

Post-Quantum Cryptography

Quantum computing is increasingly seen as a threat to communications security: rapid progress towards realizing practical quantum computers has drawn attention to the long understood potential of such machines to break fundamentals of contemporary cryptographic infrastructure. While this potential is so far firmly theoretical, the cryptography community is preparing for this possibility by developing Post-Quantum Cryptography (PQC), that is, cryptography resisting the increased capabilities of quantum computers.

In this white paper, we explore the background, impact, and urgency of this threat, and summarize the cryptographic schemes being evaluated. We also provide recommendations on what steps should be taken today to be prepared for the changes to come, and discuss how Arm is approaching PQC.

In two technical appendices, the interested reader learns about how quantum computers could break RSA and ECC and gets an overview of the main ideas behind lattice-based cryptography, a promising candidate for quantum safe cryptography.

Short on time? See → [What to do now?](#) for concrete guidance.

How cryptography secures today's network traffic – A primer

To understand the impact of quantum computing on cryptography, it is useful to have an overview of the different kinds of cryptography mainly used to secure today's network traffic. This section provides a short summary.

Secure communication over insecure links

The breadth of today's connectivity technologies (such as BLE, Cellular, WiFi) allows communication to happen almost everywhere, between everyone and everything. However, while these technologies provide channels for the exchange of information, those channels are *a priori insecure* – that is, they lack some or all of the following properties:

- **Confidentiality:** Nobody except the designated communicating parties can infer anything about the information that is being exchanged.¹
- **Integrity:** Information cannot be modified in transit without the modification being detected.
- **Authentication:** The parties know whom they talk to.

If we call a channel with the above properties *secure*, the problem thus arises to find constructions for secure channels from the insecure links provided by the various connectivity technologies. Cryptography can be viewed as offering tools for a variety of such constructions², as we will recall now.

Symmetric Cryptography

Symmetric cryptography builds secure channels from the assumption of a single piece of pre-shared confidential information, the *symmetric key*:

- *Symmetric ciphers* such as AES or ChaCha construct *confidential* channels from a symmetric key.
- *Message authentication codes* (MACs) such as the hash-based HMAC or the cipher-based CMAC construct *authenticated* and *integrity protected* channels from a symmetric key.
- Finally, combined *Authenticated encryption* schemes such as AES-GCM or ChaCha/Poly establish *secure* channels from a symmetric key.

¹Except for a bound on the amount of information that has been communicated.

²This is also called *Constructive Cryptography* [Mau12].

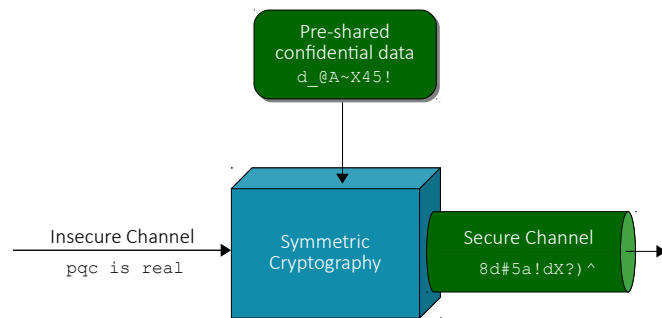


Figure 1: Symmetric Cryptography as a black box transforming an insecure channel into a secure channel on the basis of pre-shared confidential data.

Historically, secure communication could *only* be established via symmetric cryptography, that is, on the basis of a pre-shared secret piece of information which the communicating parties had exchanged upfront. The primary problem with this approach, of course, is the need to establish the shared symmetric key in the first place.

Public Key Cryptography

*Public Key Cryptography*³ builds secure channels without any confidentiality assumption (such as the existence of a pre-shared symmetric key).

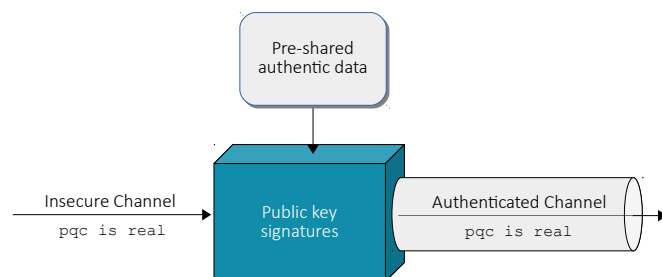


Figure 2: Public Key Signatures as a black box transforming an insecure channel into an authenticated channel on the basis of pre-shared authentic data.

- *Signature schemes* such as RSA-PSS, ECDSA or EdDSA construct *authenticated* and *integrity protected* channels from a piece of *authentic* public data associated with each communicating party, called the *public key*.
- *Key establishment protocols* such as (EC)DHE construct shared *confidential* data from *authenticated* and *integrity protected* channels.

Note that it is highly remarkable that such constructions exist in the first place: for example, a key establishment protocol resembles a magic conversation by which the two communicating parties agree on something that a passive listener cannot figure out.

³Public Key Cryptography is also called *asymmetric cryptography*.

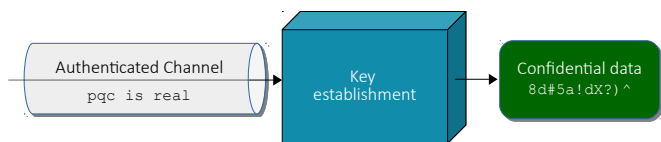


Figure 3: Key Establishment as a black box transforming an authenticated channel into shared confidential data.

The means to reliably distribute public keys is the *public key infrastructure (PKI)*. The most prevailing PKI are X.509 certificate chains, which bootstrap public key distribution using signatures and a small number of public keys distributed out-of-band.

Putting it together: TLS and friends

Combining the constructions that symmetric and public key cryptography provide, one arrives at the following two-step scheme for establishing secure communication channels over insecure links:

- An initial *authentication and key establishment phase* based on public key cryptography authenticates one or both parties in the communication and establishes a shared secret between them.
- The actual communication happens in the *bulk encryption phase*, where the established symmetric key is used to construct a secure channel using symmetric cryptography.

For example, this approach is taken by the popular *Transport Layer Security (TLS)* protocol, and following TLS terminology the initial key establishment phase is often called *handshake*. In this language, the above two-step approach essentially says: Shake hands first, then talk.⁴

(How) Do we know it is secure?

Cryptographic schemes are not usually proved secure in absolute terms. Instead, modern cryptography develops precise notions of security and uses those notions to formally reason how the claimed security of a particular scheme follows from an underlying *hardness assumption*. Such hardness assumption is usually the statement that a particular computational problem cannot be solved efficiently,

⁴In addition to signatures and key establishment, there are also public key *encryption* schemes such as RSA-OAEP which could theoretically be used to protect the actual traffic, but their inferior performance compared to symmetric primitives renders this approach impractical. Instead, where still in use, public key encryption realizes the key establishment phase in the above hybrid strategy, with one party choosing a secret and sending it to the peer after encryption with the peer's public key. This approach, however, comes at the danger of the sending side choosing insufficiently random secrets, and is nowadays mostly replaced by *Key Encapsulation Mechanisms (KEM)*, which combine *random* key generation and public-key encryption into a single primitive.

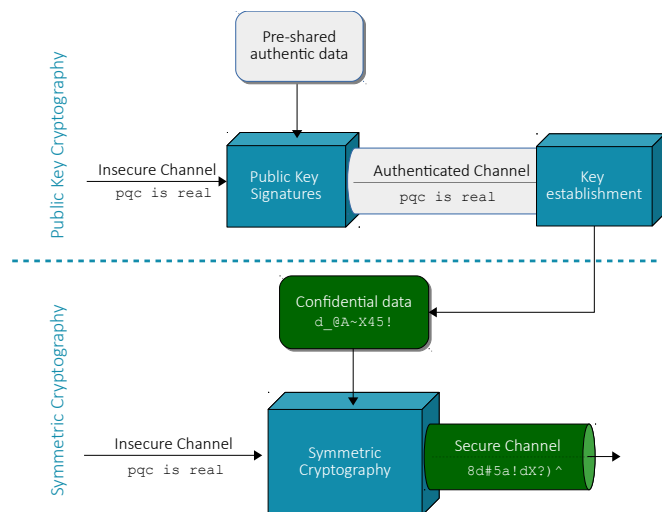


Figure 4: Combining public key cryptography and asymmetric cryptography to establish secure communication channels from insecure ones.

and trust in its validity derives from time and effort spent on them, while a formal proof of their hardness is considered infeasible. For example, schemes from the RSA family rely on the hardness of integer factorization for which no polynomial time algorithm is known.

It is common for cryptographic schemes to have a finite lifetime due to gradually decreasing practical security resulting from advances in algorithmic research and increased computational power:

- The hash function SHA-1 was standardized in 1995, a theoretical collision attack was found in 2004 and finally put to practice with increased compute power in 2017.
- Increasingly large RSA Factoring Challenges are being solved (the latest one being the factorization of an 829-bit modulus in February 2020) raising the bar for what is considered to be a safe use of RSA.

This gradual degradation of security is somewhat expected and can sometimes, as in the example of RSA, be counteracted by increased key sizes of the underlying schemes.

The impact of quantum computing on cryptography, however, is of a different nature: the reason why the empirical evidence of security is crumbling with the advent of quantum computing is that the latter introduces entirely new computational capabilities that haven't been considered by classical algorithmic research, and whose limits are hence still to be understood.

How quantum computing impacts cryptography

A new computational model

Quantum computing uses phenomena from quantum physics to perform computation, and the *quantum computational model* is an abstraction of the state and capabilities of such quantum computations which allows us to ignore the physical details and their realizability – in the same way as, for example, the model of Turing machines provides a way to study classical computation independently from its physical realization. This opens a new field of quantum algorithmic research, and specifically the question arises to what extent the quantum computational model allows for algorithms solving problems in fewer operations than classical algorithms.

The impact on public key cryptography

In 1994, Peter Shor [Sho94] discovered that the quantum computational model allows for the construction of an efficient algorithm for integer factorization, a highly remarkable insight for multiple reasons:

- Theoretically, it demonstrated the superiority of quantum computing by exhibiting a problem for which no efficient classical algorithm is known but which a hypothetical quantum computer *can* solve efficiently.
- Practically, it is remarkable that the problem demonstrated to be amenable to significant speedup on a quantum computer happens to be the problem underlying the widely used RSA cryptosystems.

Moreover, Shor also exhibited an efficient quantum algorithm for the discrete logarithm problem underlying the popular (EC)DHE key exchange mechanisms.

As a result, Shor's algorithms therefore demonstrate that the most popular public key cryptosystems which our ability to establish secure communication channels relies upon, are no longer secure within the hypothetical model of quantum computing. This discovery triggered interest in quantum algorithms and the question of what it takes to actually build a quantum computer.

It is important to note that increasing key sizes, which can be an acceptable response to gradual degradation of practical security due to increased compute power, does *not* apply to make RSA and (EC)DHE quantum safe: Shor's algorithm puts those problems in a different complexity class altogether, and keys scaled to accommodate for this would render the schemes impractical.⁵

⁵This was entertainingly demonstrated by the pqRSA (post-quantum RSA) proposal, which uses multi-GB keys.

For the interested reader, [Appendix A: How quantum computers threaten RSA & ECC](#) provides a high-level description of the quantum computational model and an outline of how Shor's algorithm breaks RSA and (EC)DHE.

The impact on symmetric cryptography

The impact of quantum computing on *symmetric* cryptography appears to be less critical:

Grover's search algorithm [Gro96] is a quantum algorithm performing an unstructured search over n objects in \sqrt{n} steps – a quadratic speedup over the classically optimal n steps for a full traversal. This implies that brute force attacks to cryptographic schemes, such as searching through an entire key space, are potentially quadratically faster on a quantum computer than on a classical computer. As a consequence, key sizes might need to be doubled in the presence of quantum computers, for example by using AES-256 in place of AES-128. Beyond that, however, the techniques established to construct symmetric ciphers appear to remain valid even in the face of quantum computing. The same applies to the security of hash functions: there are quantum algorithms for collision search which are potentially polynomially faster than classical algorithms, but the schemes and techniques themselves appear to remain valid.

Summary

Today's network communication is secured through the combined use of public key cryptography and symmetric cryptography: the former establishes authentication and shared secrets, the latter performs the bulk encryption.

Quantum computing threatens the main public key cryptosystems in use today (RSA and ECC), while it seems that it will affect symmetric cryptography (such as AES or the SHA) only in a minor way.

Assuming that practical quantum computers can be built – the feasibility of which we will talk about below – it will therefore be necessary to find and deploy cryptosystems for authentication and key establishment that withstand the quantum computational model. The development of such “quantum safe” cryptography is called *Post-Quantum Cryptography* (PQC).

