The year in post-quantum crypto

Daniel J. Bernstein, Tanja Lange

University of Illinois at Chicago,
Eindhoven University of Technology

Post-quantum cryptography:
Cryptography designed
under the assumption that
the **attacker** (not the user!)
has a large quantum computer.

http://joakimolofsson.deviantart.com/art/Pacific-Rim-372130691

Interest builds in post-quantum cryptography

- 2015: Finally even NSA admits that the world needs post-quantum crypto.
- 2016: Every agency posts something (NCSC UK, NCSC NL, NSA).
- 2016: After public input, NIST calls for submissions to "Post-Quantum Cryptography Standardization Project". Solicits submissions on signatures and encryption.

Interest builds in post-quantum cryptography

- 2003: djb coins term "post-quantum cryptography".
- 2005–2015: 10 years of motivating people to work on post-quantum crypto.
- 2015: Finally even NSA admits that the world needs post-quantum crypto.
- 2016: Every agency posts something (NCSC UK, NCSC NL, NSA).
- 2016: After public input, NIST calls for submissions to "Post-Quantum Cryptography Standardization Project". Solicits submissions on signatures and encryption.

A year ago in the NIST competition . . .

21 December 2017: NIST posts 69 submissions from 260 people.

BIG QUAKE. BIKE. CFPKM. Classic McEliece. Compact LWE. CRYSTALS-DILITHIUM. CRYSTALS-KYBER. DAGS. Ding Key Exchange. DME. DRS. DualModeMS. Edon-K. EMBLEM and R.EMBLEM. FALCON. FrodoKEM. GeMSS. Giophantus. Gravity-SPHINCS. Guess Again. Gui. HILA5. HiMQ-3. HK17. HQC. KINDI. LAC. LAKE. LEDAkem. LEDApkc. Lepton. LIMA. Lizard. LOCKER. LOTUS. LUOV. McNie. Mersenne-756839. MQDSS. NewHope. NTRUEncrypt. pqNTRUSign. NTRU-HRSS-KEM. NTRU Prime. NTS-KEM. Odd Manhattan. OKCN/AKCN/CNKE. Ouroboros-R. Picnic. pqRSA encryption. pqRSA signature. pqsigRM. QC-MDPC KEM. qTESLA. RaCoSS. Rainbow. Ramstake. RankSign. RLCE-KEM. Round2. RQC. RVB. SABER. SIKE. SPHINCS+. SRTPI. Three Bears. Titanium. WalnutDSA.

A year ago . . . there were already attacks

By end of 2017: 8 out of 69 submissions attacked.

BIG QUAKE. BIKE. CFPKM. Classic McEliece. Compact LWE. CRYSTALS-DILITHIUM. CRYSTALS-KYBER. DAGS. Ding Key Exchange. DME. DRS. DualModeMS. Edon-K. EMBLEM and R.EMBLEM. FALCON. FrodoKEM. GeMSS. Giophantus. Gravity-SPHINCS. Guess Again. Gui. HILA5. HiMQ-3. HK17. HQC. KINDI. LAC. LAKE. LEDAkem. LEDApkc. Lepton. LIMA. Lizard. LOCKER. LOTUS. LUOV. McNie. Mersenne-756839. MQDSS. NewHope. NTRUEncrypt. pqNTRUSign. NTRU-HRSS-KEM. NTRU Prime. NTS-KEM. Odd Manhattan. OKCN/AKCN/CNKE. Ouroboros-R. Picnic. pqRSA encryption. pqRSA signature. pqsigRM. QC-MDPC KEM. qTESLA. RaCoSS. Rainbow. Ramstake. RankSign. RLCE-KEM. Round2. RQC. RVB. SABER. SIKE. SPHINCS+. SRTPI. Three Bears. Titanium. WalnutDSA.

Some less security than claimed; some really broken; some attack scripts.

Do cryptographers have any idea what they're doing?

By end of 2018: 22 out of 69 submissions attacked.

BIG QUAKE. BIKE. CFPKM. Classic McEliece. Compact LWE. CRYSTALS-DILITHIUM. CRYSTALS-KYBER. DAGS. Ding Key Exchange. DME. DRS. DualModeMS. Edon-K. EMBLEM and R.EMBLEM. FALCON. FrodoKEM. GeMSS. Giophantus. Gravity-SPHINCS. Guess Again. Gui. HILA5. HiMQ-3. HK17. HQC. KINDI. LAC. LAKE. LEDAkem. LEDApkc. Lepton. LIMA. Lizard. LOCKER. LOTUS. LUOV. McNie. Mersenne-756839. MQDSS. NewHope. NTRUEncrypt. pqNTRUSign. NTRU-HRSS-KEM. NTRU Prime. NTS-KEM. Odd Manhattan. OKCN/AKCN/CNKE. Ouroboros-R. Picnic. pqRSA encryption. pqRSA signature. pqsigRM. QC-MDPC KEM. qTESLA. RaCoSS. Rainbow. Ramstake. RankSign. RLCE-KEM. Round2. RQC. RVB. SABER. SIKE. SPHINCS+. SRTPI. Three Bears. Titanium. WalnutDSA.

Some less security than claimed; some really broken; some attack scripts.

Some attempts to explain the situation

People often categorize submissions. Examples of categories:

- Code-based encryption and signatures.
- Hash-based signatures.
- Isogeny-based encryption.
- Lattice-based encryption and signatures.
- Multivariate-quadratic encryption and signatures.

Some attempts to explain the situation

"What's safe is lattice-based cryptography." — Are you sure about that?

Some attempts to explain the situation

"What's safe is lattice-based cryptography." — Are you sure about that?

Lattice-based submissions: Compact LWE. CRYSTALS-DILITHIUM. CRYSTALS-KYBER. Ding Key Exchange. DRS. EMBLEM and R.EMBLEM. FALCON. FrodoKEM. HILA5. KINDI. LAC. LIMA. Lizard. LOTUS. NewHope. NTRUEncrypt. NTRU-HRSS-KEM. NTRU Prime. Odd Manhattan. OKCN/AKCN/CNKE. pqNTRUSign. qTESLA. Round2. SABER. Titanium.

Some attempts to explain the situation

"What's safe is lattice-based cryptography." — Are you sure about that?

Lattice-based submissions: Compact LWE. CRYSTALS-DILITHIUM. CRYSTALS-KYBER. Ding Key Exchange. DRS. EMBLEM and R.EMBLEM. FALCON. FrodoKEM. HILA5. KINDI. LAC. LIMA. Lizard. LOTUS. NewHope. NTRUEncrypt. NTRU-HRSS-KEM. NTRU Prime. Odd Manhattan. OKCN/AKCN/CNKE. pqNTRUSign. qTESLA. Round2. SABER. Titanium.

Important progress in lattice attacks this decade—even this year.
e.g. D'Anvers–Vercauteren–Verbauwhede papers in November+December: "On the impact of decryption failures on the security of LWE/LWR based schemes"; "The impact of error dependencies on Ring/Mod-LWE/LWR based schemes".

