Postquantum Cryptography

Sophie Stevens

October 2024

If computers that you build are quantum,
Then spies everywhere will all want 'em.
Our codes will all fail
And they'll read our email,
Till we get crypto that quantum, and daunt 'em.
- Jennifer and Peter Shor

1 Introduction

The advent of quantum computing represents huge strides for humanity, but challenges for cryptography. Quantum computing is fundamentally different from classical computing: in a classical computer, operations are performed on bits that are deterministically 0 or 1; a quantum computer performs operations on *qubits*. We can think of a qubit as a probabilistic combination of 0 and 1, whose binary value is only fixed upon measurement. This means that we can perform operations on a superposition of 0 and 1 before we measure the outcome. A consequence of this is that we can compute a function on several different inputs simulatenously (although we will only see one of the possible corresponding outputs upon measuring). This means that there are some functions that we think are hard to compute on a classical computer that we know are easy to compute on a quantum computer. This sentence needs some caveats: firstly, by 'we think are hard', we mean that it's a problem that we have studied for decades and don't know how to solve on a classical computer efficiently (i.e. in polynomial time); secondly 'easy to compute on a quantum computer' assumes that our real quantum computer functions as its ideal abstraction – in reality, we must take into account the effect of noise and errors.

2 Shor's algorithm and Grover's algorithm

There are two quantum algorithms that are most relevant for cryptography today: Shor's algorithm and Grover's algorithm.

Grover's algorithm is a very generic search algorithm. It makes a brute-force attack quicker, because you search inputs until you find one that matches the output you're looking for.. It is optimal in the sense that no search algorithm on unstructured data will perform asymptotically quicker. A classical unstructured search takes $O(N)$ evaluations, whereas Grover's takes $O(\sqrt{N})$ evaluations we have a quadratic speedup.

Shor's algorithm attacks the algebraic structure of some schemes by using quantum mechanics (specifically the power of the Quantum Fourier Transform) for order finding: given $g^x \pmod{y}$ recover x. This is directly applicable to Diffie-Hellman (and it's in an arbitrary group, so automatically extends to Elliptic Curve Diffie-Hellman); using a number theory trick (difference of two squares), it's also applicable to RSA. Classically factoring is assumed to be subexponential, whereas Shor's algorithm is polynomial (i.e. to factor an integer N takes times $O(\text{poly}(\log N))$.

2.1 Impact on symmetric cryptography

As symmetric cryptography does not have much algebraic structure (or at least if it does, we haven't yet discovered it), Shor's algorithm does not affect it. Grover's algorithm however does – searching the key space is now significantly quicker. To fix this, we need longer keys. As an overly cautious solution, we could double the key size (to account for the square root loss of security). However, the actual details behind algorithms such as AES means that this gives us more security than the classical counterpart.

2.2 Impact on asymmetric cryptography

Grover's algorithm also applies to asymmetric cryptography because of its genericity. However, a brute-force attack is rarely the state-of-the-art attack of a public key system. Thus, although we must of course consider the effect of Grover, in reality there's a far greater issue at hand: Shor's algorithm.

Shor's algorithm completely destroys the security of Diffie-Hellman. That's because given a public key $g^x \pmod{p}$, Shor's algorithm recovers the order of the public key, namely x. This can be extended to any group, so Elliptic Curve Cryptography is also not a post-quantum candidate.

Shor's algorithm also destroys the security of RSA. At a very high level, we sample an element x and consider its order mod N. We do this until we find an element x with even order r so that $x^{r/2} \pmod{N} \neq -1$. Then we find that $x^r - 1 \equiv 0 \pmod{N}$ so that $(x^{r/2} - 1)(x^{r/2} + 1) \equiv 0 \pmod{N}$; using Euclid's algorithm we find a non-trivial factor of N by considering the gcd of $(N, x^{r/2} - 1)$ or $(N, x^{r/2} + 1)$.

3 Postquantum cryptography - KEMs

A KEM is a key encapsulation mechanism: at the end of the process, both parties should share a key.