Post-Quantum Cryptography vs. Quantum-Cryptography

Post-Quantum Cryptography is to be distinguished from *Quantum Cryptography*, which concerns cryptographic algorithms which make use of quantum phenomena. Post-Quantum Cryptography, in turn, is concerned with algorithms that *run* on a classical computer and cannot be *broken* even with a quantum computer. We will not discuss quantum-cryptography in this paper. The study of quantum algorithms attacking cryptographic schemes is called *quantum cryptanalysis*.

(When) Does it matter?

Today's public key cryptography will need replacing in the face of practical quantum computing. But when do we need to act, and will we be ready? Answering these questions mainly depends on three factors:

- The characteristics of the system to be protected.
- The progress towards practical quantum computing.
- The development of Post-Quantum Cryptography.

The present section describes each aspect in more detail, but here is the bottom line, in terms of confidentiality:

Suppose we require confidentiality of our data for c years and that quantum computers may break RSA/ECC in q years. If $c > q$, we're in trouble *today*. Otherwise, we need to transition to PQC within the next $q - c$ years. So, if s is the time it takes to bring PQC to life, we are in trouble if $s > q - c$. Or, in the words of Michele Mosca: If $c + s > q$, then worry.

So, should we worry? The confidentiality requirement c depends on the data, and estimates for q and s are hard to obtain – we'll go into details below. However, cautious estimates such as $c = 20$ (data confidentiality for 20 years) and $s = 10$ (PQC standardization and deployment in 10 years) show that even with $q = 30$ (quantum computers breaking RSA and ECC in 30 years) it's time to think about and prepare the transition to PQC *now*.

What are you protecting?

Confidentiality of data in transit

Data sent over an encrypted channel may be intercepted, stored and retroactively decrypted if key material is later exposed through information leakage or through the advent of an effective attack against the underlying cryptographic scheme. Confidentiality is therefore at risk as soon as an attack against the underlying cryptography is conceivable within the timeframe during which the data needs to stay confidential.

It is important to understand that preventing such a breach of confidentiality requires changing cryptographic systems *ahead of time*, well before they can actually be practically attacked.

Example: Suppose a system uses a security protocol run over an insecure channel, such as TLS over the internet. Assume that the data has to remain confidential for 20 years, and that a practical threat to the protocol or the underlying cryptography is conceivable in 25 years. Then the system needs to be upgraded no later than 5 years in the future to maintain confidentiality of the data it will handle at that point. If the data has to remain confidential for more than 25 years, it is at risk *today* and systems should be upgraded immediately.

Concretely, quantum computing puts the security of today's prominent key establishment mechanisms RSA and (EC)DHE at risk. Those schemes must therefore be replaced as soon as the data they protect is required to be confidential for longer than the minimal feasible time for quantum computing to become practical. We will look into this in [Practicality of quantum computing](#) below.

Confidentiality of data at rest

In addition to the security of ephemeral encryption used in protocols such as TLS, we have to consider the confidentiality of encrypted data *at rest*. Such data will usually be encrypted with a *symmetric* encryption scheme such as AES, and as discussed in [The impact on symmetric cryptography](#), those schemes are currently expected to remain suitable in principle in the face of quantum computing, but may require an increase of key sizes.

Confidentiality of encrypted data at rest is thus at risk if both (a) and either (b.1) or (b.2) hold in the following:

- (a) An attacker has gained access to the encrypted data itself (for example, through a data breach) or intercepted a non quantum safe channel through which the encrypted data was communicated (for example, an RSA/ECC based TLS communication between systems storing the encrypted data).
- (b.1) The attacker has also intercepted a non quantum safe channel establishing or communicating the symmetric key protecting the encrypted data at rest.
- (b.2) The symmetric encryption scheme uses key sizes which could make it vulnerable to polynomially improved quantum algorithms such as Grover's algorithm – for example, AES-128.

Preventing (b.1) is an instance of protecting data in transit: at some point *before* the lifetime of the data at rest surpasses the time towards practical quantum attacks, it needs to be re-encrypted with a key that is established and communicated solely through quantum safe cryptographic mechanisms. This also prevents (b.2) if the re-established keys are sufficiently long. Note that in this approach, the encrypted data chunks themselves need not be considered confidential.⁶

Data access

An authentication mechanism may fail to provide access control in the future if key material is exposed or if an attack against the underlying primitive is found. Concretely,

⁶An alternative approach is to treat the encrypted data itself as confidential and only exchange it either out of band or through quantum safe channels. In this case, it is sufficient to re-encrypt the data with a key established and communicated through quantum safe mechanisms at any point prior to quantum computing practically threatening today's public key cryptography.

quantum computing threatens today's prominent signature schemes RSA and (EC)DSA, so it must be ensured that those schemes are no longer in use when quantum computing has become practical.

To mitigate the threat to authentication, authentication software should be kept upgradable to allow replacement once practical attacks become available.

Note that in contrast to the threat to confidentiality, it is not necessary to upgrade the software ahead of time. However, where software upgrades are not possible, a quantum safe authentication mechanism needs to be deployed today.

Example: Consider a long-lived IoT device. To allow software to be patched over time the system allows remote firmware upgrade. The firmware itself has to be signed by the vendor to be accepted by the device, and special firmware verification code is responsible for checking the validity of the signature. If this firmware verification code itself is immutable, a quantum safe signature mechanism needs to be deployed *today*. Section [Problem: Firmware updates](#) elaborates on this.

Practicality of quantum computing

Quantum computing received widespread attention when Google [\[Aru+19\]](#) announced *quantum supremacy*: they built a 54-qubit quantum computer capable of performing an artificial computational task in 200s for which it was argued that no available classical computer would be able to solve it in less than 10,000 years.

While quantum supremacy was an important milestone, today's quantum computers are still far away from being able to break RSA or ECC. Specifically, from an engineering perspective, the following problems need to be overcome to reach the point where quantum computers can run complex quantum algorithms such as Shor's:

- (a) Controlling a larger number of physical qubits.
- (b) Controlling errors accumulated during operation via *quantum error correction*, ideally giving rise to error-free *logical* qubits built from a set of physical qubits.
- (c) Providing quantum random access memory (QRAM) for intermediate results and for efficient conversion of classical data into quantum states.

Solving those challenges is going to be a very expensive, multi-year process, funding and enthusiasm for which will also depend on when "practical" quantum supremacy will demonstrate that quantum computing can solve computational problems of commercial value. Interestingly, the very transition to Post-Quantum Cryptography might be a weakening force in the development of quantum computing which PQC protects against.

Ultimately, there are too many technical, economical and political unknowns to allow for an accurate prediction of when we might see quantum computers capable of running Shor's algorithm, but estimates range from 10 years in the worst case to 30 years or more in the best case. While this might seem far away, it is important to realize that dimensions of the same order of magnitude apply to the confidentiality requirements of data and communications – as discussed in the previous section [What are you protecting?](#) – and to the development and transitioning to Post-Quantum Cryptography – as we'll discuss in the next section.

We highly recommend [\[NM19\]](#) for further reading.

Availability of quantum safe cryptography

Given a system's security requirements, as well as a guess for when quantum computing could break RSA/ECC, we infer when Post-Quantum Cryptography needs to be available and put in place. For example, if we predict a quantum computer running Shor's algorithm in 30 years, and we would like our data to remain confidential for 25 years, Post-Quantum Cryptography should be made available in the next 5 years. But what does availability entail?

Bringing cryptography to life

Introducing cryptographic change is a long-lived and multifaceted process, encompassing at least the following:

- (a) Cryptographic research
- (b) Standardization
- (c) Development of secure implementations
- (d) Platform development (e.g. accelerators or ISA)
- (e) Integration into existing infrastructure
- (f) Public Awareness
- (g) Education
- (h) Deployment

What's more, most of those aspects multiply with the number of proposals for "quantum safe" cryptography, of which there are dozens, as we will see. In other words: developing and transitioning to PQC is a lot of work.

In order to guide and structure these parallel streams of research on Post-Quantum Cryptography and narrow down the range of PQC primitives, standards bodies have launched various projects, working groups and competitions around PQC, the most prominent of which is the NIST PQC project which we will look into next.

NIST PQC project

Overview

In December 2016, the US National Institute of Standards and Technology (NIST) initiated the PQC project guiding the development, evaluation and standardization of public key cryptography secure in the advent of quantum computers. We will refer to this process as the [NIST PQC](#)


[project](#). The goal is the standardization of a set of quantum safe key encapsulation and signature schemes.

Background

NIST has a history of conducting processes leading to the standardization of cryptography: examples are the standardization of the Rijndael block cipher as AES in 2001 and the standardization of the Keccak hash function as SHA-3 in 2015. NIST currently also runs the [Lightweight Cryptography competition](#) as well as [The NIST hash-based signatures project](#) described below.

Timeline

The timeline for the NIST PQC project is as follows:

- ✓ [Call for proposals](#): Dec '16 - Nov '17
- ✓ [Round 1](#): Dec '17 - Jan '19, 69 complete submissions.
- ✓ [Round 2](#): Jan '19 - Jul '20, 26 candidates remaining.
-  [Round 3](#): Since July 2020, 15 candidates remaining, 7 “finalists”, and 8 “alternate candidates”.
- ⇒ [Draft standards from finalists track available](#): '22-'23.
- ⇒ [Round 4](#): TBD, focusing on “alternate” track
- ⇒ [Potential amendment of standard](#): TBD

Current status and expected outcome

On July 22nd 2020, NIST [announced](#) the 15 candidates for Round 3 of the NIST PQC project. They're split in two separate tracks: a 'finalists' track comprising 7 schemes – 4 key encapsulation mechanisms and 3 signature schemes – and an 'alternate' track comprising 8 schemes – 5 key encapsulation mechanisms and 3 signature schemes.

At the end of Round 3, NIST expects to standardize one or two KEMs and one or two signature schemes from the finalists track. The candidates from the alternate track will be subject to further analysis in a fourth round, and may be added to the standard at a later point.

Considering the impact of previous projects, e.g. the standardization of AES and SHA-3, it is expected that the outcome of the NIST PQC project will have a major impact on which PQC schemes will find their way into widespread use.

The NIST hash-based signatures project

Problem: Firmware updates

As explained in Section [What are you protecting?](#), there is potential need to use PQC *today* when deploying long-lived systems that cannot be updated.

One important example of long-lived immutable *authentication* arises in the deployment of IoT devices with firmware update mechanisms: in this context, the firmware verification code itself might not be updatable. To

prevent those devices from being compromised when quantum computers become a reality, a quantum safe signature mechanism should be used for firmware updates.

Luckily, while general-purpose quantum safe signature schemes are still in development under the NIST PQC project, the infrequent use of firmware signatures allows us to consider the use of a restricted but more mature class of signature schemes called *stateful signatures*.

The project

NIST runs the [Stateful Hash-based Signatures project](#), the goal of which is to standardize one or more *stateful hash-based signature schemes*. Those are very mature signature schemes, but they come with two severe limitations:

- The private key evolves with every signature, and accidental duplicated use of the same private key breaks the security of the scheme.
- They are *finite-use*: after a pre-defined number of signatures and associated private key evolutions, the private key becomes unusable.

While those properties render stateful hash-based signatures unsuitable for many domains – and exclude them from being considered in the NIST PQC project – they are potentially acceptable in the aforementioned use cases.

Expected Outcome

It is strongly expected that the project will result in the standardization of two schemes called LMS and XMSS, and NIST has already published a [draft standard](#).

The NIST PQC project is only the beginning

A standard is only one step in the transition towards PQC: hardware and software implementations need to be provided, integration of PQC into standards such as TLS and X.509 developed and implemented, tests run at various scales. Then, with sufficient public awareness of the need for PQC, it will gradually find its way into mainstream use.

The potential duration of this process should not be underestimated: for example, ECC was proposed in 1984, standardized as a standalone primitive in 1999 and 2000, integrated into TLS in 2006, and despite smaller keys and lower computational complexity than RSA, it took over a decade for it to become widely supported and used. Even today, the majority of server certificates on the internet use RSA signatures. A more encouraging example is TLS 1.3, which was standardized in 2018 and as of August 2020 is already supported by 32% of major web servers⁷. In contrast to ECC and TLS 1.3, though, PQC affects performance in a negative way (see [Resource usage: No “one size fits all”](#)), which will likely slow down its adoption further.

⁷Source: [SSL Pulse](#)

Libraries for PQC

There are a number of libraries to choose from when benchmarking and testing PQC primitives, including:

- [libpqcrypto](#) consolidates implementations from NIST PQC submissions, providing a unified API and testing framework. It is part of the [PQCRYPTO](#) project.
- [OpenQuantumSafe](#) (OQS) provides both a PQC library [libOQS](#) and an integration of the supported PQC primitives into OpenSSL and OpenSSH.
- [pqm4](#) is a PQC library for Cortex-M4, including a testing and benchmarking framework for the STM32F4 Discovery board. It too is part of the [PQCRYPTO](#) project.
- The [SUPERCOP benchmarking framework](#) contains numerous optimized implementations of PQC primitives.

