6

Cryptanalysis

Computation and communication are secured by cryptography. For example, a user's data can be made private, along with messages that they send or receive, from malicious agents who interfere to try to learn the sensitive information. A set of algorithms collectively called a *cryptosystem* endows the security. The attempt to break security is known as *cryptanalysis*, which has its own set of algorithms. Historically, both cryptography and cryptanalysis considered classical, polynomial-time algorithms as the only realistic ones. The advent of quantum computation forces us to consider attacks via quantum algorithms. Generally, we want to know what is the best algorithm for cryptanalysis, in order to understand the effect on the cryptosystem in the worst case. Quantum attacks can void the security of widely used cryptosystems (see Section 6.1 on breaking cryptosystems). More broadly, quantum cryptanalysis can reduce a cryptosystem's security (see Section 6.2 on weakening cryptosystems), such that it becomes more expensive to implement in a secure manner. While the properties of quantum mechanics can also be used to devise more secure cryptosystems (e.g., quantum key distribution) [121, 834, 1058], we consider this area of cryptography to be outside the scope of the present discussion on quantum algorithms.

The authors are grateful to Matthew Campagna and Samuel Jaques for reviewing this chapter.

6.1 Breaking cryptosystems

Overview

Much of modern cryptography relies on computational assumptions.[1] A cryptosystem is a protocol to achieve some security goal, such as hiding information, ensuring integrity of information, or computing a function correctly. A cryptosystem is secure if, assuming a particular mathematical problem is hard to solve (or assuming the existence of certain functions, e.g., pseudorandom), an adversary cannot compromise the security goal. The earliest such cryptosystems used particular problems from number theory, and variants are widely deployed to this day [605]. These cryptosystems are in the class of public-key or asymmetric cryptography. Public-key cryptography uses key pairs: a private key known only to one user, and a public key that can be widely distributed to allow any user to perform tasks like encryption. In contrast, symmetric-key cryptography uses a single secret key that must be preshared between communicating parties.

Quantum computers use quantum algorithms to solve computational problems, and in some cases they provide a speedup over the best known classical techniques. When they are applied to the underlying computational task in a cryptosystem, a large speedup over classical methods can break the cryptosystem, in that an adversary efficiently learns the secret key or the encrypted information to a non-negligible degree. One of the first discovered and most famous applications of quantum computing is Shor's algorithm [937], which breaks or solves the integer factorization problem, and both the general discrete logarithm problem and the elliptic curve discrete logarithm problem. These problems are believed to be classically hard, and are the basis of security for the most common public-key cryptosystems, like Diffie–Hellman, Rivest–Shamir–Adleman (RSA), and elliptic curve cryptography (ECC). The applications of these public-key cryptosystems include encryption to hide the contents of a message, signatures that prevent tampering and impersonation, and key exchange to generate a key for symmetric-key cryptography [126]. In this section, we restrict our focus to two of the most widely used cryptosystems: RSA and ECC.

Actual end-to-end problem(s) solved

The RSA cryptosystem [875] relies on a user choosing a large number N that is the product of two prime numbers; arithmetic is done modulo N. Denote by $n = \lceil \log_2(N) \rceil$ the number of bits specifying N. Along with the modulus N, two integers e and d are used as exponents, such that $(m^e)^d = m \bmod N$ for all val-

[1] An example of a cryptosystem not requiring computational assumptions is the one-time pad.

ues $0 \leq m < N$. The pair (N, e) is the public key, and the value d is the private key. A message m is encrypted as $m^e \bmod N$. Exponentiation with d performs decryption, recovering m as specified in the relation above. Some applications store the factors of N, which must also be kept private. The user generates e and d using their knowledge of the prime factors of N. First, a suitable e is generated, and then d is computed from the prime factors of N and e using some number-theoretic facts together with the extended Euclidean algorithm. However, if an adversary is able to find the factors of N after the construction by the user, they can also solve for d and thereby decrypt messages. The security of RSA is based on the observed difficulty of factoring large numbers like N, that is, the integer factorization problem.