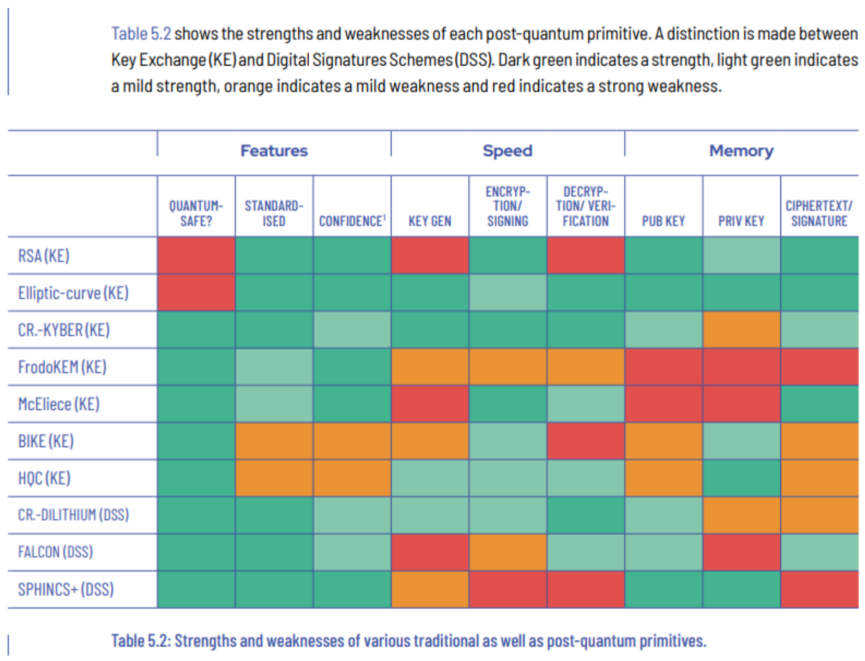

Table 5.2 shows the strengths and weaknesses of each post-quantum primitive. A distinction is made between Key Exchange (KE) and Digital Signatures Schemes (DSS). Dark green indicates a strength, light green indicates a mild strength, orange indicates a mild weakness and red indicates a strong weakness.

	Features			Speed			Memory		
	QUANTUM-SAFE?	STANDARD-ISED	CONFIDENCE[1]	KEY GEN	ENCRYP-TION/ SIGNING	DECRYP-TION/ VERI-FICATION	PUB KEY	PRIV KEY	CIPHERTEXT/ SIGNATURE
RSA (KE)	🟥	🟩	🟩	🟥	🟩	🟥	🟩	🟢	🟩
Elliptic-curve (KE)	🟥	🟩	🟩	🟩	🟢	🟩	🟩	🟩	🟩
CR.-KYBER (KE)	🟩	🟩	🟢	🟩	🟩	🟩	🟢	🟧	🟢
FrodoKEM (KE)	🟩	🟢	🟩	🟧	🟧	🟧	🟥	🟥	🟥
McEliece (KE)	🟩	🟢	🟩	🟥	🟩	🟢	🟥	🟥	🟩
BIKE (KE)	🟩	🟧	🟧	🟧	🟢	🟥	🟧	🟢	🟧
HQC (KE)	🟩	🟧	🟧	🟢	🟢	🟢	🟧	🟩	🟧
CR.-DILITHIUM (DSS)	🟩	🟩	🟢	🟢	🟢	🟩	🟢	🟧	🟧
FALCON (DSS)	🟩	🟩	🟢	🟥	🟧	🟢	🟢	🟥	🟢
SPHINCS+ (DSS)	🟩	🟩	🟩	🟧	🟥	🟥	🟩	🟩	🟥

Table 5.2: Strengths and weaknesses of various traditional as well as post-quantum primitives.

Figure 1: This figure from the Netherlands National Communications Security Agency https://publications.tno.nl/publication/34641918/oicFLj/attema-2023-pqc.pdf nicely summarises the PQC options we have today.

3.1 NIST standardisation process

Lattice-based cryptography The ultimate 'winner' of the process - Kyber (now called ML-KEM) was the scheme chosen as the KEM for standardisation, and its corresponding signature scheme (Dilthium) also chosen. More on lattices in a future lecture.

- Hardness: Shortest vector problem (find a short vector in a high-dimensional lattice)
- Practical: fastest encryption
- At NIST Level 3 parameters (Kyber):
  - 1184 bytes for a public key
  - 2400 bytes for a secret key
  - 1088 bytes for a ciphertext

Code-based cryptography The first code-based public-key cryptosystem was introduced in 1978 by McEliece.

A linear code $\mathcal{C}$ is a k-dimensional linear subspace in $\mathbb{F}_q^n$. This means that if $c, d \in \mathcal{C}$ and $\alpha, \beta \in \mathbb{F}_q$, then $\alpha c + \beta d \in \mathcal{C}$.
An example of a code in $\mathbb{F}_2^3$ is vectors of weight 2: the codewords are $\{110, 101, 011\}$ and $110 + 101 = 011$, $101 + 011 = 110$, $110 + 011 = 101$.

A t-error correcting (binary) code means that any pair of 'codewords' (elements of the subspace) differ in at at least $2t + 1$ elements.
Our above example is not error correcting - for example, if we see one error and get 111, then we don't know which codeword it originally was. Compare this to the triple-repetition code: codewords are 111 and 000; if we experience a single error we know which codeword ought to have been transmitted.

