INFOSEC GLOBAL

WHAT DOES AGILITY MEAN FOR POST-QUANTUM CRYPTOGRAPHY SOLUTIONS?

Dr. Basil Hess, InfoSec Global Research Team

The ability to encrypt sensitive information is the key to everything from our digitally driven economy to our national security. Within a decade, however, quantum computing could make existing cryptographic solutions obsolete.

At InfoSec Global, we are working alongside the world's best cryptographers to come up with solutions that will extend the effectiveness of encryption in a post-quantum world. But the key to our success is the concept of crypto-agility – the ability to transition to different cryptographic solutions as new vulnerabilities are revealed.

OUR QUANTUM FUTURE

Why are quantum computers such a threat to existing crypto solutions?

Quantum computers take advantage of the unique characteristics of photons, which can exist in multiple states at the same time. In contrast to a binary computer bit, whose value is always either 0 or 1, a quantum bit (or qubit) can be a 0, 1, or both simultaneously – a quality known as superposition.

Superposition allows quantum computers to perform vast numbers of mathematical operations at the same time, which could make them orders of magnitude more powerful than today's fastest supercomputers.

Most cryptographic schemes are difficult to crack because doing so requires enormous amounts of time and computing power. But a quantum computer of sufficient size could be used to break the most common cryptographic schemes in use today, particularly public-key cryptosystems such as RSA and elliptic-curve cryptography (ECC), by running Shor's algorithm.

While questions remain about how practical large-scale quantum computing may ultimately be, IBM, Google, Microsoft and others have been steadily pushing the envelope. Last March, Google introduced Bristlecone, the largest quantum processor yet with 72 qubits. In July it released an open source framework called CIRQ that will allow researchers to begin programming quantum computers.

Because conventional computers and quantum machines will likely co-exist for many years to come, the world needs quantum-safe crypto alternatives now.

This is why, in 2016, the National Institute of Science and Technology (NIST) issued a call for new post-quantum cryptographic algorithms (PQC). Nearly 70 proposals were submitted for Round 1 of the competition. Two of those proposals – Supersingular Isogeny Key Encapsulation (SIKE) and an enhancement to an existing stateless hash-based signature scheme (SPHINCS+) – were submitted by InfoSec Global, in collaboration with researchers at Amazon, Microsoft, and elsewhere. Ten have since been withdrawn, leaving 59 submissions to undergo further scrutiny in Round 1.

Most of the submissions are not based on new techniques but rather on mathematical principles that have been known and studied for years. However, more time and studies are required to improve the practicality of these schemes and increase our confidence in them. After all, they will replace critical security components of the world's infrastructure. Luckily, as with previous competitions run by NIST, the crypto research community will thoroughly examine all of these submissions.

AGILITY IS FUNDAMENTAL

While we wait for quantum computers to mature and post-quantum crypto primitives to emerge, cryptographic lifecycle management has become increasingly essential for every organization.

As noted above, a key concept embedded within lifecycle management is agility – the ability to transition from one cryptographic scheme to another as vulnerabilities emerge or as schemes are phased out. It's especially important to have cryptographic solutions that can eventually migrate to the post-quantum world.

The PQC algorithmn that have been submitted vary in the degree of agility they provide. The level of agility is also influenced by the crypto solutions an organization currently deploys. The cost of ripping out and replacing legacy software and hardware is likely to be considerable, so one needs a solid understanding of each crypto scheme and how compatible it is with existing applications, in order to choose schemes that minimize the burden.

To help with this task, we looked at the NIST candidates and evaluated their degree of cryptographic agility. To simplify matters, we've broken down the agility concept into four fundamental aspects -- API, keys, performance, and hardware support -- and will discuss each of these in turn.

AGILITY ASPECT 1: API

No matter how or where they are applied, all cryptographic schemes are accessible via an application programming interface (API), using an abstraction layer such as signatures or key agreement schemes. As an example, the Diffie-Hellman (DH) protocol and its elliptic-curve analogue ECDH can both be abstracted with a common Key Agreement API. These two schemes are called Diffie-Hellman style protocols.

