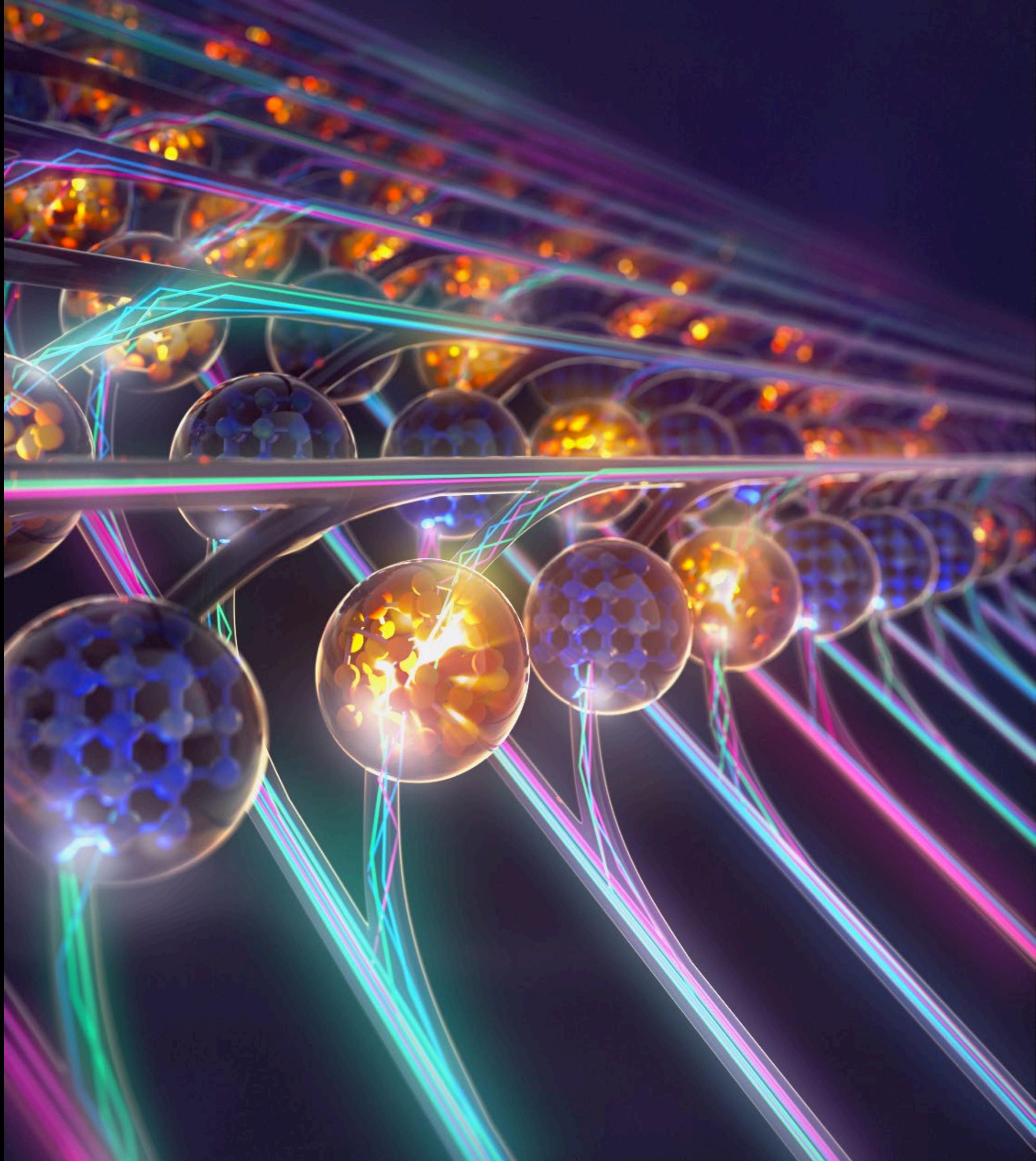
IBM Research

# Quantum Safe Cryptography for Long Term Security

SDC 2021

Basil Hess
IBM Research Europe - Zurich
bhe@zurich.ibm.com



# Agenda

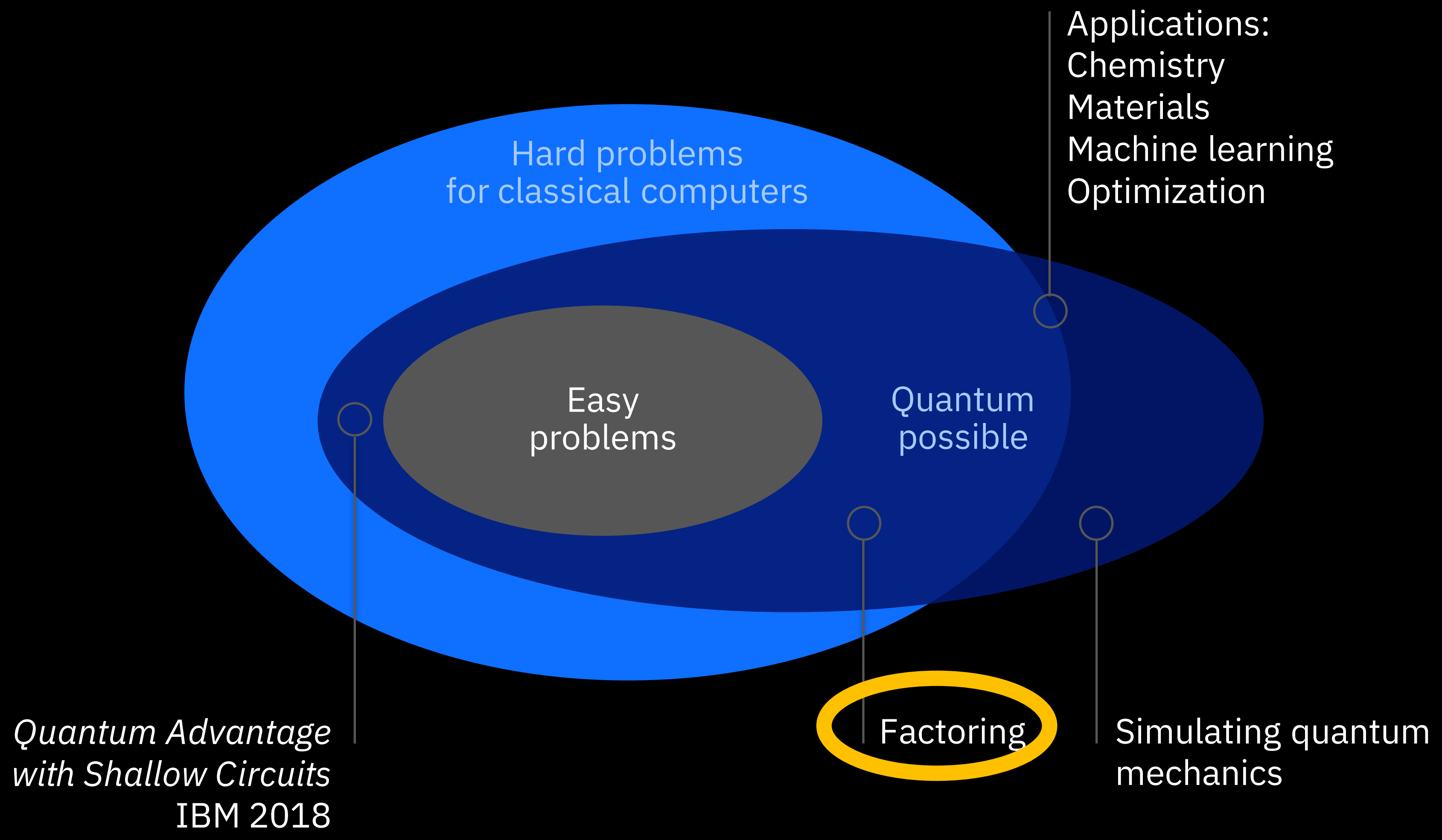
Quantum Risk

Quantum Safe Cryptography

Quantum Safe Applications and Libraries

# Quantum Risk

# Quantum Computing
*Applications*



# ***Quantum risk***

*Exponential speedup for some algorithms*

A quantum computer can solve certain problems much **faster**.

