)^{1/3})$ and $J_g := J(|Y|, (|X||Y|)^{1/3})$ (provided that $|Y| \le |X|^2$, which we will assume since our target case is $|X| = |Y|$). A vertex $(F, G) \in J_f \times J_g$ is marked if there is a pair $(x, y) \in F \times G$ such that $f(x) = f(y)$ (i.e. $F \times G$ contains a claw). To check if (F, G) is a marked vertex, we sort all elements of $F \cup G$ with respect to their function value (for us the function values are j-invariants in $Z = \mathbb{F}_{p^2}$). Therefore, Cost(Setup) is the cost of evaluating f and g on $X \cup Y$ and then sorting all the elements. However, Cost (Update) only consists in the deletion of one element and the insertion of a new element on an already sorted list, which is efficient. Note that the memory used by this algorithm is QRAQM (due to the quantum walk framework).

Proposition 25 (Tani). *The claw finding problem can be solved by a quantum algorithm using a circuit* $\tilde{O}\left((|X||Y|)^{1/3}\right)$ *calls to the comparison oracle and with* O($(|X||Y|)^{1/3}$) *quantum memory.*

When searching for a degree-ℓ^k isogeny between representatives of $\overline{E}, \overline{E'}$, we can choose $|X| = |Y| = \ell^{k/2}$, which yields a circuit with depth and width in $\tilde{O}\left(\ell^{k/3}\right)$ (to simplify notations, we incorporate the cost of the comparison oracle in $\tilde{O}$). In the case of SIDH/SIKE, $\ell = 2$ and $k = \frac{1}{2}\log(p)$, which yields depth and width in $\tilde{O}\left(p^{1/6}\right)$, but a Depth $\times$ Width cost of $\tilde{O}\left(p^{1/3}\right)$, which is worse than a Grover search that achieves $\tilde{O}\left(p^{1/4}\right)$.

There exists a natural time-memory tradeoff, which interpolates between Tani's algorithm and Grover search: One simply sets the vertex size of the Johnson graphs to a given parameter $R \le (|X||Y|)^{1/3}$, so that only R elements of X and R elements of Y are stored. This reduces the probability to be marked to $\frac{R^2}{|X||Y|}$ and the time complexity changes to $\tilde{O}\left(R + \sqrt{\frac{|X||Y|}{R}}\right)$. The result of Proposition 25 is simply the point $R = (|X||Y|)^{1/3}$, which balances the setup and walk steps and achieves the minimum complexity. Storing a single pair of elements in the vertex ($R = 1$) reduces this algorithm to a Grover search on all the pairs, corresponding to Proposition 24: the complexity becomes $\tilde{O}(\sqrt{|X||Y|})$.

More tradeoffs between time, depth and width can be done, and optimisations of the resulting Depth $\times$ Width and Gate costs (thus without QRAM) have been studied in Refs. [16, 180]. In all cases, due to the techniques being used, the quantum time complexity achievable for a given amount of quantum hardware (e.g. memory or processors) achieves at most a square-root improvement on the classical time complexity achievable with the corresponding amount of classical hardware.

10.4 | Algorithm for generic isogenies

Assuming we want to find an isogeny between two supersingular curves E, E' defined over $\mathbb{F}_{p^2}$ without any guarantee of existence of a short degree map between them, there is a non-trivial method to achieve a speed-up over Grover's search without incurring the memory costs of Tani's algorithm. The generic isogeny problem is less relevant for cryptography than Problems 8 and 9, which gave rise to the most popular isogeny-based cryptosystems, but it enables the search for collisions for the Charles–Lauter–Goren hash function [181]. Aside from this, the generic isogeny problem is a fundamental problem for which quantum computers provide a non-trivial speedup.

We combine techniques from Sections 10.1 and 10.2 to compute an isogeny between two given supersingular curves defined over $\mathbb{F}_{p^2}$ without any particular property. The high level strategy we follow was first described by Delfs and Galbraith [182] and then adapted to the quantum setting by Biasse, Jao and Sankar [183]. It consists in searching for an isogeny path between E and a curve E_1 defined over $\mathbb{F}_p$ and for an isogeny path between E' and a curve E_2 defined over $\mathbb{F}_p$. Then in a second stage, we find an isogeny between E_1 and E_2 using the methods of Section 10.1. Altogether this yields an isogeny between E and E'. This approach is illustrated by Figure 10.

The cost of computing an isogeny from E_1 to E_2 is negligible and immediately derives from the analysis presented in Section 10.1. Now, we turn to the computations of isogenies $E \to E_1$ and $E' \to E_2$. There are $O(p)$ isomorphism classes of supersingular curves containing a representative defined over $\mathbb{F}_{p^2}$, among which $\tilde{\Omega}(\sqrt{p})$ have a representative defined over $\mathbb{F}_p$. This means that a fraction $\tilde{\Omega}\left(\frac{1}{\sqrt{p}}\right)$ of the isomorphism classes contain a representative defined over $\mathbb{F}_p$. The ℓ-isogeny graph for a prime $\ell \nmid p$ is a Ramanujan graph [176, Sec. 2]. This property allows us to evaluate the probability that an ℓ-isogeny path of a given length reaches a certain subset of the vertices. If the length is long enough, then the distribution of the end point of the walk is close to uniform at random.

