Quantum Cryptanalysis: Shor, Grover, and Beyond

Stephen P. Jordan and Yi-Kai Liu

1 Introduction

In 1994 Peter Shor discovered polynomial-time quantum algorithms for integer factorization and discrete logarithms. If sufficiently large-scale quantum computers are built, Shor's algorithms will completely break the RSA cryptosystem and signature scheme, Diffie-Helman key exchange, and elliptic curve cryptography such as ECDH and ECDSA. At present, there still are not any sufficiently large-scale quantum computers to perform cryptanalysis. However, there has been substantial progress since 1994 both on quantum computational hardware and algorithms. In response to accelerating progress on quantum hardware in the last few years, efforts have been intensifying to develop practical alternatives to the public key key cryptographic methods currently in use. Such "post-quantum" cryptographic schemes base their security on hard computational problems, such as finding short vectors in high-dimensional lattices or solving systems of multivariate quadratic equations over finite fields, that are not known to be susceptible to attack by quantum computers. However, since 1994, a small but dedicated community of quantum algorithm researchers has continually discovered quantum algorithms offering speedups for a widening variety of problems. Whether the various proposed post-quantum cryptosystems are truly safe from quantum attack remains an area of current research.

2 Quantum computation

Two of the most common misconceptions about quantum computers are that they can achieve exponential speedup for generic problems through exponential parallelism and that the only speedups achievable by quantum computers are for integer factorization using Shor's algorithm and searching using Grover's algorithm. The truth is far more interesting than either of these caricatures. In reality, to outperform classical algorithms, quantum algorithms rely on subtle interference effects among amplitudes representing exponentially many different computational paths. The quantum speedups discovered so far extend well beyond factoring and searching, but they much are more rare and precious than is suggested by the metaphor of exponential parallelism.

A key element of quantum mechanics, which underlies quantum computation, is the principle of superposition. Each distinguishable state of a quantum system has a corresponding amplitude, which is a complex number of magnitude at most one. Upon measurement, the

system will be found in a given state with probability given by the squared magnitude the associated amplitude. The idea of superposition is often illustrated using Schrödinger's famous thought experiment in which a cat is put into a state that simultaneously has equal amplitude to be alive and dead. However, to observe non-classical behavior arising from superposition requires very careful isolation from stray interactions with the environment. Under ordinary circumstances, at the macroscopic level, noise washes out quantum effects far too quickly for them to be noticeable.

In accordance with the superposition principle, the state of a quantum computer with n quantum bits of memory (qubits) is described by a list of 2^n amplitudes, one for each possible length-n bit string. These exponentially complicated states give some hint as to the origin of the power of quantum computers. A full description of the state of a quantum computer with only 80 qubits would already be too large to store on all the hard drives ever manufactured. However, this is not the whole story. To completely describe the state of an n-bit classical probabilistic computer one would also need to write down 2^n numbers, namely the probabilities associated with each of the 2^n bit strings. So, exponentially complicated states are not a uniquely quantum phenomenon! The crucial difference is that quantum amplitudes can be positive or negative[1], and thus can interfere constructively or destructively like waves, whereas probabilities are always nonnegative and simply add. Only by carefully designing quantum algorithms to take advantage of interference effects is it possible to achieve exponential speedup over classical computation.

Over the past 20 years, steady progress has been made in the quest to actually build quantum computers. A key obstacle is that the superpositions and interference effects necessary to achieve quantum computational speedup are extremely fragile and easily destroyed by imprecise control signals and stray influences from the environment. However, in 1996 it was shown theoretically that, if error per operation in a quantum computer could be brought below a fixed threshold, then it is possible to carry out arbitrarily long quantum computations reliably through the use of error-correcting codes. Recently, individual operations on small collections of qubits have been demonstrated in multiple quantum computing platforms, such as ion traps and superconducting circuits, that are nominally below this threshold. To bring the overhead associated with error correction down to tolerable levels for practical systems it will be necessary to decrease the error rates still further. Nevertheless, this achievement has been widely viewed as a significant milestone in the development of quantum computers.