ECC is based on a different problem, the elliptic curve discrete logarithm problem (ECDLP), which has the advantage of smaller key sizes for equivalent security levels, compared to RSA. Consequently, fewer resources (e.g., communication, complexity of encryption and decryption) are required to implement ECC. Elliptic curves are constructed over a finite field K, as the set of solutions to the equation

$$y^2 = x^3 + ax + b\,, \quad a, b \in K\,,$$

which specify points on an elliptic curve [639, 763]. The set of solutions $P = (x, y)$ forms an abelian group under a specially defined addition operation. Collectively, the set of parameters K, a, b and a so-called base point G (a solution to the equation of prime additive order N, i.e., NG is the additive identity) specify the cryptosystem. A private key is a random integer k satisfying $1 \leq k < N$, and a public key is the value $P = kG$, the result of adding G to itself k times. The assumption of hardness is in the following problem (ECDLP): *Given P and G, where $P = kG$ for some secret value k, find k.* ECC is constructed from the observation that calculating P from k, G is efficient, whereas it is computationally infeasible for an adversary to compute k from points P and G.

Dominant resource cost/complexity

Shor's algorithm [937] solves the number-theoretic problem of order finding: given n-bit positive integer N and x coprime to N, find the smallest integer r such that $x^r = 1 \bmod N$. Factoring was shown to reduce to order finding. In particular, there is an efficient, otherwise classical algorithm, of classical complexity $O(n^3)$ [801], that uses order finding as a quantum subroutine. To describe the quantum algorithm for order finding, let the function f denote modular exponentiation, that is, $f(e) = x^e \bmod N$, and note that f is periodic with (unknown) period r. Also, let L be a large integer such that an interval of

length L contains many periods, that is, $L \gg r$. It can be shown that $L \geq N^2$ is sufficient. There are three steps. First, an equal superposition over the numbers $\{0, \ldots, L-1\}$ is formed and the function f is computed into an ancilla register yielding the state $L^{-1/2} \sum_{e=0}^{L-1} |e\rangle|f(e)\rangle$. Second, a measurement is performed on the ancilla register, which, due to the periodicity of the function f, yields a state $(\lceil L/r \rceil)^{-1/2} \sum_{j=0}^{\lfloor L/r \rfloor} |rj + y\rangle$ for $0 \leq y < r$ a random and unknown integer.[2] Third, a quantum Fourier transform is performed. In the case that L is a multiple of r, the result is

$$\frac{\sqrt{r}}{L} \sum_{j=0}^{L/r} \sum_{z=0}^{L-1} \mathrm{e}^{2\pi \mathrm{i} z(rj+y)/L} |z\rangle = \frac{1}{\sqrt{r}} \sum_{\ell=0}^{r-1} \mathrm{e}^{2\pi \mathrm{i} \ell y/r} |\ell L/r\rangle \,, \tag{6.1}$$

where the equality follows since coefficients of $|z\rangle$ for which z is not equal to $\ell L/r$ for some integer ℓ vanish due to destructive interference. Measurement of this state then produces an outcome $\ell L/r$ for a randomly sampled ℓ. The value of r can be classically computed by dividing the measurement outcome by L and determining the value of the denominator of the rational number that results; repetition may be required since ℓ and r could have common divisors. If L/r is not an integer, the measurement outcome is (with high probability) an integer close to $\ell L/r$ for some integer ℓ. One can deduce the rational number ℓ/r (which allows for the determination of r) from the estimate of $\ell L/r$ by writing it as a continued fractions expansion, with classical complexity $O(n^3)$ [801].

This entire procedure can alternatively be viewed as quantum phase estimation applied to the unitary U that sends $|y\rangle \mapsto |xy \bmod N\rangle$ for all y relatively prime to N, performed with at least $2n$ bits of precision.