Proposition 26 (Prop 2.1 of Ref. [176]). *Let G be a k-regular graph such that the eigenvalues λ of the non-constant eigenvectors of its adjacency matrix satisfy $|\lambda \leq c$ for some $c < k$. Let $S \subseteq G$ be a subset of vertices and $x \in G$. Then a random walk of length at least $\frac{\log\left((2|S|/|G|)^{1/2}\right)}{\log(k/c)}$ starting from x lands in S with probability at least $\frac{|S|}{2|G||}$.*

$$\begin{tikzcd}[column sep=huge, row sep=huge]
E \arrow[r, dotted, bend left, "\text{Direct computation in } \tilde{O}(p^{1/2})"] \arrow[d, "\text{In } \tilde{O}(p^{1/4})"'] & E' \arrow[d, "\text{In } \tilde{O}(p^{1/4})"] \\
E_1 \text{ defined over } \mathbb{F}_p \arrow[r, bend right, "\text{subexponential complexity}"'] & E_2 \text{ defined over } \mathbb{F}_p
\end{tikzcd}$$

FIGURE 10 Computation of an isogeny from E to E'

The ℓ-isogeny graph is $k = \ell + 1$-regular, and the non-trivial eigenvalues satisfy $|\lambda| \leq c = 2\sqrt{\ell}$. So if we choose $\ell = 3$, G the 3-isogeny graph, and S the set of isomorphism classes with a representative over $\mathbb{F}_p$, we obtain that a random walk of length in $O(\log(p))$ hits a supersingular curve defined over $\mathbb{F}_p$ with probability at least $\tilde{\Omega}\left(\frac{1}{\sqrt{p}}\right)$. Using a Grover search over all 3-isogeny paths of the length given in Proposition 26, we obtain a cost of $\tilde{O}\left(p^{1/4}\right)$. As mentioned before, this cost is the bottleneck of the computation of an isogeny between E and E' since it is asymptotically larger than the cost of the computation of $E_1 \to E_2$.

10.5 | Open problems

In the case of SIDH, there is additional information available to the adversary that the algorithms presented so far in this section do not exploit. Indeed, the security of SIDH relies not only on the difficulty of finding isogenies but also on finding them given some additional information (the evaluation of the isogeny on a torsion subgroup) that is necessary for the key exchange.

Problem 10 (SIDH). *Let E be an elliptic curve. Let N_1, N_2 be two smooth coprime integers (powers of 2 and 3 in the case of SIKE). Let K be a cyclic subgroup of order N_1 of E chosen uniformly at random. Let $\phi : E \to E/K$. Given the supersingular elliptic curves E and E/K, given the restriction of ϕ to $E[N_2]$, compute K.*

Question 1 Can we design methods that take advantage of the information of the restriction of ϕ to $E[N_2]$ to solve Problem 10 more efficiently than the best methods to solve Problem 9?[1]

In SIKE, one has $N_1 \simeq N_2$. But the torsion points allow for some attacks in the case of unbalanced or overstretched (N_1, N_2), as seen, for example, in Ref. [184], which includes quantum attacks based on claw-finding. In Ref. [185], the authors showed the following.

[1]An efficient attack against SIDH using the restriction of φ to $E[N_2]$ was discovered by Castryck and Decru after this survey was written [186].

Theorem 27 (Theorem 4.27 in [185]). *If $N_2 > pN_1^4$, then the SIDH problem can be reduced (with additional heuristics) to the abelian hidden shift problem, and solved in quantum subexponential time.*

This is, to the best of our knowledge, the first application of quantum hidden shift algorithms in dedicated cryptanalysis of SIDH, outside the setting where a group action is already given. Another natural angle to extend the scope of quantum hidden shift algorithms would be to take advantage of orientations, which are injective morphisms $\iota : \mathcal{O} \to \mathrm{End}(E)$ where $\mathcal{O}$ is an imaginary quadratic order [[171], Def. 2]. It seems like in the case of two curves sharing the same orientation, the isogeny problem reduces to Problem 8.

Question 2 Can the quantum hidden shift framework be applied to more general classes of supersingular curves defined over $\mathbb{F}_{p^2}$ by using the concept of orientation?

ACKNOWLEDGEMENTS

We are grateful to Martin Ekerå for providing detailed material on Shor's algorithms and on the various recent derivatives of these algorithms. We also thank the anonymous reviewers for their many insightful comments that allowed significant improvements of this work.

Elena Kirshanova is supported by the Young Russian Mathematics scholarship and by the Russian Science Foundation grant N 1-71-00114, https://rscf.ru/project/22-41-04411/.

CONFLICT OF INTEREST

None.

PERMISSION TO REPRODUCE MATERIALS FROM OTHER SOURCES

None.

DATA AVAILABILITY STATEMENT

Data sharing not applicable to this article as no datasets were generated or analysed during the current study.

ORCID

J.-F. Biasse https://orcid.org/0000-0001-8591-8408
E. Kirshanova https://orcid.org/0000-0001-8924-7605