At the same time as universal quantum computers are being developed, other labs are developing more specialized devices such as quantum simulators for mimicking materials that are hard to simulate on conventional supercomputers, and quantum annealers for solving discrete optimization problems. These specialized quantum computers have been scaled up much further than universal digital quantum computers. In particular, D-wave's quantum current annealer has over 1000 qubits, whereas current prototypes of universal quantum computers use only tens of qubits. However, quantum simulators and quantum annealers

[1]More generally, quantum amplitudes can have arbitrary complex phases. However, it turns out that real positive and negative amplitudes suffice for universal quantum computation.

are single-purpose devices, unable to run, for example, Shor's algorithm, and do not have any known applications to cryptanalysis. The remainder of this article will focus only on algorithms for universal digital quantum computers.

3 Quantum algorithms

In classical computing we are used to describing computations at different levels of abstraction. At the bottom level we have elementary logic operations such as AND, OR, and NOT gates. At an intermediate level we talk of arithmetic operations, if statements, for loops, and so on. At a high level we talk of commonly used subroutines, such as searching, sorting, inverting matrices, and fast Fourier transforms. The same holds for quantum computation. What follows is a brief tour of quantum algorithms from a high-level perspective, emphasizing quantum-algorithmic primitives of relevance to cryptanalysis.

3.1 Reversible Computing

In the earliest days of quantum computation research the main question being asked was not whether quantum effects could be used for computational advantage but rather whether they posed a barrier to continued miniaturization of computers. In particular, the dynamics of closed quantum systems are fully reversible, that is, information cannot be erased. Consequently, logic gates such as the AND gate, which takes two bits of input and produces one bit of output, cannot be directly implemented in a quantum-coherent way. However, pioneering works by Lecerf, Bennett, Toffoli, and Fredkin in the 60s, 70s, and 80s, showed how universal classical computing could be performed in a reversible way, such as by using universal sets of logic gates that implement reversible maps from their input bits to their output bits. Consequently, any classical subroutine can be recompiled into reversible logic gates and queried in quantum superposition within a quantum algorithm. This recompilation incurs an efficiency cost of only constant-factor overhead.

Many quantum algorithms are usefully phrased in terms of oracle (or "black-box") problems. One is given access to some efficiently computable function, and one wishes to deduce some property of it by making queries. Grover search and period finding, discussed in detail below, are two examples of this. Reversible computing underlies these quantum algorithms in that the oracles are ultimately classical subroutines that have been recompiled into reversible gate sets so that they may be run on a quantum computer and queried on superpositions of inputs.

3.2 Discrete Period finding

One of the most important high-level primitives in quantum algorithm design is period finding. The period finding problem is to find the period of a periodic function f, given the ability to evaluate f at arbitrary elements of its domain. That is, there is some s such that $f(x+s) = x$ for all x, and the goal is to find s. Access to the source code for f may make

this problem easier. For example, if one examines the code for f and sees that it simply returns the residue of x modulo s, then it is trivial to read off that the period is s. However, in many situations, the computation of f is sufficiently complicated that there may be no better way to find the period than by treating f as a black-box to be queried at various x until the period can be found. Such a black-box framework is frequently a useful simplifying assumption in the analysis of quantum algorithms.

On a quantum computer, it is possible to make queries to a black box function in superposition. For some problems, such as period finding, this ability is greatly advantageous. For others, such as computing the overall parity of the set of outputs produced by f, quantum superposition queries yield very little advantage. If the period s of f is exponentially large then classically one requires exponentially many queries, on average, to find s. After a number of queries small compared to $\sqrt{s}$, one will not encounter any pair x, y such that $f(x) = f(y)$ and consequently one will have learned nothing about the period. In contrast, by making only polynomially many superposition queries, a quantum computer can learn the period s, even if it is exponentially large. This is done by computing the quantum Fourier transform — an analogue of the classical fast Fourier transform, which operates on quantum superposition states. This discovery, made by Peter Shor in 1994, formed the heart of his quantum algorithms for integer factorization and discrete logarithms.