The number of qubits for order finding—and hence for Shor's factoring algorithm—is $O(n)$, which stems from the number of bits specifying the problem: the first register has size $2n$, and the ancilla register holding the result $f(e)$ has size n. Naively, the number of gate operations is $O(n^2)$ for the quantum Fourier transform and $O(n^3)$ for implementing the coherent modular exponentiation $|e\rangle|0\rangle \mapsto |e\rangle|x^e \bmod N\rangle$. The $O(n^3)$ arises from decomposing modular exponentiation into $O(n)$ modular multiplications, one for each bit of e—using schoolbook multiplication, the gate cost per multiplication is $O(n^2)$. Implementing this modular arithmetic with reversible circuits represents the bottleneck in the complexity. These circuits are closely related to those in classical computing that have been optimized. Still, improvements can lead to better resource counts [592]. The best scaling in theory is achieved with algorithms that have large prefactors in their complexity, making them impractical to implement except when n is large: total gate complexity of $O(n^2 \log(n))$ is possi-

[2] If $r\lfloor L/r \rfloor + y \geq L$, then the $j = \lfloor L/r \rfloor$ term does not appear in the expression.

ble asymptotically, using integer multiplication with $O(n\log(n))$ scaling [502], although analyses optimized for $n \approx 2048$ use methods for which the total gate complexity scales as $\widetilde{O}(n^3)$ [424]. Alternatively, optimization may be performed to, for example, increase qubit count and decrease gate count or gate depth. For example, an approximate version of the quantum Fourier transform is implemented with $O(n\log(n))$ gates and allows factoring with $O(\log(n))$-depth quantum circuits [298], at the cost of extra overhead in number of qubits and gates; allowing for $O(\log^2(n))$-depth preserves the circuit size $O(n^3)$.

A related approach proposed by Regev [869] for quantum factoring has quantum circuit size of only $\widetilde{O}(n^{3/2})$ gates (assuming fast integer multiplication at cost $\widetilde{O}(n)$), but the circuit has to be run $O(n^{1/2})$ times. Thus, it achieves the same overall asymptotic gate complexity as Shor's algorithm. The idea of the algorithm is to extend period finding to higher dimensions and optimize resource counts. Unlike Shor's algorithm, which is proven to succeed, Regev's approach relies on a number-theoretic assumption, albeit a plausible one. The reduction in quantum circuit depth and natural parallelism of the approach may lead to more favorable resource counts in practice. In particular, a constant fraction of the runs can fail [849], which may allow it to be implemented fault tolerantly with less overhead. Initial work on optimizing the algorithm has established a tradeoff between the number of qubits and the number of gates. Linear qubit cost of $O(n)$ is possible (although the constant prefactor is larger than that of Shor's algorithm) while still maintaining $\widetilde{O}(n^2)$ total gate complexity [849, 848].

Essentially the same quantum algorithm of Shor is readily applied to elliptic curves, as well as the discrete logarithm problem (i.e., find r such that $a^r = b$ for $a, b \in G$ where G is a group), which is also used as a computationally hard problem for cryptography. These applications are all instances of the *hidden subgroup problem*: Find the generators for subgroup K of a finite group G, given a quantum oracle performing $U|g\rangle|h\rangle = |g\rangle|h \oplus f(g)\rangle$, where $f : G \to X$ (X is a finite set) is a function that is promised to be constant on the cosets of K and take unique values on each coset. In the case of period finding, G is the group $\mathbb{Z}/L\mathbb{Z}$ under addition, and the hidden subgroup is $K = \{0, r, 2r, \ldots, L - r\}$ (technically a subgroup only if r divides L); one can verify that $f(g) = x^g \bmod N$ is constant on each coset of K. The procedure outlined above for period finding can be applied to other groups, where it is called "the standard method" [277], which requires generalizing the quantum Fourier transform to arbitrary groups. A simple example is Simon's problem [940]—indeed, historically speaking Simon's algorithm inspired Shor's [938]—where G is the abelian group $(\mathbb{Z}/2\mathbb{Z})^n$ of bit strings of length n under addition, $K = \{0, c\}$ for some hidden bit string c, and the generalized Fourier

transform is simply the Hadamard transform $H^{\otimes n}$. For abelian groups, the hidden subgroup K can be determined with polylog($|G|$) queries to f, but the method does not work for nonabelian groups, such as the symmetric group and the dihedral group.