REFERENCES

1. Shor, P.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Comput. 26(5), 1484–1509 (1997). https://doi.org/10.1137/s0097539795293172
2. Rivest, R., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM 21(2), 120–126 (1978). https://doi.org/10.1145/359340.359342
3. NIST.: Fips Pub 186-4: Digital Signature Standard (Dss) (2013)
4. Diffie, W., Helman, M.: New directions in cryptography. IEEE Transactions on Information Society 22(6), 644–654 (1976). https://doi.org/10.1109/tit.1976.1055638
5. Koblitz, N.: Elliptic curve cryptosystems. Math. Comput. 48(177), 203–209 (1987). https://doi.org/10.1090/s0025-5718-1987-0866109-5
6. Miller, V.: Use of elliptic curves in cryptography. In: Williams, H. (ed.) Advances in Cryptology—CRYPTO '85 Proceedings, pp. 417–426. Springer Berlin HeidelbergBerlin (1986)
7. National Institute of Standards and Technology (NIST).: Report on Post-quantum Cryptography. Online document (2016). https://csrc.nist.gov/publications/detail/nistir/8105/final
8. National Security Agency.: NSA Suite B Cryptography. Online document (2015). https://www.nsa.gov/ia/programs/suiteb_cryptography/
9. Cooper, D., et al.: Recommendation for Stateful Hash-Based Signature Schemes (2020)
10. NIST.: Post-Quantum Cryptography: Workshops and Timeline (2021)
11. Grover, L.: A fast quantum mechanical algorithm for database search. In: Miller, G. (ed.) Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996, pp. 212–219. ACM (1996)
12. Nielsen, M., Chuang, I.: Quantum Computation and Quantum Information. Cambridge University Press (2010)
13. Kaye, P., Laflamme, R., Mosca, M.: An Introduction to Quantum Computing. Oxford University Press (2006)
14. Bennett, C.: Time/space trade-offs for reversible computation. SIAM J. Comput. 18(4), 766–776 (1989). https://doi.org/10.1137/0218053
15. Fowler, A., et al.: Surface codes: towards practical large-scale quantum computation. Phys. Rev. 86(3), 032324 (2012). https://doi.org/10.1103/physreva.86.032324
16. Jaques, S., Schanck, J.: Quantum cryptanalysis in the RAM model: claw-finding attacks on SIKE. In: CRYPTO (1), volume 11692 of Lecture Notes in Computer Science, pp. 32–61. Springer (2019)
17. Gouzien, E., Sangouard, N.: Factoring 2048-bit rsa integers in 177 days with 13 436 qubits and a multimode memory. Phys. Rev. Lett. 127(14), 140503–2021. https://doi.org/10.1103/physrevlett.127.140503
18. Kuperberg, G.: Another subexponential-time quantum algorithm for the dihedral hidden subgroup problem. In: Severini, S., Brandão, F. (eds.) 8th Conference on the Theory of Quantum Computation, Communication and Cryptography, TQC 2013, May 21-23, 2013, Guelph, Canada, volume 22 of LIPIcs, pp. 20–34. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2013)
19. Kimmel, S., Lin, Y.-Y., Lin, H.-H.: Oracles with costs. In: Beigi, S., König, R. (eds.) 10th Conference on the Theory of Quantum Computation, Communication and Cryptography, TQC 2015, May 20-22, 2015, Brussels, Belgium, volume 44 of LIPIcs, pp. 1–26. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2015)
20. Magniez, F., et al.: Search via quantum walk. SIAM J. Comput. 40(1), 142–164 (2011). https://doi.org/10.1137/090745854
21. Montanaro, A.: Quantum-walk speedup of backtracking algorithms. Theor. Comput. 14(15), 1–24 (2018). https://doi.org/10.4086/toc.2018.v014a015
22. Kuperberg, G.: A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. SIAM J. Comput. 35(1), 170–188 (2005). https://doi.org/10.1137/s0097539703436345
23. Regev, O.: A Subexponential Time Algorithm for the Dihedral Hidden Subgroup Problem with Polynomial Space. arXiv:quant-ph/0406151
24. Shor, P.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35th Annual Symposium on Foundations of Computer Science, pp. 124–134 (1994)
25. Riemann's, G.M.: Hypothesis and tests for primality. J. Comput. Syst. Sci. 13(3), 300–317 (1976)
26. Rabin, M.: Probabilistic algorithm for testing primality. J. Number Theor. 12(1), 128–138 (1980). https://doi.org/10.1016/0022-314x(80)90084-0
27. Long, D.: Random equivalence of factorization and computation of orders. In: Technical Report (Undated, seemingly Unpublished), pp. X–XX. Princeton University
28. Ekerå, M.: On completely factoring any integer efficiently in a single run of an order-finding algorithm. Quant. Inf. Process. 20(6), 205 (2021). https://doi.org/10.1007/s11128-021-03069-1
29. Ekerå, M.: On the Success Probability of Quantum Order Finding (2022)