This list is for convenience of the reader only and is not an endorsement by Arm.

Taxonomy of Post-Quantum Cryptography

In this section, we give an overview over the main classes of cryptographic primitives that have been suggested for Post-Quantum Cryptography. The purpose is solely to give the reader an impression of the varied landscape of PQC and it is not assumed that the names of the different classes mean anything to the reader. Readers interested in some technical content will find a short introduction to the ideas behind lattice-based cryptography in [Appendix B: An introduction to lattice-based cryptography](#).

Overview

While numerous proposals for quantum safe public key cryptography have been brought forward during the NIST PQC project, promising candidates can roughly be categorized as follows:

- Lattice-based cryptography*
 - Unstructured
 - Structured*
- Code-based cryptography*
 - Goppa-codes*
 - Quasi-Cyclic codes
- Supersingular elliptic curve cryptography
- Multivariate cryptography*
- Public-key cryptography from symmetric primitives

The 15 candidates remaining in Round 3 of the NIST PQC project cover all of those categories, but only the categories marked with an * are represented by candidates in the finalist track of Round 3 — see Figure 5.

As can be seen, 5 of the 7 finalists of the NIST PQC project are based on structured lattices, and it is considered highly likely that at least one structured-lattice based key encapsulation mechanism and one structured-lattice based signature scheme will be standardized at the end of Round 3.

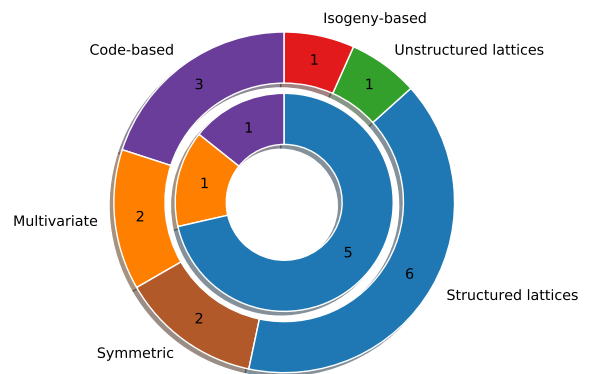


Figure 5: Categorization of remaining candidates of Round 3 of the NIST PQC project. Outer circle: Finalist + Alternate track. Inner circle: Finalist track only. Note the dominance of Structured Lattices.

Maturity

Most PQC families are new in the *practical* sense that they are not currently in widespread use, nor do they rest on the same foundations as today's prevalent cryptography — the only exception is public key cryptography based on symmetric primitives, such as hash-based signatures. *Theoretically*, however, they all predate the increased focus on Post-Quantum Cryptography as triggered by the NIST PQC project by multiple years:

- Code-based cryptography goes back to McEliece's work in 1978 [[McE78](#)], using so-called Goppa-codes. It is considered secure but has not found widespread use because of its large keys. The use of quasi-cyclic codes to reduce key sizes was first considered in 2005 [[Gab05](#)].
- Hash-based signatures (which falls under the last category) go back to Lamport-Diffie and Merkle in 1979 [[Lam79](#); [Mer79](#)], and they are attractive since their security is based on the existence of secure hash functions alone.
- "Unstructured" lattice-based cryptography has its roots in a seminal paper by Ajtai in 1996 [[Ajt96](#)]. The famous "learning with errors" (LWE) problem, which lies at the heart of many lattice-based PQC schemes, was introduced by Regev in 2005 [[Reg09](#)].

- The more efficient sibling of unstructured lattices are so-called “structured” lattices. The NTRU family of cryptographic schemes was introduced in 1996, while the number-theoretic “ring learning with errors” (RLWE) [LPR13] variant of the LWE problem was first considered for cryptographic purposes in 2010. The increased efficiency of structured lattice based schemes comes at the cost of a less well studied hardness assumption, and quantum algorithmic progress has been made in this field recently [CDW16], sending a note of caution.
- Supersingular elliptic curve cryptography is a relatively young field: the use of supersingular elliptic curves for cryptography was first proposed in 2011 [JD11].
- Multivariate cryptography was first proposed by Matsumoto and Imai in 1988 [MI88], and the main ideas underlying some of the NIST PQC candidates in this field were developed in the late 1990’s.

The *ideas* behind many PQC schemes are therefore not new. Most concrete instantiations proposed for the NIST PQC project, however, are young, and carefully reviewing parameters and security arguments for each candidate is a time and resource consuming collaborative undertaking. During the two completed rounds of the NIST PQC project, the review process has uncovered numerous issues in proposed PQC schemes, and it may well continue to do so in the remainder of the NIST PQC project.

Beyond the abstract security of cryptographic schemes, questions of *implementation security* arise, for example: Which schemes lend themselves to side-channel resistant (for example, constant-time) implementations? What are potential pitfalls? How can we test or verify the correctness of an implementation? Answering those questions is essential for the development of trustworthy PQC implementations and an active area of research. See e.g. [PQC19] for an interesting discussion.

Until the review process has led to sufficient confidence in a set of PQC schemes *and* their implementations, the use of Post-Quantum Cryptography today has to be approached with great caution.

Hybrid modes

Where long-lived data requires protection by quantum safe cryptographic mechanisms today or in the near future, so-called *hybrid schemes* should be used. Those hybrids combine a classical scheme like ECC with one or more conjecturally quantum safe PQC schemes and are therefore expected to be at least as secure as the chosen classical scheme.⁸

Various standards for the integration of such hybrid modes into security protocols such as TLS [CC20; SFG20;

⁸Germany’s Federal Office for Information Security has recently recommended the use of hybrid modes with the mature but resource-expensive schemes McEliece (Round 3 finalist) and FrodoKEM (Round 3 alternate) where quantum safe cryptography is required already today [BSI20].

CC20] or X.509 [Kam+18; Bin+17; Tru+18] are in active development.

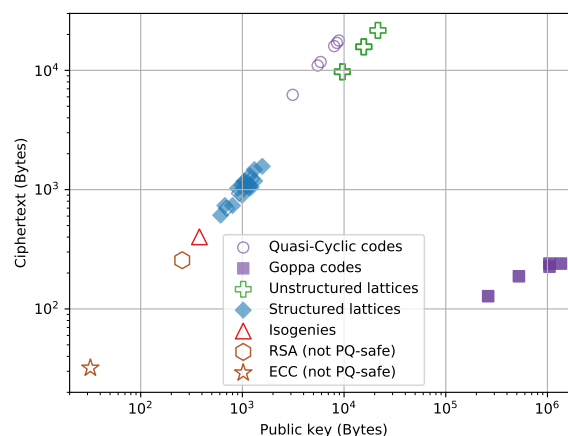


Figure 6: Public key size and Encapsulated Secret size for Key Encapsulation Mechanisms from Round 3 of the NIST PQC project. Filled markers represent the finalist track.

Resource usage: No “one size fits all”

In this section we survey the performance characteristics of the PQC families mentioned above. The takeaway is that there are no candidates for PQC which come close to current public key cryptography in terms of *both* size of cryptographic material and performance. Instead, each family has its strength and weaknesses, making it more suitable for some applications, and less so for others.

As a reference, we consider RSA and ECC: RSA-2048 uses keys and ciphertexts/signatures of size 256B, while for Curve25519 and Ed25519 they are as small as 32B. Moreover, optimized implementations allow for the use of ECC on microcontrollers which are constrained both in terms of their memory and computational abilities.

Now to the numbers for PQC: Figures 6, 7, 8, 9, 10 give an impression of the resource characteristics for the Key Encapsulation and Signature schemes considered in the two tracks of Round 3 of the NIST PQC project, grouped by the family of schemes.^{9,10}

Note: The numbers underlying those graphs are for one specific platform. Moreover, NIST PQC candidates are evolving over time, both in terms of their specification and in terms of optimized implementations for various platforms. Different platforms and/or better (future) implementations will likely lead to improvements.

⁹Source: SUPERCOP benchmarking framework, version supercop-20200702, machine aarch64; A53 (410fd034); 2018 Broadcom BCM2837B0; 4 x 1400MHz; pi3bplus.

¹⁰The *logical* size of cryptographic material is not necessarily a lower bound on an implementation’s RAM usage, since it might be possible to process data gradually. For example, it has been demonstrated that the SPHINCS signature scheme can be implemented using 16kB of RAM despite signatures being 41kB in size [HRS15].

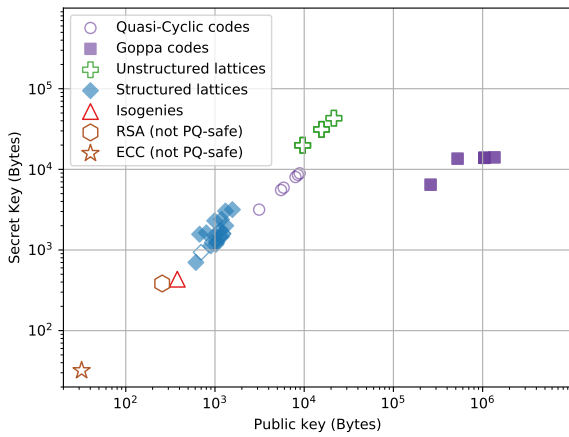


Figure 7: Public key size and Secret key size for Key Encapsulation Mechanisms from Round 3 of the NIST PQC project. Filled markers represent the finalist track.

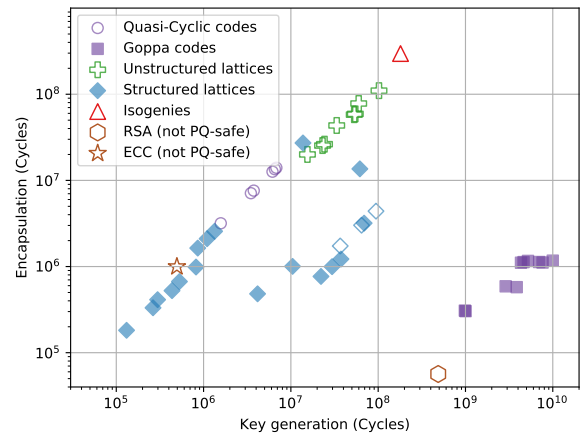


Figure 9: Cycles for Key Generation and Key Encapsulation for Key Encapsulation Mechanisms from Round 3 of the NIST PQC project, measured on a Cortex-A53. Filled markers represent the finalist track.

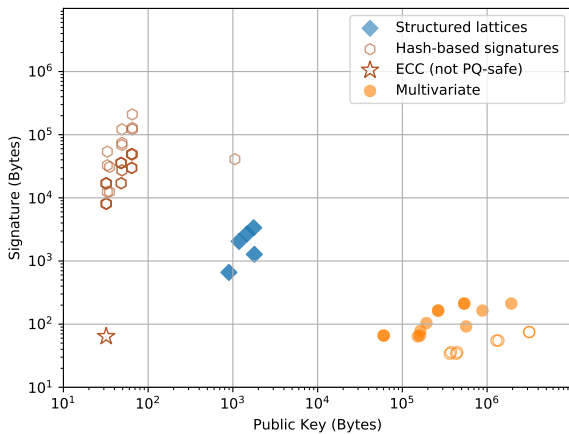


Figure 8: Public Key Size and Signature size for Signature schemes from Round 3 of the NIST PQC project. Filled markers represent the finalist track.

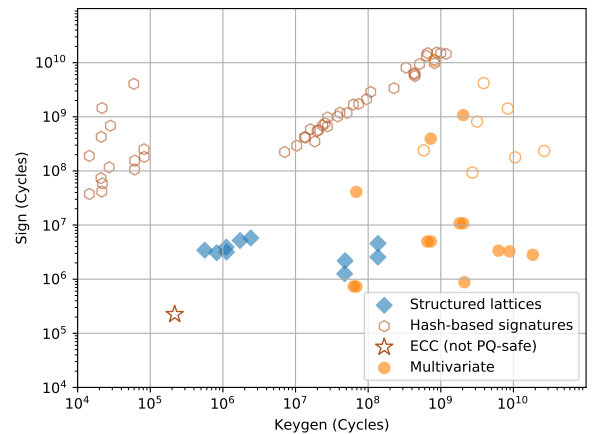


Figure 10: Cycles for Key Generation and Signing for Signature schemes from Round 3 of the NIST PQC project, measured on a Cortex-A53. Filled markers represent the finalist track.