Some attempts to explain the situation

"What's safe is using the portfolio from the European PQCRYPTO project."
— Are you sure about that?

Some attempts to explain the situation

"What's safe is using the portfolio from the European PQCRYPTO project."
— Are you sure about that?

The portfolio: BIG QUAKE. BIKE. Classic McEliece. CRYSTALS-DILITHIUM. CRYSTALS-KYBER. DAGS. FrodoKEM. Gui. KINDI. LUOV. MQDSS. NewHope. NTRU-HRSS-KEM. NTRU Prime. Picnic. qTESLA. Rainbow. Ramstake. SABER. SPHINCS+.

Some attempts to explain the situation

"What's safe is using the portfolio from the European PQCRYPTO project."
— Are you sure about that?

The portfolio: BIG QUAKE. BIKE. Classic McEliece. CRYSTALS-DILITHIUM. CRYSTALS-KYBER. DAGS. FrodoKEM. Gui. KINDI. LUOV. MQDSS. NewHope. NTRU-HRSS-KEM. NTRU Prime. Picnic. qTESLA. Rainbow. Ramstake. SABER. SPHINCS+.

69 submissions = **denial-of-service attack against security evaluation**.
Maybe cryptanalysts have been focusing on submissions from outside the project.

April 2018: PQCrypto 2018, and NIST conference

New RaCoSS parameters

Kirill Morozov (UNT)

UNT
UNIVERSITY OF NORTH TEXAS®
EST. 1890

RaCoSS – Random-code-based signature scheme

- Submitted to NIST Competition [Roy, M, Fukushima, Kiyomoto, Takagi ‘17]
- Adaptation of “Fiat-Shamir with abort” from [Lyubashevsky ‘09]
- [Hülsing, Bernstein, Panny, Lange: Nov’17] Attack on original parameters
- Updated secure parameters coming soon, but the keys and signature sizes are terabytes
- Quasi-cyclic (QC) variant: possibly megabytes
- # signatures (life-time of keys) may be limited
- Design improvements needed to shift from theoretical to practical security

Courtois-Finiasz-Sendrier code-based signature variant is SEUF-CMA

[M, Roy, Steinwandt, Xu ‘18]

https://www.degruyter.com/downloadpdf/j/math.2018.16.issue-1/math-2018-0011/math-2018-0011.pdf

- Problem: EUF-CMA security proof by [Dallot ‘07] does not apply due to Goppa-code distinguisher [Faugere, Gauthier, Otmani, Perret, Tillich, ‘11]
- Way around: Assume hardness of the underlying Niederreiter problem
- Extra: Security against key-substitution attack via hashing pk [Menezes Smart ‘04]

Framework for efficient adaptively secure UC oblivious transfer (OT) in ROM

[Barreto, David, Dowsley, M, Nascimento, Crypto ePrint ‘17] https:// ia.cr/2017/993

- Efficient universally composable (UC) protocol for OT secure against active adaptive adversaries from special type of OW-CPA secure PKE in ROM
- Covered: Low-noise LPN, McEliece, QC-MDPC, and CDH assumptions
- The first UC-secure OT protocols based on coding assumptions to achieve: 1) adaptive security, 2) low round complexity, 3) low communication and computational complexities

RaCoSS – Random-code-based signature scheme

- Submitted to NIST Competition [Roy, M, Fukushima, Kiyomoto, Takagi ‘17]
- Adaptation of “Fiat-Shamir with abort” from [Lyubashevsky ‘09]
- [Hülsing, Bernstein, Panny, Lange: Nov’17] Attack on original parameters
- Updated secure parameters coming soon, but the keys and signature sizes are terabytes
- Quasi-cyclic (QC) variant: possibly megabytes
- # signatures (life-time of keys) may be limited

C
c
is

htt
issu

Call for merged submissions

"NIST would like to encourage any submissions which are quite similar to consider merging."

Call for merged submissions

"NIST would like to encourage any submissions which are quite similar to consider merging."

"While the selection of candidates for the second round will primarily be based on the original submissions, NIST may consider a merged submission more attractive than either of the original schemes if it provides improvements in security, efficiency, or compactness and generality of presentation. At the very least, NIST will accept a merged submission to the second round if either of the submissions being merged would have been accepted."

Call for merged submissions

"NIST would like to encourage any submissions which are quite similar to consider merging."

"While the selection of candidates for the second round will primarily be based on the original submissions, NIST may consider a merged submission more attractive than either of the original schemes if it provides improvements in security, efficiency, or compactness and generality of presentation. At the very least, NIST will accept a merged submission to the second round if either of the submissions being merged would have been accepted."

"Submissions should only merge which are similar, and the merged submission should be in the span of the two original submissions."

August 2018: first merge announcement

4 August: HILA5 and Round2 merge to form Round5.

"The papers show that Round5 is a leading lattice-based candidate in terms of security, bandwidth and CPU performance."

August 2018: first merge announcement

4 August: HILA5 and Round2 merge to form Round5.

"The papers show that Round5 is a leading lattice-based candidate in terms of security, bandwidth and CPU performance."

24 August: Hamburg announces major vulnerability in Round5.

- Decryption failures in Round5 are much more likely than claimed.
- For many earlier lattice systems, presumably also for Round5: can break system using a small number of decryption failures.
- Underlying mistake wasn't in HILA5, wasn't in Round2.

August 2018: first merge announcement

4 August: HILA5 and Round2 merge to form Round5.

"The papers show that Round5 is a leading lattice-based candidate in terms of security, bandwidth and CPU performance."

24 August: Hamburg announces major vulnerability in Round5.

- Decryption failures in Round5 are much more likely than claimed.
- For many earlier lattice systems, presumably also for Round5: can break system using a small number of decryption failures.
- Underlying mistake wasn't in HILA5, wasn't in Round2.

Round5 response: "proposed fix" ... "looking at the security proof adjustments" ... "The actual Round5 proposal to NIST is still months away."

October–November 2018: More merge announcements

22 October: pqRSA encryption and pqRSA signatures merge to form pqRSA.

"This merged submission is a leading candidate in terms of depth of security analysis, amount of network traffic, and flexibility."

October–November 2018: More merge announcements

22 October: pqRSA encryption and pqRSA signatures merge to form pqRSA.

"This merged submission is a leading candidate in terms of depth of security analysis, amount of network traffic, and flexibility."

15 November: LEDAkem merges with LEDApkc.

October–November 2018: More merge announcements

22 October: pqRSA encryption and pqRSA signatures merge to form pqRSA.

"This merged submission is a leading candidate in terms of depth of security analysis, amount of network traffic, and flexibility."