While today's most commonly used key agreement schemes are DH and ECDH, the PQC key agreement submissions all fall in another API category, so-called Key Encapsulation Mechanisms (KEM). Migrating to a post-quantum key agreement scheme will thus in many cases require applications to use a KEM API instead of an API for Diffie-Hellman style protocols. The agility of these schemes is therefore limited.

There is one exception in PQCs that falls in the same category as DH-style schemes: the Supersingular Isogeny Diffie Hellman (SIDH) scheme. Below, we briefly describe ECDH and SIDH to show their similarities. In this scenario, users Alice and Bob agree on a shared secret key. This process involves two steps that the cryptographic library exposes with an abstracted API:

- **Key generation.** The key generation API outputs a public/private keypair.
- **Key derivation**. The key derivation API takes Alice's private key and Bob's public key, and outputs a shared secret key.

ECDH

The global public parameters are as follows:

- Prime p, for example a Mersenne prime of the form $p = 2^n - 1$
- Elliptic curve $E: y^2 = x^3 + ax + b$ with coefficients (a, b) defined over F_p(finite field of size p)
- The order n of the curve E.
- A base point P, which is a generator of E.

For key generation, Alice does the following:

- Randomly selects $d_A \in [0, n-1]$, which constitutes the private key.
- Computes $Q_A = d_A P$, which constitutes the public key.
- Sends the public key Q_A to Bob.

Bob does the same, symmetrically, obtaining the private/public keypair (d_B, Q_B)(   ).

For key derivation, Alice does the following.

- Using her private key value d_A and Bob's public key Q_B, computes $K_{AB} = d_A Q_B$.
- Observation: $d_A Q_B = d_A d_B P = d_B d_A P = d_B Q_A$

Bob does the same, symmetrically, obtaining $K_{AB} = d_B Q_A$.

SIDH

The global public parameters are as follows:

- Prime $p = l_A^{e_A} \cdot l_B^{e_B} \cdot f \pm 1$, where l_A and l_B are small primes and f is a small cofactor
- Elliptic curve $E: y^2 = x^3 + ax + b$ with coefficients (a, b) defined over F_{p^2}(finite field of size p^2)
- Bases points $\{P_A, Q_A\}$ and $\{P_B, Q_B\}$, which are generators of E[$l_A^{e_A}$] and E[$l_B^{e_B}$], respectively

For key generation, Alice does the following:

- Randomly selects m_A,n_A ∈ $Z_{l_A^{e_A}}$. They constitute the private key.
- Computes $K_A = m_A P_A + n_A Q_A$.
- Obtains E_A using the kernel ⟨ K_A ⟩ for corresponding isogeny φ_A: E → E_A.
- Computes the values of P_B and Q_B under her isogeny φ_A, namely $\varphi_A(P_B)$ and $\varphi_A(Q_B)$. These values are referred to as auxiliary points.
- Sends E_A and auxiliary points $\varphi_A(P_B)$ and $\varphi_A(Q_B)$ to Bob. They constitute the public key.

Bob does the same, symmetrically having the private key m_B,n_B ∈ $Z_{l_B^{e_B}}$ and the public key E_B and auxiliary points $\varphi_B(P_A)$ and $\varphi_B(Q_A)$.

For key derivation, Alice does the following:

- Using her private key values m_A,n_A and Bob's auxiliary points $\varphi_B(P_A)$, $\varphi_B(Q_A)$ on E_B, computes $m_A \cdot \varphi_B(P_A) + n_A \cdot \varphi_B(Q_A)$.
- Observation: $m_A \cdot \varphi_B(P_A) + n_A \cdot \varphi_B(Q_A) = \varphi_B(m_A \cdot P_A) + \varphi_B(n_A \cdot Q_A) = \varphi_B(m_A P_A + n_A Q_A) = \varphi_B(K_A)$.
- Hence, it is the image of Alice's kernel generator point in Bob's curve EB.
- Using that value as the generator point for the kernel, ⟨$\varphi_B(K_A)$⟩, Alice maps E_B → E_{AB}.
- Computes the j-invariant of E_{AB} and uses that as a value of the common key.

Bob does the same, symmetrically, and obtains the j-invariant of E_{BA}.