Shor's algorithms for factoring and discrete logarithms completely break the RSA cryptosystem and signature scheme, and Diffie-Hellman key exchange. Furthermore, the quantum algorithms for period finding can straightforwardly generalize to other domains. In the examples above, the inputs to x have been integers taken modulo N. However, if the inputs to x are thought of as elements of some arbitrary Abelian group then the quantum algorithm for period finding still applies with little modification. The period may be described by a set of generators for a subgroup of the inputs which leave the value of f invariant. Quantum algorithms for this "Abelian hidden subgroup problem" can in particular be applied to the groups associated with elliptic curves, thereby breaking Elliptic Curve Diffie-Hellman (ECDH) key exchange and the Elliptic Curve Digital Signature Algorithm (ECDSA).

Encouraged by these successes, many researchers have attempted to devise quantum algorithms for "hidden subgroup problems" (HSP) over various non-Abelian groups, such as the symmetric group, the dihedral group, and the general linear group over a finite field. These HSP's are connected to a number of other well-known problems that seem to be hard for classical computers, namely the graph isomorphism problem, the shortest vector problem in lattice-based cryptography, and certain problems underlying the security of multivariate cryptosystems. To date there has not been a breakthrough in solving any of these famous problems on a quantum computer. But there have been a number of interesting results, such as Kuperberg's algorithm, which solves the dihedral HSP in subexponential time [1].

3.3 Searching

Like period finding, searching can be cast in a black box query framework. For example, given a function $f : D \to \{0, 1\}$ on some finite set D, find an $x \in D$ such that $f(x) = 1$. The hardest instances of such a problem are when $f(x) = 1$ only for a unique x. Classically solving this

requires $|D|$ queries in the worst case and $|D|/2$ queries on average. Lov Grover discovered in 1996 that quantum computers can solve this problem using $O(\sqrt{|D|})$ superposition queries. Earlier lower bounds showed that this is optimal, that is, fewer than order $\sqrt{|D|}$ quantum queries will not suffice.

Although stated abstractly, this primitive is easily adapted to many real-world problems. For example, suppose $g : D \to R$ is a one-way function, and you wish to invert it, that is, calculate $g^{-1}(r)$ for some $r \in R$. From g one can easily construct $f : D \to \{0, 1\}$ which outputs 1 if $g(x) = r$ and 0 otherwise. Such a function can be compiled into a reversible circuit that can be run on a quantum computer and queried in superposition. In this way Grover's algorithm can perform brute-force inversion of one-way functions in $O(\sqrt{|D|})$ time, a quadratic improvement over classical brute-force. This has led to a rule of thumb for post-quantum symmetric-key cryptography, which advocates doubling key sizes (and thereby squaring the size of D) to maintain a given security level against attack by Grover search. However, this rule of thumb likely overestimates the advantage that can be gained from running Grover search in practice, since brute-force attacks are often run in parallel on large clusters of computers, and the Grover speedup is less significant in this setting. Also, more detailed analysis has yielded more precise guidance on choice of key sizes in response to the speedups afforded by replacing search-like subroutines in state of the art classical cryptanalysis algorithms with Grover search [2, 3].

A quantum algorithm based on Grover search can also speed up collision finding, that is, finding $x, y \in D$ such that $f(x) = f(y)$. Classically, this requires, in the worst case $O(\sqrt{|D|})$ queries, by the birthday paradox, whereas on a quantum computer, it can be achieved using $O(|D|^{1/3})$ queries. However, this quantum algorithm has large memory requirements and is probably not of practical relevance for finding collisions in hash functions [4].

3.4 Hidden Shift

Hidden shift problems are another class of idealized oracle-query problems which have real-world implications. In a hidden shift problem, we are given oracle access to some function f, and we know that $f(x) = g(x+s)$ for some fixed known function g and unknown shift s. The goal is to find s by making queries to f. For the hardest instances of hidden shift, quantum computers can achieve only a quadratic speedup. For example, suppose the domain of f and g is the integers modulo N and let $g(0) = 1$ and $g(x) = 0$ for all $x \neq 1$. Then, finding s is exactly the Grover search problem, for which quantum computers can achieve only a quadratic speedup over classical computers

However, for more structured instances of hidden shift, quantum algorithms have been discovered yielding exponential speedup. In particular, the Legendre symbol, for any prime p, is defined as follows.