Existing resource estimates

The minimum recommended key size for RSA is 2048 bits [96]. Optimizations in the circuits [109, 493] and incorporation of hardware constraints [398] have led to decreasing but also more realistic resource estimates. For key size $n = 2048$, an optimized resource estimate was performed in [424], geared toward implementation on a device with nearest-neighbor connectivity in 2D, where logical qubits are encoded with the surface code. Their implementation used roughly $3n \approx 6000$ logical qubits and roughly $0.3n^3 \approx 3 \times 10^9$ non-Clifford gates (a mixture of Toffoli and T gates). Accounting for magic state distillation and routing in 2D, it was shown how the computation could be laid out on a 2D grid of 14,000 logical qubits.

For ECC, the minimum recommended key size to ensure 128-bit security (quantifying the number of attacks needed to learn the encrypted information; see Section 6.2 on weakening cryptosystems for details), is $n = 256$ bits [96] (achieving the same level of security with RSA requires a key size of 3072 bits [175, 876]). For breaking 256-bit ECC, an early resource esimate concluded that around three times fewer logical qubits, and 100 times fewer Toffoli gates are required (compared to 3072-bit RSA) [876]. Similar to factoring, improvements have been made in logical circuit compilation [495] and how this translates into hardware implementations [1030, 450, 695]. In [695], a method was given requiring 1.1×10^8 Toffoli gates and 6000 logical qubits. Additionally, by offloading some of the work to a brute-force classical computation and exploiting a simplification that arises when computing multiple ECC keys in parallel, the total Toffoli count per key was shown to approach 4.4×10^7 [695]. As a conclusion, breaking elliptic curve cryptography is easier than factoring for quantum computers in practice [846], relative to their practical difficulty on classical computers.

The physical resources required to implement these logical circuits fault tolerantly depends on many details of the hardware, including the error rate, the physical gate speed, and the available connectivity. In both cases (2048-bit RSA [424, 476] and 256-bit ECC [1030, 450, 695]), given current hardware schemes restricted to nearest-neighbor 2D connectivity with logical qubits encoded into surface codes, the number of physical qubits is estimated to be on the order of 10 million and the computation runs for at least 3–10 hours (significantly longer than this for platforms with relatively slower physical gate

speeds). For a discussion on how to convert between logical and physical resources, see Part III on fault-tolerant quantum computation. Optimization based on the particular architecture can give improvements to these estimates. For example, if one assumes a logarithmic number of nonlocal links, as could be envisaged in photonic implementations, the estimated runtime can be reduced to less than one minute per 256-bit ECC key [695]. The algorithms considered in the resource estimates above do not achieve the best known asymptotic scaling, which comes at the cost of large constant prefactors.

Caveats

While the popular cryptosystems based on number-theoretic problems are rendered insecure for public-key cryptography, there exist alternatives that are believed to be secure against quantum computers: for example, based on error correcting codes or lattices [126]. These alternative computational problems are believed to be hard for both classical and quantum computers. The National Institute of Standards and Technology (NIST) of the United States has provided standards and encouraged implementation [15]. The class of symmetric-key cryptography (see a standard text [605] for details) involves computations that do not have much structure, and also is not broken by quantum computers.[3] Instead, the number of bits of security is reduced.

Prior experimental demonstrations of Shor's algorithm have used knowledge of the answer in order to optimize the circuit and thus lead to sizes that are experimentally feasible on non-error-corrected devices. Meaningful demonstration should avoid such shortcuts [943], which are not available in realistic cryptographic scenarios.

Comparable classical complexity and challenging instance sizes

The best known classical algorithm for factoring is the number field sieve, which has time complexity superpolynomial in number of bits n: namely, it scales as $O(\exp(p \cdot n^{1/3} \log^{2/3}(n)))$, where $p > 1.9$. With a hybrid quantum-classical algorithm applying amplitude amplification on the number field sieve, $p = 1.387$ can be achieved using a number of qubits scaling only as $O(n^{2/3})$ [128]. Classically, problems of size 795 bits have been factored, taking 76 computer core-years, which distributed in parallel over a cluster took 12 days; the same team then extended the record to 829 bits [175].

Several algorithms attacking elliptic curve cryptography have complexity $O(2^{n/2})$ [1026], leading to the recommended doubling of key size compared to

[3] If the adversary can query the cryptosystem's algorithms in superposition, some symmetric-key cryptography can be broken using period finding to extract the key [599]. However, this capability of the adversary is not considered realistic.

bits of security. In practice, a problem of size 117 bits was solved [127]. The corresponding security is estimated [676] to be about 60 bits, compared to 70 bits for the RSA record above.