The following rough observations can be made:

- Isogeny-based cryptography comes closest to RSA/ECC in terms of the size of cryptographic material, with key and ciphertext size of around 400B for SIKE-p503. Its weakness is the large computational complexity.
- Code-based schemes often have large public keys (mostly between 10kB and 1MB), but score with smaller private keys or ciphertexts. For example, the McEliece schemes have short 128B ciphertexts, while schemes based on quasi-cyclic codes have short private keys.
- Most multivariate signatures have large public keys (between 10kB and 1MB) but offer very small signatures.
- Structured Lattices have given rise to a large number of PQC proposals with acceptable performance characteristics in any metric.
- Candidates based on unstructured lattices pay for their stronger mathematical foundations with performance

penalties of around an order of magnitude compared to their structured siblings.

- Hash-based signatures schemes have small public keys, but comparatively large signatures.

We conclude that for every metric there are candidates performing well in that metric, but that no candidate PQC scheme comes close to the characteristics of classical public key cryptography in all of them.

Judging resource characteristics only, structured lattices appear to hit a sweet spot: their computational complexity is in the range of ECC, and keys of a few kB are likely practical for most platforms, despite being more than an order of magnitude larger than those of ECC.

What to do now?

Quantum computing threatens the public key cryptosystems in use today, and a variety of proposals for quantum safe cryptography are being developed and scrutinized under the guidance of the NIST PQC project. However, draft standards are not expected before 2022/2023, and while candidates have been narrowed down significantly for Round 3 of the NIST PQC project, one cannot yet predict which *precise* schemes will ultimately prevail.

In this state of uncertainty, it is premature to deploy systems statically bound to the use of a particular PQC primitive: doing so not only introduces the risk of the system failing to meet its security goal due to advances in cryptanalysis, but also endangers interoperability if the chosen scheme does not prevail.

Recommendation:

Do not rely on pre-standardized cryptography.

Step 1: Know your data

Define and track which data needs protecting and for how long it has to stay confidential. Understand which systems are involved in generating, processing and communicating the data, and estimate those systems' lifetimes. Finally, assess which key establishment, encryption and authentication schemes protect against unsolicited access, and estimate their lifetime.

The availability of such data and algorithm 'expiration labels' can then be used to implement security infrastructure which ensures that data is protected by cryptographic mechanisms that are expected to be secure for the lifetime of the data. As a result, systems would in particular:

- Switch to quantum safe schemes for the establishment of keys used to secure the communication of confidential data, *before* the lifetime of the data surpasses the time towards practical quantum attacks.
- Switch to quantum safe authentication schemes *before* classical authentication schemes become vulnerable to practical quantum attacks.
- Re-encrypt encrypted data at rest with a fresh key established and communicated through quantum safe schemes, *before* the lifetime of the data surpasses the time towards practical quantum attacks.

See [What are you protecting?](#) for more information.

Step 2: Crypto-agility

Ensure that security infrastructure supports changing the available cryptographic schemes and their assessment of security. For example, designers of IoT systems should ensure that devices support remote firmware upgrades.

Step 3: Overprovision resources

Crypto-agility is only useful if your system is able to accommodate future cryptographic primitives. As we have seen in [Resource usage: No "one size fits all"](#), PQC is more resource hungry than classical crypto, so it is important to overprovision systems with sufficient resources so they can host whatever PQC scheme(s) prevail.

It is likely that Round 3 of the NIST PQC project will lead to the standardization of structured lattice schemes: other Round 3 finalists have large public keys which limit their uses, while NIST aims to offer schemes that cover a wide range of applications. Schemes based on structured lattices have similar performance to ECC, and keys of a few kB seem acceptable even on smaller systems:

Recommendation: At the least, we recommend systems be overprovisioned to be able to host the Round 3 finalists based on structured lattices.

However, as mentioned before, schemes based on structured lattices pay for their comparably small key sizes with their reliance on assumptions whose hardness on a quantum computer is not yet well understood, and for which recent progress has been made.

Recommendation: Where possible, we recommend systems be overprovisioned to also be able to host alternate track candidates of Round 3 of the NIST PQC project, in particular as the unstructured lattice key encapsulation scheme FrodoKEM or the hash-based signature scheme SPHINCS+.

Step 4: Use stateful hash-based signatures for immutable authentication

Use the stateful hash-based signatures LMS/XMSS for authentication mechanisms that cannot be updated, such as immutable firmware verification code. See [Problem: Firmware updates](#).

Step 5: Trial the use of PQC

Assess crypto-agility and the ability to run various PQC primitives, as well as security infrastructure such as TLS building on them, in a test environment.

[Hybrid modes](#) are potentially even suitable for in-field use.¹¹ However, don't forget about implementation security when using hybrid implementations and ensure that the underlying classical implementation is well-established and considered secure.

See [Libraries for PQC](#) for references.

¹¹ A well-known example for such a test are Google's and Cloudflare's experiments trialling the use of hybrid modes in TLS.

What is Arm doing?

Arm is following the numerous aspects of research on PQC and the NIST PQC project very closely. Moreover, Arm is actively working on the following topics:

- Arm is contributing to the development of the lattice-based CRYSTALS-Kyber key encapsulation scheme, which reached the finalist track of Round 3 of the NIST PQC project. Arm is also involved in NewHope, which reached Round 2 of the NIST PQC project.
- Arm is working to ensure that promising PQC schemes work well on Arm technology.
- Arm is exploring support for PQC in the Mbed TLS security software stack, with emphasis on usability on resource constrained IoT devices.
- Arm is investigating PQC within the context of the [PSA Certified](#) security framework and open security standards such as the IETF working group for Software Updates for Internet of Things (SUIT).

For more information, contact pqc-whitepaper@arm.com.

Appendix A: How quantum computers threaten RSA & ECC

In this section, we give an overview of how the quantum computational model allows for the construction of polynomial-time algorithms breaking RSA and ECC.

The exposition is occasionally deliberately imprecise at the benefit of brevity and intuition, and the goal is that the reader will get an impression of the main ideas and share our fascination for the field. We refer to [NC11; NM19] for more information.

We assume familiarity with and readiness for some mathematical notation, as well as familiarity with the notion of a group. $\mathbb{Z}/n\mathbb{Z}$ denotes the ring of integers modulo n , where addition and multiplication are computed modulo n : for example, $3 \cdot 5 = 15 = 2$ in $\mathbb{Z}/13\mathbb{Z}$, or $3 \cdot 5 = 15 = 0$ in $\mathbb{Z}/15\mathbb{Z}$. \mathbb{F}_p is another name for $\mathbb{Z}/p\mathbb{Z}$ in case p is a prime.

The hidden subgroup problem

Both RSA and the discrete logarithm problem can be reduced to special cases of what is called the *hidden subgroup problem*, and Shor's algorithm provides a strategy for solving the latter. We begin by recalling the RSA and discrete logarithm problems.

Problem (RSA factoring problem). *Given the product $n = p \cdot q$ of two large primes p, q , find p, q .*

Problem (Discrete Logarithm Problem — DLP). *Given a group G and elements $g, h \in G$ with $h = g^a$ for some $a \in \mathbb{Z}$, find such an a .*

Classically, the group G is either a subgroup of the multiplicative group of a prime field \mathbb{F}_p , or a subgroup of the group of points on an elliptic curve.

The main observation is that both the RSA factoring and the DLP problem can be reduced to finding the periodicity of a known function:

Observation. *The Discrete Logarithm Problem for (G, g, h) reduces to finding the periodicity of the function*

$$f_{G,g,h} : (\alpha, \beta) \mapsto g^\alpha / h^\beta : \mathbb{Z} \times \mathbb{Z} \rightarrow G.$$

Namely, for $h = g^\gamma$ the periodicity of $f_{G,g,h}$ is $(\gamma, 1)$:

$$\begin{aligned} f_{G,g,h}((\alpha, \beta) + (\gamma, 1)) &= g^{\alpha+\gamma} / h^{\beta+1} \\ &= g^\alpha / h^\beta \cdot g^\gamma / h \\ &= f_{G,g,h}(\alpha, \beta). \end{aligned}$$

Observation. *The RSA factoring problem reduces to finding the periodicity of the function*

$$f_{n,x} : e \mapsto x^e : \mathbb{Z} \rightarrow \mathbb{Z}/n\mathbb{Z}$$

for $x \in \mathbb{Z}/n\mathbb{Z}$ with $\gcd(x, n) = 1$.

Here is the argument in a nutshell: If $f_{n,x}$ is d -periodic and $1 \neq x \in \mathbb{Z}/n\mathbb{Z}$ with $\gcd(x, n) = 1$, we have

$$1 = f_{n,x}(0) = f_{n,x}(0 + d) = x^d.$$

Now, for 75% of all x , all such d will be even — we omit the proof here — so going through $d, d/2, d/4, \dots$ we then find f s.t. $y := x^f \neq 1$ and $y^2 = x^{2f} = 1$. In other words, $y^2 - 1 = (y - 1)(y + 1)$ is a multiple of n , and for 50% of choices of x this will give away the factorization of n via $p = \gcd(y - 1, n)$ or $q = \gcd(y + 1, n)$ — again we omit the details.

Abstracting from both observations, we therefore gain interest in understanding the following problem:

Problem (Hidden Subgroup Problem). *Assume that we are given a function $f : G \rightarrow X$ from some abelian group G to a set X , and that there is a subgroup $H \subseteq G$ such that f is H -periodic, i.e. $f(x + t) = f(x)$ if and only if $t \in H$. The hidden subgroup problem is to find H .*

The above observations can therefore be restated as saying that the RSA factoring and the DLP problem can be reduced to the hidden subgroup problem for $G = \mathbb{Z}$ and $G = \mathbb{Z} \times \mathbb{Z}$, respectively.

We next explain how the hidden subgroup problem can be approached in the quantum computational model.

The state space of quantum computing

The unit of information in classical computing is the *bit*, which can have value 0 or 1 — in other words, the state space of a single bit is $\{0, 1\}$. Similarly, the state space of n bits is $\{0, 1\}^n$, with each bit being either 0 or 1.

In (universal) quantum computing, in turn, the basic unit of information is the *qubit*. In contrast to a bit which is either in state 0 or in state 1, a qubit is in state 0 or 1 only with some *probability* — as a simplified model, one can think of the state space of a single qubit as the set of probability distributions on $\{0, 1\}$. The classical states 0 or 1 can be viewed as “pure” qubit states where the qubit is certainly in state 0 or 1, respectively, but equally a qubit might be 0/1 with probability 75%/25% — those states are said to be a *superposition* of the classical states. Similarly, a configuration of n qubits can be in any classical/pure state $\underline{x} \in \{0, 1\}^n$ with some probability, and a simplified model for the state space of n qubits is thus the space of probability distributions on $\{0, 1\}^n$.¹²

The actual state space model for a qubit adds a phase shift to the probabilistic model: for a single qubit, the state is not a pair (p_0, p_1) of probabilities for state 0 and 1, respectively, but a pair of complex numbers $(\alpha_0, \alpha_1) \in \mathcal{H} := \mathbb{C}^2$ such that $|\alpha_0|^2 + |\alpha_1|^2 = 1$, and the p_i are recovered as $|\alpha_i|^2$.¹³ The usual notation is $\alpha_0|0\rangle + \alpha_1|1\rangle$ with $|0\rangle = (1, 0)$ and $|1\rangle = (0, 1)$. Similarly, the state space for n qubits is a linear combination $\sum_{\underline{x} \in \{0, 1\}^n} \alpha_{\underline{x}} |\underline{x}\rangle$ such that $\sum_{\underline{x}} |\alpha_{\underline{x}}|^2 = 1$, and the underlying probabilities are recovered as $|\alpha_{\underline{x}}|^2$. Valid state transitions between n -qubit states are modeled as length-preserving (*unitary*) linear maps.¹⁴

A summary of the various views on classical and quantum state spaces is given in Figure 16.

For the rest of this chapter, we will geometrically depict qubits by displaying pure states as points and superpositions as sets of colored points. For example, the state of three qubits can be interpreted as a subset of cube of unit length as depicted in Figure 11, or the state of four qubits can be interpreted as a subset of a square of length 4 as depicted in Figure 12. In general, roughly speaking an entire subset of a k -dimensional object with l bits precision in each dimension can be represented by a single kl -qubit configuration. It is this ability to represent entire *subsets* where classical computing allows us to represent

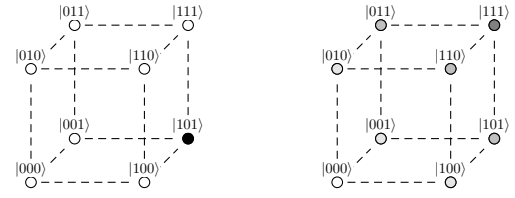


Figure 11: 3-qubit states as subsets of the cube $\{0, 1\}^3$. Left: pure. Right: non-pure/entangled/in superposition

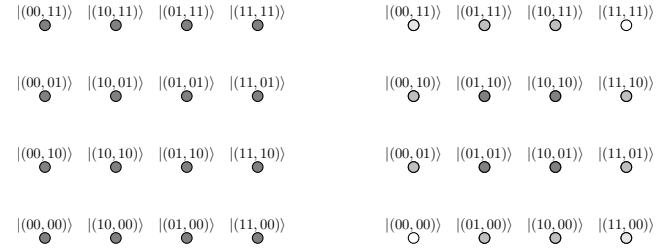


Figure 12: 4-qubit states as subsets of the 4×4 -square $\{0, 1\}^4 \cong \{0, 1, 2, 3\}^2$. Left: Uniform subset with equal probability for all points. Right: Non-uniform distribution.

only individual points what gives quantum computing its power.