15 November: LEDAkem merges with LEDApkc.

29 November: Ouroboros-R, LAKE, LOCKER merge to form ROLLO.

October–November 2018: More merge announcements

22 October: pqRSA encryption and pqRSA signatures merge to form pqRSA.

"This merged submission is a leading candidate in terms of depth of security analysis, amount of network traffic, and flexibility."

15 November: LEDAkem merges with LEDApkc.

29 November: Ouroboros-R, LAKE, LOCKER merge to form ROLLO.

29 November: NTRU-HRSS-KEM and NTRUEncrypt merge.

December 2018: field will be narrowed soon

13 December: "NIST will be announcing the 2nd round candidates at the Real World Crypto conference, Jan 9-11, in San Jose, California."

December 2018: field will be narrowed soon?

13 December: “NIST will be announcing the 2nd round candidates at the Real World Crypto conference, Jan 9-11, in San Jose, California.”

21 December: “We just wanted to alert you that in the case of a partial US government shutdown (which may start tonight), NIST will not be funded by Congress. As such, NIST employees will not be able to do any work. This includes the NIST PQC team. So in case of a shutdown, we will not be checking our email, monitoring the pqc-forum, doing analysis, etc. So you will hear silence from us if this occurs. We just wanted to let everybody know.”

Computer Security Resource Center

Due to the lapse in government funding, csrc.nist.gov and all associated online activities will be unavailable until further notice. Learn more.

Home Notifications Messages

Search Twitter

Tweet

Donald J. Trump
@realDonaldTrump

Follow

Have the Democrats finally realized that we desperately need Border Security and a Wall on the Southern Border. Need to stop Drugs, Human Trafficking,Gang Members & Criminals from coming into our Country. Do the Dems realize that most of the people not getting paid are Democrats?

6:06 AM - 27 Dec 2018

28,532 Retweets **120,240** Likes

53K 29K 120K

Donald J. T
@realDonaldTrun

March 2018: quantum cyber blockchain

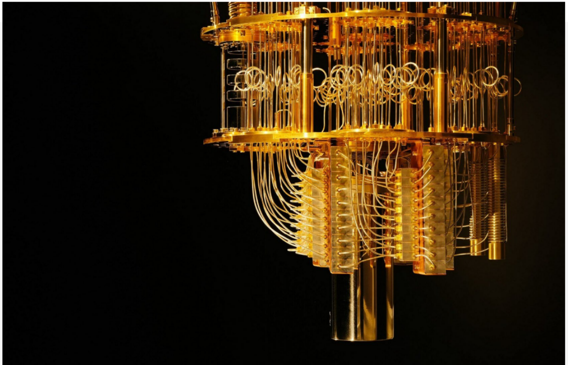
79 qubits from IonQ

SCIENCE INDUSTRY QUANTUM COMPUTING

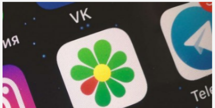
A new type of quantum computer has smashed every record

Quantum computing is progressing in leaps and bounds

By **Isaiah Mayersen** on December 16, 2018, 7:28 AM | 31 comments



MOST READ



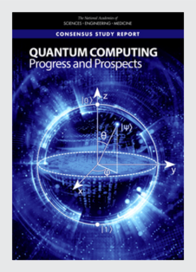
Quantum computer or microbrewery?

nap.edu report on quantum computing

The National Academies of SCIENCES ENGINEERING MEDICINE THE NATIONAL ACADEMIES PRESS

This PDF is available at http://nap.edu/25196

SHARE



Quantum Computing: Progress and Prospects (2018)

DETAILS

202 pages | 6 x 9 | PAPERBACK
ISBN 978-0-309-47969-1 | DOI 10.17226/25196

nap.edu report on quantum computing

Don't panic. "Key Finding 1: Given the current state of quantum computing and recent rates of progress, it is highly unexpected that a quantum computer that can compromise RSA 2048 or comparable discrete logarithm-based public key cryptosystems will be built within the next decade."

nap.edu report on quantum computing

Don't panic. "Key Finding 1: Given the current state of quantum computing and recent rates of progress, it is highly unexpected that a quantum computer that can compromise RSA 2048 or comparable discrete logarithm-based public key cryptosystems will be built within the next decade."

Panic. "Key Finding 10: Even if a quantum computer that can decrypt current cryptographic ciphers is more than a decade off, the hazard of such a machine is high enough—and the time frame for transitioning to a new security protocol is sufficiently long and uncertain—that prioritization of the development, standardization, and deployment of post-quantum cryptography is critical for minimizing the chance of a potential security and privacy disaster."