In Classic McEliece, we use a random binary Goppa code. The 'binary' part means that it's defined over $\mathbb{F}_2$; the 'Goppa' part means that it the code is specified in a special algebraic way (corresponding to an irreducible polynomial over $\mathbb{F}_{2^m}$ of degree t), that makes the scheme secure and efficient. A binary Goppa code is t-error correcting - this means that any code word is separated by Hamming distance $t/2$.

To encrypt, pick $n = 2^m$ and t and randomly select an irreducible[1] polynomial g of degree t over $\mathbb{F}_{2^m}$ and a sequence of elements $L_1, \ldots, L_k \in \mathbb{F}_{2^m}$; from this we construct a code: codewords are elements that satisfy:

$$\left\{ c \in \{0,1\}^k : \sum_{i=1}^{k} \frac{c_i}{x - L_i} = 0 \mod g(x) \right\}.$$

Because codewords form a vector space, they have a basis, from which we can extract a $k \times n$ generator matrix G (which we can put into canonical form after

[1]Irreducibility can be relaxed to g having no repeating roots

a basis change, so that the top left of G is the $k \times k$ identity matrix). This is computationally very straightforward.

The public key is a random 'scrambling' of G: a random dense non-singular matrix S and a random permutation matrix[2] P is chosen; the public key is $G' = SGP$. This is essentially a 'bad' basis of G.

To encode a message m, we choose an 'error' vector z containing exactly t ones, and send $c = mG' + z$.

The private key is a basis of the code that enables efficient decoding (using 'Patterson's algorithm'). It enables an efficient way to recover m given c.

- Hardness: decoding problem for linear codes is NP-complete. BUT this harness is only true for an arbitrary linear code, whereas a Goppa code has structure.
- McEliece is one of the oldest schemes – no progress has been made on the decoding problem for Goppa codes, which lends weight to security.
- Practical: Short ciphertexts - importantly under 256 bytes; this is great for fiting ciphertexts inside single network packets.
- At NIST Level 3 parameters (Classic McEliece):
  - 524160 bytes for a public key
  - 13608 bytes for a secret key
  - 156 bytes for a ciphertext
- Problem: Size of public key is between 0.25MB and 1.35MB. For context, the plaintext version of *Little Women* comes in at 1MB, so the public key requirement of McEliece demands exchanging novels in advance.

BIKE (Bit FlIpping Key Encapsulation) and HQC (Hamming quasi-cyclic codes) are other examples of code-based schemes that are currently alternate candidates in the NIST competition. Because the codewords have different structures, they use different decoding algorithms.

BIKE has shorter keys (significantly shorter public key, slightly shorter private key) but larger ciphertexts; HQC has small public and private keys (similar in size to BIKE's public key) but a larger ciphertext again.

4 Signature Schemes

There are three signature schemes that were selected for standardisation: Crystals-Dilithium (LWE-based), Falcon (NTRU-based) and SPHINCS+ (hash-based). None of the candidates are ideal drop-in replacements, so NIST have created another standardisation process aiming to seek more diversity in signature schemes.

[2] A permutation matrix is a binary matrix with exactly one non-zero entry in each row and each column.

Lattice-based Not really adding diversity

Symmetric-based Signatures Promising - SPHINCS+ has been standardised, but it's quite big (because it's using Merkle tables)

MPC-in-the-Head Signatures MPC = Multi-party computation. This is a way to create a zero-knowledge proof for a relation R, given a secure multiparty computation protocol to compute R.

Multivariate Signatures Multivariate cryptography suffered a catastrophic reputation blow with the break of Rainbow, a finalist in the NIST competition. However, other Multivariate Signature schemes could still be promising....

Isogeny-based cryptography Isogeny-based cryptography is a relatively new field. However, the final round of the process saw a devastating break for the isogeny-based scheme SIKE that was put forward for standardisation. What does this mean for isogeny schemes? Find out in a future lecture!

Additional Reading[3]

[1] Martin R Albrecht et al. "Classic McEliece: conservative code-based cryptography". In: (2022). URL: https://inria.hal.science/hal-04288769/document.

[2] Nicolas Aragon et al. "BIKE: bit flipping key encapsulation". In: (2022). URL: https://inria.hal.science/hal-04278509/document.

[3] Robert J McEliece. "A public-key cryptosystem based on algebraic". In: *Coding Thv* 4244 (1978), pp. 114–116. URL: https://ipnpr.jpl.nasa.gov/progress_report2/42-44/44N.PDF.

[4] Carlos Aguilar Melchor et al. "Hamming quasi-cyclic (HQC)". In: *NIST PQC Round* 2.4 (2018), p. 13. URL: https://pqc-hqc.org/doc/hqc-specification_2023-04-30.pdf.

[3]NIST host a series of seminars about the PQC standardisation process - here you can find seminars on BIKE/McEliece/HQC, which might help with the additional reading. Link: https://csrc.nist.gov/Projects/post-quantum-cryptography/workshops-and-timeline/pqc-seminars