**2048-bit composite integer**

25195908475657893494027183240048398571429282126204032027777
13783604366202070759555626401852588078440691829064124951508
21892985591491761845028084891200728449926873928072877767359
71418347270261896375014971824691165077613379859095700097330
45974880842840179742910064245869181719511874612151517265463
22822168699875491824224336372590851418654620435767984233871
84774447920739934236584823824281198163815010674810451660377
30605620161967625613384414360383390441495263443219011465754
44541784240209246165157233507787077498171257724679629263863
56373289912154831438167899885040445364023527381951378636564
39212010397122822120720357

**Problem: find prime factors**

= p x q

**Expected computation time:**

Most powerful computer today: millions of years

Shor's Quantum Algorithm: Some hours

# Quantum risk

*Quantum algorithms and impact on classical cryptography*

Shor's algorithm completely breaks many asymmetric Public Key Cryptography schemes:

RSA, ECC, ECDH, EDSA

| Factoring Algorithm (RSA) | | EC Discrete logarithm (ECC) | |
|---|---|---|---|
| N bits | Approx #qubits | N bits | Approx #qubits |
| | 2n | | F'(n) |
| 512 | 1024 | 110 | 700 (800) |
| 1024 | 2048 | 163 | 1000 (1200) |
| 2048 | 4096 | 224 | 1300 (1800) |
| 3072 | 6144 | 256 | 2800 (3600) |

Grover's algorithm halves the security of symmetric algorithms:

AES, SHA 2, SHA 3

# *Quantum risk*

*Outlook*

Physical Qubits Required for RSA 2048

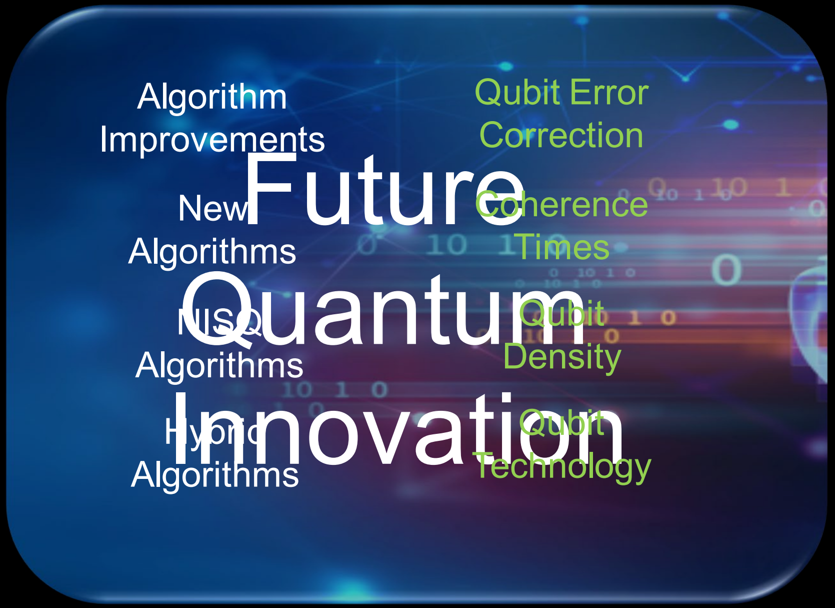
2012 : 1,000,000,000

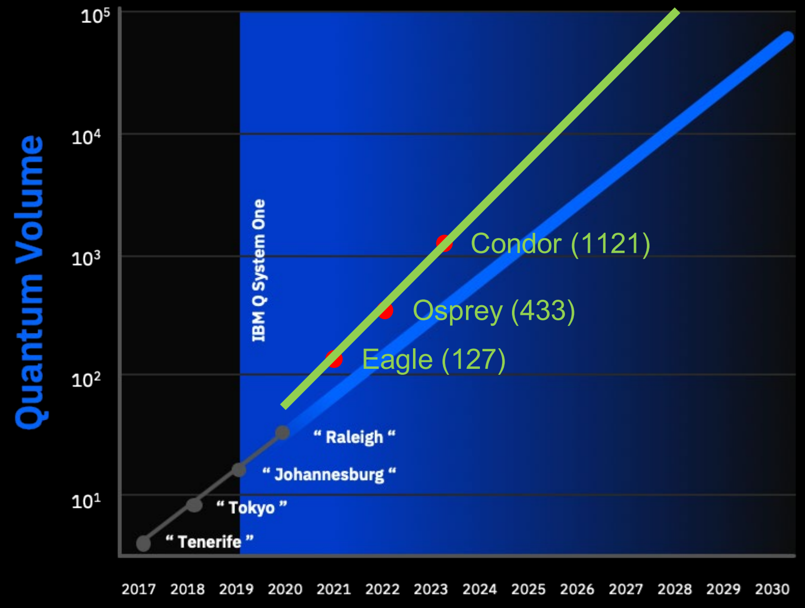
2017: 230,000,000

2019: 170,000,000

2019: 16,000,000

ECC requires fewer logical Qubits – likely to be at risk earlier




1. Significant progress on quantum algorithm efficiency
2. Quantum technology roadmaps becoming more tangible

# Quantum risk

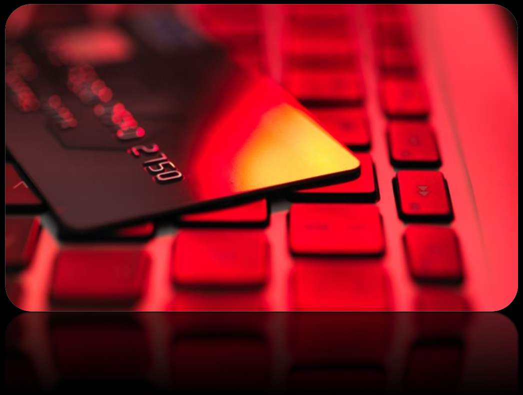
*What will a cyber criminal be able to do ?*