30. Ekerå, M.: On post-processing in the quantum algorithm for computing short discrete logarithms. Des. Codes Cryptogr. 88(11), 2313–2335 (2020). https://doi.org/10.1007/s10623-020-00783-2
31. Hardy, G., Wright, E.: An Introduction to the Theory of Numbers, ed. Clarendon PressOxford (1975)
32. Ekerå, M.: On the Success Probability of Quantum Order Finding. (In preparation. To appear on ArXiv.) (2021)
33. Ekerå, M.: Quantum algorithms for computing general discrete logarithms and orders with tradeoffs. J. Math. Cryptol. 15(1), 359–407 (2021). https://doi.org/10.1515/jmc-2020-0006
34. Griffiths, R., Niu, C.-S.: Semiclassical Fourier transform for quantum computation. Phys. Rev. Lett. 76(17), 3228–3231 (1996). https://doi.org/10.1103/physrevlett.76.3228
35. Mosca, M., Ekert, A.: The hidden subgroup problem and eigenvalue estimation on a quantum computer. In: Williams, C. (ed.) Quantum Computing and Quantum Communications, pp. 174–188. Springer Berlin HeidelbergBerlin (1999)
36. Parker, S., Plenio, M.: Efficient factorization with a single pure qubit and $\mathrm\{log\}\mathit\{n\}$ mixed qubits. Phys. Rev. Lett. 85(14), 3049–3052 (2000). https://doi.org/10.1103/physrevlett.85.3049
37. Seifert, J.-P.: Using fewer qubits in shor's factorization algorithm via simultaneous diophantine approximation. In: Naccache, D. (ed.) Topics in Cryptology —CT-RSA 2001, pp. 319–327. Springer Berlin HeidelbergBerlin (2001)
38. Ekerå, M., Håstad, J.: Quantum algorithms for computing short discrete logarithms and factoring rsa integers. In: Lange, T., Takagi, T. (eds.) Post-quantum Cryptography, pp. 347–363. Springer International PublishingCham (2017)
39. Barker, E., et al.: Recommendation for Pair-wise Key-Establishment Schemes Using Discrete Logarithm Cryptography. National Institute of Standards and Technology (2018)
40. Gillmor, D.: RFC 7919: Negotiated Finite Field Diffie–Hellman Ephemeral Parameters for Transport Layer Security. TLS (2016)
41. Kivinen, T., Kojo, M.: RFC 3526: More Modular Exponentiation (MODP) Diffie–Hellman Groups for Internet Key Exchange (2003)
42. Proos, J., Zalka, C.: Shor's discrete logarithm quantum algorithm for elliptic curves. Quant. Inf. Comput. 3(4), 317–344 (2003). https://doi.org/10.26421/qic3.4-3
43. Ekerå, M.: Revisiting Shor's Quantum Algorithm for Computing General Discrete Logarithms (1905). ArXiv.09084
44. Kaliski, B.: A Quantum "Magic Box" for the Discrete Logarithm Problem. IACR ePrint 2017/745
45. Blum, M., Micali, S.: How to generate cryptographically strong sequences of pseudorandom bits. SIAM J. Comput. 13(4), 850–864 (2021/11/11 1984). https://doi.org/10.1137/0213053
46. Gidney, C., Ekerå, M.: How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits. Quantum 5, 433 (2021). https://doi.org/10.22331/q-2021-04-15-433
47. Häner, T., et al.: Improved quantum circuits for elliptic curve discrete logarithms. In: PQCrypto 2020, pp. 425–444 (2020)
48. Roetteler, M., et al.: Quantum resource estimates for computing elliptic curve discrete logarithms. In: Proc. ASIACRYPT 2017, pp. 241–270. Springer (2017). Lecture Notes in Computer Science
49. Webber, M., et al.: The Impact of Hardware Specifications on Reaching Quantum Advantage in the Fault Tolerant Regime (2021). ArXiv 2108.12371
50. Gheorghiu, V., Mosca, M.: Benchmarking the Quantum Cryptanalysis of Symmetric, Public-Key and Hash-Based Cryptographic Schemes (2019). ArXiv 1902.02332
51. Mosca, M., Piani, M.: Quantum Threat Timeline Report 2020 (2021)
52. Grumbling, E., Horowitz, M. (eds.) Quantum Computing: Progress and Prospects. The National Academies PressWashington (2019)
53. Willhelm, F., et al.: Status of Quantum Computer Development (2020)
54. Hallgren, S.: Polynomial-time quantum algorithms for Pell's equation and the principal ideal problem. J. ACM 54(1), 4–19 (2007). https://doi.org/10.1145/1206035.1206039