$$\left(\frac{x}{p}\right) = \begin{cases} 0 & \text{if } x \equiv 0 \bmod p \\ 1 & \text{if } x \equiv m^2 \bmod p \text{ for some } m \\ -1 & \text{otherwise} \end{cases}$$

The shifted Legendre symbol problem is, given oracle access to $f(x) = \left(\frac{x+s}{p}\right)$ for known p, to find s. As shown in [5], quantum computers can solve this in time that scales only polynomially in $\log p$, whereas it is believed that no classical algorithms can achieve this. In fact, the presumed hardness of this problem was used as the basis for a proposed cryptographic random number generator [6]. At its core, the quantum algorithm of [5] relies on the same quantum Fourier transform that powers Shor's algorithm.

Another problem that has beyond-Grover quantum speedup is the hidden shift problem for injective functions. In this case, the number of quantum queries that are information-theoretically sufficient to extract the shift is only poly$(\log N)$. However, no quantum algorithm has been found to solve hidden shift in poly$(\log N)$ total runtime (which includes not just the queries but also pre- and post-processing). The quantum algorithm with the smallest total runtime for solving the injective hidden shift is Kuperberg's sieve, which runs in $2^{O(\sqrt{N})}$ time [1]. Interestingly, this quantum algorithm is based on fundamentally different algorithmic techniques than the other known quantum algorithms achieving superpolynomial speedups for algebraic and number-theoretic problems, which are all essentially based on Fourier transforms and period finding. Subsequent work improved upon Kuperberg's sieve by giving a quantum algorithm for the injective hidden shift problem that runs in $2^{O(\sqrt{N})}$ time and also uses only poly$(\log N)$ qubits of quantum memory.

In [7] a subexponential-time quantum algorithm for finding isogenies between ordinary (*i.e.* non-supersingular) elliptic curves was constructed through a nontrivial reduction to the injective hidden shift problem. Roughly speaking, such isogenies are maps from one elliptic curve to another which preserve the associated group structure. This quantum algorithm can be used to attack certain public-key cryptosystems that base their security on the assumed difficulty of finding isogenies of ordinary elliptic curves. However, this quantum algorithm is not applicable to attacking the cryptosystems based on the difficulty of finding isogenies between supersingular elliptic curves.

3.5 Quantum Walks

A very different kind of computational speedup can be obtained using an algorithmic technique known as a quantum walk. Quantum walks are a technique for exploring a graph or network. They are defined by analogy with classical random walks, in different ways — there are several variants. As an illustration, we describe the continuous-time quantum walk, which is particularly simple.

In a classical random walk, a particle moves around a graph, hopping from one vertex to another, where each step is chosen at random, i.e., if the particle currently resides at vertex v, it moves to one of the neigboring vertices of v, chosen at random. This defines a stochastic process, with a Markov transition matrix A. The continuous-time quantum walk is defined as the quantum dynamics that is generated by the Schrodinger equation, with the Hamiltonian given by A.

Quantum walks can be used to explore graphs that are exponentially large, provided that the list of neighbors of any given vertex can be computed efficiently on a quantum computer.

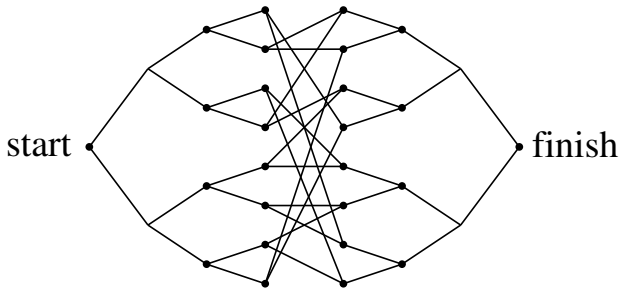

Figure 1: For glued trees, as illustrated above, the quantum walk algorithm of [8] finds the finish in time polynomial in the depth of the trees, whereas the fastest possible classical algorithm requires time polynomial in the number of vertices. Thus quantum walks achieve an exponential speedup for this problem.