June 2018: quantum cyber blockchain

EDGY_

Science Technology Marketing Culture About Edgy

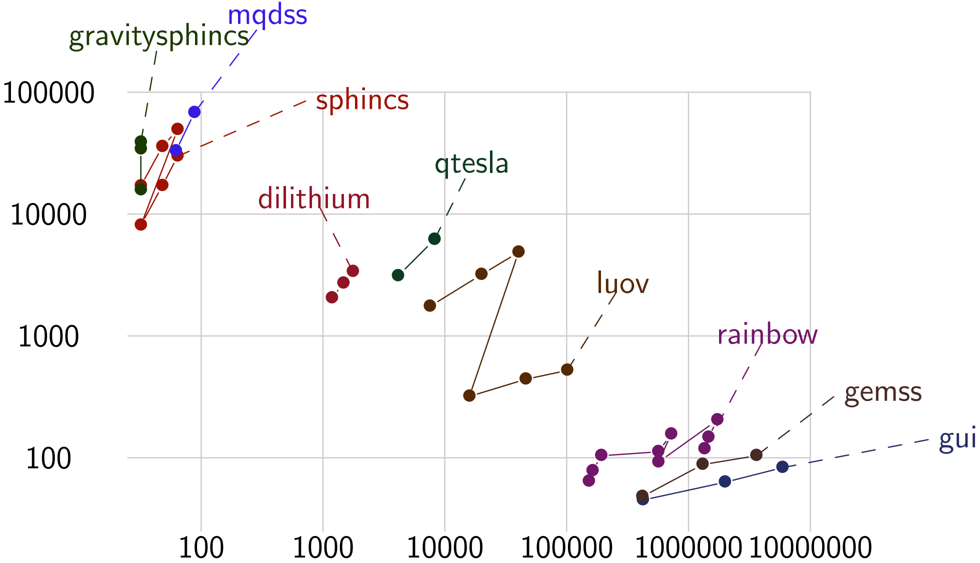
Technology 3 mins read

Quantum Resistant Ledger Could Make Current Blockchain Models Obsolete

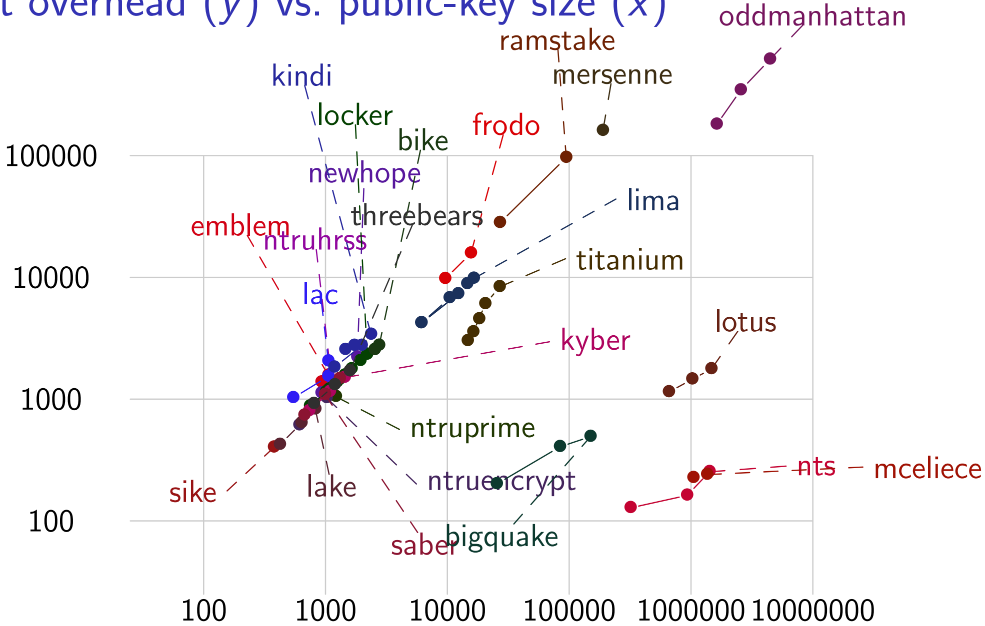
Quantum computing is just around the corner. To stay secure, cryptocurrencies need to adapt to this fact -- and fast. Enter the quantum resistant ledger, which could make contemporary blockchain systems obsolete.

Zayan Guedim
Jun 27

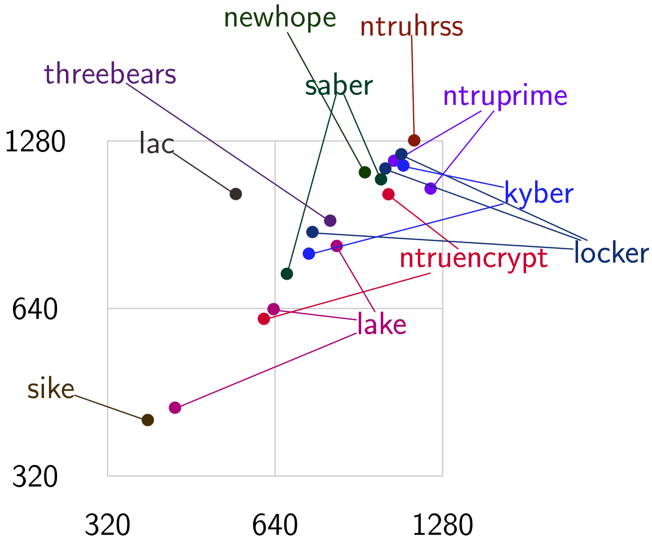
Signature size (y axis) vs. public-key size (x axis)



Ciphertext overhead (y) vs. public-key size (x)



Ciphertext overhead (y) vs. public-key size (x)



April 2018: Google–Cloudflare experiment

- Supersingular isogenies (SI): 400 bytes.
- Structured lattices (SL): 1 100 bytes.
- Unstructured lattice stand-in (ULS): 3 300 bytes (as placeholder, too many pages dropped at 10 000 bytes).

Configuration	Additional latency over control group		
	SI	SL	UL (estimated)
Desktop, Full, Median	4.0%	6.4%	71.2%
Desktop, Full, 95%	4.7%	9.6%	117.0%
Desktop, Resume, Median	4.3%	12.5%	118.6%
Desktop, Resume, 95%	5.2%	17.7%	205.1%
Mobile, Full, Median	-0.2%	3.4%	34.3%
Mobile, Full, 95%	0.5%	7.2%	110.7%
Mobile, Resume, Median	0.6%	7.2%	66.7%
Mobile, Resume, 95%	4.2%	12.5%	149.5%

December 2018: Google starts an NTRU-HRSS experiment

ImperialViolet

CECPQ2 (12 Dec 2018)

CECPQ1 was the experiment in post-quantum confidentiality that my colleague, Matt Braithwaite, and I ran in 2016. It's about time for CECPQ2.

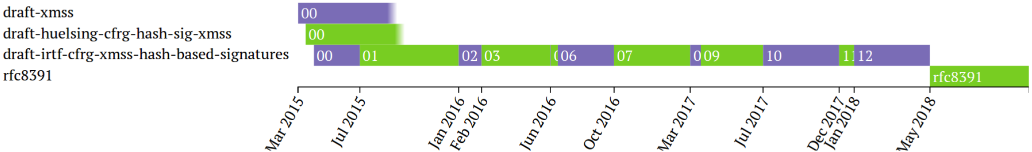
I've previously written about the experiments in Chrome which lead to the conclusion that structured lattices were likely the best area in which to look for a new key-exchange mechanism at the current time. Thanks

May 2018: XMSS RFC

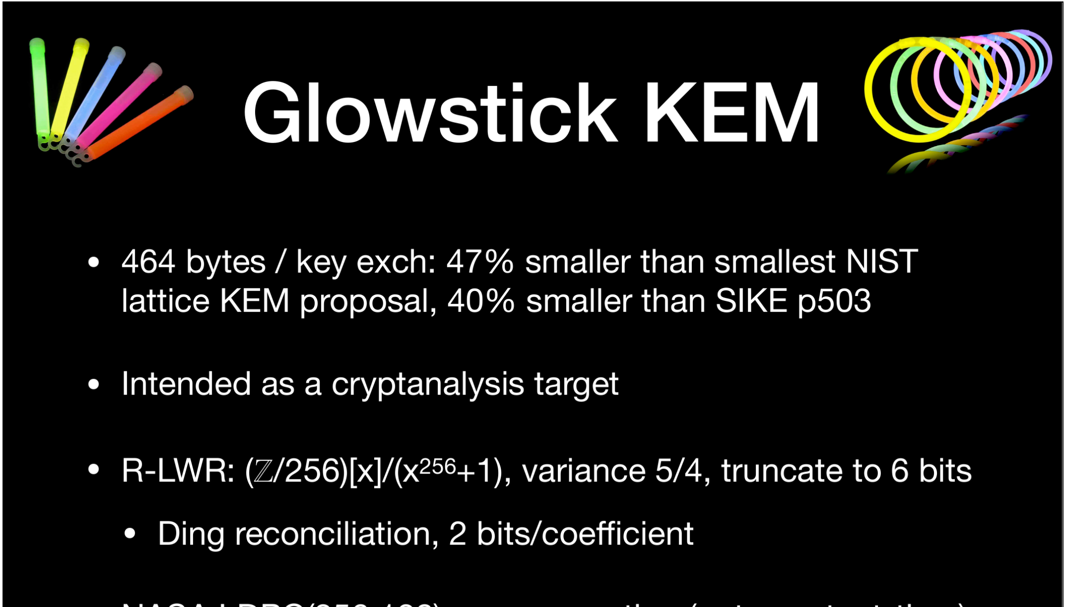
Datatracker Groups Documents Meetings Other User

```
Internet Research Task Force (IRTF)                          A. Huelsing
Request for Comments: 8391                                 TU Eindhoven
Category: Informational                                         D. Butin
ISSN: 2070-1721                                            TU Darmstadt
                                                               S. Gazdag
                                                              genua GmbH
                                                             J. Rijneveld
                                                      Radboud University
                                                             A. Mohaisen
                                           University of Central Florida
                                                                May 2018

                XMSS: eXtended Merkle Signature Scheme
```



Glowstick: how small can lattices go?