Finally, a qubit can be *measured*. Measurement randomly collapses a qubit into one of the pure states $|0\rangle$ or $|1\rangle$, with the probabilities for each represented by the qubit state itself. For example, on measurement, the qubit $\frac{3}{5}|0\rangle + \frac{4}{5}|1\rangle$ would collapse to $|0\rangle$ with probability $\frac{9}{25} = 36\%$ and to $|1\rangle$ with probability $\frac{16}{25} = 64\%$.

The HSP on a Quantum Computer

The blueprint to approach the hidden subgroup problem in the quantum computational model is remarkably simple and we will describe it in this section.

Assume the context of the HSP: $f : G \rightarrow X$ is a function with periodicity $H \subseteq G$, and we would like to find H . Recall that f having periodicity H means that for $x, t \in G$ we have $f(x + t) = f(x)$ precisely if $t \in H$.

Step 1: Uniform superposition over domain

We start with a qubit state representing G uniformly. That is, each $x \in G$ occurs equally likely with probability $1/|G|$. The state is depicted on the left in Figure 13.¹⁵

Step 2: Compute f in a separate register

In the second step, we ‘tag’ each x with its image $f(x)$ under f . This is, we apply $x \mapsto (x, f(x))$ to the uniform state constructed in the previous step. This state is depicted on the right in Figure 13.¹⁶

¹²Already in this simplified probabilistic model one can observe the important property of *quantum entanglement*: describing the state of $n + m$ qubits is not equivalent to individually describing an n -qubit and an m -qubit state. In probabilistic terms, the states that can be decomposed in this way correspond to probability distribution on $\{0, 1\}^{n+m}$ where the first n and last m coordinates are *independent*, but not every distribution on $\{0, 1\}^{n+m}$ has this property.

¹³The state space of just a single qubit is of remarkable complexity and beauty: it is a 3-dimensional sphere, and ignoring phase shifts reduces it to the 2-dimensional *Bloch sphere* by means of the *Hopf fibration*.

¹⁴The complex-linear model allows us to observe another important property, namely *no-cloning*: the cloning map $x \mapsto x \otimes x : \mathcal{H} \rightarrow \mathcal{H} \otimes \mathcal{H}$ is not linear.

¹⁵Algebraically, this is the state $\frac{1}{\sqrt{|G|}} \sum_{x \in G} |x\rangle$.

¹⁶Algebraically, this is $\frac{1}{\sqrt{|G|}} \sum_{x \in G} |x\rangle |f(x)\rangle$.

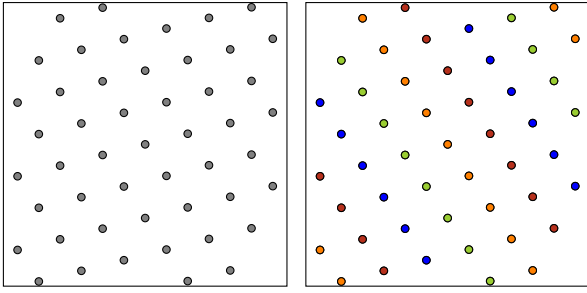


Figure 13: Left: Uniform superposition of domain G . Right: Applying $x \mapsto (x, f(x))$ ‘tagging’ each point with its image under f . Different colors represent different values under f .

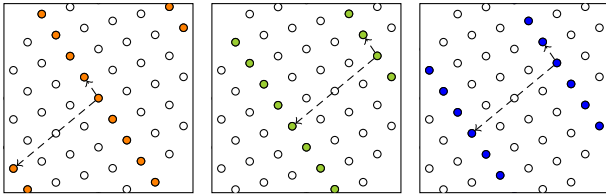


Figure 14: Various level-sets of periodic function after measuring value-register in uniform superposition of $(x, f(x))$.

Step 3: Measure the tag to obtain level-set

Given the superposition over $(x, f(x))$, we now ‘measure’ the qubits describing the value $f(x)$. This collapses the superposition of *all* $(x, f(x))$ to a superposition of those $(x, f(x))$ where $f(x)$ is a random but fixed value in the image of f . That is, we obtain a random, non-empty level-set $f^{-1}(f(x)) = x + H$ for some $x \in G$. Some of those states are depicted in Figure 14.¹⁷

Step 4: The Quantum Fourier Transform

At this point, we have constructed the superposition of a random level set $x + H$. Those level sets are all H -periodic in the sense that shifting them by $t \in H$ does not change them — this is depicted by the dashed arrows in Figure 14 — and the goal is to extract this periodicity.

Extraction of periodicity is a job for the *Fourier Transform*, offering conversion between time to frequency domains, the numerous applications of which the reader might have come across before — as a nice example, see the [Tide-predicting machine](#). In the context of quantum computing, it is a particular powerful tool since the Fourier Transform can be implemented in polylogarithmic time on a quantum computer — this is known as the *Quantum Fourier Transform* (QFT).

We are omitting the details here as they would go beyond the scope of this article, but the interested reader is encouraged to look into [RP11, Chapter 7.8] or [NC11, Chapter 5] for details.

Step 5: Measure & Repeat

At this point, we have constructed the superposition over the periodicity subgroup H that we are trying to determine. Measuring this state gives a single random element of H . Repeating Steps 1-5 produces arbitrarily many elements of H .

Keeping it honest

The previous sections have swept many important details under the rug, and readers interested in a more precise treatment may consult e.g. [RP11, Chapter 8] or [NC11, Chapter 5.3]. Most importantly, implementing the blueprint above in the case of RSA requires the knowledge of a finite quotient $\mathbb{Z}/L\mathbb{Z}$ s.t. the map $f_{n,x} : e \mapsto x^e : \mathbb{Z} \rightarrow \mathbb{Z}/n\mathbb{Z}$ is well-defined on $\mathbb{Z}/L\mathbb{Z}$, but it can be seen that this is as hard as the original problem. Instead, one chooses a sufficiently large L and observes that, albeit no longer exact, the blueprint still allows us to extract the desired periodicity with sufficiently high probability.

Appendix B: An introduction to lattice-based cryptography

In this appendix, we give a brief introduction to some ideas behind lattice-based cryptography. We begin with a gentle and hopefully intuitive journey from the classical Caesar and Vigenère ciphers to the main idea behind symmetric encryption based on the fundamental *learning with errors* (LWE) problem. We will then explain that this scheme is a *homomorphic encryption* scheme and how this property allows to construct a *public key* encryption scheme from it. In the final, slightly more technical sections, we explain the relation between LWE and classical lattice problems, which is a main source of confidence in the hardness of LWE, and hence the security of lattice based cryptography.

In addition to modular arithmetic in $\mathbb{Z}/n\mathbb{Z}$ already used [Appendix A: How quantum computers threaten RSA & ECC](#), we assume that the reader is familiar with basic notions from linear algebra such as vectors, matrices, and scalar products. RLWE will be explained at an intuitive level only, and beyond the notion a polynomial ring, no knowledge of number theory is necessary.

¹⁷Algebraically, this is $\frac{1}{\sqrt{|H|}} \sum_{t \in H} |x + t\rangle |f(x)\rangle$ for some $x \in G$.

From the Caesar cipher to weak pseudo-random functions

Caesar and Vigenère

The reader will undoubtedly have come across the famous *Caesar cipher*: a plaintext is encrypted by shifting each letter a fixed number of positions. For example,

pqcisreal
↓ Shift by 6
vwioyxlgr

Of course, this is easily reversed. A slightly better approach is the *Vigenère cipher*, which uses different yet repeating shifts depending on which letter is being encrypted and encodes those shifts in a passphrase: For example, the passphrase `pqcisreal` corresponds to the repeated shift sequence 15,16,2,8,18,17,4,0,11, giving

shorsalgorithmbreaksrsaonaquantumcomputer
↓ pqcisreal
hxqzkrpgzgyvpesvelzitasfrabjqpbgdgoxekvmj

This is better than the Caesar cipher, but there are many problems remaining:

- Knowing any pair of corresponding plaintext and ciphertext reveals the passphrase and allows an attacker to decrypt any other message protected by the same passphrase.
- The repetitive nature of the shifts allows us to recover the plaintext by analyzing letter frequencies.
- Human readable passphrases are prone to brute force attack.

Alice goes to the library

All of the above problems with the Vigenère cipher could be fixed by choosing fresh and truly random passphrases with every encryption and by splitting the plaintext into chunks as large as the passphrase. Those changes, however, would render the scheme impractical because all those passphrases would need to be pre-agreed.

An alternative is to stick with one passphrase per encryption, but to dynamically derive those passphrases from a public hint and some shared secret: for example, each passphrase could be the first word on a random page in a secret book both sides have agreed upon upfront, the hint to which would be the page number. Or, to avoid brute forcing the passphrase, as well as to harden the recovery of the book from the knowledge of which words appear on which pages, one could form the passphrases from the first letters of the words on the chosen page.

Symmetric encryption from weak PRFs

While the above sounds — and, taken literally, is indeed — very naïve, it is at heart a sound procedure:

A key k defines a secret function F_k , and encryption $\text{enc}_k(m) := (h, m + F_k(h))$ consists in choosing a random hint h and masking the message m with the fresh 'passphrase' $F_k(h)$. In our example, k is the secret book, h is a page number, $F_k(h)$ is the lookup of the initial letters on page h in book k , and $+$ is Caesar shifting. Decryption is simply given by $\text{dec}_k(h, c) := c - F_k(h)$.

Intuitively, this procedure is safe if the mask $F_k(h)$ appears random, assuming the hints h are chosen randomly. And indeed, the above is an informal description of a well-known construction of a symmetric encryption scheme from a *weak pseudo-random function*.¹⁸

Definition. A *weak pseudo-random function (weak PRF)* is a family of functions $\{F_k\}_{k \in \mathcal{K}}$ indexed by a key-space \mathcal{K} such that an attacker cannot distinguish the stream

$h_1, F_k(h_1), h_2, F_k(h_2), \dots, k, h_i$ randomly chosen from truly random data.

Weak PRFs and Block Ciphers

Weak PRFs are called 'weak' since they are a relaxed variant of the definition of a *pseudo-random function* (PRF), where an attacker can choose the arguments h_i to query F_k on freely and adaptively, and on the basis of which they should decide whether they are querying one of the F_k , or rather a random function.

The most popular examples of PRFs are *block ciphers*, which are PRFs where all F_k are efficiently invertible. The most popular example of a block cipher in turn is the *Advanced Encryption Standard* AES.

Practically relevant block ciphers often have an ad-hoc construction and heuristic, informal arguments of security. Block ciphers with formal proofs (or rather, reductions) of security do exist, but are not usually of practical significance. As we shall see, lattice-based cryptography opens the door to constructions of (weak) PRFs and their associated cryptographic schemes which are both provably secure and of acceptable performance.

The LWE problem

Weak PRFs from Linear Algebra?

At the heart of lattice-based cryptography is the following extremely simple function family from linear algebra:

Definition. The function family $\{F_s\}$ is indexed by vectors s of a fixed length, and defined by

$$F_s(t) := st^\top = \sum_i s_i t_i,$$

¹⁸The momentarily considered approach of choosing for every encryption a truly random passphrase of the same size as the message to be encrypted, is nothing but the famous *one-time pad*.

where t is another vector of the same length as s .

This family $\{F_k\}$ is *not* a weak PRF, since each F_s is linear while a randomly chosen function likely is not, so an attacker can easily distinguish F_k from a random function by performing a linearity check.

Somewhat surprisingly, adding noise/errors converts $\{F_k\}$ into what *does* seem to be a weak PRF:

Definition (Informal). Let s be a secret vector, and

$$F_s(t) := st^\top + \varepsilon, \quad \varepsilon \text{ small fresh random error term.}$$

The learning with errors problem (LWE) [Reg09] asks us to distinguish

$$t_1, F_s(t_1), t_2, F_s(t_2), \dots, \quad s, t_i \text{ random}$$

from truly random data.

More precisely, the LWE problem $\text{LWE}_{q,\alpha}$ is stated for vectors over $\mathbb{Z}/q\mathbb{Z}$ for some q , and the noise ε is chosen from a discrete Gaussian distribution with standard deviation αq and mean 0. We refer to [Reg09] for the details, which are not relevant for our purpose.

