



Radboud University



The transition to post-quantum cryptography

Peter Schwabe

peter@cryptojedi.org

<https://cryptojedi.org>

October 15, 2018



5 building blocks for a “secure channel”

Symmetric crypto

- Block or stream cipher (e.g., AES, ChaCha20)
- Authenticator (e.g., HMAC, GMAC, Poly1305)
- Hash function (e.g., SHA-2, SHA-3)



5 building blocks for a “secure channel”

Symmetric crypto

- Block or stream cipher (e.g., AES, ChaCha20)
- Authenticator (e.g., HMAC, GMAC, Poly1305)
- Hash function (e.g., SHA-2, SHA-3)

Asymmetric crypto

- Key agreement / public-key encryption (e.g., RSA, Diffie-Hellman, ECDH)
- Signatures (e.g., RSA, DSA, ECDSA, EdDSA)



5 building blocks for a “secure channel”

Symmetric crypto

- Block or stream cipher (e.g., AES, ChaCha20)
- Authenticator (e.g., HMAC, GMAC, Poly1305)
- Hash function (e.g., SHA-2, SHA-3)

Asymmetric crypto

- Key agreement / public-key encryption (e.g., RSA, Diffie-Hellman, ECDH)
- Signatures (e.g., RSA, DSA, ECDSA, EdDSA)

The asymmetric monoculture

- All widely deployed asymmetric crypto relies on
 - the **hardness of factoring**, or
 - the **hardness of (elliptic-curve) discrete logarithms**



Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*

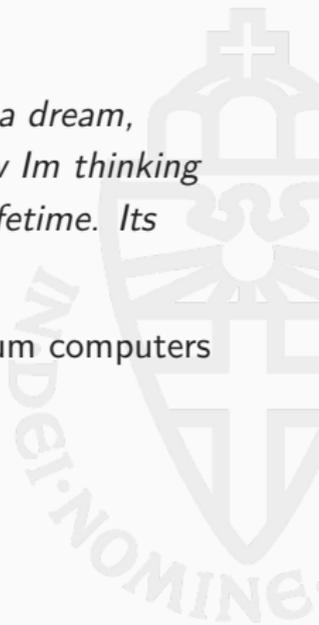
Peter W. Shor[†]

Abstract

A digital computer is generally believed to be an efficient universal computing device; that is, it is believed able to simulate any physical computing device with an increase in computation time by at most a polynomial factor. This may not be true when quantum mechanics is taken into consideration. This paper considers factoring integers and finding discrete logarithms, two problems which are generally thought to be hard on a classical computer and which have been used as the basis of several proposed cryptosystems. Efficient randomized algorithms are given for these two problems on a hypothetical quantum computer. These algorithms take a number of steps polynomial in the input size, e.g., the number of digits of the integer to be factored.

“In the past, people have said, maybe its 50 years away, its a dream, maybe itll happen sometime. I used to think it was 50. Now Im thinking like its 15 or a little more. Its within reach. Its within our lifetime. Its going to happen.”

—Mark Ketchen (IBM), Feb. 2012, about quantum computers



Definition

Post-quantum crypto is (asymmetric) crypto that resists attacks using classical *and quantum* computers.



Definition

Post-quantum crypto is (asymmetric) crypto that resists attacks using classical *and quantum* computers.

5 main directions

- Lattice-based crypto (PKE and Sigs)
- Code-based crypto (mainly PKE)
- Multivariate-based crypto (mainly Sigs)
- Hash-based signatures (only Sigs)
- Isogeny-based crypto (so far, mainly PKE)



The NIST PQC “not-a-competition”

- Inspired by two earlier NIST crypto competitions:
 - AES, running from 1997 to 2000
 - SHA3, running from 2007 to 2012