NIST submission Classic McEliece

- Security asymptotics unchanged by 40 years of cryptanalysis.
- Short ciphertexts.
- Efficient & straightforward conversion OW-CPA PKE → IND-CCA2 KEM.
- Open-source (public domain) implementations.
  - Constant-time software implementations.
  - FPGA implementation of full cryptosystem.
- No patents.

Metric	**mceliece6960119**	**mceliece8192128**
Public-key size	1047319 bytes	1357824 bytes
Secret-key size	13908 bytes	14080 bytes
Ciphertext size	226 bytes	240 bytes
Key-generation time	1108833108 cycles	1173074192 cycles
Encapsulation time	153940 cycles	188520 cycles
Decapsulation time	318088 cycles	343756 cycles

See `https://classic.mceliece.org` for more details.

Goodness, what big keys you have!

- Public keys look like this:

$$K = \begin{pmatrix} 1 & 0 & \dots & 0 & 1 & \dots & 1 & 0 & 1 \\ 0 & 1 & \dots & 0 & 1 & \dots & 0 & 1 & 1 \\ \vdots & \vdots & \ddots & \vdots & 1 & \dots & 1 & 1 & 0 \\ 0 & 0 & \dots & 1 & 0 & \dots & 1 & 1 & 1 \end{pmatrix}$$

  Left part is $(n-k) \times (n-k)$ identity matrix (no need to send)
  right part is random-looking $(n-k) \times k$ matrix.
  E.g. $n = 6960$, $k = 5413$, so $n - k = 1547$.
- Encryption xors secretly selected columns.

Big issues with big keys

- Sending 1MB takes time and bandwidth.

Big issues with big keys

- Sending 1MB takes time and bandwidth.
- Google–Cloudlare experiment:

  in some cases the public-key + ciphertext size was too large to be viable in the context of TLS

  and even 10KB messages dropped.

Big issues with big keys

- Sending 1MB takes time and bandwidth.
- Google–Cloudlare experiment:

  in some cases the public-key + ciphertext size was too large to be viable in the context of TLS

  and even 10KB messages dropped.
- If server accepts 1MB of public key from any client,
  an attacker can easily flood memory.
  This invites DoS attacks.

Can servers avoid storing big keys?

$$K = \begin{pmatrix} 1 & 0 & \dots & 0 & 1 & \dots & 1 & 0 & 1 \\ 0 & 1 & \dots & 0 & 1 & \dots & 0 & 1 & 1 \\ \vdots & \vdots & \ddots & \vdots & 1 & \dots & 1 & 1 & 0 \\ 0 & 0 & \dots & 1 & 0 & \dots & 1 & 1 & 1 \end{pmatrix} = (I_{n-k}|K')$$

- Encryption xors secretly selected columns.
- With some storage and trusted environment:
  Receive columns of K' one at a time, store and update partial sum.

Can servers avoid storing big keys?

$$K = \begin{pmatrix} 1 & 0 & \dots & 0 & 1 & \dots & 1 & 0 & 1 \\ 0 & 1 & \dots & 0 & 1 & \dots & 0 & 1 & 1 \\ \vdots & \vdots & \ddots & \vdots & 1 & \dots & 1 & 1 & 0 \\ 0 & 0 & \dots & 1 & 0 & \dots & 1 & 1 & 1 \end{pmatrix} = (I_{n-k}|K')$$

- Encryption xors secretly selected columns.
- With some storage and trusted environment:
  Receive columns of K' one at a time, store and update partial sum.
- On the real Internet, without per-client state:

Can servers avoid storing big keys?

$$K = \begin{pmatrix} 1 & 0 & \dots & 0 & 1 & \dots & 1 & 0 & 1 \\ 0 & 1 & \dots & 0 & 1 & \dots & 0 & 1 & 1 \\ \vdots & \vdots & \ddots & \vdots & 1 & \dots & 1 & 1 & 0 \\ 0 & 0 & \dots & 1 & 0 & \dots & 1 & 1 & 1 \end{pmatrix} = (I_{n-k}|K')$$

- Encryption xors secretly selected columns.
- With some storage and trusted environment:
  Receive columns of K' one at a time, store and update partial sum.
- On the real Internet, without per-client state:
  Don't reveal intermediate results! It's a secret, which columns are picked!
  Intermediate results show whether a column was used or not.

McTiny (Bernstein/Lange, 2018?)

Partition key

$$K' = \begin{pmatrix} K_{1,1} & K_{1,2} & K_{1,3} & \dots & K_{1,\ell} \\ K_{2,1} & K_{2,2} & K_{2,3} & \dots & K_{2,\ell} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ K_{r,1} & K_{r,2} & K_{r,3} & \dots & K_{r,\ell} \end{pmatrix}$$

- Each submatrix $K_{i,j}$ small enough to fit + cookie into network packet.
- Server does computation on $K_{i,j}$, puts partial result into cookie.
- Cookies are encrypted by server to itself using some temporary symmetric key (same key for all server connections). No per-client memory allocation.
- Client feeds the $K_{i,j}$ to server & handle storage for the server.
- Cookies also encrypted & authenticated to client.
- More stuff to avoid replay & similar attacks.

McTiny (Bernstein/Lange, 2018?)

Partition key

$$K' = \begin{pmatrix} K_{1,1} & K_{1,2} & K_{1,3} & \dots & K_{1,\ell} \\ K_{2,1} & K_{2,2} & K_{2,3} & \dots & K_{2,\ell} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ K_{r,1} & K_{r,2} & K_{r,3} & \dots & K_{r,\ell} \end{pmatrix}$$

- Each submatrix $K_{i,j}$ small enough to fit + cookie into network packet.
- Server does computation on $K_{i,j}$, puts partial result into cookie.
- Cookies are encrypted by server to itself using some temporary symmetric key (same key for all server connections). No per-client memory allocation.
- Client feeds the $K_{i,j}$ to server & handle storage for the server.
- Cookies also encrypted & authenticated to client.
- More stuff to avoid replay & similar attacks.
- Several round trips, but no per-client state on the server.