In contrast to DH-style algorithms, KEM schemes have three functions. The first one is the Key Generation phase, which is analogous to DH-style schemes. Then there are two more functions:

- KEM Encapsulation: Takes the public key as an input, creates a random shared secret 's', encrypts it and outputs a ciphertext 'ct'
- KEM Decapsulation: Takes the private key as an input, decrypts the ciphertext `ct` and returns the shared secret `s`.

Thus, because of its API agility, SIDH can be seen as an excellent candidate as a quantum-safe replacement of ECDH. There are only two limitations: First, SIDH must only be used for ephemeral keys. Second, Alice "the sender" and Bob "the responder" don't operate on the same subgroup, so one must explicitly pass an argument to specify "Alice" or "Bob" to the API.

InfoSec Global provides solutions for both limitations: It has co-submitted the KEM scheme SIKE as part of the NIST submission. It further provides a SIDH-variation called PQDH that doesn't suffer from the second limitation.

API agility also makes SIDH a good candidate to be included in protocols like TLS. Cloudflare has recently published a draft for hybrid ECDHE-SIDH key-exchanges.

AGILITY ASPECT 2: KEYS

Another important aspect of a cryptosystem is the public/private key pairs used in the asymmetric PQCs. Public keys are always specific to a cryptosystem, making it difficult to migrate them to a new cryptosystem. On the other hand, private keys are typically just a series of bytes selected uniformly and at random. Reusing those keys in an agile way among the cryptosystems is therefore possible, though one may need to adjust for different key sizes.

In Tables 1 and 2 we list different categories of PQCs for KEM and Signature schemes, respectively. Under PQC categories we list isogeny-based ECC, lattice-based schemes, multivariate-based schemes, and code-based schemes for KEM, along with one promising candidate for each. For Signatures, we additionally included the category of hash-based signatures.

WHAT IS THE STRUCTURE OF PQC'S PRIVATE KEYS?

Candidates where the private keys are indeed strings chosen uniformly and at random include SIKE, DME, SPHINCS+, and MQDSS. The other candidates employ a different structure for their private keys.

For example, the private keys of the PQC called New Hope are encoded polynomials that are sampled from a binomial distribution. In the submission Classic McEliece, the private key includes a uniformly random string, but also a monic irreducible polynomial. The private keys of Dilithium are sampled from a Gaussian distribution. The private keys of pqsigRM contain two random matrices, as well as a set of matrix indexes.

In most current cryptographic schemes, it is quite straight-forward to generate private keys because they are just random values. Schemes that use non-uniformly distributed keys, such as those that employ Gaussian or binomial distribution, can be seen as less agile.

WHAT ARE THE SIZES OF THE PQC KEYS?

The agility and applicability of a PQC greatly depends on its application, and an important aspect are the key sizes. That's especially true of embedded systems, which often have limited storage and bandwidth for exchanging keys. This is why it's better to select a PQC with minimal key sizes for these applications.

Table 1 and Table 2 list the public/private key sizes for the submissions that fall in NIST category 1 or 2. The numbers are taken from the SafeCrypto PQLounge.

Table 1: Keys in KEM schemes (sizes in bytes)

Category	Scheme	Public key size	Private key size	Private key uniformly random
Isogeny ECC	SIKEp503	378	434	Yes
Lattice	NewHope512-CCA	929	1’888	No
Multivariate	DME-(3,2,48)	2’304	288	Yes
Code	Classic McEliece 6960119	1’047’319	13’908	No

Table 2: Keys in Signature schemes (sizes in bytes)

Category	Scheme	Public key size	Private key size	Private key uniformly random
Hash-based	SPHINCS+-Haraka	32	64	Yes
Lattice	Dilithium	1’184	2’800	No
Multivariate	MQDSS	62	32	Yes
Code	pqsigRM	336’804	1’382’118	No

The relatively short keys generated by isogeny-based, lattice-based, and hash-based schemes makes them good candidates for memory- and bandwidth-constrained applications.

Here, their agility mainly depends on the size of the keys. Isogeny-based schemes and hash-based schemes with quite short key sizes are more applicable to memory-constrained environments, while other schemes with keys of 1MB or more would be too heavy. Another interesting aspect is the structure of the keys, and if the structure can be reused for new PQC schemes. Uniformly random keys can make the migration to PQC easier, which will however in most cases not be a big issue since new key-pairs have to be generated for PQC algorithms.