Speedup

The number of gates to implement Shor's algorithm is $\widetilde{O}(n^2)$ asymptotically using fast multiplication on large numbers [111]. More practically, without incurring the time overhead and additional storage space of fast multiplication, the scaling is $O(n^3)$. Assuming classical and quantum gates are polynomially related in time complexity, the speedup for solving the factoring problem is superpolynomial, and the speedup for solving the ECDLP is exponential. However, there are no tight lower bounds on the classical complexity of factoring or ECDLP; it remains possible that more efficient classical algorithms could be discovered.

NISQ implementations

The large circuit depth, complicated operations, and high number of qubits needed to implement Shor's algorithm make faithful NISQ implementation challenging. However, there have been several attempts to ease implementation at the expense of losing the guarantees of Shor's algorithm, in the hope that the output is still correct with some nonzero probability, which could be vanishing.

One approach [885] is to simplify several operations and make them approximate. The outcome is that the circuit depth is $O(n^2)$, saving a factor of n [493]. The depth is then about 10^8 to factor a 1024-bit instance of RSA, so for relevant sizes, error correction is still required. Implementation of the approximate algorithm, including experimentally, allowed for the successful factorization of larger problem instances than had been possible before. This approximate version is not NISQ in the usual sense of involving noisy circuits, but rather introduces some uncontrolled approximation error in return for reducing the depth, for the possibility of a useful result. Another approach is to encode the factoring problem in a variational optimization circuit. Again, performance is not guaranteed; moreover, variational optimization applied to generic problems is expected to have, at best, a quadratic improvement compared to classical methods, leaving no hope for breaking cryptography. Classical simulation on small problem sizes shows that the algorithm can succeed [39], as does experimental implementation on a superconducting quantum processor [600]. We emphasize that, generally, these NISQ approaches have no evidence or arguments for scaling to cryptographically relevant system sizes.

Outlook

The existence of Shor's algorithm implies common RSA and elliptic curve schemes are theoretically not secure, and resource estimates have made clear what scale of quantum hardware would break them. While such hardware does not exist currently, progress toward such a device can be used to inform the speed of transitioning to quantum-resistant encryption [263]. Currently, from a hardware perspective, the field of quantum computing is far from implementing algorithms that would break encryption schemes used in practice. The estimates above suggest that the resources required would be millions of physical qubits performing billions of Toffoli gates running on the timescale of hours or days. In contrast, the current state of the art is on the order of one hundred noisy physical qubits, with progress toward demonstration of a single logical qubit. Running fault-tolerant quantum computation requires extra overhead, such as magic state factories (see Chapter 26 on quantum error correction and Chapter 27 on lattice surgery). Thus, the gap between state-of-the-art hardware and the requirements for breaking cryptosystems is formidable. Moreover, a linear increase in key size will increase, for example, the number of Toffoli gates by a power of three, which can be substantial. Therefore, considering the experimental challenges, likely only the most sensitive data will be at risk first, rather than common transactions. Consequently, these highly confidential communications will likely adopt post-quantum cryptography first to avoid being broken. However, insecure protocols often linger in practice, so quantum computers can exploit any vulnerabilities in deployed systems that have not been addressed. For example, RSA keys of size 768 bits have been found in commercial devices (note that such key sizes can already be broken classically [175]). In addition, intercepted messages, encrypted with RSA or elliptic curves, can be stored now and decrypted later, once large-scale quantum computers become available.

The resilience of candidates for post-quantum cryptography is under active investigation. In particular, specialized quantum attacks [274, 831] can reduce the number of bits of security, weakening the cryptosystem. Relaxing to toy variants of relevant cryptosystems in order to find new attacks, quantum algorithms can provide polynomial-time solutions [318]. Classical algorithms have even broken certain candidate cryptosystems [145, 239]. Note that these attacks affect the feasibility of particular proposals, but there exist other post-quantum candidates that have no known weaknesses. With the completion of NIST's standardization process, approved post-quantum cryptography is being rapidly deployed. For example, several popular messaging platforms [645, 378] recently adopted post-quantum key derivation hybridized

with ECC, so that the scheme is secure as long as at least one of the underlying cryptosystems is secure.