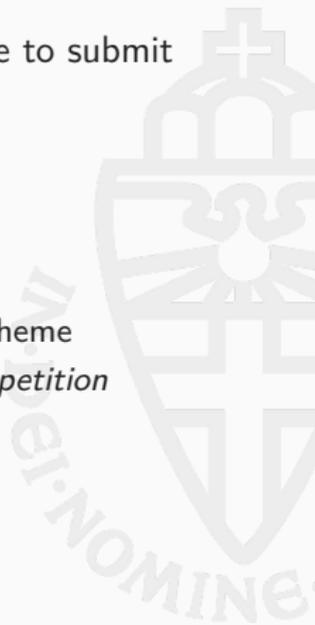
The NIST PQC “not-a-competition”

- Inspired by two earlier NIST crypto competitions:
 - AES, running from 1997 to 2000
 - SHA3, running from 2007 to 2012
- Approach: NIST specifies criteria, everybody is welcome to submit proposals
- Selection through an open process and multiple rounds
- Actual decisions are being made by NIST



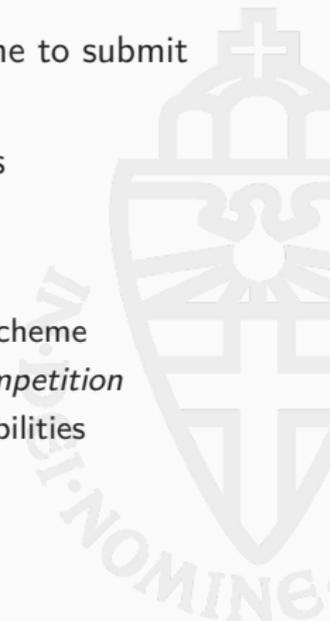
The NIST PQC “not-a-competition”

- Inspired by two earlier NIST crypto competitions:
 - AES, running from 1997 to 2000
 - SHA3, running from 2007 to 2012
- Approach: NIST specifies criteria, everybody is welcome to submit proposals
- Selection through an open process and multiple rounds
- Actual decisions are being made by NIST
- Widely successful in the past, but also some criticism:
 - Small tweaks are typically allowed, but standardized scheme represents state of the art *at the beginning of the competition*



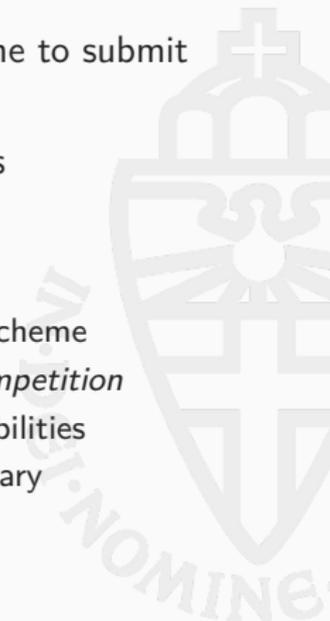
The NIST PQC “not-a-competition”

- Inspired by two earlier NIST crypto competitions:
 - AES, running from 1997 to 2000
 - SHA3, running from 2007 to 2012
- Approach: NIST specifies criteria, everybody is welcome to submit proposals
- Selection through an open process and multiple rounds
- Actual decisions are being made by NIST
- Widely successful in the past, but also some criticism:
 - Small tweaks are typically allowed, but standardized scheme represents state of the art *at the beginning of the competition*
 - AES standardization unaware of cache-timing vulnerabilities



The NIST PQC “not-a-competition”

- Inspired by two earlier NIST crypto competitions:
 - AES, running from 1997 to 2000
 - SHA3, running from 2007 to 2012
- Approach: NIST specifies criteria, everybody is welcome to submit proposals
- Selection through an open process and multiple rounds
- Actual decisions are being made by NIST
- Widely successful in the past, but also some criticism:
 - Small tweaks are typically allowed, but standardized scheme represents state of the art *at the beginning of the competition*
 - AES standardization unaware of cache-timing vulnerabilities
 - SHA-3 criterion of 512-bit preimage security unnecessary



The NIST PQC “not-a-competition”