## 1

Attack systems through fraudulent authentication

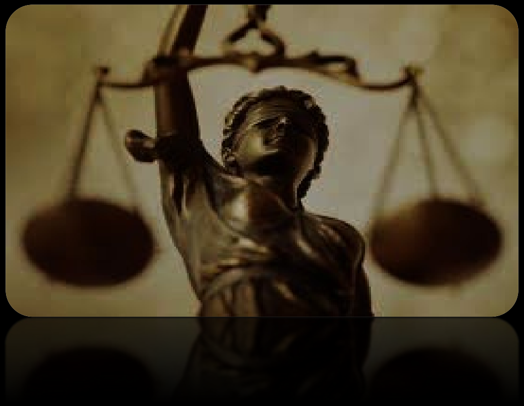
## 2

Harvest then later decrypt confidential information



## 3

Manipulate legal history through forged digital signatures



**Implications ->**

**The Systems that we are building today are vulnerable**

**The Data that we are protecting today is vulnerable**

**The legal underpinnings of digitalization are vulnerable**

# Quantum vulnerable legacy

*Systems live on after IT is no longer supported*

"About 75 percent of the devices that are control systems are on Windows XP or other non supported operating systems," said Daryl Haegley, program manager for the Office of the Assistant Secretary of Defense for Energy, Installations and Environment.

https://www.defenseone.com/technology/2017/04/pentagons-bug-bounty-program-should-be-expanded-americas-military-bases-dod-official-says/137229/

# Quantum vulnerable legacy

*Computer Systems that still run after 50 years*

" Treasury Department/Internal Revenue Service"

Individual Master File: A massive application that receives taxpayer data and dispenses refunds. "This investment is written in assembly language code -- a low-level computer code that is difficult to write and maintain -- and operates on an IBM mainframe

https://www.nextgov.com/cio-briefing/2016/05/10-oldest-it-systems-federal-government/128599/

https://www.computerhistory.org/timeline/1961/

# Quantum vulnerable technology

*Long term communication and storage confidentiality*

Encrypted data can be harvested to be decrypted later, when the quantum computer will be available

Healthcare data:

- Clinical trials (US): 25 years
- Health records (Jap): 100 years

Government data:

- Census records (UK): 100 years
- Toxic Substances Control Act: 30 years

Finance data

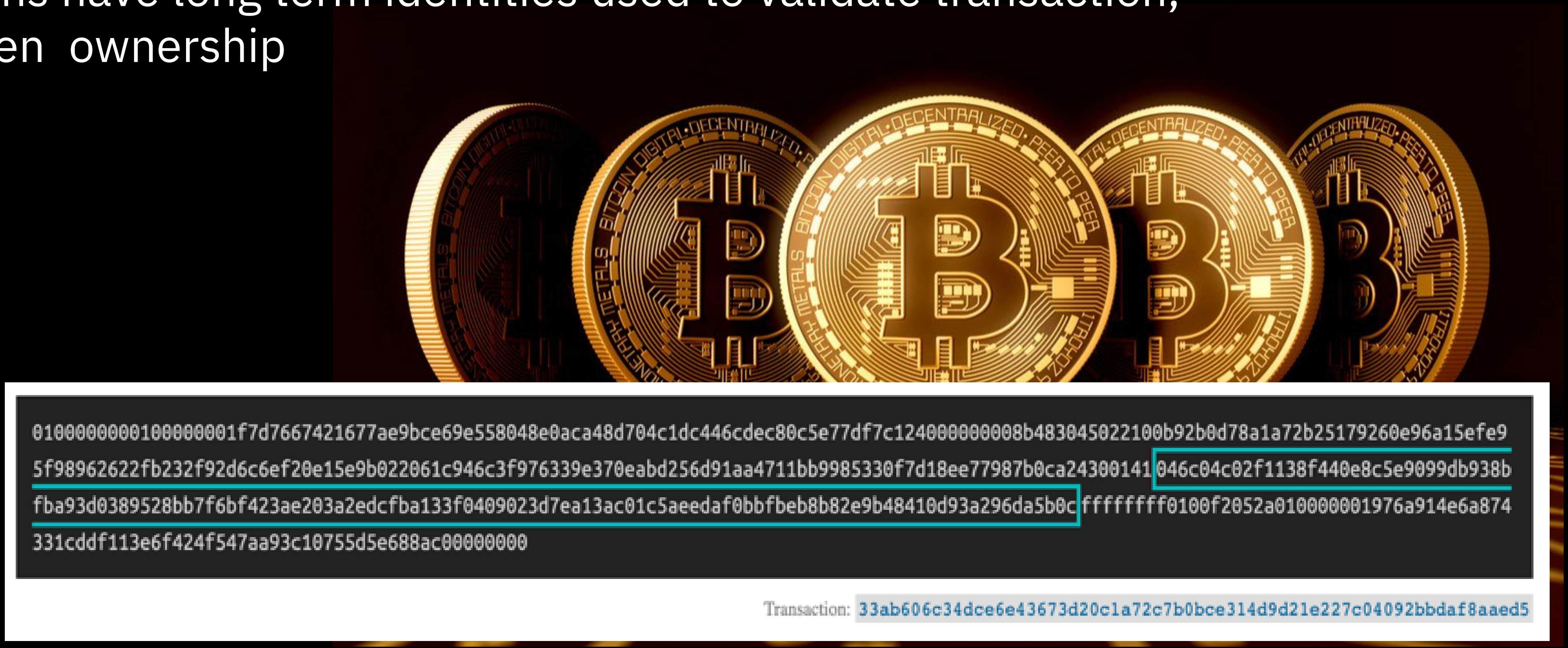
- Tax records: 7-10 years in most countries
- Payroll records (Rou): 50 years

# Quantum vulnerable technology

*We are still creating vulnerable legacy*

Bitcoin is over a decade old

Blockchains have long term identities used to validate transaction, asset/token ownership



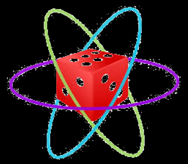
# Quantum risk

*Summary*

1. *The impact is in the future but the problem is NOW*
2. *Need for new cryptography*
3. *Need for transition to new cryptography*