A sensitive area that warrants additional discussion is cryptocurrency, since much of it relies on the compromised public-key cryptography based on abelian groups. Moreover, changing the cryptographic protocol of the currency requires that most of the users reach a consensus to do so, which can be challenging to coordinate, even if the technical hurdles of adopting post-quantum encryption are overcome. Cryptocurrency wallets that have revealed their public key (e.g., via a transaction reusing a public key assigned to that wallet previously) can be broken using Shor's algorithm. An attack is also possible during the short time window in which the key is revealed during a single transaction [8]. Different cryptocurrencies have different levels of susceptibility to these types of attacks [97, 98]. Nevertheless, the mining of cryptocurrency is not broken, but only weakened by quantum computers.

6.2 Weakening cryptosystems

Overview

The discovery of Shor's algorithm (see Section 6.1 on breaking cryptosystems) prompted interest in post-quantum cryptography, the study of cryptosystems assuming the presence of large-scale, working quantum computers [126]. While some existing systems retained confidence in their security, others that were broken by quantum algorithms were superseded by those that accomplish the same task, but are believed to maintain a high level of security against quantum attacks.

Even if a cryptosystem is not broken altogether, its degree of security can be weakened by quantum algorithms. The strength of a cryptosystem is typically quantified by the number of bits of security (also called the security parameter), that is, n bits corresponds to guessing the desired information with probability $1/2^n$ and accessing what is being protected. When considering computational assumptions, a simplified definition of the security parameter n is that cryptanalysis requires a computational cost of 2^n operations, captured by the best known attack. *Breaking* a cryptosystem means only an efficient number of operations (i.e., $\text{poly}(n)$) are needed, while an attack that *weakens* a cryptosystem still takes $2^m > \text{poly}(n)$ operations, for some $m < n$.

In contrast to public-key cryptosystems, symmetric-key cryptography was discovered earlier and has fewer capabilities. However, it relies less on the presumed hardness of underlying mathematical problems, and correspondingly

has only been weakened by quantum cryptanalysis, as discussed in more detail below.

Actual end-to-end problem(s) solved

In symmetric-key cryptography, two communicating parties share the same key K, which is used both in encryption Enc_K and decryption Dec_K. As usual, the cryptographic algorithm (Enc_K, Dec_K) is known to everyone, including adversaries. The most significant break of a symmetric-key algorithm is an adversary learning the key, given r pairs of plaintext (the message m) and corresponding ciphertext c (its encryption).[4] Such a pair can be accessed by, for example, forcing a certain test message to be transmitted. Precisely, an input K is sought for which the following function outputs 1:

$$f(K) = ((Enc_K(m_1) = c_1) \wedge \cdots \wedge (Enc_K(m_r) = c_r)),$$

that is, find a key such that all the messages encrypt correctly. A straightforward attack is to use brute force and test every key; in practice, sophisticated classical attacks do not perform better than this approach in asymptotic scaling.

Dominant resource cost/complexity

The main, generic quantum attack is to use amplitude amplification: given a classical algorithm with success probability $O(2^{-n})$ of finding a solution, the probability is increased quadratically to $O(2^{-n/2})$. Thus, applying amplitude amplification to the task of solving for the key, the security of cryptosystems goes from n bits to $n/2$.

The function queried in superposition must be efficient to evaluate with a quantum circuit, which is often the case in cryptography [126]. However, the operations are typically long sequences of Boolean arithmetic. As such, a universal gate set and fault-tolerant quantum computation are still required. To store the key, $O(n)$ register qubits are needed, and many more ancilla qubits are used for the reversible arithmetic.