Of course, classical random walks can also be used to explore large graphs. But the behavior of a quantum walk is strikingly different. For instance, a classical random walk converges to the stationary or equilibrium distribution, which is often spread over the whole graph. A quantum walk, on the other hand, is a reversible process, so it never converges. Instead, the quantum particle "hits" different parts of the graph, and then returns. Which parts of the graph get "hit" depends in part on the patterns of constructive and destructive interference that are created by the complex-valued wavefunction of the quantum particle.

This unusual behavior is advantageous for solving certain problems. A famous example is the problem of finding a path through two "glued trees" [8]. Here, the graph consists of two full binary trees, whose leaves are connected by a random, alternating cycle. The problem is to find a path from the root of one tree to the root of the other tree.[2] This problem can be solved in polynomial time using a continuous-time quantum walk, but it requires exponential time using any classical algorithm. Intuitively, the reason is that any classical algorithm gets lost in the middle section of the graph, because there are exponentially many vertices, and exponentially few paths out of that section. The quantum walk, meanwhile, will propagate straight across the graph, due to constructive interference at each "level" of the tree.

The above algorithm is an intriguing example of an exponential quantum speedup that has nothing to do with the quantum Fourier transform, the hidden subgroup problem, or any kind of algebraic structure at all. This suggests that there may be other approaches to designing quantum algorithms for problems that are hard for classical computers.

[2]This is formulated as an oracle problem: one is given the name of the vertex at the root of the first tree, and an oracle that computes the neighbors of any vertex in this graph, and one is asked to discover the name of the vertex at the root of the other tree.

4 Some Current Frontiers in Quantum Algorithms

Quantum algorithms research is a highly technical subject in which major new algorithmic techniques are developed only once or twice per decade. Some of the most recent such developments are a quantum algorithm for the principal ideal problem, which has applications to lattice-based cryptography, a new method for solving discrete optimization problems on quantum computers called the Quantum Approximate Optimization Algorithm (QAOA), and a method for solving solving exponentially large systems of linear equations under certain conditions called the Harrow-Hassidim-Lloyd (HHL) algorithm. It is not yet clear what, if any, implications these primitives may have for quantum cryptanalysis.

4.1 Quantum Algorithms for Lattice Problems

There have been many attempts to devise quantum algorithms for finding short vectors in high-dimensional lattices. Early on, researchers noted a number of tantalizing connections between the mathematical methods used in lattice-based cryptography, and those used in quantum algorithms. In particular, many of the security proofs for lattice-based cryptosystems make use random samples from certain periodic distributions over $\mathbb{R}^n$, as well as the Fourier transforms of these periodic distributions. Could there be a way of using the quantum Fourier transform to prepare the quantum superposition states corresponding to these distributions? These superposition states, sometimes called "quantum samples," contain more information than classical random samples, and could be used to solve lattice problems.

In fact, this line of thinking did not lead to quantum algorithms for lattice problems. Instead, this idea was used by Oded Regev to prove one of the strongest security guarantees for lattice-based cryptography [9]. Specifically, Regev showed that breaking a certain public key encryption scheme is at least as hard as solving an intractable lattice problem, on a *quantum* computer. And so in this case, a technique originally intended for quantum cryptanalysis was ultimately used to provide evidence that a cryptosystem is actually secure.

More recently, however, there has been more progress on the side of quantum cryptanalysis. Specifically, there has been progress in developing quantum algorithms for solving problems involving lattices that have algebraic structure. (Lattices with algebraic structure are often used in cryptography, as they have more compact descriptions than general lattices. This makes the resulting cryptosystems more efficient.)

Recently, researchers have discovered efficient quantum algorithms for finding short generators of certain principal ideals in cyclotomic rings [10, 11, 12]. These quantum algorithms demonstrate an exponential speedup over the fastest classical algorithms, as well as a potential security weakness of certain algebraically-structured lattices as compared to general lattices. In particular, these quantum algorithms break the Buchman-Williams key exchange system, as well as lattice-based cryptosystems that use principal ideal lattices with unusually short generators, such as SOLILOQUY [10]. Other lattice-based cryptosystems based on ideal lattices, such as NTRUEncrypt and ring-LWE schemes, are not broken by any

presently-known quantum algorithm[3].

