The Road to Post-Quantum Cryptography

Martin R. Albrecht

https://malb.io

Introduction

About Me

- Reader in the Information Security Group, Royal Holloway, University of London
- Working on post-quantum cryptography with a focus on lattice-based cryptography[1]
- Also working on analysing cryptographic protocols such as SSH[2] and TLS[3]

[1] Martin R. Albrecht, Léo Ducas, Gottfried Herold, Elena Kirshanova, Eamonn W. Postlethwaite, and Marc Stevens. The General Sieve Kernel and New Records in Lattice Reduction. to appear at *Eurocrypt 2019*. 2019.

[2] Martin R. Albrecht, Kenneth G. Paterson, and Gaven J. Watson. Plaintext Recovery Attacks against SSH. In: *2009 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2009, pp. 16–26. DOI: 10.1109/SP.2009.5.

[3] Martin R. Albrecht, Jake Massimo, Kenneth G. Paterson, and Juraj Somorovsky. Prime and Prejudice: Primality Testing Under Adversarial Conditions. In: *ACM CCS 2018*. Ed. by David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang. ACM Press, Oct. 2018, pp. 281–298. DOI: 10.1145/3243734.3243787.

Essential Cryptographic Primitives

Symmetric Primitives

- Block and stream ciphers (AES, ChaCha20, ...)
- Authentication codes (HMAC, Poly1305, ...)
- Hash functions (SHA-2, SHA-3, ...)

Asymmetric Primitives

- Key agreement and public-key encryption (RSA, Diffie-Hellman, ECDH, ...)
- Digital signatures (RSA, DSA, ECDSA, ...)

Applications

TLS, SSH, banking, smart cards, hard disk encryption . . .

Essential Cryptographic Primitives: Theoretical Perspective

Minicrypt

- Block and stream ciphers
- Hash functions
- Authentication codes
- **Digital signatures**

Cryptomania

- Key agreement and public-key encryption
- ...

Essential Cryptographic Primitives: Theoretical Perspective

Minicrypt

- Block and stream ciphers
- Hash functions
- Authentication codes
- **Digital signatures**

Cryptomania

- Key agreement and public-key encryption
- ...

Very slow one-time digital signatures from hash functions

KeyGen $H(\cdot)$ is a hash function with 256 bits of output. Sample random numbers $(a_{0,0}, a_{0,1}), (a_{1,0}, a_{1,1}), \ldots, (a_{255,0}, a_{255,1})$. Publish $H(a_{i,j})$ for all $a_{i,j}$.

Sign Let b_i be the bits of $H(m)$. For each bit b_i, publish a_{i,b_i}.

Verify check that a_{i,b_i} indeed hashes to $H(a_{i,j})$ in the public key.

Symmetric v Asymmetric Primitives

Symmetric Primitives

Indeed, it seems that "you can't throw a rock without hitting a one-way function" in the sense that, once you cobble together a large number of simple computational operations then, unless the operations satisfy some special property such as linearity, you will typically get a function that is hard to invert. [Bar17]

Asymmetric Primitives

All widely deployed asymmetric cryptography relies on the hardness of **factoring:**

$$\text{Given } N = p \cdot q \text{ find } p, \text{ or}$$

(elliptic-curve) discrete logarithms:

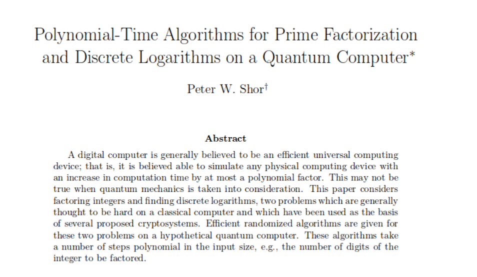
$$\text{Given } g^a \bmod q \text{ and } g \text{ find } a.$$

Symmetric v Asymmetric Primitives: Quantum Computing Perspective

Symmetric Primitives

- Best known quantum algorithms for attacking symmetric cryptography are based on **Grover's algorithm**
- At best quadratic speed-up: 256 bits → 128 "bits"
- This estimate is too optimistic, [Amy+16] suggests 256 bits → 166 bits
- Grover's algorithm does not parallelise: have to wait for 2^{128} steps, cannot buy 2^{32} computers and wait 2^{96} steps

Asymmetric Primitives



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.

Post-Quantum Cryptography

Post-Quantum Cryptography

Definition

Asymmetric cryptographic algorithms run on classical computers that resist attacks using classical and quantum computers.