October 2018: quantum cyber blockchain

BLOCKCHAIN TECHNOLOGY

NEO and Arqit form quantum resistant blockchain research partnership

6 Shares

WEDNESDAY, OCTOBER 17, 2018

NEO, the global Chinese originated public blockchain project with a steadily growing ecosystem today announced that NEO Global Development (NGD) will be partnering with Arqit on quantum resistance research with a shared vision of facilitating blockchain industry development. Arqit is a non-profit community driven blockchain project based in Singapore

As quantum resistance is a core feature and significant aspect of the NEO network, NEO Global Development (NGD) has been pushing forward with relevant research on quantum-proof technology represented by NeoQS (Quantum Safe), a lattice-based cryptographic mechanism.

May 2018: NIST publishes patent statements

NIST required each submission team to declare its patents (and patent applications) on that submission.

- BIKE☢
- Compact LWE☢
- Ding Key Exchange☢
- DME☢
- FALCON☢
- Gui☢
- HQC☢
- Lizard☢
- MQDSS☢
- OKCN/AKCN/CNKE☢
- Ouroboros-R☢
- pqNTRUSign☢
- QC-MDPC KEM☢
- Rainbow☢
- RLCE-KEM☢
- Round2☢
- RQC☢
- WalnutDSA☢

Warning: More submissions are covered by patents

(12) **United States Patent**
Gaborit et al.

(10) **Patent No.: US 9,094,189 B2**
(45) **Date of Patent: Jul. 28, 2015**

(54) **CRYPTOGRAPHIC METHOD FOR COMMUNICATING CONFIDENTIAL INFORMATION**

(75) Inventors: **Philippe Gaborit**, Feytiat (FR); **Carlos Aguilar Melchor**, Limoges (FR)

(73) Assignee: **CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE-CNRS**, Paris (FR)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 319 days.

(21) Appl. No.: **13/579,682**

(22) PCT Filed: **Feb. 17, 2011**

(86) PCT No.: **PCT/FR2011/050336**

§ 371 (c)(1), (2), (4) Date: **Feb. 4, 2013**

(87) PCT Pub. No.: **WO2011/101598**

PCT Pub. Date: **Aug. 25, 2011**

(65) **Prior Publication Data**

US 2013/0132723 A1 May 23, 2013

(30) **Foreign Application Priority Data**

Feb. 18, 2010 (FR) 10 51190

(52) **U.S. Cl.**
CPC .. ***H04L 9/08*** (2013.01); ***G09C 1/00*** (2013.01); ***H04L 9/0841*** (2013.01); ***H04L 9/304*** (2013.01)

(58) **Field of Classification Search**
CPC H04L 9/08; G09C 1/00
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,144,740	A *	11/2000	Laih et al.	380/2
7,010,738	B2 *	3/2006	Morioka et al.	714/752
7,080,255	B1 *	7/2006	Kasahara et al.	713/182

(Continued)

OTHER PUBLICATIONS

Regev, "On Lattices, Learning with Errors, Random Linear Codes, and Cryptography", May 24, 2005, pp. 84-93, XP002497024.

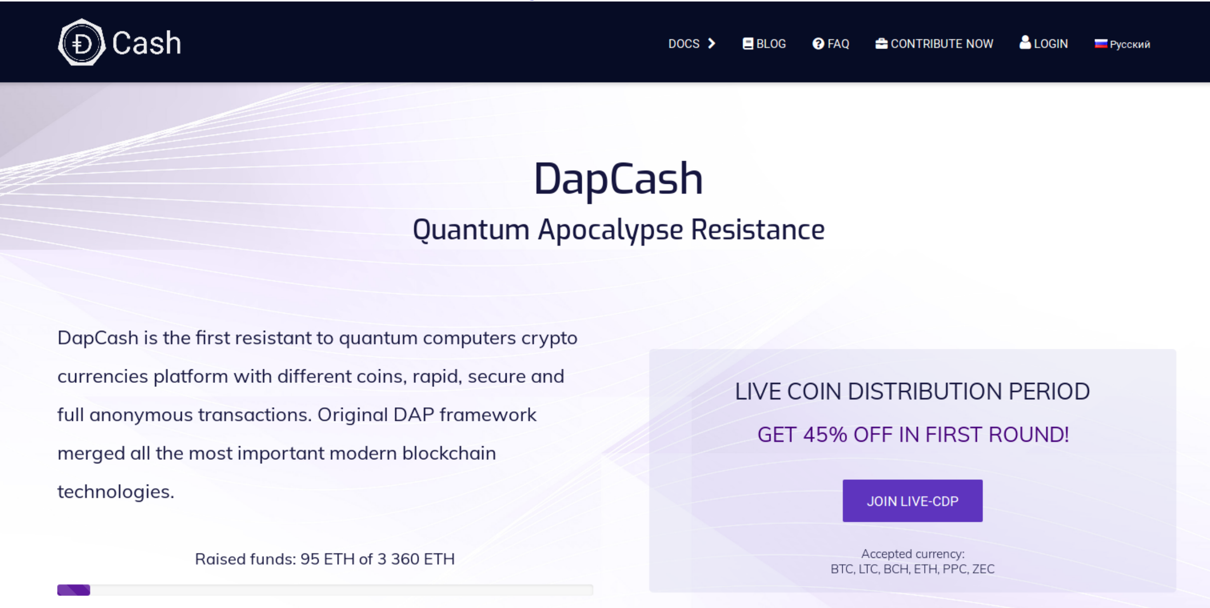
(Continued)

Primary Examiner — Dede Zecher
Assistant Examiner — Jason C Chiang
(74) *Attorney, Agent, or Firm* — Young & Thompson

(57) **ABSTRACT**

A cryptographic method for communicating confidential information m between a first electronic entity (A) and a second electronic entity (B), includes a distribution step and a reconciliation step, the distribution step including a plurality of steps, one of which consists of the first entity (A) and the second entity (B) calculating a first intermediate value P_A and a second intermediate value P_B, respectively, such that: $P_A=Y_A \cdot S_B=Y_A \cdot X_B+Y_A \cdot f(Y_B)$, and $P_B=Y_B \cdot S_A=Y_B \cdot X_A+Y_B$

December 2018: quantum cyber blockchain




[ˈsiːˌsaɪd]

CSIDH: An Efficient Post-Quantum Commutative Group Action

CSIDH: An Efficient Post-Quantum Commutative Group Action

Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, Joost Renes

- Closest thing we have in PQC to normal Diffie–Hellman key exchange: Keys can be reused, blinded; no difference between initiator &responder.
- Public keys are represented by some $A \in \mathbf{F}_p$; p fixed prime.
- Alice computes and distributes her public key A.
  Bob computes and distributes his public key B.
- Alice and Bob do computations on each other's public keys to obtain shared secret.
- Fancy math: computations start on some elliptic curve $E_A : y^2 = x^3 + Ax^2 + x$, use *isogenies* to move to a different curve.
- Computations need arithmetic (add, mult, div) modulo p and elliptic-curve computations.