55. Hallgren, S.: Fast quantum algorithms for computing the unit group and class group of a number field. In: Proceedings of the thirty-seventh annual ACM symposium on Theory of Computing, pp. 468–474 (2005)
56. Schmidt, A., Vollmer, U.: Polynomial time quantum algorithm for the computation of the unit group of a number field. In: Proceedings of the thirty-seventh annual ACM symposium on Theory of Computing, pp. 475–480 (2005)
57. Eisenträger, K., et al.: A quantum algorithm for computing the unit group of an arbitrary degree number field. In: Proceedings of the forty-sixth annual ACM Symposium on Theory of Computing, STOC 2014, pp. 293–302. ACM (2014)
58. Biasse, J.-F., Song, F.: Efficient quantum algorithms for computing class groups and solving the principal ideal problem in arbitrary degree number fields. In: Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, pp. 893–902. SIAM (2016)
59. Campbell, P., Groves, M., Soliloquy, D.D.S.: A cautionary tale. In: ETSI 2nd Quantum-Safe Crypto Workshop, pp. 1–9 (2014)
60. Cramer, R., et al.: Recovering short generators of principal ideals in cyclotomic rings. In: Advances in Cryptology - EUROCRYPT, vol. 2016, pp. 559–585. Springer (2016)
61. Cramer, R., Ducas, L., Wesolowski, B.: Short stickelberger class relations and application to ideal-svp. In: Advances in Cryptology - EUROCRYPT 2017, pp. 324–348. Springer (2017)
62. Ducas, L., Plançon, M., Wesolowski, B.: On the shortness of vectors to be found by the ideal-svp quantum algorithm. In: Advances in Cryptology - CRYPTO 2019, pp. 322–351. Springer (2019)
63. Daemen, J., Rijmen, V.: The design of rijndael - the advanced encryption standard (AES). In: Information Security and Cryptography, ed, pp. X–XX. Springer (2020)
64. Derbez, P., Fouque, P.-A., Jean, J.: Improved key recovery attacks on reduced-round AES in the single-key setting. In: EUROCRYPT, volume 7881 of Lecture Notes in Computer Science, pp. 371–387. Springer (2013)
65. Grassl, M., et al.: Applying grover's algorithm to AES: quantum resource estimates. In: PQCrypto, volume 9606 of Lecture Notes in Computer Science, pp. 29–43. Springer (2016)
66. Jaques, S., et al.: Implementing grover oracles for quantum key search on AES and lowmc. In: EUROCRYPT (2), volume 12106 of Lecture Notes in Computer Science, pp. 280–310. Springer (2020)
67. Kaplan, M., et al.: Breaking symmetric cryptosystems using quantum period finding. In: CRYPTO (2), volume 9815 of Lecture Notes in Computer Science, pp. 207–237. Springer (2016)
68. Bonnetain, X., Naya-Plasencia, M., Schrottenloher, A.: Quantum security analysis of AES. IACR Trans. Symmetric Cryptol. 2019(2), 55–93 (2019). https://doi.org/10.46586/tosc.v2019.i2.55-93
69. Brassard, G., Høyer, P., Tapp, A.: Quantum cryptanalysis of hash and claw-free functions. In: LATIN, volume 1380 of Lecture Notes in Computer Science, pp. 163–169. Springer (1998)
70. Aaronson, S., Shi, Y.: Quantum lower bounds for the collision and the element distinctness problems. J. ACM 51(4), 595–605 (2004). https://doi.org/10.1145/1008731.1008735
71. Chailloux, A., Naya-Plasencia, M., Schrottenloher, A.: An efficient quantum collision search algorithm and implications on symmetric cryptography. In: ASIACRYPT (2), volume 10625 of Lecture Notes in Computer Science, pp. 211–240. Springer (2017)
72. van Oorschot, P.C., Wiener, M.J.: Parallel collision search with cryptanalytic applications. J. Cryptol. 12(1), 1–28 (1999). https://doi.org/10.1007/pl00003816
73. Hosoyamada, A., Sasaki, Y.: Finding hash collisions with quantum computers by using differential trails with smaller probability than birthday bound. In: EUROCRYPT (2), volume 12106 of Lecture Notes in Computer Science, pp. 249–279. Springer (2020)
74. Aumasson, J.-P., Endignoux, G.: Improving stateless hash-based signatures. In: Cryptographers' Track at the RSA Conference, pp. 219–242. Springer (2018)