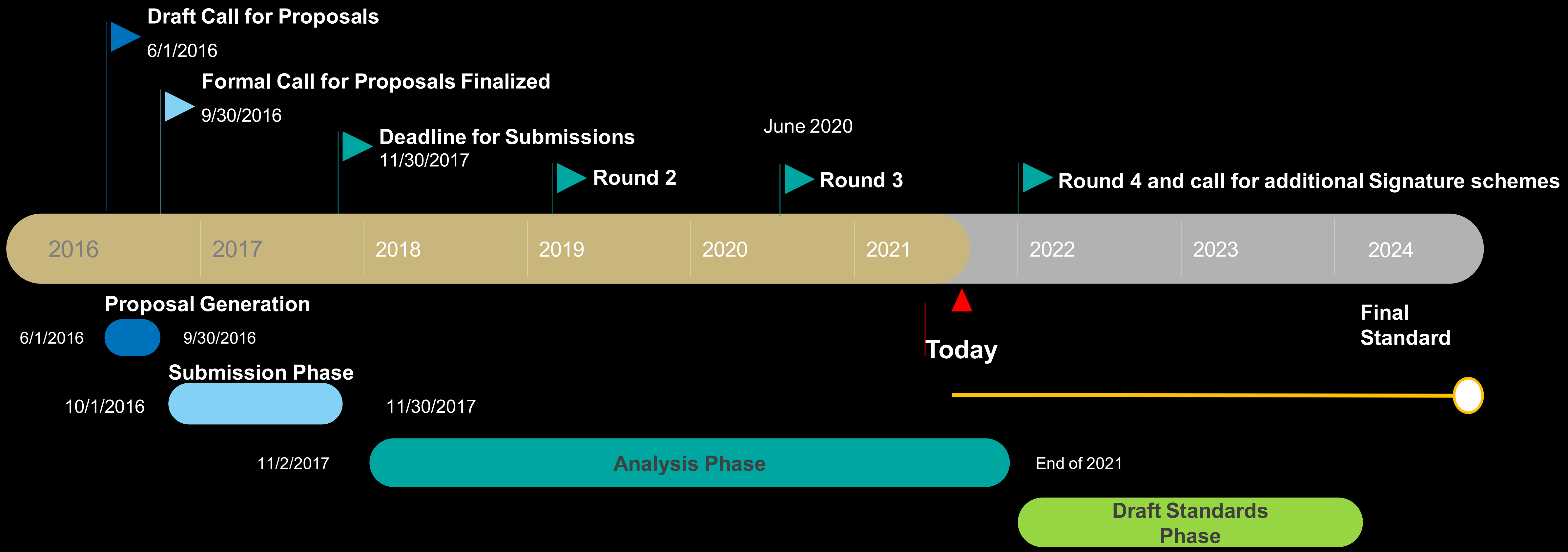
# Quantum risk

*Quantum Safe Technologies*

| | | Confidentiality | Identification | Integrity | Repudiation |
|---|---|---|---|---|---|
| NIST National Institute of Standards and Technology | Quantum Safe Cryptography QSC) Cryptography based mathematical problems difficult for classical and quantum computers to solve efficiently | ✓ | ✓ | ✓ | ✓ |
|  | Quantum key distribution (QKD) allows the exchange of a cryptographic key between two remote parties | ✓ | ✗ | ✗ | ✗ |
|  | Quantum Random Number Generator (QRNG) Generates randomness from a quantum effect | ✗ | ✗ | ✗ | ✗ |

# Quantum Safe Cryptography

# Quantum Safe Algorithms
## NIST PQC Standardization



# Quantum Safe Algorithms

*Standards – NIST – Round 3*

Finalists (some will be standardized after Round 3)

KEMs
1. CRYSTALS–Kyber (Lattice)
2. Classic McEliece (Code)
3. NTRU (Lattice)
4. SABER (Lattice)

Signatures
1. CRYSTALS–Dilithium (Lattice)
2. FALCON (Lattice)
3. Rainbow (Multivariate)

Alternates (some may be standardized as backup or after a further round)

KEMs
1. BIKE (Code)
2. Frodo (Unstructured Lattice)
3. HQC (Code)
4. NTRU Prime (Lattice)
5. SIKE (Isogenies)

Signatures
1. GeMSS (Multivariate)
2. Picnic (Hash functions)
3. Sphincs+ (Hash functions)

# ***Quantum Safe Algorithms***

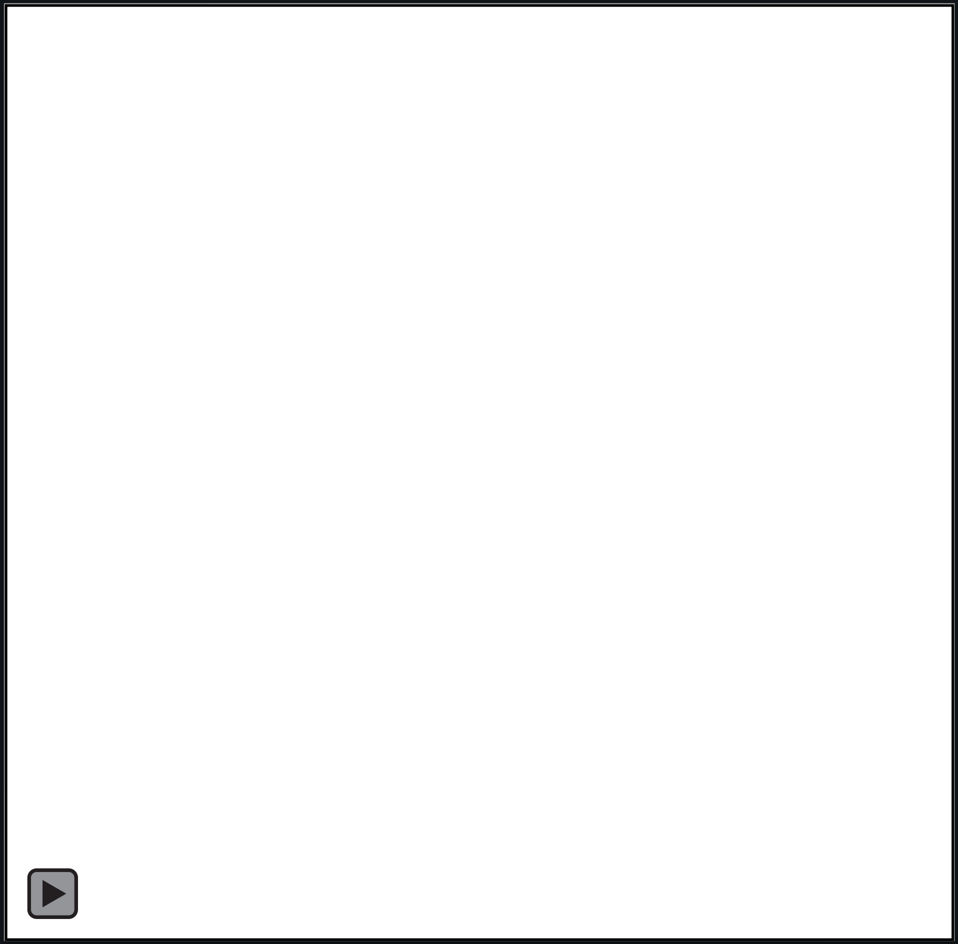
*Lattice cryptography*

1 From the field of mathematics called ‘the geometry of numbers’

2 Allows us to formulate problems such as “given the blue vectors find the red secret”

Lattice cryptography combines many advantages for quantum safe cryptography:

- Extremely efficient and fast implementations
- Keys and Signatures larger – but not massively
- The least  impact on protocols and applications that need to migrate
- A versatile building block for more complex cryptographic schemes, e.g. FHE