The underlying technique used in these quantum algorithms is period-finding, but over a continuous (rather than discrete) domain. For functions on continuous domains, such as the real numbers, the problem of finding (approximate) periodicities becomes more subtle. Nevertheless, quantum algorithms have been discovered which achieve exponential speedup over classical algorithms for this task. This in turn leads to polynomial-time quantum algorithms for various number theoretic problems, such as solving Pell's equation, finding generators of principal ideals, and computing the class groups and unit groups of algebraic number fields, in time polynomial in the degree of the number field [14].

4.2 Quantum Approximate Optimization Algorithm

QAOA was proposed in 2014 as a quantum algorithm for finding approximate solutions to discrete optimization problems such as MAX3SAT [15]. Such problems have wide-ranging practical applications. Furthermore, QAOA may be implementable on near-term quantum computing hardware that is not yet a universal error-corrected quantum computer. (QAOA shares these features with adiabatic quantum computation and quantum annealing to which it is closely related.) As a consequence, QAOA has attracted substantial interest from researchers since its introduction.

The difficulty of solving discrete optimization problems such as MAX3SAT varies greatly between instances. Consequently, proving meaningful bounds on the runtime of quantum or classical algorithms for these problems is extremely difficult. Furthermore, finding fully optimal solutions to these problems is NP-complete, and most computer scientists expect this task to have exponential complexity for worst case instances on both quantum and classical computers. However, in 2014 it was proven that QAOA can in polynomial-time find approximate solutions to a discrete optimization problem called Max E3LIN2, at an approximation scale that no polynomial-time classical algorithm had achieved. Prompted by this discovery, research intensified on classical approximation algorithms for Max E3LIN2, and in 2015 a classical algorithm provably outperforming QAOA on this problem was obtained. Nevertheless, the performance of QAOA on various discrete optimization problems remains a topic of intense research interest, both in terms of provable worst-case runtime bounds, and as a heuristic quantum algorithm.

4.3 Solving Linear Systems

Algorithms for solving linear algebra problems are among the most widely used primitives in all of scientific computing. Thus the 2009 announcement by Harrow, Hassidim, and Lloyd (HHL) of a quantum algorithm for solving exponentially large systems of linear equations in polynomial time was met with great excitement [16]. Specifically, given an $N \times N$ matrix A and an N-dimensional vector $\vec{b}$, over the real or complex numbers, the HHL algorithm can

[3]See [13] for a very nice journalistic description of recent developments in the quantum cryptanalysis of lattice-based cryptosystems.

under certain circumstances construct in time poly($\log N$) a quantum state proportional to $\vec{x}$, the solution to $A\vec{x} = \vec{b}$. Subsequently, measurements on this quantum state can yield partial information about the solution $\vec{x}$. Clearly, simply reading all the entries of an $N \times N$ matrix takes time of order N^2 and no algorithm (quantum or classical) that uses N^2 time to read its input can possibly have total running time of only poly($\log N$). Instead, the HHL algorithm is of interest in the case that A is given implicitly by a subroutine which, when queried with a row index $j \in \{1, 2, \ldots, N\}$, outputs a list of its nonzero matrix elements and their locations. Under the condition that A is highly sparse, *i.e.* each row has only poly($\log N$) nonzero entries, and has condition number at most poly($\log N$), it is possible for the HHL algorithm to construct a quantum state proportional to the solution $\vec{x}$ in polynomial time even when N is exponentially large.

The problem solved by the HHL algorithm is significantly different from the problem solved by most conventional classical linear algebra packages; the input matrix is given by an oracle rather than explicitly stored, and the output is in the form of a quantum state. Furthermore, exponential speedup is only achievable if the condition number (*i.e.* ratio of largest to smallest eigenvalue) of the input matrix scales as a power of the logarithm of the dimension of the matrix. In short, this quantum algorithm is highly powerful but its applicability has limitations of an unfamiliar type. At present the boundaries of what can and cannot be achieved by quantum algorithms based on the HHL primitive are still being mapped out. Potential applications explored so far range from machine learning to calculating electromagnetic scattering crossections.