Security

Size of key space:

- About $\sqrt{p}$ of all $A \in \mathbf{F}_p$ are valid keys.

Without quantum computer:

- Meet-in-the-middle variants: Time $O(\sqrt[4]{p})$.

Security

Size of key space:

- About $\sqrt{p}$ of all $A \in \mathbf{F}_p$ are valid keys.

Without quantum computer:

- Meet-in-the-middle variants: Time $O(\sqrt[4]{p})$.

With quantum computer:

- Hidden-shift algorithms apply: Subexponential complexity.
  - Literature contains mostly asymptotics.
  - Recent work analyzing cost: see https://quantum.isogeny.org.

CSIDH security:

- Public-key validation:
  Quickly check that $E_A : y^2 = x^3 + Ax^2 + x$ has $p + 1$ points.

CSIDH-512

Sizes:

- Private keys: 32 bytes. (37 in current software for simplicity.)
- Public keys: 64 bytes.

Performance on typical Intel Skylake laptop core:

- Wall-clock time: 32ms per operation.
- Clock cycles: about 10^8 per operation.
- Memory usage: about 4 kilobytes.

Security:

- Pre-quantum: at least 128 bits.

CSIDH-512

Sizes:

- Private keys: 32 bytes. (37 in current software for simplicity.)
- Public keys: 64 bytes.

Performance on typical Intel Skylake laptop core:

- Wall-clock time: 32ms per operation.
- Clock cycles: about 10^8 per operation.
- Memory usage: about 4 kilobytes.

Security:

- Pre-quantum: at least 128 bits.
- Post-quantum: complicated. AFAWK similar to AES-128.

Website:

- https://csidh.isogeny.org/

October 2018: quantum AI blockchain

Quantum AI Blockchain

News

24-Oct-2018

Quantum AI Blockchain T-Shirt available! Grab yours here!

14-May-2018 Quantum Computing for Computer Scientists

18-Apr-2018 Quantum Blockchain using entanglement in time

10-Apr-2018 Quantum Algorithm Implementations for Beginners

7-Mar-2018 Google Engineers Think This 72-Qubit Processor Can Achieve Quantum Supremacy

7-Mar-2018 'BLOCKCHAIN' IS MEANINGLESS

27-Feb-2018 UChicago scientists to lead $10 million NSF 'expedition' for practical quantum computing

13-Feb-2018 Blockchain and Artificial Intelligence - Tshilidzi Marwala, Bo Xing

8-Sep-2017 Machine learning & artificial intelligence in the quantum domain - Vedran Dunjko, Hans J. Briegel

Board of Directors

Noon van der Silk, Lyndon Maydwell, Sean Seefried, Fabion Kauker, Susie Sheldrick and Dr Ruth Pearson.

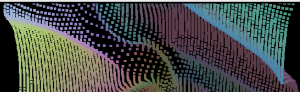
Investors/Contact

We're only interested in investment from the wholesale quantum investment, bitcoin or AI communities; all of whom can get in contact with us directly by standard means. If you're outside of these communities, we belive you will be able to get the details of one of us. Sorry. Our contact details are not listed directly due to overwhelming interest.

Talky Form

Welcome to Talky Form! Cutting-edge AI chat.

Hi!
What is your name?



The evolution of cryptographic software quality

Good

Bad

Terrible

Horrifying

1978 1988 1998 2008 2018

The evolution of cryptographic software quality

Good

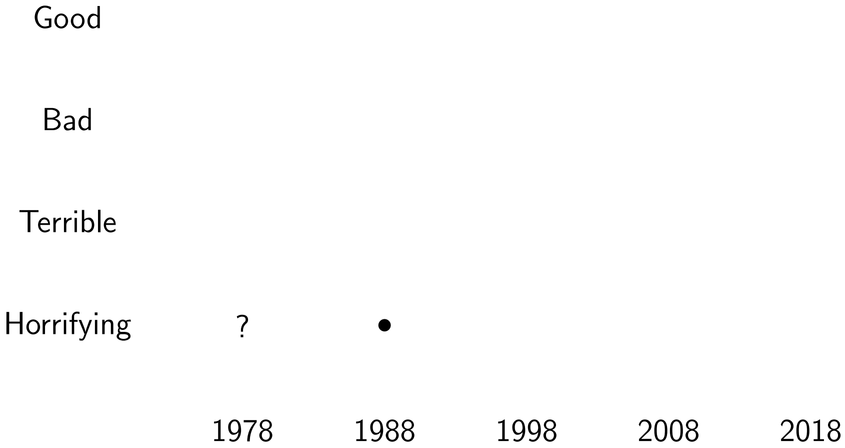
Bad

Terrible

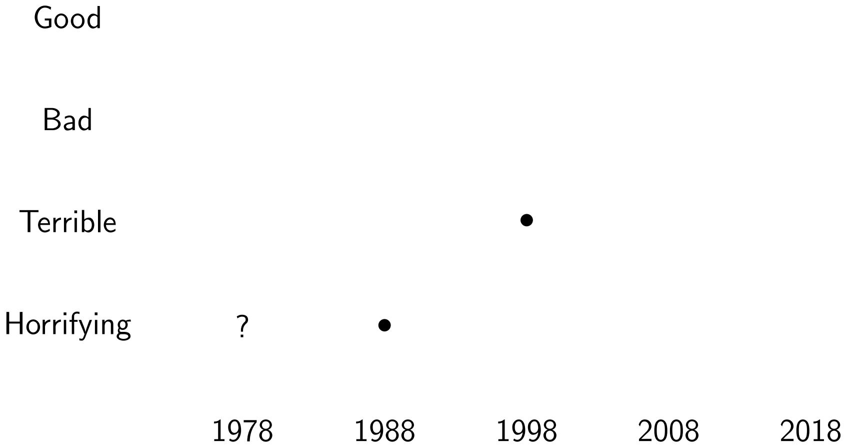
Horrifying ?