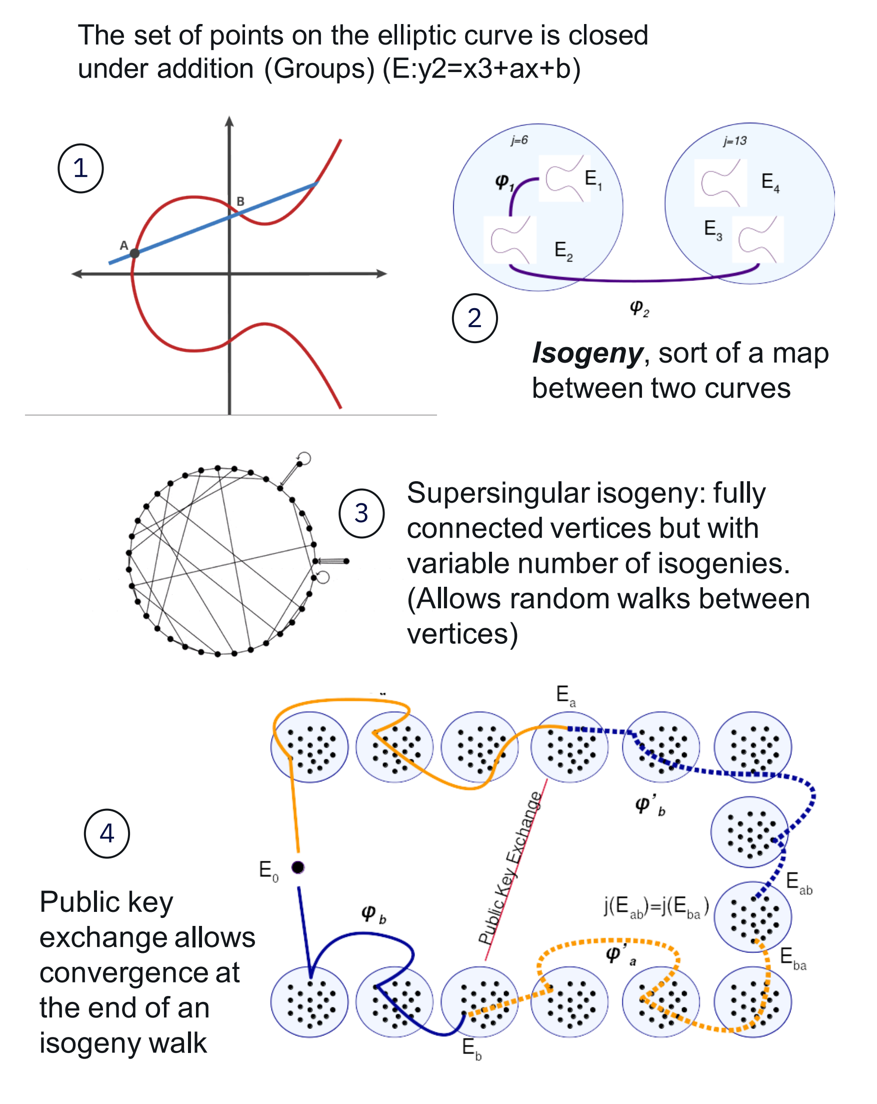
# Quantum Safe Algorithms

*Elliptic Curve Supersingular Isogenies*

**1** The core idea in SIDH is to compose two random walks on an isogeny graph of elliptic curves in such a way that the end node of both ways of composing is the same.

SIDH cryptography is interesting for a number of reasons:

- One of the few post quantum schemes that allows a Diffie Hellman type key exchange
- Operation based on well known elliptic curve functions
- Public keys are very small (but the performance is as well slow)
- Interesting for certain networks – for example high latency



# *Quantum Safe Algorithms*

*What Changes*

**There will not be a single new scheme but a small 'portfolio' of 4 or 5 schemes**

- They will be based on different hard problems
- There is a backup list just in case

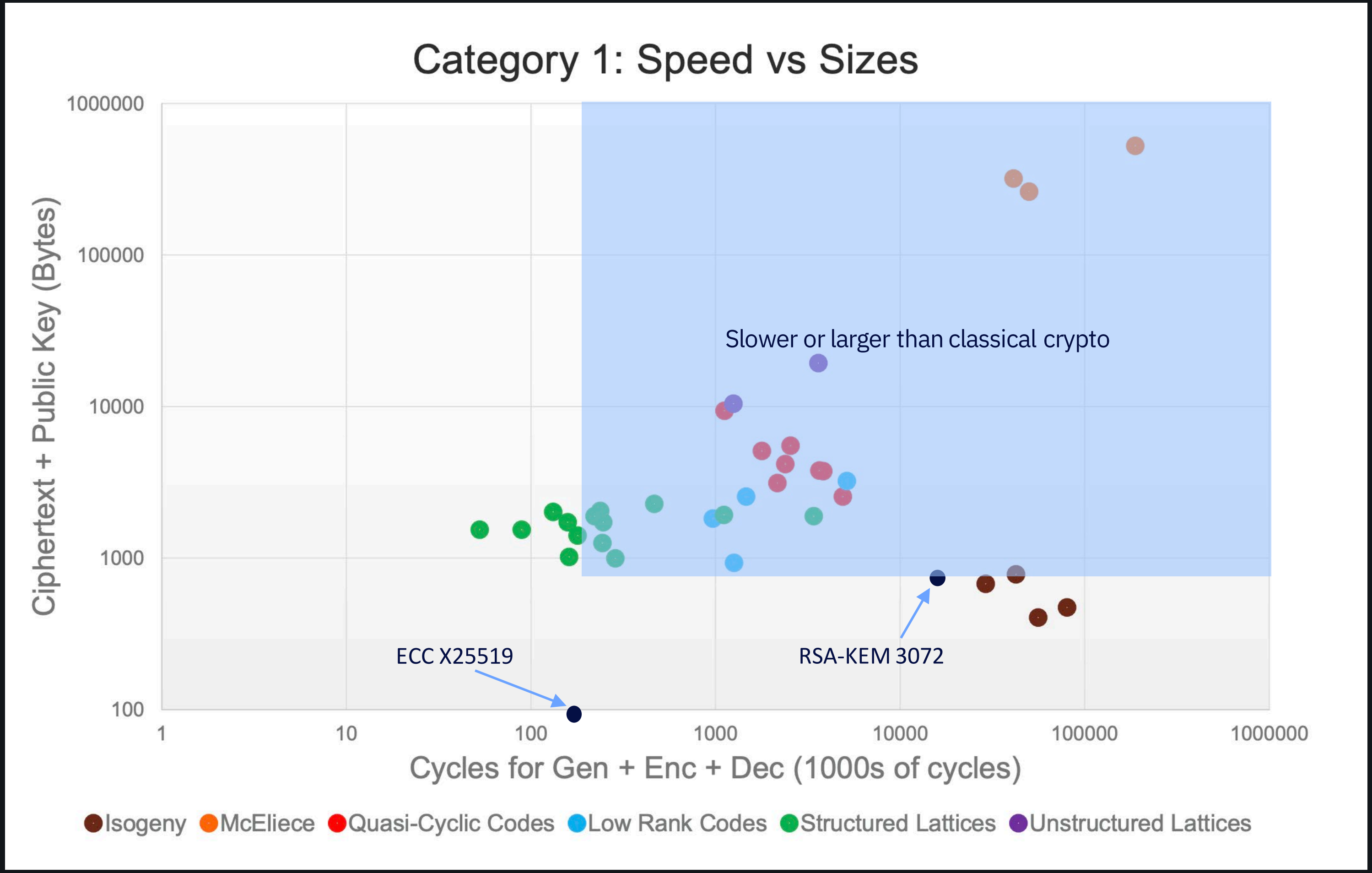
**The finalists are selected because they are based on different properties**

- Some schemes suitable only for specific use cases, for example 'code based crypto' with very large keys
- Structured Lattice Based Systems are the most versatile quantum safe alternatives