- Inspired by two earlier NIST crypto competitions:
 - AES, running from 1997 to 2000
 - SHA3, running from 2007 to 2012
- Approach: NIST specifies criteria, everybody is welcome to submit proposals
- Selection through an open process and multiple rounds
- Actual decisions are being made by NIST
- Widely successful in the past, but also some criticism:
 - Small tweaks are typically allowed, but standardized scheme represents state of the art *at the beginning of the competition*
 - AES standardization unaware of cache-timing vulnerabilities
 - SHA-3 criterion of 512-bit preimage security unnecessary
- PQC project:
 - Announcement: Feb 2016
 - Call for proposals: Dec 2016 (based on community input)
 - Deadline for submissions: Nov 2017



Submission categories

- Cryptographic signatures (only stateless)
 - Security for at least 2^{64} signatures per key
- Public-key encryption / key encapsulation
 - Passive or active security (CPA or CCA2)



Submission categories

- Cryptographic signatures (only stateless)
 - Security for at least 2^{64} signatures per key
- Public-key encryption / key encapsulation
 - Passive or active security (CPA or CCA2)

Security categories

- Level 1: Equivalent to AES-128 (pre- and post-quantum)
- Level 2: Equivalent to SHA-256 (pre- and post-quantum)
- Level 3: Equivalent to AES-192 (pre- and post-quantum)
- Level 4: Equivalent to SHA-512 (pre- and post-quantum)
- Level 5: Equivalent to AES-256 (pre- and post-quantum)



The NIST competition, initial overview

Count of Problem Category	Column Labels		
Row Labels	Key Exchange	Signature	Grand Total
?	1		1
Braids	1	1	2
Chebychev	1		1
Codes	19	5	24
Finite Automata	1	1	2
Hash		4	4
Hypercomplex Numbers	1		1
Isogeny	1		1
Lattice	24	4	28
Mult. Var	6	7	13
Rand. walk	1		1
RSA	1	1	2
Grand Total	57	23	80

4 31 27

Overview tweeted by Jacob Alperin-Sheriff on Dec 4, 2017.

“Key exchange”

- What is meant is **key encapsulation mechanisms (KEMs)**
 - $(pk, sk) \leftarrow \text{KeyGen}()$
 - $(c, k) \leftarrow \text{Encaps}(pk)$
 - $k \leftarrow \text{Decaps}(c, sk)$

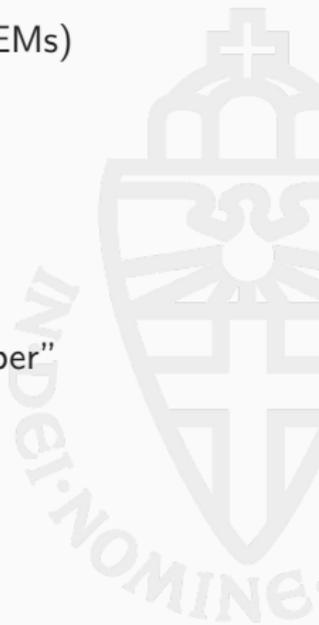


“Key exchange”

- What is meant is **key encapsulation mechanisms** (KEMs)
 - $(pk, sk) \leftarrow \text{KeyGen}()$
 - $(c, k) \leftarrow \text{Encaps}(pk)$
 - $k \leftarrow \text{Decaps}(c, sk)$

Status of the NIST competition

- In total 69 submissions accepted as “complete and proper”
- Several broken, 5 withdrawn
- Jan 2019: NIST announces 26 round-2 candidates
 - 17 KEMs and PKEs
 - 9 signature schemes



Signature schemes

- 3 lattice-based
- 2 symmetric-crypto based
- 4 MQ-based



Signature schemes

- 3 lattice-based
- 2 symmetric-crypto based
- 4 MQ-based

KEMs/PKE

- 9 lattice-based
- 7 code-based
- 1 isogeny-based



We care about 10% difference in performance



We care about 10% difference in performance

The baseline: ECC

- Today: build asymmetric crypto from elliptic-curve arithmetic
- Given P on a curve, $s \in \mathbb{Z}$, compute $Q = sP$
- ECDLP: hard to compute s , given P and Q
- Use for ECDH for key encapsulation and encryption
- Use for ECDSA or Schnorr signatures
- Use same curves, same parameters