Post-Quantum Cryptography

Definition

Asymmetric cryptographic algorithms run on classical computers that resist attacks using classical and quantum computers.

Note

Post-quantum cryptography is entirely distinct from quantum cryptography such as a quantum key exchange (QKD). The latter uses quantum effects to achieve security.

Post-Quantum Standardisation

NIST Post Quantum Competition Process[4]

ETSI Cyber Working Group for Quantum Safe Cryptography

ISO WG2 Standing Document 8 (SD8): Survey

IETF Standardisation of **stateful** hash-based signatures, nothing further

CSA Quantum-safe Security Working Group: position papers

[4] “NIST anticipates that the evaluation process for these post-quantum cryptosystems may be significantly more complex than the evaluation of the SHA-3 and AES candidates. . . . NIST believes that its post-quantum standards development process should not be treated as a competition; in some cases, it may not be possible to make a well-supported judgment that one candidate is ‘better’ than another.“

Post-Quantum Standardisation

NIST Post Quantum Competition Process[4]

ETSI Cyber Working Group for Quantum Safe Cryptography

ISO WG2 Standing Document 8 (SD8): Survey

IETF Standardisation of **stateful** hash-based signatures, nothing further

CSA Quantum-safe Security Working Group: position papers

Status

Essentially, everyone is waiting for NIST.

[4] “NIST anticipates that the evaluation process for these post-quantum cryptosystems may be significantly more complex than the evaluation of the SHA-3 and AES candidates. . . . NIST believes that its post-quantum standards development process should not be treated as a competition; in some cases, it may not be possible to make a well-supported judgment that one candidate is ‘better’ than another.“

NIST PQC Competition Process

Timeline

- Submission deadline was November 2017.
- Round 2 selection announced January 2019.
- Final standard expected 2022-2024.

“Key Exchange”/Key Encapsulation

- `(pk,sk)` ← `KeyGen()`
- `(c,k)` ← `Encaps(pk)`
- `k` ← `Decaps(c,sk)`

Digital Signature

- `(vk,sk)` ← `KeyGen()`
- `s` ← `Sig(m, sk)`
- `{0,1}` ← `Verify(s,m,vk)`

NIST PQC Competition Process

Timeline

- Submission deadline was November 2017.
- Round 2 selection announced January 2019.
- Final standard expected 2022-2024.

“Key Exchange”/Key Encapsulation

- `(pk,sk) ← KeyGen()`
- `(c,k) ← Encaps(pk)`
- `k ← Decaps(c,sk)`

Digital Signature

- `(vk,sk) ← KeyGen()`
- `s ← Sig(m, sk)`
- `{0,1} ← Verify(s,m,vk)`

NIST also asked for public-key encryption but this is less important as it can be built generically from a KEM and a block cipher.

Security Notions

KEM **IND-CCA**: Given some challenge ciphertext c and some key k, the adversary gets an oracle to decapsulate (“decrypt”) any other ciphertext but still cannot decide if c encapsulates (“encrypts”) the key k.

SIG **EUF-CMA**: Given access to some oracle that signs arbitrary messages, the adversary still cannot produce a valid signature not previously submitted to the signing oracle.

Security Notions

KEM **IND-CCA**: Given some challenge ciphertext *c* and some key *k*, the adversary gets an oracle to decapsulate (“decrypt”) any other ciphertext but still cannot decide if *c* encapsulates (“encrypts”) the key *k*.

SIG **EUF-CMA**: Given access to some oracle that signs arbitrary messages, the adversary still cannot produce a valid signature not previously submitted to the signing oracle.

Computational Security

“cannot“ → “computationally infeasible even given access to a quantum computer.”

Security Notions

KEM **IND-CCA**: Given some challenge ciphertext c and some key k, the adversary gets an oracle to decapsulate (“decrypt”) any other ciphertext but still cannot decide if c encapsulates (“encrypts”) the key k.

SIG **EUF-CMA**: Given access to some oracle that signs arbitrary messages, the adversary still cannot produce a valid signature not previously submitted to the signing oracle.

Computational Security

“cannot“ → “computationally infeasible even given access to a quantum computer.”

Conditional Security

“cannot” → “…assuming some mathematical problem is hard on a quantum computer”

Post-Quantum Candidate Families

- Code-based (key encapsulation)
- Multivariate-based (signatures)
- OWF-based (signatures)
- Isogeny-based (key encapsulation)
- Lattice-based (key encapsulation, signatures)

NIST PQC 2nd Round

17 KEMs BIKE, Classic McEliece, CRYSTALS-KYBER, FrodoKEM, @@beamer:<1>*HQC, LAC, LEDAcrypt, NewHope, NTRU, NTRU Prime, NTS-KEM, ROLLO, Round5, RQC, SABER, SIKE, Three Bears.