Hybrid Schemes promise a gradual transition

# Quantum Safe Algorithms

*KEM: Impact of Public Key and Ciphertext Sizes*

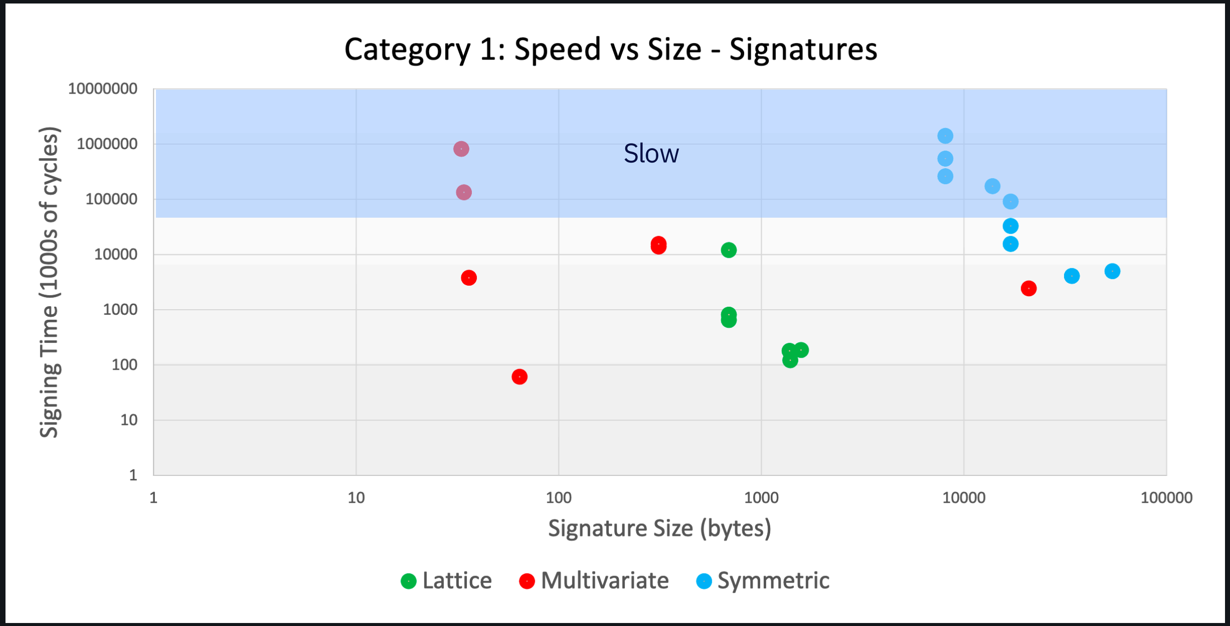


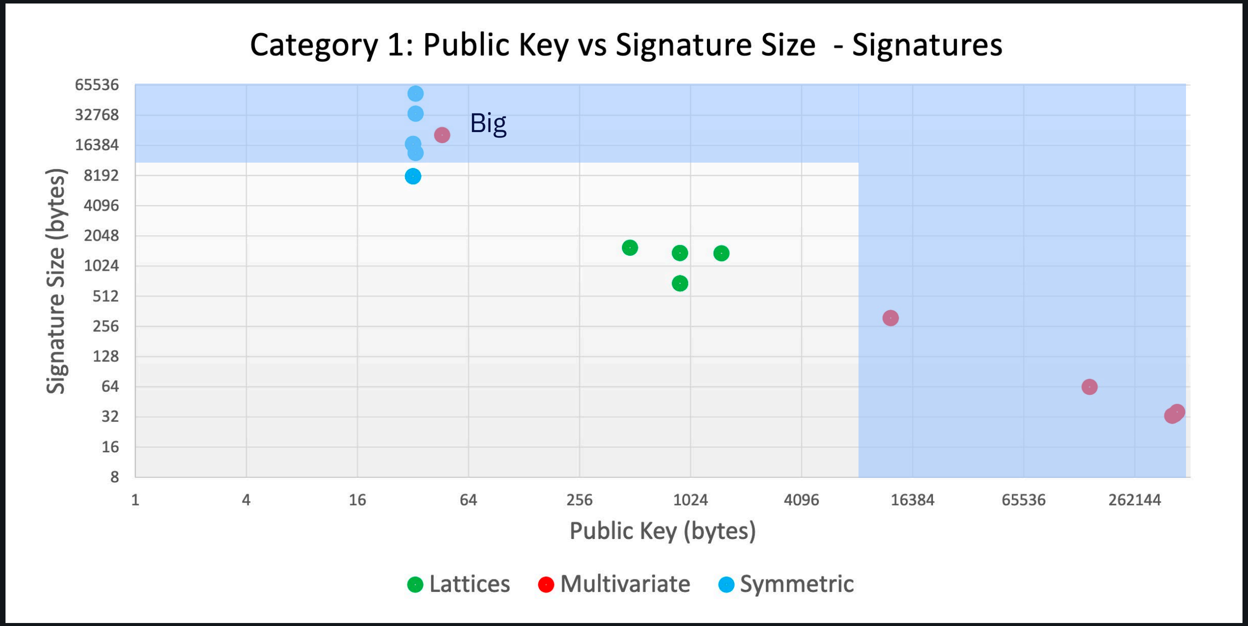

Charts based on: The 2nd Round of the NIST PQC Standardization Process Dustin Moody, Lilly Chen, NIST

# Quantum Safe Algorithms

## *Signatures: Impact of Public Key and Signature Size*

- Size is as important as performance for most networks
  - Fragmentation, buffer sizes, protocol limits
- Example: Public Key + Signature used in TLS 1.3 handshake




Lattice based schemes are well balanced (performance and bandwidth)

Charts based on: The 2nd Round of the NIST PQC Standardization Process Dustin Moody, Lilly Chen, NIST