We care about 10% difference in performance

The baseline: ECC

- Today: build asymmetric crypto from elliptic-curve arithmetic
- Given P on a curve, $s \in \mathbb{Z}$, compute $Q = sP$
- ECDLP: hard to compute s , given P and Q
- Use for ECDH for key encapsulation and encryption
- Use for ECDSA or Schnorr signatures
- Use same curves, same parameters
- Performance (64-bit Intel CPU):
 - All operations between 50 000 and 200 000 cycles
 - Keys and ciphertexts: 32 bytes
 - Signatures: 64 bytes



PQ performance, some examples

- Supersingular-isogeny-based key agreement:
 - Public key/ciphertext: < 500 bytes each



PQ performance, some examples

- Supersingular-isogeny-based key agreement:
 - Public key/ciphertext: < 500 bytes each
 - Keygen: ≈ 2.6 Mio cycles
 - Encaps: ≈ 3.8 Mio cycles
 - Decaps: ≈ 4.5 Mio cycles



PQ performance, some examples

- Supersingular-isogeny-based key agreement:
 - Public key/ciphertext: < 500 bytes each
 - Keygen: ≈ 2.6 Mio cycles
 - Encaps: ≈ 3.8 Mio cycles
 - Decaps: ≈ 4.5 Mio cycles
- McEliece code-based key agreement:
 - Encapsulation: $\approx 90\,000$ cycles
 - Decapsulation: $\approx 270\,000$ cycles
 - Key generation: ≈ 300 Mio cycles
 - Cipher text: 188 bytes



PQ performance, some examples

- Supersingular-isogeny-based key agreement:
 - Public key/ciphertext: < 500 bytes each
 - Keygen: ≈ 2.6 Mio cycles
 - Encaps: ≈ 3.8 Mio cycles
 - Decaps: ≈ 4.5 Mio cycles
- McEliece code-based key agreement:
 - Encapsulation: $\approx 90\,000$ cycles
 - Decapsulation: $\approx 270\,000$ cycles
 - Key generation: ≈ 300 Mio cycles
 - Cipher text: 188 bytes
 - Public key: ≈ 0.5 MB



PQ performance, some examples

- Supersingular-isogeny-based key agreement:
 - Public key/ciphertext: < 500 bytes each
 - Keygen: ≈ 2.6 Mio cycles
 - Encaps: ≈ 3.8 Mio cycles
 - Decaps: ≈ 4.5 Mio cycles
- McEliece code-based key agreement:
 - Encapsulation: $\approx 90\,000$ cycles
 - Decapsulation: $\approx 270\,000$ cycles
 - Key generation: ≈ 300 Mio cycles
 - Cipher text: 188 bytes
 - Public key: ≈ 0.5 MB
- \mathcal{MQ} -based signatures (e.g., GeMSS):
 - Signature: ≈ 50 bytes
 - Verification: $\approx 580\,000$ cycles



PQ performance, some examples

- Supersingular-isogeny-based key agreement:
 - Public key/ciphertext: < 500 bytes each
 - Keygen: ≈ 2.6 Mio cycles
 - Encaps: ≈ 3.8 Mio cycles
 - Decaps: ≈ 4.5 Mio cycles
- McEliece code-based key agreement:
 - Encapsulation: $\approx 90\,000$ cycles
 - Decapsulation: $\approx 270\,000$ cycles
 - Key generation: ≈ 300 Mio cycles
 - Cipher text: 188 bytes
 - Public key: ≈ 0.5 MB
- \mathcal{MQ} -based signatures (e.g., GeMSS):
 - Signature: ≈ 50 bytes
 - Verification: $\approx 580\,000$ cycles
 - Signing: ≈ 2.7 billion cycles
 - Public key: ≈ 1.2 MB