9 SIGs CRYSTALS-DILITHIUM, FALCON, GeMSS, LUOV, MQDSS, Picnic, qTESLA, Rainbow, SPHINCS+.

Post-Quantum Candidate Families

- Code-based (key encapsulation)
- Multivariate-based (signatures)
- OWF-based (signatures)
- Isogeny-based (key encapsulation)
- Lattice-based (key encapsulation, signatures)

NIST PQC 2nd Round

17 KEMs BIKE, Classic McEliece, CRYSTALS-KYBER, FrodoKEM, @@beamer:<1>*HQC, LAC, LEDAcrypt, NewHope, NTRU, NTRU Prime, NTS-KEM, ROLLO, Round5, RQC, SABER, SIKE, Three Bears.

9 SIGs CRYSTALS-DILITHIUM, FALCON, GeMSS, LUOV, MQDSS, Picnic, qTESLA, Rainbow, SPHINCS+.

Post-Quantum Candidate Families

- Code-based (key encapsulation)
- Multivariate-based (signatures)
- OWF-based (signatures)
- Isogeny-based (key encapsulation)
- Lattice-based (key encapsulation, signatures)

NIST PQC 2nd Round

17 KEMs BIKE, Classic McEliece, CRYSTALS-KYBER, FrodoKEM, @@beamer:<1>*HQC, LAC, LEDAcrypt, NewHope, NTRU, NTRU Prime, NTS-KEM, ROLLO, Round5, RQC, SABER, SIKE, Three Bears.

9 SIGs CRYSTALS-DILITHIUM, FALCON, GeMSS, LUOV, MQDSS, Picnic, qTESLA, Rainbow, SPHINCS+.

Post-Quantum Candidate Families

- Code-based (key encapsulation)
- Multivariate-based (signatures)
- OWF-based (signatures)
- Isogeny-based (key encapsulation)
- Lattice-based (key encapsulation, signatures)

NIST PQC 2nd Round

17 KEMs BIKE, Classic McEliece, CRYSTALS-KYBER, FrodoKEM, @@beamer:<1>*HQC, LAC, LEDAcrypt, NewHope, NTRU, NTRU Prime, NTS-KEM, ROLLO, Round5, RQC, SABER, SIKE, Three Bears.

9 SIGs CRYSTALS-DILITHIUM, FALCON, GeMSS, LUOV, MQDSS, Picnic, qTESLA, Rainbow, SPHINCS+.

Post-Quantum Candidate Families

- Code-based (key encapsulation)
- Multivariate-based (signatures)
- OWF-based (signatures)
- Isogeny-based (key encapsulation)
- Lattice-based (key encapsulation, signatures)

NIST PQC 2nd Round

17 KEMs BIKE, Classic McEliece, CRYSTALS-KYBER, FrodoKEM, @@beamer:<1>*HQC, LAC, LEDAcrypt, NewHope, NTRU, NTRU Prime, NTS-KEM, ROLLO, Round5, RQC, SABER, SIKE, Three Bears.

9 SIGs CRYSTALS-DILITHIUM, FALCON, GeMSS, LUOV, MQDSS, Picnic, qTESLA, Rainbow, SPHINCS+.

Post-Quantum Candidate Families

- Code-based (key encapsulation)
- Multivariate-based (signatures)
- OWF-based (signatures)
- Isogeny-based (key encapsulation)
- Lattice-based (key encapsulation, signatures)

NIST PQC 2nd Round

17 KEMs BIKE, Classic McEliece, CRYSTALS-KYBER, FrodoKEM, @@beamer:<1>*HQC, LAC, LEDAcrypt, NewHope, NTRU, NTRU Prime, NTS-KEM, ROLLO, Round5, RQC, SABER, SIKE, Three Bears.

9 SIGs CRYSTALS-DILITHIUM, FALCON, GeMSS, LUOV, MQDSS, Picnic, qTESLA, Rainbow, SPHINCS+.

Post-Quantum Candidate Families

- Code-based (key encapsulation)
- Multivariate-based (signatures)
- OWF-based (signatures)
- Isogeny-based (key encapsulation)
- Lattice-based (key encapsulation, signatures)

NIST PQC 2nd Round

17 KEMs BIKE, Classic McEliece, CRYSTALS-KYBER, FrodoKEM, @@beamer:<1>*HQC, LAC, LEDAcrypt, NewHope, NTRU, NTRU Prime, NTS-KEM, ROLLO, Round5, RQC, SABER, SIKE, Three Bears.