75. Kuwakado, H., Morii, M.: Quantum distinguisher between the 3-round feistel cipher and the random permutation. In: ISIT, pp. 2682–2685. IEEE (2010)
76. Kuwakado, H., Morii, M.: Security on the quantum-type even-mansour cipher. In: ISITA, pp. 312–316. IEEE (2012)
77. Simon, D.: On the power of quantum computation. SIAM J. Comput. 26(5), 1474–1483 (1997). https://doi.org/10.1137/s0097539796298637
78. Bonnetain, X., Naya-Plasencia, M.: Hidden shift quantum cryptanalysis and implications. In: ASIACRYPT (1), volume 11272 of Lecture Notes in Computer Science, pp. 560–592. Springer (2018)
79. Bonnetain, X., et al.: Quantum linearization attacks. In: ASIACRYPT, Lecture Notes in Computer Science, pp. X–XX (2021). to appear
80. Bonnetain, X., et al.: Quantum attacks without superposition queries: the offline simon's algorithm. In: ASIACRYPT (1), volume 11921 of Lecture Notes in Computer Science, pp. 552–583. Springer (2019)
81. Bonnetain, X., Schrottenloher, A., Sibleyras, F.: Beyond quadratic speedups in quantum attacks on symmetric schemes. In: IACR Cryptol. ePrint Arch, pp. 1348–X (2021)
82. Leander, G., May, A.: Grover meets simon - quantumly attacking the fx-construction. In: ASIACRYPT (2), volume 10625 of Lecture Notes in Computer Science, pp. 161–178. Springer (2017)
83. Lenstra, A., Lenstra, H., Lovász, L.: Factoring polynomials with rational coefficients. Math. Ann. 261(4), 515–534 (1982). https://doi.org/10.1007/bf01457454
84. Korkine, A., Zolotarev, G.: Sur les formes quadratiques. Math. Ann. 6(3), 366–389 (1873). https://doi.org/10.1007/bf01442795
85. Schnorr, C.-P.: A hierarchy of polynomial time lattice basis reduction algorithms. Theor. Comput. Sci. 53(2-3), 201–224 (1987). https://doi.org/10.1016/0304-3975(87)90064-8
86. Hanrot, G., Pujol, X., Stehlé, D.: Analyzing blockwise lattice algorithms using dynamical systems. In: Advances in Cryptology – CRYPTO 2011, pp. 447–464 (2011)
87. Kannan, R.: Minkowski's convex body theorem and integer programming. Math. Oper. Res. 12(3), 415–440 (1987). https://doi.org/10.1287/moor.12.3.415
88. Buchmann, J.: Reducing lattice bases by means of approximations. In: Algorithmic Number Theory, pp. 160–168 (1994)
89. Alkim, E., et al.: Post-quantum key exchange—a new hope. In: 25th USENIX Security Symposium (USENIX Security 16), pp. 327–343 (2016)
90. Ducas, L., et al.: Crystals-dilithium: a lattice-based digital signature scheme. In: Transactions on Cryptographic Hardware and Embedded Systems, pp. X–XX (2018) 01 2018
91. Smart, N., Vercauteren, F.: Fully homomorphic encryption with relatively small key and ciphertext sizes. In: Nguyen, P., Pointcheval, D. (eds.) Public Key Cryptography - PKC 2010, 13th International Conference on Practice and Theory in Public Key Cryptography, Paris, France, May 26-28, 2010. Proceedings, volume 6056 of Lecture Notes in Computer Science, pp. 420–443. Springer (2010)
92. The FPLLL development team.: FPLLL, a Lattice Reduction Library (2018). https://github.com/fplll/fplll
93. Gama, N., Nguyen, P., Regev, O.: Lattice enumeration using extreme pruning. In: Advances in Cryptology – EUROCRYPT 2010, pp. 257–278 (2010)
94. Kannan, R.: Improved algorithms for integer programming and related lattice problems. In: Proceedings of STOC, pp. 193–206 (1983)
95. Albrecht, M., et al.: Faster enumeration-based lattice reduction: root hermite factor $k^{1/(2k)}$ time $k^{k/8+o(k)}$. In: Advances in Cryptology – CRYPTO 2020, vol. 12171 (2020).186
96. Aono, Y., Nguyen, P., Shen, Y.: Quantum lattice enumeration and tweaking discrete pruning. In: Advances in Cryptology – ASIACRYPT 2018, pp. 405–434 (2018)
97. Ajtai, M., Kumar, R., Sivakumar, D.: A sieve algorithm for the shortest lattice vector problem. In: Proceedings of the 33rd Annual ACM Symposium on Theory of Computing, STOC '01, pp. 601–610 (2001)
98. Becker, A., Laarhoven, T.: Efficient ideal lattice sieving using cross-polytope lsh. In: AFRICACRYPT 2016, vol. 9646, pp. 3–23 (2016)
99. Laarhoven, T.: Sieving for shortest vectors in lattices using angular locality-sensitive hashing. In: Advances in Cryptology – CRYPTO 2015, pp. 3–22 (2015)
100. Micciancio, D., Voulgaris, P.: Faster exponential time algorithms for the shortest vector problem. In: Proceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1468–1480. SODA '10 (2010)
101. Nguyen, P., Vidick, T.: Sieve algorithms for the shortest vector problem are practical. J. Math. Cryptol. 2(2), 181–207 (2008). https://doi.org/10.1515/jmc.2008.009
102. Wang, X., et al.: Improved nguyen-vidick heuristic sieve algorithm for shortest vector problem. In: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, pp. 1–9. ASIACCS '11 (2011)
103. Becker, A., et al.: New directions in nearest neighbor searching with applications to lattice sieving. In: Proceedings of the Twenty-seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '16, pp. 10–24 (2016)
104. Aggarwal, D., Ursu, B., Vaudenay, S.: Faster sieving algorithm for approximate svp with constant approximation factors. In: Cryptology ePrint Archive, Report 2019/1028, pp. X–XX (2019). https://ia.cr/2019/1028
105. Aggarwal, D., et al.: Improved (Provable) Algorithms for the Shortest Vector Problem via Bounded Distance Decoding (2020). https://arxiv.org/abs/2002.07955
106. Aggarwal, D., Stephens-Davidowitz, N.: Just Take the Average! an Embarrassingly Simple 2^n-Time Algorithm for Svp (And Cvp). SOSA (2018)
107. Aggarwal, D., Li, Z., Stephens-Davidowitz, N.: A $2^{n/2}$-time algorithm for n-SVP and n-Hermite SVP, and an improved time-approximation tradeoff for (H)SVP. In: Eurocrypt – Advances in Cryptology—2021, pp. 467–497 (2021)
108. Hanrot, G., Stehlé, D.: Improved analysis of kannan's shortest lattice vector algorithm. In: Advances in Cryptology - CRYPTO 2007, pp. 170–186 (2007)
109. Klein, P.: Finding the closest lattice vector when it's unusually close. In: SODA '00, pp. 937–941 (2000)
110. Bai, S., Laarhoven, T., Stehlé, D.: Tuple lattice sieving. LMS J. Comput. Math. 19(A), 146–162 (2016). https://doi.org/10.1112/s1461157016000292
111. Herold, G., Kirshanova, E., Laarhoven, T.: Speed-ups and time–memory trade-offs for tuple lattice sieving. In: Public-Key Cryptography – PKC 2018, pp. 407–436 (2018)
112. Kirshanova, E., et al.: Quantum algorithms for the approximate k-list problem and their application to lattice sieving. In: Advances in Cryptology - ASIACRYPT 2019, pp. 521–551 (2019)
113. Aggarwal, D., et al.: Solving the shortest vector problem in 2^n time using discrete Gaussian sampling: extended abstract. In: STOC '15, pp. 733–742 (2015)
114. Pujol, X., Stehlé, D.: Solving the shortest lattice vector problem in time $2^{2.465n}$. In: Cryptology ePrint Archive, Report 2009/605, pp. X–XX (2009). https://eprint.iacr.org/2009/605
115. Herold, G., Kirshanova, E.: Improved algorithms for the approximate k-list problem in Euclidean norm. In: PKC 2017, pp. 16–40 (2017)
116. Laarhoven, T., Mosca, M., van de Pol, J.: Finding shortest lattice vectors faster using quantum search. Des. Codes Cryptogr. 77(2), 375–400 (2015). https://doi.org/10.1007/s10623-015-0067-5
117. Laarhoven, T.: Search Problems in Cryptography. PhD Thesis. Eindhoven University of Technology (2015)
118. Chailloux, A., Loyer, J.: Lattice sieving via quantum random walks. In: Advances in Cryptology – ASIACRYPT, vol. 2021, pp. 63–91 (2021)
119. Heiser, M.: Improved quantum hypercone locality sensitive filtering in lattice sieving. In: Cryptology ePrint Archive, Report 2021/1295, pp. X–XX (2021). https://eprint.iacr.org/2021/1295