Example: The simplest example of the LWE problem is the case where $q = 2$, where s, t are bitvectors and

$$F_s(t) = (s_1 \text{ AND } t_1) \text{ XOR } \dots \text{ XOR } (s_n \text{ AND } t_n) \text{ XOR } \varepsilon.$$

This is called the *Learning Parity with Noise* problem.

The reader is invited to pause and reflect on the remarkable simplicity of the above candidate weak PRF, being ‘just’ randomized linear algebra over $\mathbb{Z}/q\mathbb{Z}$.

Symmetric Encryption from LWE: A first try

We leave the question of hardness of LWE aside for now and consider how to instantiate the construction of a symmetric encryption scheme from a weak PRF using LWE.

For concreteness, we consider $q = 26$ and identify $\mathbb{Z}/26\mathbb{Z} = \{a, b, c, \dots, x, y, z\}$ as in the Caesar and Vigenère ciphers — with respect to this identification, Caesar shifting is just addition modulo 26. The secret key s as well as the ‘hint’ t are fixed-size vectors of letters, say

$$s = \begin{bmatrix} \text{p,q,c,i,s,r,e,a,l} \\ 15,16,2,8,18,17,4,0,11 \end{bmatrix}, \quad t = \begin{bmatrix} \text{j,u,s,t,a,h,i,n,t} \\ 9,20,18,19,0,7,8,13,19 \end{bmatrix}.$$

To encrypt a letter ℓ , we pick a small random noise term ε biased towards $\mathbf{a} \triangleq 0$, and compute

$$\begin{aligned} \text{enc}_s(\ell) &= st^\top + \varepsilon + \ell \\ &= \begin{bmatrix} \text{p,q,c,i,s,r,e,a,l} \\ 15,16,2,8,18,17,4,0,11 \end{bmatrix} \cdot \begin{bmatrix} \text{j,u,s,t,a,h,i,n,t} \\ 9,20,18,19,0,7,8,13,19 \end{bmatrix}^\top + \varepsilon + \ell \\ &= \frac{1003}{p} + \varepsilon + \ell = \mathbf{p} + \varepsilon + \ell. \end{aligned}$$

But now a problem becomes apparent which the attentive reader will likely have spotted much earlier: our mask-generating function F_s is not deterministic¹⁹, and depending on our choice of ε — for example, we could pick $\varepsilon = \mathbf{a}$ or $\varepsilon = \mathbf{b}$ — the result of the encryption will be different. Concretely, the decryption routine evaluates $F_s(t)$ itself and might pick a different noise ε' , leading to

$$\text{dec}_s(\text{enc}_s(\ell)) = \ell + (\varepsilon - \varepsilon').$$

This term equals ℓ in average, but may slightly deviate from it each time.

To summarize: on the one hand, the addition of noise is essential to make LWE hard. On the other hand, it prevents instantiation of the transformation of weak PRFs into symmetric ciphers as-is. We will study ways around this problem in the next section.

How to handle the noise?

There are multiple ways to handle non-determinism in the mask-generating function F_s .

Method 1: Error correcting codes per letter

Recall that encrypting and decrypting leads to

$$\text{dec}_s(\text{enc}_s(\ell)) = \ell + (\varepsilon - \varepsilon'),$$

so we only recover the letter ℓ *approximately*, up to a small left/right shift. However, if we do not allow *any* letter ℓ as the plaintext, but restrict to a subset of letters at sufficient distance, we are able to remove the noise $\varepsilon - \varepsilon'$ after decryption.

For example, if we only allow the noise ε to be $\mathbf{a} \triangleq 0$ or $\mathbf{b} \triangleq 1$, then $\text{dec}_s(\text{enc}_s(\ell))$ differs from ℓ by at most one shift to the left or right, and restricting ℓ to the letters $\mathbf{a}, \mathbf{d}, \mathbf{g}, \mathbf{j}, \mathbf{m}, \mathbf{p}, \mathbf{s}, \mathbf{v}$ allows to remove the decryption noise: we would e.g. ‘round down’ \mathbf{n} to \mathbf{m} , or ‘round up’ \mathbf{u} to \mathbf{v} . Of course, with growing noise we need to widen the distances between letters, too, but the idea of essentially applying an error correcting code in the plaintext alphabet stays the same. For example, if we would allow the noise ε to shift up to 6 letters we would need to restrict the plaintext letters to just $\ell \in \{\mathbf{a}, \mathbf{n}\}$ to be able to remove the noise.

This idea appears in [GGH97] (based on [AD97]) and in [Reg04], and is used in the first LWE-based encryption scheme by Regev [Reg09]. It also underlies the NIST PQC Round 3 Finalist CRYSTALS-Kyber: the ciphertext alphabet is a large $\mathbb{Z}/q\mathbb{Z}$, while the plaintext alphabet is reduced to just $\{0, \frac{q}{2}\} \subset \mathbb{Z}/q\mathbb{Z}$, allowing for a large noise term ε which in turn hardens the underlying LWE problem. That is, the plaintext is broken down into bits, and each individual bit is represented as either 0 or $\frac{q}{2}$ and encrypted under the LWE procedure.

¹⁹This is called a *randomized weak PRF* in the literature [App+09].

Note that the width of the noise is a tradeoff between efficiency and security: the larger the noise, the harder (at least intuitively) the LWE problem. However, at the same time a large noise requires a larger gap between the allowed plaintext letters, and hence implies a larger plaintext to ciphertext expansion factor.

Method 2: Leave gaps in the noise

The previous approach uses small noise terms and spreads out plaintext letters. Dually, we can use small plaintext letters and spread out the noise terms instead: For example, we could choose the noise ε from $\{a, d, g, j, m, p, s, v\}$ and the plaintext letter ℓ from $\{a, b, c\}$.

This approach is popular in the context of fully homomorphic encryption [Gen09; BV11; BGV11].

Method 3: Error correcting codes per word

Error correction at the level of individual letters requires a sufficiently large alphabet $\mathbb{Z}/q\mathbb{Z}$. In particular, it does not work in the Learning Parity with Noise case $q = 2$. Instead, one can apply error correction at the level of words by approximately decrypting a sequence of letters and then using an error correcting code at the level of such words. This idea was implemented in [GRS08].

Method 4: Remove the noise - Learning with Rounding

One way to look at adding small amount of noise to a mask is that it hides low bits, and the same could be achieved by *removing* those low-bits instead of randomizing them. This is the idea of the *Learning with Rounding* (LWR) problem [BPR11], which replaces the randomized weak PRF $F_s(t) := st^\top + \varepsilon$ by $F_s(t) := \lfloor \frac{st^\top}{r} \rfloor$, where r measures how many low-bits should be omitted from st^\top .

This approach is natural and brings us back to the world of deterministic weak PRFs. It is at the heart of the NIST PQC Round 3 Finalist SABER.

Public Key Encryption from LWE

So far, we have only considered *symmetric* encryption schemes based on the LWE problem, but — as detailed in [The impact on symmetric cryptography](#) — it is *public key* cryptography that is threatened by quantum computers. In this section we outline how LWE can be used as the basis for a public key encryption scheme.

Homomorphic Encryption

The LWE-based symmetric encryption scheme has some remarkable properties:

Observation. For encryptions of plaintexts ℓ_1 and ℓ_2 ,

$$\begin{aligned} c_1 &:= \text{enc}_s(\ell_1) = (t_1, st_1^\top + \varepsilon_1 + \ell_1) \\ c_2 &:= \text{enc}_s(\ell_2) = (t_2, st_2^\top + \varepsilon_2 + \ell_2), \end{aligned}$$

the sum of the two ciphertexts

$$c_1 + c_2 = (t_1 + t_2, s(t_1 + t_2)^\top + \varepsilon_1 + \varepsilon_2 + (\ell_1 + \ell_2)),$$

is a valid encryption of the sum $\ell_1 + \ell_2$ of the two plaintexts — at least if the noise $\varepsilon_1 + \varepsilon_2$ is not too large.

This extends to arbitrary sums and, more generally, linear combinations of more than two ciphertexts, as long as the coefficients of those combinations are small enough to keep the noise value within its bounds — we do not go into details. Considering encryptions of 0 as a special case, we in particular obtain:

Observation. Small linear combinations of encryptions of 0 are again encryptions of 0.

We will also need the following:

Observation. $(0, \ell)$ is a valid LWE-encryption of ℓ , obtained by choosing 0 for both the hint t and the noise ε .

Encryption which supports computations on encrypted data which have a controlled effect on the underlying plaintexts is called *homomorphic encryption*. If arbitrary plaintext computations can be performed on the corresponding ciphertexts, the scheme is called *fully homomorphic*. The above observation can therefore be phrased as saying that LWE-based encryption is naturally “additively homomorphic” — and in fact, one can even construct *fully* homomorphic encryption based on LWE.

Depending on how one looks at it, homomorphic encryption is either a defect or a feature: it is a defect in the sense that it makes the scheme *malleable* — if you do online banking, you would not want an attacker to be able to fiddle with the encrypted transfer request to turn a \$1,000 into a \$1,000,000. On the other hand, it is a feature in the sense that it allows one to offload computation to untrusted third parties.

Fully homomorphic encryption is a fascinating field, but out of scope of this paper, so we will not go further into it here. Suffice to say that apart from the dominance of (structured) lattice-based cryptography in the Round 3 Finalist Track of the NIST PQC project, its use for fully homomorphic encryption is yet another reason why lattice-based cryptography is here to stay.

Encryptions of 0 as public keys

In the last section we have seen that:

- For any plaintext letter ℓ , the sum of $(0, \ell)$ and an arbitrary encryption of 0 is an encryption of ℓ .
- ‘Small’ linear combinations of encryptions of 0 are again encryptions of 0.

This leads to the following idea of turning LWE into a *public key* cryptosystem:

- As a public key, publish a set $\{c_i\}$ of encryptions of 0.
- To encrypt ℓ , form a fresh encryption of 0 via a random ‘small’ linear combination c_{fresh} of the c_i . The encryption of ℓ is $(0, \ell) + c_{\text{fresh}}$.
- Decryption is unchanged: given (t, c) , the receiver computes the mask $F_s(t) = st^\top + \varepsilon$ and removes the noise from $c - F_s(t)$ to recover the plaintext (the noise ε can in fact be chosen to be 0 here).

Note that decryption does not involve the public key, nor does it require recovery of the coefficients used to combine the c_i . Indeed, as we shall see below, the latter is a hard problem.

Some notation

At this point it’s useful to introduce some common notation, simplifying subsequent discussions and helping the reader interested in further literature study.

Each encryption c_i of 0 is of the form $(t_i, st_i^\top + \varepsilon_i)$, so the public key $\{c_i\}$ can be expressed in matrix form as

$$\mathbf{pk} = (A, b := sA + \varepsilon),$$

where $A = [t_1 \dots t_n]^\top$ and $\varepsilon = [\varepsilon_1 \dots \varepsilon_m]$. The encryption of a letter ℓ is then given by

$$\text{enc}_{(A,b)}(\ell) = (A\lambda^\top, b\lambda^\top + \ell),$$

for a ‘small’ random λ , and decryption is given by

$$\text{dec}_s(h, c) = \text{decode}(c - sh^\top),$$

where **decode** removes the noise as one of the ways discussed in [How to handle the noise?](#).

This is the way LWE-based cryptosystems are usually presented in the literature.

As a sanity check, let us ask: can an attacker learn anything about s from the public key $\{(t_i, st_i^\top + \varepsilon_i)\}$? No, since the very LWE assumption says that the latter is indistinguishable from truly random data.

The Short Integer Solution problem

The reader might wonder: given a public key (A, b) and an encryption $(A\lambda^\top, b\lambda^\top + \ell)$ of some letter ℓ , can’t we recover λ^\top from $A\lambda^\top$ by linear algebra, and thereby deduce also the mask $b\lambda^\top$ and finally ℓ ?

The hidden complexity here is that while linear algebra allows us to find *some* vector μ s.t. $A\mu^\top = A\lambda^\top$, we need a *small* one: assuming only $A\mu^\top = A\lambda^\top$, the real mask $b\lambda^\top$ and the candidate mask $b\mu^\top$ differ by

$$\begin{aligned} b\lambda^\top - b\mu^\top &= sA\lambda^\top + \varepsilon\lambda^\top - sA\mu^\top - \varepsilon\mu^\top \\ &= \varepsilon(\mu - \lambda)^\top, \end{aligned}$$

which is only within the allowed (and removable) range of noise provided that μ is a small vector.

LWE-based public key encryption — in fact, the LWE problem itself — therefore rests on the following problem:

Problem. Given a random matrix $A \in (\mathbb{Z}/q\mathbb{Z})^{n \times m}$ with $m \gg n$, the Short Integer Solution (SIS) [Ajt96] problem asks us to find ‘small’ vectors $t \in (\mathbb{Z}/q\mathbb{Z})^m$ s.t. $At^\top = 0$.