9 SIGs CRYSTALS-DILITHIUM, FALCON, GeMSS, LUOV, MQDSS, Picnic, qTESLA, Rainbow, SPHINCS+.

KEM: Code-based

Idea: Take error-correcting code for up to t errors. Keep decoding algorithm secret, hide structure of the code.

- Encapsulated key: error vector with t error indices
- Most prominent example: McEliece (1978), uses binary Goppa codes
- Alternatives: QCMDPC codes (e.g. BIKE)
  - Less studied, less conservative, problems with CCA security

NTS-KEM(13, 136) NIST submission:

Key generation	$\approx$ 240,000,000 cycles
Encapsulation	$\approx$ 280,000 cycles
Decapsulation	$\approx$ 2,000,000 cycles
Ciphertext	253 bytes
Public key	1,419,704 bytes

`https://bench.cr.yp.to/results-kem.html`

KEM: Lattice-based

Idea: Noisy linear algebra mod q is hard and equivalent to finding short vectors in lattices. Encrypt as solution to noisy linear equations.

- Learning with Errors: given $\mathsf{A}, \mathsf{b} \equiv \mathsf{A} \cdot \mathsf{s} + \mathsf{e} \bmod q$ where e is a vector with small entries, find s
- Most submissions use structured A
  - Faster but less conservative
- Frodo uses plain, unstructured LWE

Kyber-768 NIST submission:

Key generation	$\approx$ 80,000 cycles
Encapsulation	$\approx$ 100,000 cycles
Decapsulation	$\approx$ 100,000 cycles
Ciphertext	1,152 bytes
Public key	1,088 bytes

`https://bench.cr.yp.to/results-kem.html`

KEM: SIKE

Idea: Hard problem is finding a rational map that preserves structure **between** elliptic curves.

- “Supersingular-Isogeny Diffie-Hellman” (SIDH) proposed in 2011
- Security related to claw/collision finding, but no reduction from it
- Rather young construction, more study needed

SIKE NIST submission:

Key generation	$\approx$ 13,000,000 cycles
Encapsulation	$\approx$ 20,000,000 cycles
Decapsulation	$\approx$ 20,000,000 cycles
Ciphertext	402 bytes
Public key	378 bytes

`https://bench.cr.yp.to/results-kem.html`

SIG: OWF-based

Idea: Start from one-time digital signature based on hash functions. Build Merkle trees on top to produce many-time signature schemes.

- Many tradeoffs possible
- Secure if there exist collision/pre-image resistant hash functions

SPHINCS256 NIST submission:

Key generation	≈ 3,500,000 cycles
Signing	≈ 65,000,000 cycles
Verifying	≈ 1,600,050 cycles
Signature	41,000 bytes
Verification key	1,056 bytes

`https://bench.cr.yp.to/results-sign.html`

SIG: Lattice-based (Hash-and-Sign)

Idea: Verification key is matrix A. Hash message m to vector $H(m)$. Signature is a **short** vector s such that $H(m) = \mathsf{A} \cdot \mathsf{s}$.

- Can be instantiated from structured and unstructured A
- Typically uses structured lattices
- Falcon uses NTRU problem: Given $\mathsf{A} = f/g$ where both f, g are small. Find f

Falcon-768 NIST submission

Key generation	$\approx$ 43,000,000 cycles
Signing	$\approx$ 930,000 cycles
Verifying	$\approx$ 160,000 cycles
Signature	994 bytes
Verification key	1441 bytes

Falcon submission document

SIG: MQ-BASED

Idea: Hard problem is to find solution to system of **quadratic** equations in many variables over a finite field.

- All but one submissions use structured systems and assume attacker cannot exploit structure
- No reduction from standard MQ problem
- MQDSS reduces to unstructured MQ

Rainbow NIST submission

Key generation	$\approx$ 7,000,000 cycles
Signing	$\approx$ 10,000 cycles
Verifying	$\approx$ 6,000 cycles
Signature	42 bytes
Verification key	30,240 bytes

`https://bench.cr.yp.to/results-sign.html`

The Road Ahead

Parameters Matter

One cannot hope to simply "plug in" a key of 10^6 or 10^9 bits into a protocol designed to work for keys of 10^3 bits and expect it to work as is, and so such results could bring about significant changes to the way we do security over the Internet. For example, it could lead to a centralization of power, where key exchange will be so expensive that users would share public-keys with only a few large corporations and governments, and smaller companies would have to route their communication through these larger corporations.[5]

[5] Boaz Barak. The Complexity of Public-Key Cryptography. Cryptology ePrint Archive, Report 2017/365. `http://eprint.iacr.org/2017/365`. 2017.