# Quantum Safe Applications and Libraries

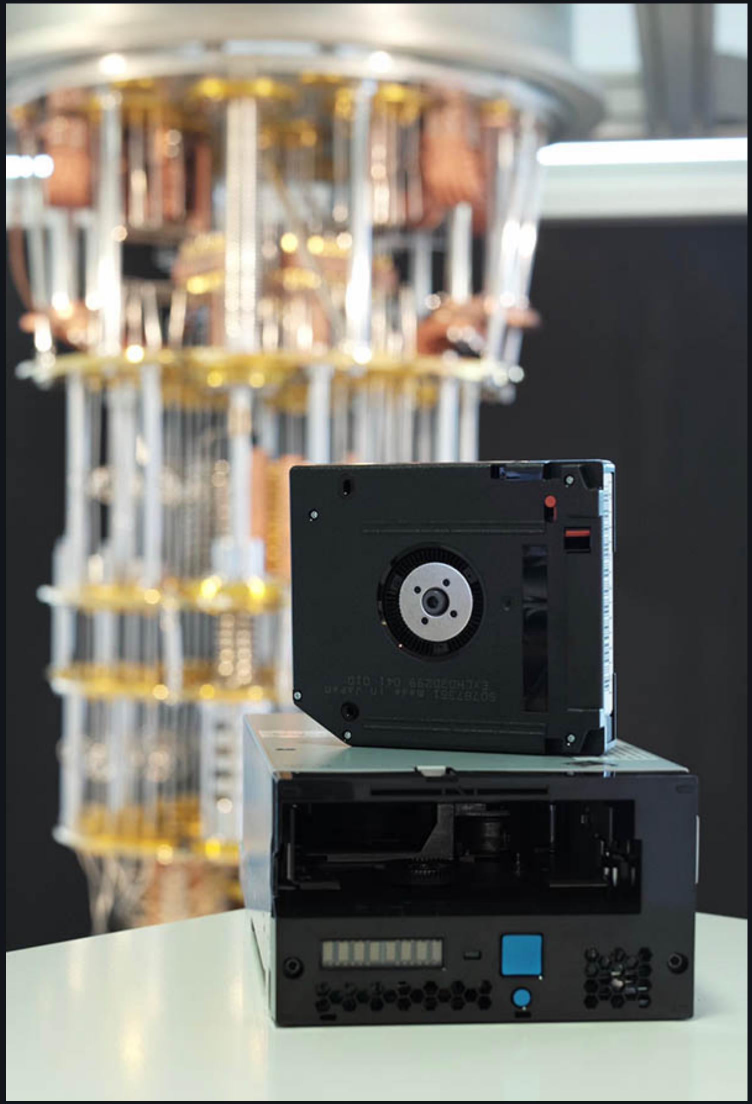
# *Application: Quantum Safe Tape*

Combination of asymmetric and symmetric cryptography

- Asymmetric cryptography for key establishment, e.g. with KMIP, and for firmware verification
- Symmetric cryptography for data encryption

Tape systems used to store data of periods up to decades

Quantum Safe Cryptography allows to establish long term security



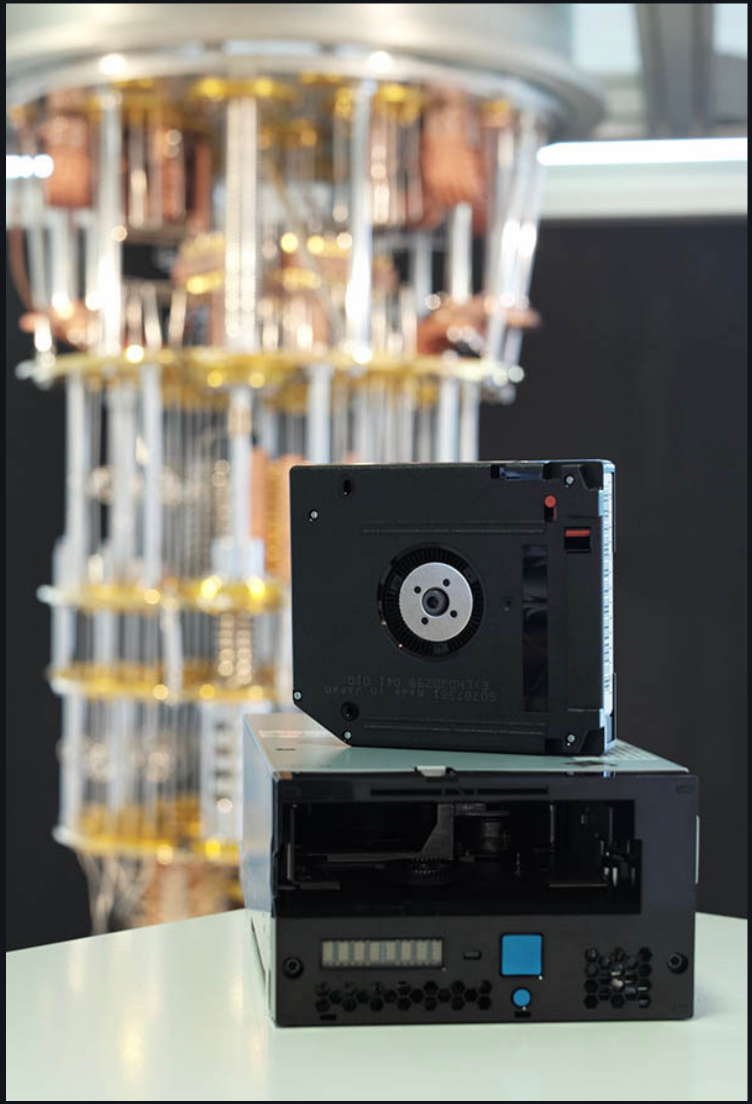
# Application: Quantum Safe Tape

In 2019, IBM announced a prototype of the world's first quantum safe tape drive. Implemented in firmware of TS1160 tape drive.

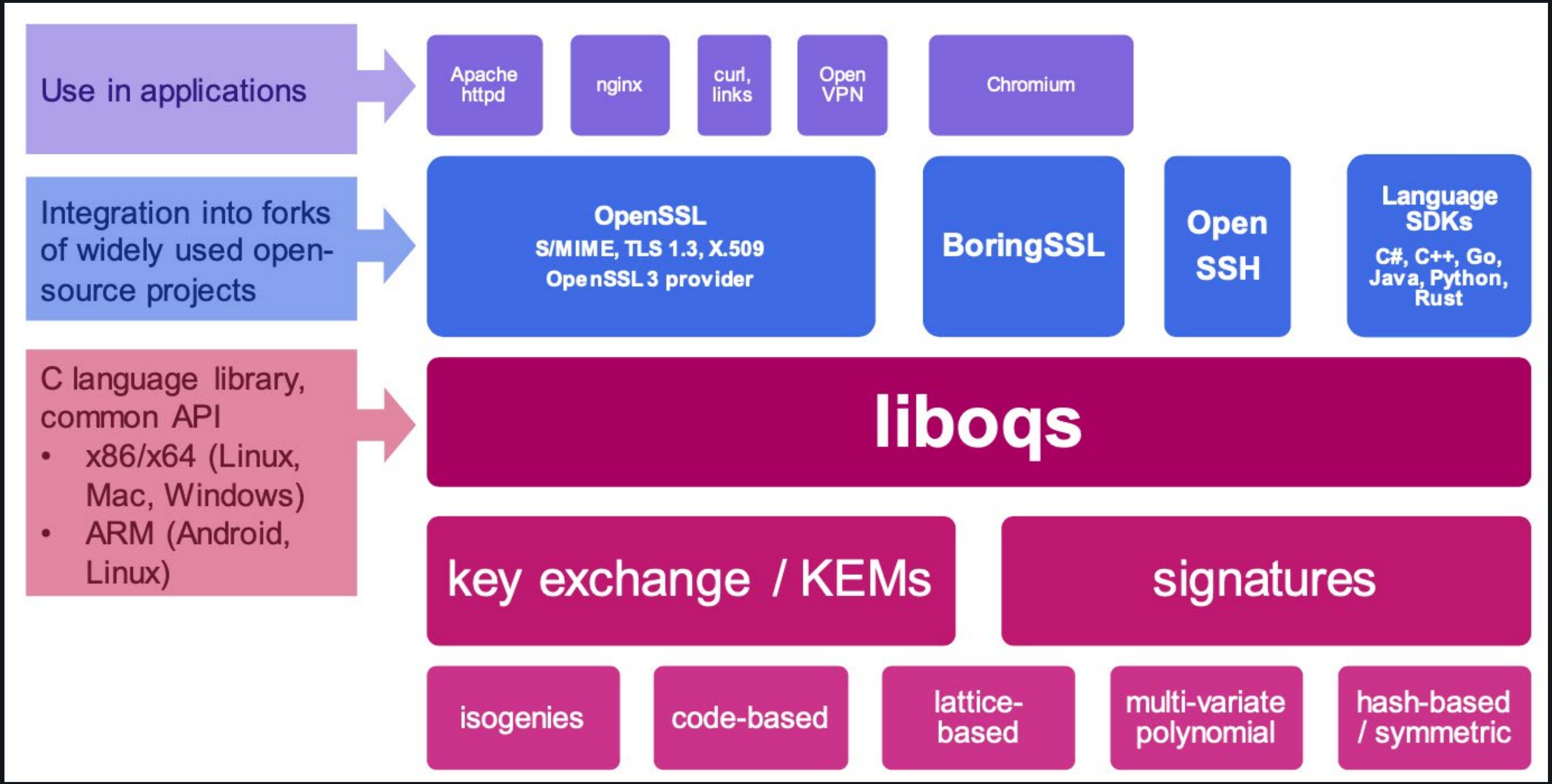
Uses:

- Kyber for secure key transport between tape drive and key manager
- Dilithium signatures for authentication and firmware verification

Data encryption on tape with AES256



# Open Quantum Safe




Academic contributors:

- University of Waterloo

Industry partners:

- IBM Reesearch
- Microsoft Research
- AWS
- evolutionQ

Individual contributors

https://openquantumsafe.org

# Open Quantum Safe

*liboqs*

| | |
|---|---|
| OQS_API const char * | OQS_KEM_alg_identifier(size_t i) |
| OQS_API int | OQS_KEM_alg_count(void) |
| OQS_API int | OQS_KEM_alg_is_enabled(const char *method_name) |
| OQS_API OQS_KEM * | OQS_KEM_new(const char *method_name) |
| OQS_API OQS_STATUS | OQS_KEM_keypair(const OQS_KEM *kem, uint8_t *public_key, uint8_t *secret_key) |
| OQS_API OQS_STATUS | OQS_KEM_encaps(const OQS_KEM *kem, uint8_t *ciphertext, uint8_t *shared_secret, const uint8_t *public_key) |
| OQS_API OQS_STATUS | OQS_KEM_decaps(const OQS_KEM *kem, uint8_t *shared_secret, const unsigned char *ciphertext, const uint8_t *secret_key) |
| OQS_API void | OQS_KEM_free(OQS_KEM *kem) |

| | |
|---|---|
| OQS_API const char * | OQS_SIG_alg_identifier(size_t i) |
| OQS_API int | OQS_SIG_alg_count(void) |
| OQS_API int | OQS_SIG_alg_is_enabled(const char *method_name) |
| OQS_API OQS_SIG * | OQS_SIG_new(const char *method_name) |
| OQS_API OQS_STATUS | OQS_SIG_keypair(const OQS_SIG *sig, uint8_t *public_key, uint8_t *secret_key) |
| OQS_API OQS_STATUS | OQS_SIG_sign(const OQS_SIG *sig, uint8_t *signature, size_t *signature_len, const uint8_t *message, size_t message_len, const uint8_t *secret_key) |
| OQS_API OQS_STATUS | OQS_SIG_verify(const OQS_SIG *sig, const uint8_t *message, size_t message_len, const uint8_t *signature, size_t signature_len, const uint8_t *public_key) |
| OQS_API void | OQS_SIG_free(OQS_SIG *sig) |

## Key encapsulation mechanisms (KEMs)

### Round 3 finalists

- **Classic McEliece**: Classic-McEliece-348864†, Classic-McEliece-348864f†, Classic-McEliece-460896†, Classic-McEliece-460896f†, Classic-McEliece-6688128†, Classic-McEliece-6688128f†, Classic-McEliece-6960119†, Classic-McEliece-6960119f†, Classic-McEliece-8192128†, Classic-McEliece-8192128f†
- **Kyber**: Kyber512, Kyber768, Kyber1024, Kyber512-90s, Kyber768-90s, Kyber1024-90s
- **NTRU**: NTRU-HPS-2048-509, NTRU-HPS-2048-677, NTRU-HPS-4096-821, NTRU-HRSS-701
- **SABER**: LightSaber-KEM, Saber-KEM, FireSaber-KEM

Note that algorithms marked with a dagger (†) have large stack usage and may cause failures when run on threads or in constrained environments.

### Round 3 alternate candidates

- **BIKE**: BIKE1-L1-CPA, BIKE1-L3-CPA, BIKE1-L1-FO, BIKE1-L3-FO
- **FrodoKEM**: FrodoKEM-640-AES, FrodoKEM-640-SHAKE, FrodoKEM-976-AES, FrodoKEM-976-SHAKE, FrodoKEM-1344-AES, FrodoKEM-1344-SHAKE
- **HQC**: HQC-128-1-CCA2, HQC-192-1-CCA2, HQC-192-2-CCA2, HQC-256-1-CCA2†, HQC-256-2-CCA2†, HQC-256-3-CCA2†
- **NTRU-Prime**: ntrulpr653, ntrulpr761, ntrulpr857, sntrup653, sntrup761, sntrup857
- **SIKE**: SIDH-p434, SIDH-p503, SIDH-p610, SIDH-p751, SIKE-p434, SIKE-p503, SIKE-p610, SIKE-p751, SIDH-p434-compressed, SIDH-p503-compressed, SIDH-p610-compressed, SIDH-p751-compressed, SIKE-p434-compressed, SIKE-p503-compressed, SIKE-p610-compressed, SIKE-p751-compressed

## Signature schemes

### Round 3 finalists

- **Dilithium**: Dilithium2, Dilithium2-AES, Dilithium3, Dilithium3-AES, Dilithium5, Dilithium5-AES
- **Falcon**: Falcon-512, Falcon-1024
- **Rainbow**: Rainbow-I-Classic, Rainbow-I-Circumzenithal, Rainbow-I-Compressed, Rainbow-III-Classic†, Rainbow-III-Circumzenithal†, Rainbow-III-Compressed†, Rainbow-V-Classic†, Rainbow-V-Circumzenithal†, Rainbow-V-Compressed†

Note that algorithms marked with a dagger (†) have large stack usage and may cause failures when run on threads or in constrained environments.

### Round 3 alternate candidates

- **Picnic**: Picnic-L1-FS, Picnic-L1-UR, Picnic-L1-full, Picnic-L3-FS, Picnic-L3-UR, Picnic-L3-full, Picnic-L5-FS, Picnic-L5-UR, Picnic-L5-full, Picnic3-L1, Picnic3-L3, Picnic3-L5
- **SPHINCS+**:

# *Open Quantum Safe*

- Implementations from PQClean and reference implementations
- MIT license
- Builds for macOS, Linux, x86_64, ARM32v7, ARM64v8, ppc64le
- Language bindings for C++, Go, Java, .NET, Python, Rust

Protocol stack integration:

- CMS/SMIME, SMIME, SSH, TLS 1.3

Support for QSC/classical hybrid key exchange and signatures:

- Combines classical and QSC algorithms

https://openquantumsafe.org/liboqs/

# Summary

- *We are in the final round of the NIST selection process for QSC standards: final standards expected in 2022-2024*
- *We will see more than one algorithm standardized*
- *Some QSC standards will based on lattice cryptography*
- *Ongoing prototyping and adoption of quantum-safe cryptography in areas such as tape storage already now*

# Thank You!