Note that finding a small t with $At^\top = 0$ is the same as finding collisions for the map

$$\{t \in (\mathbb{Z}/q\mathbb{Z})^m \text{ small}\} \xrightarrow{t \mapsto At^\top} (\mathbb{Z}/q\mathbb{Z})^n. \quad (*)$$

Not only is this a remarkably simple candidate for a collision-resistant function family, but note also that it is ‘almost’ additively homomorphic — that is, $At_0^\top + At_1^\top = A(t_0 + t_1)^\top$ — with the caveat that the addition of elements of the left hand side of $(*)$ is only a partially defined operation due to the smallness constraint.

This ‘almost homomorphic’ one-way function turns out to be a powerful replacement for the classical homomorphic one-way function $e \mapsto g^e$ defining the Discrete Logarithm Problem: For example, the “Fiat-Shamir with Abort” [Lyu09] approach to lattice signature schemes used in the Round 3 Finalist CRYSTALS-Dilithium essentially arises from the classical Schnorr signature scheme by replacing $e \mapsto g^e$ with $t \mapsto At^\top$.

Hardness of LWE and lattices

We have not touched on two obvious questions:

- What evidence do we have that LWE/SIS are hard?
- Why is this appendix called “An introduction to lattice-based cryptography” — where are the lattices?

Lattices and lattice problems

An n -dimensional *lattice* is a discrete subgroup Λ of \mathbb{R}^n which spans it. Equivalently, it is the set of \mathbb{Z} -linear combinations of a basis $\{b_1, \dots, b_n\}$ of \mathbb{R}^n . Figure 15 shows an example of a two-dimensional lattice together with one possible choice of basis. Two things can already be observed in that picture:

Observation. The lattice generated by a basis can contain vectors that are shorter than the basis: in the figure, the shortest vector $3b_1 - 2b_2$ is shorter than both b_1 and b_2 .

Observation. The lattice point closest to a vector given as an \mathbb{R} -linear combination of a lattice basis is not linked to the basis coefficients in an obvious way: in the figure, the lattice point closest to $v \approx \frac{2}{3}b_1$ is $2b_1 - b_2$.

Those observations are at the heart of the following two central algorithmic problems about lattices:

Problem. Given Λ , the shortest vector problem (SVP) asks us to find the shortest non-zero vector in Λ .

Problem. Given Λ , the closest vector problem (CVP) asks us to find the lattice point closest to a given vector.

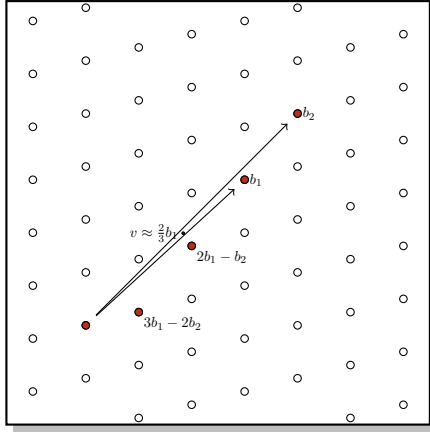


Figure 15: A 2-dimensional lattice generated by $\{b_1, b_2\}$.

Both problems are usually formulated as a continuous spectrum of decision problems indexed by an approximation factor: for Gap-SVP_γ , we are given (Λ, d) and need to decide if the length of the shortest vector in Λ is smaller than d or larger than γd ; the problem Gap-CVP_γ is defined similarly.

Gap-SVP_γ and Gap-CVP_γ get easier with larger approximation factor γ . In the extreme cases, the [LLL algorithm](#) (a lattice reduction technique) solves Gap-SVP_γ in polynomial time, for an approximation factor γ which is exponential in the lattice-dimension n , e.g. $\gamma = 2^n$. At the other end of the spectrum, it is known that Gap-SVP_γ is NP-hard for any constant γ . The middle ground of a γ which is *polynomial* in n is what many LWE-based cryptosystems relate to.

See [\[AR05\]](#) for a graph illustrating the hardness of lattice problems as the approximation factor varies.

From LWE to lattices

The fundamental result about the hardness of LWE is the following relation to Gap-SVP_γ . Recall that $\text{LWE}_{q,\alpha}$ asks us to distinguish samples $st^\top + \varepsilon$ from random, where $s \in (\mathbb{Z}/q\mathbb{Z})^n$ is secret and fixed, $t \in (\mathbb{Z}/q\mathbb{Z})^n$ is uniformly random, and the noise $\varepsilon \in \mathbb{Z}/q\mathbb{Z}$ is chosen from a discrete Gaussian distribution of standard deviation αq .

Theorem 1 (Informal). *There is an efficient quantum reduction from $\text{Gap-SVP}_{\tilde{O}(n/\alpha)}$ to $\text{LWE}_{q,\alpha}$ provided α and q aren't too small: $\alpha q > 2\sqrt{n}$.*

For example, if $q = n^2$ and $\alpha = 1/n$, we see that an efficient quantum algorithm $\text{LWE}_{n^2, 1/n}$ yields an efficient quantum algorithm for $\text{Gap-SVP}_{\tilde{O}(n^2)}$ with *polynomial* approximation factor, and no such algorithm is known. Note how an increased width α of the noise in LWE leads to a better approximation factor in Gap-SVP .

Main idea

At the heart of the reduction is a quite intuitive idea to use LWE to solve instances of the closest vector problem:

Suppose $\Lambda \subset \mathbb{R}^n$ is a lattice with basis $\{b_1, \dots, b_n\}$, and $v \in \mathbb{R}^n$ is a vector close to the lattice point $s_1 b_1 + \dots + s_n b_n$, $s_i \in \mathbb{Z}$. We want to find $s = [s_1 \dots s_n]$.

To reduce the problem to LWE, we use the following “measurement approach”: we pick an arbitrary “coordinate function” $\varphi : \mathbb{R}^n \rightarrow \mathbb{R}$ and consider $\varphi(v)$. Since $v \approx \sum_i s_i b_i$, we should also have

$$\varphi(v) \approx \sum_i s_i \varphi(b_i) = s \cdot [\varphi(b_1) \dots \varphi(b_n)]^\top. \quad (*)$$

Doesn't this look like a sample from the mask-generating function $F_s(t) := st^\top + \varepsilon$ underlying LWE? If we generate a large number of those samples and invoke the assumed LWE-solver, we should be able to find s . Ultimately, this idea turns out to be fruitful, but to make it work, numerous technical obstacles have to be overcome. For the benefit of the reader interested in diving into the details, we briefly sketch those obstacles.

Firstly, LWE lives over $\mathbb{Z}/q\mathbb{Z}$ for some q , and in particular, we need $\varphi(b_i) \in \mathbb{Z}$. The set of $\varphi : \mathbb{R}^n \rightarrow \mathbb{R}$ satisfying $\varphi(b_i) \in \mathbb{Z}$ — or, equivalently, $\varphi(\Lambda) \subset \mathbb{Z}$ — is called the *dual lattice* of Λ , and denoted Λ^* . Moreover, for an arbitrary $v \in \mathbb{R}^n$, we have $\varphi(v) \in \mathbb{R}$, and we need to *discretize* it to a value in \mathbb{Z} .

Secondly, how do we choose φ from the dual lattice Λ^* ? There are opposing constraints: on the one hand, we need $[\varphi(b_1) \dots \varphi(b_n)] \in (\mathbb{Z}/q\mathbb{Z})^n$ to be uniformly random modulo q , which suggests choosing φ from a sufficiently wide distribution on Λ^* . On the other hand, the longer φ , the larger the error in $(*)$, and hence φ should be kept small. The solution turns out to be to choose φ from a suitable Gaussian distribution on Λ^* . The construction of such samples is a difficult problem in itself, and two approaches are described in [\[Reg10, Proposition 2.1\]](#). We do not go into further details here.

Thirdly and finally, there's a correlation between φ and the noise in $(*)$, while LWE requires hint and noise generation to be independent. This needs to be fixed by ‘smoothing’ the noise by adding Gaussian noise to $\varphi(v)$.

Working out the details here is rather technical, but we hope that we could convince the reader that the underlying idea behind the reduction is quite natural.

Efficiency considerations

The inefficiency of plain LWE

Let us get a rough estimate of the efficiency of the above LWE-based public key cryptosystem, assuming hint and secret vectors of length n . The expansion from plaintext to ciphertext is determined by three factors:

- The mask generation hint, a vector in $(\mathbb{Z}/q\mathbb{Z})^n$.
- The mask itself, an element of $\mathbb{Z}/q\mathbb{Z}$.
- The size Σ of the plaintext alphabet.

This gives an expansion factor of $\approx (n+1) \log_2(q) / \log_2(\Sigma)$. The secret key has size $\approx n \log_2(q)$ bits.

Example: Let us consider the parameters used by FrodoKEM, a NIST PQC Round 3 Alternate Track candidate, and moreover the only candidate remaining whose security is based on unstructured lattices. We will see that the result would be of limited practical use, and indeed FrodoKEM applies some modifications to the LWE-encryption blueprint to improve efficiency, which we will discuss afterwards.

FrodoKEM provides three choices for (Σ, q, n) , namely $(4, 2^{15}, 640)$, $(8, 2^{16}, 976)$, and $(16, 2^{16}, 1344)$. If those parameters were used for *plain* LWE encryption, the ciphertext overhead would be ≈ 650 bytes per bit, leading e.g. to ciphertexts of ≈ 83 kB for the encryption of a randomly chosen 128-bit secret — too much for general purpose use. In contrast, the secret keys have a smaller size, from ≈ 1.2 kB for the first parameter set to ≈ 2.6 kB for the third parameter set.

One idea to improve efficiency of plain LWE is to balance the sizes of secret key and cipher text: for key generation, we generate multiple independent secret key vectors s_1, \dots, s_m , and during encryption, a *single* hint h is used to generate masks $F_{s_1}(h), \dots, F_{s_m}(h)$ for *all* of the s_i . This reduces the plaintext to ciphertext expansion by the factor m , but increases the secret key length by the same amount. This approach is e.g. applied in FrodoKEM with $m = 8$, leading to ciphertext lengths between 10 kB and 20 kB and secret keys between ≈ 10 kB and ≈ 21 kB (excluding the public key).

Ring Learning With Errors

One major reason for the large key sizes in plain LWE is the fact that generating a single scalar mask in $\mathbb{Z}/q\mathbb{Z}$ requires the choice and transmission of a mask generating hint *vector*, since mask generation involves the scalar product $F_s(t) = st^\top + \varepsilon$. At the same time, the only structural property of the scalar product that we actually used in the construction of LWE-based encryption was its bilinearity: *any* mask generation of the form $F_s(t) = s \bullet t + \varepsilon$ with bilinear \bullet would yield a candidate cryptosystem (though the question of *security* will very much depend on the specific choice of \bullet).

Those considerations motivated the study of the *Ring Learning With Errors* (RLWE) problem, in which the (dimension reducing) scalar product in the mask generation function of LWE is replaced by a (dimension preserving) multiplication (more precisely, a ring structure) on $(\mathbb{Z}/q\mathbb{Z})^n$. As a result, the mask itself is an n -dimensional vector which can be used to mask n plaintext letters, and so the plaintext to ciphertext expansion factor drops from $(n+1) \log_2(q)/\Sigma$ to $2 \log_2(q)/\Sigma$.

Example: Let us consider the parameters used by the RLWE-based scheme NewHope, a popular NIST PQC Round 2 candidate. It uses $q = 12289$ and encrypts the plaintext bit-wise ($\Sigma = 1$), which would give an expansion factor of 28 and e.g. a $28 \cdot 32 = 896$ byte ciphertext for a 256-bit plaintext, if the RLWE blueprint was used unmodified. The actual numbers for NewHope differ slightly since the each plaintext bit is masked two (for NewHope-512) or four (for NewHope-1024) times, and NewHope applies some ciphertext compression, giving ciphertext sizes of 1088B and 2176B.

Choice of rings

RLWE is often used over polynomial rings $\mathbb{Z}[X]/(P(X))$, where $P(X)$ is an irreducible polynomial. The choice of the 2^{k+1} -th cyclotomic polynomial $P(X) = X^{2^k} + 1$ is particularly popular and used in the NIST PQC Round 3 Finalists CRYSTALS-Kyber, CRYSTALS-Dilithium and SABER. More generally, RLWE is studied for rings of integers in number fields.

The choice of $\mathbb{Z}[X]/(X^n + 1)$ with n a power of 2 is attractive for performance: for a prime q with $2n \mid q-1$ (as is the case for CRYSTALS-Kyber and CRYSTALS-Dilithium, but not for SABER), the *Number Theoretic Transform* allows us to represent elements $(\mathbb{Z}/q\mathbb{Z})[X]/(X^n + 1)$ in a way that multiplication has complexity which is linear in n , as opposed to the quadratic complexity of naïve polynomial multiplication.