PQ Candidates are not DH

Diffie-Hellman is extremely versatile:

- Non-Interactive Key Exchange (NIKE)
  - Bob knows Alice's long-term pk g^a
  - Alice knows Bob's long-term pk g^b
  - Agree on a shared key before exchanging any messages
  - Expensive to instantiate post-quantum (SIDH-based)
- Oblivious PRF:
  - Alice sends h^r to Bob
  - Bob computes $(h^r)^b$
  - Alice computes $(h^{r \cdot b})^{(1/r)}$
  - Not clear how to instantiate post-quantum

Lattices are extremely versatile:

- Fully-Homomorphic Encryption (FHE)
  - Computing on encrypted data
  - Only from lattices
- Identity-Based Encryption (IBE)
  - Names are the public keys
- Attribute-Based Encryption (ABE)
  - Encrypt to all doctors in an organisation etc.
- ...

Alternatives: QKD?

> *QKD: has fundamental practical limitations; does not address large parts of the security problem; is poorly understood in terms of potential attacks.*
> *By contrast, post-quantum public key cryptography appears to offer much more effective mitigations for real-world communications systems from the threat of future quantum computers.*[6]

- attacks on implementations/instantiations
- limited range, dedicated hardware
- limited speed → keys then used in AES
- authentication required: MAC or digital signature

[6] National Cyber Security Centre. Quantum Key Distribution. `https://www.ncsc.gov.uk/information/quantum-key-distribution`. 2016.

The Road Ahead

- We need to understand the underlying hard problems better
- Resistance to side-channel attacks
- Efficient, safe implementations
- How fast is fast enough? How small is small enough?
- How do existing protocols interact with post-quantum primitives? Should we change protocols?

Don't Jump the Gun!

- Temptation to pick one of the NIST candidates as drop-in replacement for deployment in existing protocols now
- This is a terrible idea!
  - mediocre performance
  - non-optimal security properties
- Bad cryptography is very hard to get rid of (think MD5)
- Will also need to think carefully about changes to protocols
- Let's get this one right!

Don't Jump the Gun!

- Temptation to pick one of the NIST candidates as drop-in replacement for deployment in existing protocols now
- This is a terrible idea!
  - mediocre performance
  - non-optimal security properties
- Bad cryptography is very hard to get rid of (think MD5)
- Will also need to think carefully about changes to protocols
- Let's get this one right!

Proof of Concept Code

...even worse idea: pick **source code** of one of the NIST candidates to deploy

Fin

Thank You

References I

[Alb+18] Martin R. Albrecht, Jake Massimo, Kenneth G. Paterson, and Juraj Somorovsky. Prime and Prejudice: Primality Testing Under Adversarial Conditions. In: *ACM CCS 2018*. Ed. by David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang. ACM Press, Oct. 2018, pp. 281–298. DOI: `10.1145/3243734.3243787`.

[Alb+19] Martin R. Albrecht, Léo Ducas, Gottfried Herold, Elena Kirshanova, Eamonn W. Postlethwaite, and Marc Stevens. The General Sieve Kernel and New Records in Lattice Reduction. to appear at *Eurocrypt 2019*. 2019.

[Amy+16] Matthew Amy, Olivia Di Matteo, Vlad Gheorghiu, Michele Mosca, Alex Parent, and John M. Schanck. Estimating the Cost of Generic Quantum Pre-image Attacks on SHA-2 and SHA-3. In: *SAC 2016*. Ed. by Roberto Avanzi and Howard M. Heys. Vol. 10532. LNCS. Springer, Heidelberg, Aug. 2016, pp. 317–337. DOI: `10.1007/978-3-319-69453-5_18`.

[APW09] Martin R. Albrecht, Kenneth G. Paterson, and Gaven J. Watson. Plaintext Recovery Attacks against SSH. In: *2009 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2009, pp. 16–26. DOI: `10.1109/SP.2009.5`.

[Bar17] Boaz Barak. The Complexity of Public-Key Cryptography. Cryptology ePrint Archive, Report 2017/365. `http://eprint.iacr.org/2017/365`. 2017.

[NCSC:QKD16] National Cyber Security Centre. Quantum Key Distribution. `https://www.ncsc.gov.uk/information/quantum-key-distribution`. 2016.