Cryptographic hardness and proofs

- Need better understanding of attacks and their complexity
- Security reductions (“proofs”) help



Cryptographic hardness and proofs

- Need better understanding of attacks and their complexity
- Security reductions (“proofs”) help, but
 - they are almost never tight



Cryptographic hardness and proofs

- Need better understanding of attacks and their complexity
- Security reductions (“proofs”) help, but
 - they are almost never tight
 - they are too often wrong



Cryptographic hardness and proofs

- Need better understanding of attacks and their complexity
- Security reductions (“proofs”) help, but
 - they are almost never tight
 - they are too often wrong
- **Try to break schemes and check proofs!**



Cryptographic hardness and proofs

- Need better understanding of attacks and their complexity
- Security reductions (“proofs”) help, but
 - they are almost never tight
 - they are too often wrong
- **Try to break schemes and check proofs!**

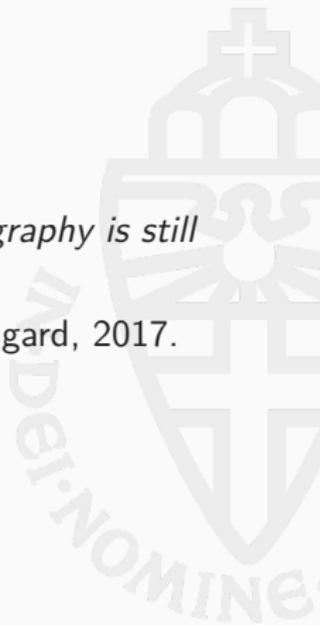
Secure implementations

- Implementations of secure schemes are not necessarily secure:
 - Subtle mistakes/bugs in implementations
 - Side-channel attacks
 - Fault attacks



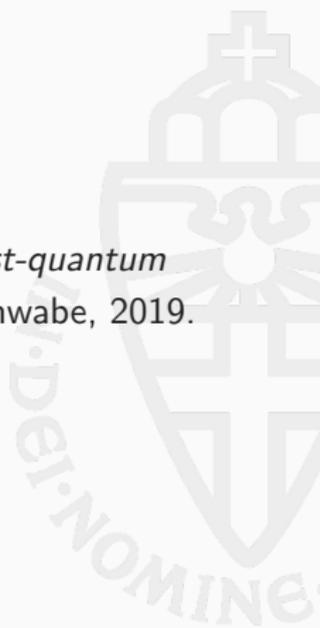
“the implementation security aspect of lattice-based cryptography is still a vastly unexplored and open topic”

Primas, Pessl, Mangard, 2017.



“... this isn't very different for any of the other areas of post-quantum crypto”

Schwabe, 2019.



Challenges part 3: The case of DH

- Diffie-Hellman is extremely versatile:
- Can use it, for example, for *non-interactive key exchange (NIKE)*
 - Bob knows Alice' long-term public key A
 - Alice knows Bob's long-term public key B
 - They can each compute $k = h(A, B, aB) = h(A, B, bA)$
 - Used in various protocols, e.g., WireGuard



Challenges part 3: The case of DH

- Diffie-Hellman is extremely versatile:
- Can use it, for example, for *non-interactive key exchange (NIKE)*
 - Bob knows Alice' long-term public key A
 - Alice knows Bob's long-term public key B
 - They can each compute $k = h(A, B, aB) = h(A, B, bA)$
 - Used in various protocols, e.g., WireGuard
- Only one practical post-quantum proposal for NIKE: **CSIDH** (Wouter Castryck, Tanja Lange, Chloe Martindale, Joost Renes, Lorenz Panny. Asiacrypt 2018)
 - Very new and not well studied
 - Heavy debates about post-quantum security of proposed parameters
 - Small public keys, but rather slow (≈ 300 Mio. cycles)