Security of RLWE

The reduction of SVP to LWE carries over to RLWE. However, in contrast to LWE, it does not target SVP for general lattices, but only lattices which are contained within the chosen ring — so-called *ideal lattices*, examples of the “structured” lattices giving the field its name^{20,21}.

The reduction of Ideal-SVP to RLWE gives a less satisfactory lower bound on the hardness of RLWE than the one we obtain for LWE: generally, since SVP for structured lattices is less studied than for general lattices, and specifically, since recently [CDW16] a polynomial quantum algorithm was found for Ideal-SVP_{exp(√n)}. No such algorithm is known for general lattices, so there is — at least in asymptotical terms — a gap between the hardness of lattice problems for unstructured and structured lattices that was not expected at first.

Further variants of (R)LWE

Numerous other choices of bilinear maps \bullet for the mask generation have been studied: For example, as an “interpolation” between LWE (using vectors with scalar en-

²⁰The term “structured” lattice can also refer to lattices obtained from other algebraic variants of LWE, see [Further variants of \(R\)LWE](#).

²¹The “measurement approach” from [Main idea](#) goes through for RLWE in an analogous fashion, provided it is formulated in the right generality — see [LPR13].

tries) and RLWE (using ring elements) one can consider *Module-LWE* (MLWE), where the mask generating hints are vectors with ring entries. We recommend [\[PP19\]](#) for a survey and general treatment of those approaches.

Concrete security of (R)LWE

While the reductions from (Ideal-)SVP to (R)LWE give confidence in the *asymptotic* complexity of (R)LWE, they do not provide *concrete* security guarantees for specific cryptosystems and parameter choices. While those could be obtained by estimating the complexity of the reduction itself as well as the complexity of (Ideal-)SVP, the concrete security for lattice cryptosystems is usually directly estimated in terms of concrete cryptanalysis using known lattice algorithms. The interested reader can find more information on the [Estimate all the {LWE,NTRU} schemes!](#).

	Classical	Quantum Model	
		Probabilistic view	Algebraic view
Single unit	$\{0, 1\}$	Probability space on $\{0, 1\}$ $\left\{ p \cdot \underline{0} + q \cdot \underline{1} \mid \begin{array}{l} 0 \leq p, q \leq 1 \\ p + q = 1 \end{array} \right\}$ Convex combinations of 'pure' states	$\left\{ \alpha \cdot 0\rangle + \beta \cdot 1\rangle \mid \begin{array}{l} \alpha, \beta \in \mathbb{C} \\ \alpha ^2 + \beta ^2 = 1 \end{array} \right\}$ \mathbb{C} -linear combinations of 'pure' states
Multiple units	$\{0, 1\}^n$	Probability space on $\{0, 1\}^n$ $\left\{ \sum_{x \in \{0,1\}^n} p_x \underline{x} \mid \begin{array}{l} 0 \leq p_x \leq 1 \\ \sum_{x \in \{0,1\}^n} p_x = 1 \end{array} \right\}$ Convex combinations of 'pure' states	$\left\{ \sum_{x \in \{0,1\}^n} \alpha_x x\rangle \mid \begin{array}{l} \alpha_x \in \mathbb{C} \\ \sum_{x \in \{0,1\}^n} \alpha_x ^2 = 1 \end{array} \right\}$ \mathbb{C} -linear combinations of 'pure' states

Figure 16: State spaces in classical and quantum computing

References

- [AD97] Miklós Ajtai and Cynthia Dwork. "A Public-Key Cryptosystem with Worst-Case/Average-Case Equivalence". In: *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*. STOC '97. El Paso, Texas, USA: Association for Computing Machinery, 1997, pp. 284–293. ISBN: 0897918886. DOI: [10.1145/258533.258604](https://doi.org/10.1145/258533.258604). URL: <https://doi.org/10.1145/258533.258604>.
- [Ajt96] M. Ajtai. "Generating Hard Instances of Lattice Problems (Extended Abstract)". In: *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*. STOC '96. ACM, 1996, pp. 99–108.
- [App+09] Benny Applebaum et al. "Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems". In: *Advances in Cryptology - CRYPTO 2009*. Ed. by Shai Halevi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 595–618. ISBN: 978-3-642-03356-8.
- [AR05] Dorit Aharonov and Oded Regev. "Lattice Problems in $\text{NP} \cap \text{CoNP}$ ". In: *J. ACM* 52.5 (Sept. 2005), pp. 749–765. ISSN: 0004-5411. DOI: [10.1145/1089023.1089025](https://doi.org/10.1145/1089023.1089025). URL: <https://doi.org/10.1145/1089023.1089025>.
- [Aru+19] Frank Arute et al. "Quantum Supremacy using a Programmable Superconducting Processor". In: *Nature* 574 (2019), pp. 505–510. URL: <https://www.nature.com/articles/s41586-019-1666-5>.
- [BGV11] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. *Fully Homomorphic Encryption without Bootstrapping*. Cryptology ePrint Archive, Report 2011/277. 2011.
- [Bin+17] Nina Bindel et al. *Transitioning to a Quantum-Resistant Public Key Infrastructure*. Cryptology ePrint Archive, Report 2017/460. 2017.
- [BPR11] Abhishek Banerjee, Chris Peikert, and Alon Rosen. *Pseudorandom Functions and Lattices*. Cryptology ePrint Archive, Report 2011/401. 2011.
- [BSI20] Federal Office for Information Security Germany (BSI). *Migration zu Post-Quanten-Kryptografie*. Tech. rep. 2020. URL: <https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Krypto/Post-Quanten-Kryptografie.pdf>.
- [BV11] Zvika Brakerski and Vinod Vaikuntanathan. *Efficient Fully Homomorphic Encryption from (Standard) LWE*. Cryptology ePrint Archive, Report 2011/344. 2011.
- [CC20] Matt Campagna and Eric Crockett. *Hybrid Post-Quantum Key Encapsulation Methods (PQ KEM) for Transport Layer Security 1.2 (TLS)*. Internet-Draft draft-campagna-tls-bike-sike-hybrid-03. IETF Secretariat, Mar. 2020. URL: <https://tools.ietf.org/html/draft-campagna-tls-bike-sike-hybrid-03>.

- [CDW16] Ronald Cramer, Léo Ducas, and Benjamin Wesolowski. *Short Stickelberger Class Relations and application to Ideal-SVP*. Cryptology ePrint Archive, Report 2016/885. 2016.
- [Gab05] Philippe Gaborit. “Shorter keys for code-based cryptography”. In: (Jan. 2005).
- [Gen09] Craig Gentry. “A Fully Homomorphic Encryption Scheme”. PhD thesis. Stanford, CA, USA, 2009. ISBN: 9781109444506.
- [GGH97] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. “Eliminating Decryption Errors in the Ajtai-Dwork Cryptosystem”. In: *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*. Vol. 1294. Lecture Notes in Computer Science. Springer, 1997, pp. 105–111. DOI: [10.1007/BFb0052230](https://doi.org/10.1007/BFb0052230).
- [Gro96] Lov K. Grover. “A Fast Quantum Mechanical Algorithm for Database Search”. In: *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*. STOC '96. ACM, 1996.
- [GRS08] Henri Gilbert, Matthew J. B. Robshaw, and Yannick Seurin. “How to Encrypt with the LPN Problem”. In: *Automata, Languages and Programming*. Ed. by Luca Aceto et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 679–690. ISBN: 978-3-540-70583-3.
- [HRS15] Andreas Hülsing, Joost Rijneveld, and Peter Schwabe. *ARMed SPHINCS – Computing a 41KB signature in 16KB of RAM*. Cryptology ePrint Archive, Report 2015/1042. 2015.
- [JD11] David Jao and Luca De Feo. “Towards Quantum-resistant Cryptosystems from Supersingular Elliptic Curve Isogenies”. In: *Proceedings of the 4th International Conference on Post-Quantum Cryptography*. PQCrypto'11. 2011.
- [Kam+18] Panos Kampanakis et al. *The Viability of Post-quantum X.509 Certificates*. Cryptology ePrint Archive, Report 2018/063. 2018.
- [Lam79] Leslie Lamport. *Constructing Digital Signatures from a One Way Function*. Tech. rep. CSL-98. Oct. 1979. URL: <https://www.microsoft.com/en-us/research/publication/constructing-digital-signatures-one-way-function/>.
- [LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. “On Ideal Lattices and Learning with Errors over Rings”. In: *J. ACM* 60.6 (Nov. 2013).
- [Lyu09] Vadim Lyubashevsky. “Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures”. In: *Advances in Cryptology – ASIACRYPT 2009*. Ed. by Mitsuru Matsui. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 598–616. ISBN: 978-3-642-10366-7.
- [Mau12] Ueli Maurer. “Constructive Cryptography – A New Paradigm for Security Definitions and Proofs”. In: *Theory of Security and Applications*. Ed. by Sebastian Mödersheim and Catuscia Palamidessi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 33–56.
- [McE78] R. J. McEliece. *A Public-Key Cryptosystem Based On Algebraic Coding Theory*. The Deep Space Network Progress Report. 1978.
- [Mer79] Ralph Charles Merkle. “Secrecy, Authentication, and Public Key Systems.” PhD thesis. 1979.
- [MI88] Tsutomu Matsumoto and Hideki Imai. “Public Quadratic Polynomial-Tuples for Efficient Signature-Verification and Message-Encryption”. In: *Advances in Cryptology — EUROCRYPT '88*. Ed. by D. Barstow et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 1988, pp. 419–453. ISBN: 978-3-540-45961-3.
- [NC11] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. 10th. USA: Cambridge University Press, 2011. ISBN: 1107002176.
- [NM19] Engineering National Academies of Sciences and Medicine. *Quantum Computing: Progress and Prospects*. Ed. by Emily Grumbling and Mark Horowitz. Washington, DC: The National Academies Press, 2019. ISBN: 978-0-309-47969-1. DOI: [10.17226/25196](https://doi.org/10.17226/25196). URL: <https://www.nap.edu/catalog/25196/quantum-computing-progress-and-prospects>.
- [PP19] Chris Peikert and Zachary Pepin. *Algebraically Structured LWE, Revisited*. Cryptology ePrint Archive, Report 2019/878. <https://eprint.iacr.org/2019/878>. 2019.
- [PQC19] PQC mailing list. “Footguns as an axis for security analysis”. In: 2019. URL: <https://groups.google.com/a/list.nist.gov/forum/#!topic/pqc-forum/l2iYk-8sGnI>.
- [Reg04] Oded Regev. “New Lattice-Based Cryptographic Constructions”. In: *J. ACM* 51.6 (Nov. 2004), pp. 899–942. ISSN: 0004-5411. DOI: [10.1145/1039488.1039490](https://doi.org/10.1145/1039488.1039490). URL: <https://doi.org/10.1145/1039488.1039490>.

- [Reg09] Oded Regev. “On Lattices, Learning with Errors, Random Linear Codes, and Cryptography”. In: *J. ACM* 56.6 (Sept. 2009). ISSN: 0004-5411. DOI: [10.1145/1568318.1568324](https://doi.org/10.1145/1568318.1568324). URL: <https://doi.org/10.1145/1568318.1568324>.
- [Reg10] Oded Regev. “The Learning with Errors Problem (Invited Survey)”. In: *Proceedings of the 2010 IEEE 25th Annual Conference on Computational Complexity*. CCC ’10. USA: IEEE Computer Society, 2010, pp. 191–204. ISBN: 9780769540603. DOI: [10.1109/CCC.2010.26](https://doi.org/10.1109/CCC.2010.26). URL: <https://doi.org/10.1109/CCC.2010.26>.
- [RP11] Eleanor Rieffel and Wolfgang Polak. *Quantum computing*. Scientific and Engineering Computation. A gentle introduction. MIT Press, Cambridge, MA, 2011, pp. xiv+372. ISBN: 978-0-262-01506-6.
- [SFG20] Douglas Stebila, Scott Fluhrer, and Shay Gueron. *Hybrid key exchange in TLS 1.3*. Internet-Draft draft-stebila-tls-hybrid-design-03. IETF Secretariat, Feb. 2020. URL: <https://tools.ietf.org/html/draft-stebila-tls-hybrid-design-03>.
- [Sho94] P. W. Shor. “Algorithms for Quantum Computation: Discrete Logarithms and Factoring”. In: *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*. SFCS ’94. IEEE Computer Society, 1994, pp. 124–134.
- [Tru+18] Alexander Truskovsky et al. *Multiple Public-Key Algorithm X.509 Certificates*. Internet-Draft draft-truskovsky-lamps-pq-hybrid-x509-01. IETF Secretariat, Aug. 2018. URL: <http://www.ietf.org/internet-drafts/draft-truskovsky-lamps-pq-hybrid-x509-01.txt>.