Existing resource estimates

Consider the Advanced Encryption Standard (AES) [559], a symmetric encryption algorithm that is widely used in cryptosystems, for example, for encrypting web traffic. At a high level, it mixes the plaintext and adds it to the key to obtain the ciphertext. An attack based on amplitude amplification needs

[4] There are more sophisticated attack models: for example, many communicating adversaries may have the goal of compromising one of multiple keys [89]. Correspondingly, there are other definitions of security, but this simple and powerful one generally is considered for quantum attacks.

around 3000–7000 logical qubits [453] for AES-k, where k denotes key size in bits, and $k \in \{128, 192, 256\}$. For these sizes, the number of necessary problem instances r is three to five. While the number of logical qubits roughly doubles going from AES-128 to AES-256, the number of T gates goes from $2^{86} \approx 10^{25}$ to $2^{151} \approx 10^{45}$. More recent work [566] allows for higher qubit counts (by about 70%) in exchange for much lower Toffoli depth, such that their product is reduced. Such optimization resulted in about 99% reduction in this metric.

Caveats

Since the quantum attack only halves the exponent in the complexity, a simple fix is to double the key length, for example, by adopting AES-256 instead of AES-128. This modification results in increased, but usually tolerable, cost in implementation (i.e., complexity of encryption and communication resources). In addition, there exist cryptosystems with an information-theoretic security guarantee, assuming adversaries with unlimited computational power, which covers against quantum attacks [126].

Furthermore, it is important to note that to realize the full quadratic benefit of amplitude amplification, the $O(2^{n/2})$ function queries must be performed in series. In contrast, classical brute-force attacks can exploit the parallelism available in high-performance classical computers, potentially increasing the value of n for which a quantum approach would be advantageous over classical methods.

Comparable classical complexity and challenging instance sizes

Classical algorithmic attacks on AES have reduced the security by only a few bits [160]. More practical are side-channel attacks, which make use of physical byproducts, such as energy consumption. For example, when comparing bits between a key and another string, a flipped value can result in logic that increases energy consumption, compared to the same value where nothing happens. The two cases are distinguished and information about the key is learned. Currently, 128 bits of security is roughly the minimum recommended amount [95]. The use of parallelization forces the adoption of relatively large key sizes, compared to what is necessary for a single processor ($\sim$ 60 bits).

Speedup

The basic speedup is quadratic: $O(\sqrt{N})$ function evaluations compared to $O(N)$ classically, where N denotes the number of possibilities for the key; that is, $n = \lceil \log_2(N) \rceil$. However, the function queries in amplitude amplification cannot be parallelized. Then, the evaluation time of the function sets a bottleneck [126]. That is, the problem size is limited by the number of function

evaluations T that can be run in an acceptable period of time. For $\sqrt{N} > T$, employing p parallel quantum processors, each executes $T = \sqrt{N/p}$ evaluations. Then, $p = O(N/T^2)$ and the total number of evaluations is $pT = O(N/T)$, whereas classically, the number of processors is $O(N/T)$ and total evaluations is $O(N)$. The advantage is a factor of T, which is the bottleneck, rather than the larger $\sqrt{N}$. However, the advantage can be overshadowed by faster or cheaper classical processing. That is, if classical computers evaluate the function T times faster than quantum processors, there is no advantage in runtime with using the quantum device. Furthermore, this argument assumes the same cost of parallelization for classical and quantum, which is optimistic for quantum devices. An example of this effect is in mining cryptocurrency [8]: while a quantum computer needs quadratically fewer attempts to succeed, the development of fast, specialized, classical hardware negates the advantage. Essentially, for brute-force attacks, parallelization has the most significant impact in cryptanalysis.

NISQ implementations

The key can be encoded as the ground state of a Hamiltonian, and then variational methods can be applied to solve for it. The scaling is expected to be the same as amplitude amplification. However, since the variational algorithm does not have a set time complexity, the solution may be found much slower or faster [1023]. If the fluctuations are large enough, they can potentially pose a challenge to cryptography, which makes worst-case guarantees. However, there is no reason to expect that the success probability will scale favorably with key size and compromise security in practice. Another approach is to use amplitude amplification, but adapt it to near-term devices, so that the NISQ-optimized versions perform better in real experiments [1083].

Outlook

Here, we focused on the example of symmetric-key encryption. Nonetheless, the effect of amplitude amplification to halve the effective bits of security is generic for computational problems, assuming efficient construction of the oracle. From the cryptographic standpoint, this attack is mild and can be counteracted by doubling the number of bits of security in the scheme. In practice, the increase in key size can be unwieldy in certain applications, such as cryptocurrencies, but fundamental security is not threatened.