AGILITY ASPECT 3: PERFORMANCE

Crypto agility can also depend on the application using the scheme. For example, embedded or IoT applications may have minimal performance requirements but limited bandwidth and memory. With data centers, key storage and communication are less of an issue; there, performance concerns focus on lower energy consumption and hardware cost.

We compared the performance of the same schemes described in the previous section, listing the results in Tables 3 and 4, respectively. The numbers are taken from the SafeCrypto PQLounge.

Table 3: Performance of KEM schemes (in CPU cycles)

Category	Scheme	Key generation	Encapsulation	Decapsulation
Isogeny ECC	SIKEp503	82’329’570	133’880’410	142’428’861
Lattice	NewHope512-CCA	513’054	776’525	874’199
Multivariate	DME-(3,2,48)	445’585’460	2’114’390	10’845’706
Code	Classic McEliece 6960119	2’406’818’088	1’756’816	498’750’958

Table 4: Performance of Signature schemes (in CPU cycles)

Category	Scheme	Key generation	Sign	Verify
Hash-based	Sphincs+-Haraka-128f	20’986’256	847’479’191	32’631’665
Lattice	Dilithium	227’254	910’911	291’116
Multivariate	MQDSS-48	2’579’234	252’403’091	185’066’255
Code	pqsigRM412	18’062’152’610	33’057’982’128	301’873’276

Lattice-based proposals usually provide the best performance. It should be noted that the measurements use the “Optimized Implementations” for the NIST PQC competition, with the restriction that only portable C is allowed with no assembly optimizations. SIKE provides additional assembly-optimized versions that increase the performance by an order of a magnitude, making it more competitive with the other categories.

AGILITY ASPECT 4: HARDWARE SUPPORT

Another agility aspect is hardware support. At first glance, because support for cryptographic algorithms is usually fixed for a specific algorithm, this would appear to be the least agile. Applying the same hardware to other algorithms is a non-trivial task.

However, some PQCs can use parts of existing hardware. Here we discuss two examples: The SIKE KEM and the SPHINCS+ signature scheme:

Hardware support for SIKE

The core of the SIKE KEM is elliptic-curve cryptography. ECC has already been widely implemented in hardware, and even very constrained devices such as Smartcards often provide ECC acceleration. If we have a closer look at SIKE, its implementation can be split into the following components:

1. Multi-precision arithmetic
2. Finite field arithmetic over F_P.
3. Quadratic extension field arithmetic over F_{p^2}.
4. (Montgomery) ECC arithmetic
5. Isogeny computation and evaluation

Multi-precision arithmetic is prevalent in existing cryptosystems like RSA and ECC, and existing hardware also often contains accelerators for finite field arithmetic, although for different primes than used in SIKE. New hardware can still benefit from the design experience in previous finite field implementations. The extension field arithmetic over F_{p^2} can build on top of the F_P arithmetic.

It is also worth noting that SIKE already provides an implementation for FPGA.

Hardware support for SPHINCS+

Another example of a PQC that can benefit from existing hardware support is the hash-based signature scheme SPHINCS+. The authors propose three implementations based on different hash functions: SHAKE256, SHA256, and Haraka. Haraka is especially designed with hardware reusability in mind. It uses AES, which is implemented in many current CPUs (e.g. AES-NI on Intel x86 or in the cryptographic extensions of ARM v8).

Hardware support for other schemes

For other PQC schemes, there is no commonly available hardware support yet. As an example, lattice-based schemes rely on either vector-matrix multiplications or matrix-matrix multiplications, which will need to be implemented in hardware.

Enabling support for these schemes in commonly used hardware or CPUs will help accelerate the migration to post-quantum algorithms, especially in performance-constrained embedded systems. While FPGA-based hardware implementations have already been proposed, commodity chips for handling lattice cryptographic schemes are still several years away.

THE TIME TO PREPARE IS NOW

Ultimately, cyber resilience requires cryptographic agility and interoperability.

As we have shown, the PQC schemes under consideration by NIST differ quite significantly in many aspects. As a result, there will never be a “one size fits all” replacement for today’s endangered crypto schemes. This will make the crypto world even more diverse and complex and make cryptographic lifecycle management a crucial component of every organization's security strategy.