120. Peikert, C.: A decade of lattice cryptography. Found. Trends® Theor. Comput. Sci. 10(4), 283–424 (2016). https://doi.org/10.1561/0400000074
121. Albrecht, M., Ducas, L.: Lattice attacks on NTRU and LWE: a history of refinements. In: London Mathematical Society Lecture Note Series, pp. 15–40. Cambridge University Press (2021)
122. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM, 56(6), 1–40. https://doi.org/10.1145/1568318.1568324, (2009)
123. Bos, J., et al.: Crystals - kyber: a cca-secure module-lattice-based kem. In: 2018 IEEE European Symposium on Security and Privacy (EuroS P), pp. 353–367 (2018)
124. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. In: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, pp. 309–325 (2012)
125. Boneh, D., et al.: Fully key-homomorphic encryption, arithmetic circuit abe and compact garbled circuits. In: Advances in Cryptology – EUROCRYPT 2014, pp. 533–556 (2014)
126. Postlethwaite, E., Virdia, F.: On the success probability of solving unique SVP via BKZ. In: Garay, J. (ed.) Public-Key Cryptography - PKC 2021. Lecture Notes in Computer Science, pp. 68–98 (2021)
127. Albrecht, M., et al.: Estimate all the \{LWE, NTRU\} schemes! in SCN. Lect. Notes Comput. Sci., 351–367 (2018)
128. Dachman-Soled, D., et al.: Lwe with side information: attacks and concrete security estimation. In: Advances in Cryptology – CRYPTO 2020, pp. 329–358 (2020)
129. Blum, A., Kalai, A., Wasserman, H.: Noise-tolerant learning, the parity problem, and the statistical query model. J. ACM 50(4), 506–519 (2003). https://doi.org/10.1145/792538.792543
130. Guo, Q., et al.: On the asymptotics of solving the LWE problem using coded-bkw with sieving. IEEE Trans. Inf. Theor. 65(8), 5243–5259 (2019). https://doi.org/10.1109/tit.2019.2906233
131. Kirchner, P., Fouque, P.-A.: An improved BKW algorithm for LWE with applications to cryptography and lattices. In: Advances in Cryptology - CRYPTO 2015, volume 9215 of Lecture Notes in Computer Science, pp. 43–62 (2015)
132. Arora, S., Ge, R.: New algorithms for learning in presence of errors. In: Automata, Languages and Programming, pp. 403–415 (2011)
133. Albrecht, M., et al.: Algebraic algorithms for LWE problems. ACM Commun. Comput. Algebra 49(2), 62 (2015). https://doi.org/10.1145/2815111.2815158
134. Grilo, A., Kerenidis, I., Zijlstra, T.: Learning-with-errors problem is easy with quantum samples. Phys. Rev. 99(3), 032314 (2019). https://doi.org/10.1103/physreva.99.032314
135. Brakerski, Z., et al.: Learning with errors and extrapolated dihedral cosets. In: PKC-2018, pp. 702–727 (2018)
136. Chen, Y., Liu, Q., Zhandry, M.: Quantum Algorithms for Variants of Average-Case Lattice Problems via Filtering (2021)
137. Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, STOC '96, pp. 99–108 (1996). –
138. Albrecht, M., et al.: The general sieve kernel and new records in lattice reduction. In: EUROCRYPT 2019, pp. 717–746 (2019)
139. Albrecht, M., et al.: Estimating quantum speedups for lattice sieves. In: Advances in Cryptology - ASIACRYPT 2020, pp. 583–613 (2020)
140. Bernstein, D., et al.: Classic Mceliece: Conservative Code-Based Cryptography. NIST submission (2017). https://classic.mceliece.org/nist.html
141. Aragon, N., et al.: Bike: Bit Flipping Key Encapsulation. NIST submission (2017)
142. Hoffstein, J., et al.: A ring-based public key cryptosystem. In: Algorithmic Number Theory, pp. 267–288. Springer Berlin HeidelbergBerlin (1998)
143. Debris-Alazard, T., Sendrier, N., Wave, J.-P.T.: A new family of trapdoor one-way preimage sampleable functions based on codes. In: Advances in Cryptology – ASIACRYPT 2019, pp. 21–51. Springer International Publishing (2019)
144. Barenghi, A., et al.: LESS-FM: fine-tuning signatures from the code equivalence problem. In: PQCrypto 2021, vol. 12841, pp. 23–43 (2021)
145. Bricout, R., et al.: Ternary syndrome decoding with large weight. In: Selected Areas in Cryptography – SAC 2019, pp. 437–466. Springer International Publishing (2020)
146. May, A., Meurer, A., Thomae, E.: Decoding random linear codes in. In: ASIACRYPT, volume 7073 of LNCS, pp. 107–124 (2011)$\tilde{O}(2^{0.54n})$
147. May, A., Ozerov, I.: On computing nearest neighbors with applications to decoding of binary linear codes. In: EUROCRYPT, pp. 203–228 (2015)
148. Prange, E.: The use of information sets in decoding cyclic codes. IEEE Trans. Inf. Theor. 6(5–9) (1962)
149. Stern, J.: A method for finding codewords of small weight. In: Coding Theory and Applications, vol. 106–113. 3rd International Colloquium (1989)
150. Bernstein, D.: Grover vsERSUS McEliece. In: Post-quantum Cryptography: Third International Workshop, PQCrypto, pp. 73–80. Springer Berlin Heidelberg (2010)
151. Kachigar, G., Tillich, J.-P.: Quantum information set decoding algorithms. In: Post-quantum Cryptography, pp. 69–89. Springer International Publishing (2017)
152. Kirshanova, E.: Improved quantum information set decoding. In: Post-quantum Cryptography, pp. 507–527. Springer International Publishing (2018)
153. Dumer, I.: On minimum distance decoding of linear codes. In: Proc. 5th Joint Soviet-Swedish Int. Workshop Inform. Theory, pp. 50–52 (1991)
154. Finiasz, M., Sendrier, N.: Security bounds for the design of code-based cryptosystems. In: ASIACRYPT'09, pp. 88–105 (2009)
155. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proceedings of the thirtieth annual ACM symposium on Theory of Computing, pp. 604–613 (1998)
156. Becker, A., et al.: Decoding random binary linear codes in $2^{n/20}$: how 1 + 1 = 0 improves information set decoding. In: EUROCRYPT, pp. 520–536 (2012)
157. Minder, L., Sinclair, A.: The extended k-tree algorithm. J. Cryptol. 25(2), 349–382 (2012). https://doi.org/10.1007/s00145-011-9097-y
158. Ambainis, A.: Quantum walk algorithm for element distinctness. SIAM J. Comput. 37(1), 210–239 (2007). https://doi.org/10.1137/s0097539705447311
159. Bernstein, D., et al.: Quantum algorithms for the subset-sum problem. In: PQCrypto 2013, pp. 16–33 (2013)
160. Canto Torres, R., Sendrier, N.: Analysis of information set decoding for a sub-linear error weight. In: Post-quantum Cryptography - 7th International Workshop, PQCrypto 2016, Fukuoka, Japan, February 24-26, 2016, Proceedings, volume 9606 of Lecture Notes in Computer Science, pp. 144–161. Springer (2016)
161. Both, L., May, A.: Decoding linear codes with high error rate and its impact for LPNsecurity. In: Post-quantum Cryptography - 9th International Conference, PQCrypto 2018, Fort Lauderdale, FL, USA, April 9-11, 2018, Proceedings, volume 10786 of Lecture Notes in Computer Science, pp. 25–46. Springer (2018)
162. Esser, A., Bellini, E.: Syndrome decoding estimator. In: Cryptology ePrint Archive, Report 2021/1243, pp. X–XX (2021). https://ia.cr/2021/1243
163. Silverman, J.: The Arithmetic of Elliptic Curves, Volume 106 of Graduate Texts in Mathematics. Springer-Verlag (1992)
164. Vélu, J.: Isogénies entre courbes elliptiques. C. R. Acad. Sci. Paris Sér. A-B 273 (1971). A238–A241
165. Couveignes, J.-M.: Hard Homgeneous Spaces. http://eprint.iacr.org/2006/291
166. Alamati, N., et al.: Cryptographic group actions and applications. In: Moriai, S., Wang, H. (eds.) Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part II, volume 12492 of Lecture Notes in Computer Science, pp. 411–439. Springer (2020)
167. Feo, L., Kieffer, J., Smith, B.: Towards practical key exchange from ordinary isogeny graphs. In: Peyrin, T., Galbraith, S. (eds.) Advances in

Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part III, volume 11274 of Lecture Notes in Computer Science, pp. 365–394. Springer (2018)
168. Rostovtsev, A., Stolbunov, A.: Public-key Cryptosystem Based on Isogenies. IACR Cryptol. ePrint Arch. (2006).145
169. Castryck, W., et al.: CSIDH: an efficient post-quantum commutative group action. In: Peyrin, T., Galbraith, S. (eds.) Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part III, volume 11274 of Lecture Notes in Computer Science, pp. 395–427. Springer (2018)
170. Neukirch, J.: Algebraic Number Theory. Comprehensive Studies in Mathematics. Springer-Verlag (1999). 3-540-65399-6
171. Colò, L., Kohel, D.: Orienting supersingular isogeny graphs. J. Math. Cryptol. 14(1), 414–437 (2020). https://doi.org/10.1515/jmc-2019-0034
172. Childs, A., Jao, D., Soukharev, V.: Constructing elliptic curve isogenies in quantum subexponential time. J. Math. Cryptol. 8(1), 1–29 (2013). https://doi.org/10.1515/jmc-2012-0016
173. Bonnetain, X., Schrottenloher, A.: Quantum security analysis of CSIDH. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II, volume 12106 of Lecture Notes in Computer Science, pp. 493–522. Springer (2020)
174. Peikert, C., He: Gives C-sieves on the CSIDH. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II, volume 12106 of Lecture Notes in Computer Science, pp. 463–492. Springer (2020)
175. Chávez-Saab, J., et al.: The SQALE of CSIDH: square-root vélu quantum-resistant isogeny action with low exponents. In: IACR Cryptol. ePrint Arch, pp. 1520–X (2020)
176. Jao, D., De Feo, L.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In: Proceedings of the 4th International Conference on Post-Quantum Cryptography, PQCrypto'11, pp. 19–34. Springer-VerlagBerlin (2011)
177. Azarderakhsh, R., et al.: SIKE: Supersingular Isogeny Key Encapsulation. https://sike.org/
178. Biasse, J.-F., Pring, B.: A framework for reducing the overhead of the quantum oracle for use with grover's algorithm with applications to cryptanalysis of SIKE. J. Math. Cryptol. 15(1), 143–156 (2021). https://doi.org/10.1515/jmc-2020-0080
179. Tani, S.: Claw finding algorithms using quantum walk. Theoretical Computer Science, 410(50):5285–5297 (2009)Mathematical Foundations of Computer Science (MFCS) 2007
180. Jaques, S., Schrottenloher, A.: Low-gate quantum golden collision finding. In: SAC, volume 12804 of Lecture Notes in Computer Science, pp. 329–359. Springer (2020)
181. Charles, D., Lauter, K., Goren, E.: Cryptographic hash functions from expander graphs. J. Cryptol. 22(1), 93–113 (2009). https://doi.org/10.1007/s00145-007-9002-x
182. Delfs, C., Galbraith, S.: Computing isogenies between supersingular elliptic curves over. Des. Codes Cryptogr. 78(2), 425–440 (2016). https://doi.org/10.1007/s10623-014-0010-1
183. Biasse, J.-F., Jao, D., Sankar, A.: A quantum algorithm for computing isogenies between supersingular elliptic curves. In: Meier, W., Mukhopadhyay, D. (eds.) Progress in Cryptology - INDOCRYPT 2014 - 15th International Conference on Cryptology in India, New Delhi, India, December 14-17, 2014, Proceedings, volume 8885 of Lecture Notes in Computer Science, pp. 428–442. Springer (2014)
184. Kutas, P., et al.: Weak Instances of SIDH Variants under Improved Torsion-point Attacks. IACR Cryptol. ePrint Arch. (2020).633
185. Kutas, P., et al.: One-way functions and malleability oracles: hidden shift attacks on isogeny-based protocols. In: EUROCRYPT (1), volume 12696 of Lecture Notes in Computer Science, pp. 242–271. Springer (2021)
186. Castryck, W., Decru, T.: An efficient key recovery attack on SIDH (preliminary version), Cryptology ePrint Archive, 2022/975 (2022)

How to cite this article: Biasse, J.-F., et al.: Quantum algorithms for attacking hardness assumptions in classical and post-quantum cryptography. IET Inf. Secur. 1–39 (2022). https://doi.org/10.1049/ise2.12081