5 Conclusion

Shor's polynomial-time quantum algorithms for discrete logarithms and integer factorization imply that the currently widely-deployed public-key cryptosystems will be completely insecure in an era of large-scale universal quantum computers. A number of promising public-key cryptosystems hoped to be resistant to quantum attack have been proposed. These include lattice-based, code-based, and multivariate systems, as well has hash-based signatures. Many of the classical attacks on these schemes, as well as on symmetric-key cryptography, can be sped up slightly through judicious use of Grover's search algorithm. Such quantum attacks do not preclude use of these schemes, although they may dictate increased key sizes. However, there have also been a significant number of quantum algorithms discovered subsequent to Shor's 1994 discovery of a quantum factoring algorithm and Grover's 1996 discovery of a quantum search algorithm. Some of these recent breakthroughs in quantum algorithms have resulted in complete breaks (*e.g.* polynomial-time key recovery attacks) on cryptosystems that were previously expected to be secure against quantum computers. A clear lesson is that continued attention to quantum algorithms research will be crucial to the development of the next generation of public-key cryptosystems.

References

[1] Greg Kuperberg. A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. *SIAM Journal on Computing*, 35(1):170–188, 2005. arXiv:quant-ph/0302112.

[2] T. Laarhoven, M. Mosca, and J. van de Pol. Solving the shortest vector problem in lattices faster using quantum search. In *Proceedings of PQCrypto13*, pages 83–101, 2016.

[3] Scott Fluhrer. Quantum cryptanalysis of NTRU. *IACR Cryptology ePrint Archive*, 2015:676, 2015. `https://eprint.iacr.org/2015/676`.

[4] Daniel J. Bernstein. Cost analysis of hash collisions: Will quantum computers make SHARCS obsolete? In *SHARCS'09 Proceedings of the 4th Workshop on Special-purpose Hardware for Attacking Cryptographic Systems*, pages 105–116, 2009. `https://cr.yp.to/hash/collisioncost-20090517.pdf`.

[5] Wim van Dam, Sean Hallgren, and Lawrence Ip. Quantum algorithms for some hidden shift problems. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 489–498, 2003. arXiv:quant-ph/0211140.

[6] Ivan B. Damgård. On the randomness of the Legendre and Jacobi sequences. *Lecture Notes in Computer Science*, 403:163–172, 1988.

[7] Andrew M. Childs, David Jao, and Vladimir Soukharev. Constructing elliptic curve isogenies in quantum subexponential time. *Journal of Mathematical Cryptology*, 8(1):1–29, 2014. arXiv:1012.4019.

[8] Andrew M. Childs, Richard Cleve, Enrico Deotto, Edward Farhi, Sam Gutmann, and Daniel A. Spielman. Exponential algorithmic speedup by a quantum walk. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003, San Diego, CA, USA*, pages 59–68, 2003.

[9] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93, 2005.

[10] Peter Campbell, Michael Groves, and Dan Shepherd. SOLILOQUY: a cautionary tale. `http://docbox.etsi.org/Workshop/2014/201410_CRYPTO/S07_Systems_and_Attacks/S07Groves_Annex.pdf`, 2014.

[11] Ronald Cramer, Léo Ducas, Chris Peikert, and Oded Regev. Recovering short generators of principal ideals in cyclotomic rings. *IACR Cryptology ePrint Archive*, 2016:313, 2016. `https://eprint.iacr.org/2016/313`.

[12] Jean-François Biasse and Fang Song. On the quantum attacks against schemes relying on the hardness of finding a short generator of an ideal in $\mathbb{Q}(\zeta_{p^n})$. Technical report, University of Waterloo, 2015. `http://cacr.uwaterloo.ca/techreports/2015/cacr2015-12.pdf`.

[13] Natalie Wolchover. A tricky path to quantum-safe encryption. *Quanta Magazine*, 2015. `https://www.quantamagazine.org/20150908-quantum-safe-encryption/`.

[14] Kirsten Eisenträger, Sean Hallgren, Alexei Kitaev, and Fang Song. A quantum algorithm for computing the unit group of an arbitrary degree number field. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 293–302, 2014.

[15] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv:1411.4028*, 2014.

[16] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for solving linear systems of equations. *Physical Review Letters*, 15(103):150502, 2009. arXiv:0811.3171.