Challenges part 3: The case of DH

- Diffie-Hellman is extremely versatile:
- Can use it, for example, for *non-interactive key exchange (NIKE)*
 - Bob knows Alice' long-term public key A
 - Alice knows Bob's long-term public key B
 - They can each compute $k = h(A, B, aB) = h(A, B, bA)$
 - Used in various protocols, e.g., WireGuard
- Only one practical post-quantum proposal for NIKE: **CSIDH** (Wouter Castryck, Tanja Lange, Chloe Martindale, Joost Renes, Lorenz Panny. Asiacrypt 2018)
 - Very new and not well studied
 - Heavy debates about post-quantum security of proposed parameters
 - Small public keys, but rather slow (≈ 300 Mio. cycles)
- **Think protocols in KEMs, not in DHs/NIKEs!**



Challenges part 4: stateful signatures

- Hash-based signatures are already in RFCs:
 - XMSS: RFC8391
 - LMS: RFC8554
- Also highly parametrizable, for example:
 - Signing: ≈ 12.5 Mio cycles
 - Verification: ≈ 1 Mio cycles
 - Signature: ≈ 2.8 KB
 - Public key: 64 bytes
 - Up to 2^{20} signatures



Challenges part 4: stateful signatures

- Hash-based signatures are already in RFCs:
 - XMSS: RFC8391
 - LMS: RFC8554
- Also highly parametrizable, for example:
 - Signing: ≈ 12.5 Mio cycles
 - Verification: ≈ 1 Mio cycles
 - Signature: ≈ 2.8 KB
 - Public key: 64 bytes
 - Up to 2^{20} signatures
- Issue with XMSS/LMS: it's stateful
- Security demands that secret key is updated for every signature
- Major problem, for examples, with backups



Challenges part 4: stateful signatures

- Hash-based signatures are already in RFCs:
 - XMSS: RFC8391
 - LMS: RFC8554
- Also highly parametrizable, for example:
 - Signing: ≈ 12.5 Mio cycles
 - Verification: ≈ 1 Mio cycles
 - Signature: ≈ 2.8 KB
 - Public key: 64 bytes
 - Up to 2^{20} signatures
- Issue with XMSS/LMS: it's stateful
- Security demands that secret key is updated for every signature
- Major problem, for examples, with backups
- Stateful sigs are required for forward security
- XMSS gives forward security for free



Challenges part 4: stateful signatures

- Hash-based signatures are already in RFCs:
 - XMSS: RFC8391
 - LMS: RFC8554
- Also highly parametrizable, for example:
 - Signing: ≈ 12.5 Mio cycles
 - Verification: ≈ 1 Mio cycles
 - Signature: ≈ 2.8 KB
 - Public key: 64 bytes
 - Up to 2^{20} signatures
- Issue with XMSS/LMS: it's stateful
- Security demands that secret key is updated for every signature
- Major problem, for examples, with backups
- Stateful sigs are required for forward security
- XMSS gives forward security for free
- **Start thinking systems with *stateful* signatures**



NIST resources

- NIST PQC website:
<https://csrc.nist.gov/Projects/Post-Quantum-Cryptography>
- NIST mailing list:
<https://www.safecrypto.eu/pqclounge/>



NIST resources

- NIST PQC website:
<https://csrc.nist.gov/Projects/Post-Quantum-Cryptography>
- NIST mailing list:
<https://www.safecrypto.eu/pqclounge/>

Third-party resources about NIST PQC

- Open Quantum Safe <https://openquantumsafe.org/>
- PQC Lounge: <https://www.safecrypto.eu/pqclounge/>
- PQC Wiki: <https://pqc-wiki.fau.edu>



NIST resources

- NIST PQC website:
<https://csrc.nist.gov/Projects/Post-Quantum-Cryptography>
- NIST mailing list:
<https://www.safecrypto.eu/pqclounge/>

Third-party resources about NIST PQC

- Open Quantum Safe <https://openquantumsafe.org/>
- PQC Lounge: <https://www.safecrypto.eu/pqclounge/>
- PQC Wiki: <https://pqc-wiki.fau.edu>

Shameless advertising

- pqm4: <https://github.com/mupq/pqm4>
- PQClean: <https://github.com/PQClean/PQClean>