1978 1988 1998 2008 2018

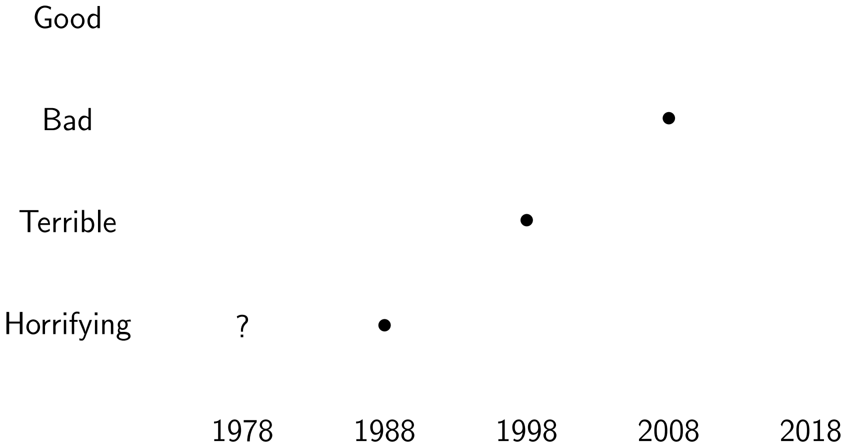
The evolution of cryptographic software quality



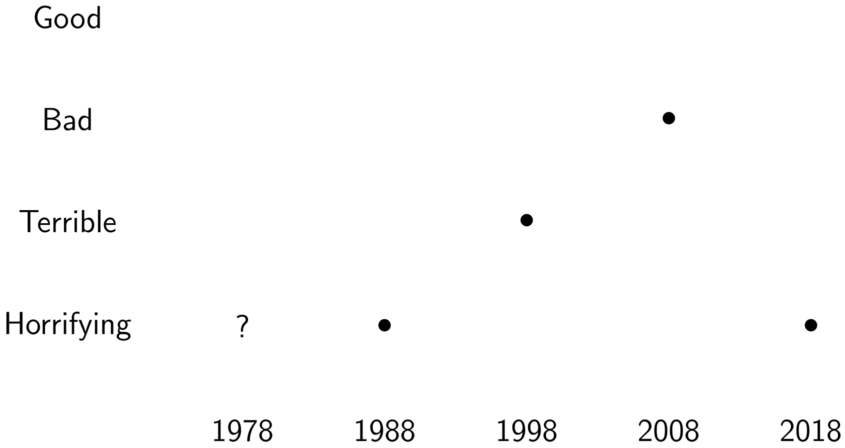
The evolution of cryptographic software quality



The evolution of cryptographic software quality



The evolution of cryptographic software quality



Where to find (scary) software for NIST submissions

- NIST (try https://archive.org during US government shutdowns): code submitted in 2017—reference code, sometimes also optimized code.

Where to find (scary) software for NIST submissions

- NIST (try https://archive.org during US government shutdowns): code submitted in 2017—reference code, sometimes also optimized code.
- Upstream web sites for 36 individual submissions.

Where to find (scary) software for NIST submissions

- NIST (try https://archive.org during US government shutdowns): code submitted in 2017—reference code, sometimes also optimized code.
- Upstream web sites for 36 individual submissions.
- SUPERCOP benchmarking framework, https://bench.cr.yp.to: 356 implementations of 170 primitives from 40 submissions.

Where to find (scary) software for NIST submissions

- NIST (try https://archive.org during US government shutdowns): code submitted in 2017—reference code, sometimes also optimized code.
- Upstream web sites for 36 individual submissions.
- SUPERCOP benchmarking framework, https://bench.cr.yp.to: 356 implementations of 170 primitives from 40 submissions.
- https://libpqcrypto.org:
  Simple C API, Python API, CLI; designed for robust production use.
  165 implementations of 77 primitives from 19 submissions.

Where to find (scary) software for NIST submissions

- NIST (try https://archive.org during US government shutdowns): code submitted in 2017—reference code, sometimes also optimized code.
- Upstream web sites for 36 individual submissions.
- SUPERCOP benchmarking framework, https://bench.cr.yp.to: 356 implementations of 170 primitives from 40 submissions.
- https://libpqcrypto.org: Simple C API, Python API, CLI; designed for robust production use. 165 implementations of 77 primitives from 19 submissions.
- https://github.com/mupq/pqm4: Some primitives for ARM Cortex-M4.
- https://github.com/mupq/pqhw: A few primitives for FPGA.
- https://openquantumsafe.org: OpenSSL/OpenSSH integrations of 59 primitives from 13 submissions.

A modern cryptographic API

Most libraries provide simple all-in-one hashing:

```
const unsigned char m[...]; unsigned long long mlen;
unsigned char h[crypto_hash_BYTES];
crypto_hash_sha256(h,m,mlen);
```

A modern cryptographic API

Most libraries provide simple all-in-one hashing:

```
const unsigned char m[...]; unsigned long long mlen;
unsigned char h[crypto_hash_BYTES];
crypto_hash_sha256(h,m,mlen);
```

Why not the same simplicity for, e.g., signing?

```
crypto_sign_ed25519(sm,&smlen,m,mlen,sk);
```

A modern cryptographic API

Most libraries provide simple all-in-one hashing:

```
const unsigned char m[...]; unsigned long long mlen;
unsigned char h[crypto_hash_BYTES];
crypto_hash_sha256(h,m,mlen);
```

Why not the same simplicity for, e.g., signing?

```
crypto_sign_ed25519(sm,&smlen,m,mlen,sk);
```

API introduced in SUPERCOP. Reused in NaCl, libsodium, `libpqcrypto`, etc.

A modern cryptographic API

Most libraries provide simple all-in-one hashing:

```
const unsigned char m[...]; unsigned long long mlen;
unsigned char h[crypto_hash_BYTES];
crypto_hash_sha256(h,m,mlen);
```

Why not the same simplicity for, e.g., signing?

```
crypto_sign_ed25519(sm,&smlen,m,mlen,sk);
```

API introduced in SUPERCOP. Reused in NaCl, libsodium, `libpqcrypto`, etc.
Usability impact: see 2017 Acar–Backes–Fahl–Garfinkel–Kim–Mazurek–Stransky.

A modern cryptographic API

Most libraries provide simple all-in-one hashing:

```
const unsigned char m[...]; unsigned long long mlen;
unsigned char h[crypto_hash_BYTES];
crypto_hash_sha256(h,m,mlen);
```

Why not the same simplicity for, e.g., signing?

```
crypto_sign_ed25519(sm,&smlen,m,mlen,sk);
```

API introduced in SUPERCOP. Reused in NaCl, libsodium, `libpqcrypto`, etc.
Usability impact: see 2017 Acar–Backes–Fahl–Garfinkel–Kim–Mazurek–Stransky.
Required by NIST. (But not enforced by test framework; many screwups.)

A modern cryptographic API

Most libraries provide simple all-in-one hashing:

```
const unsigned char m[...]; unsigned long long mlen;
unsigned char h[crypto_hash_BYTES];
crypto_hash_sha256(h,m,mlen);
```

Why not the same simplicity for, e.g., signing?

```
crypto_sign_ed25519(sm,&smlen,m,mlen,sk);
```

API introduced in SUPERCOP. Reused in NaCl, libsodium, `libpqcrypto`, etc.
Usability impact: see 2017 Acar–Backes–Fahl–Garfinkel–Kim–Mazurek–Stransky.
Required by NIST. (But not enforced by test framework; many screwups.)
Has promoted extensive code sharing. Working on reducing duplication.

50 signature systems in `libpqcrypto`

```
crypto_sign_dilithium{2,3,4}
crypto_sign_gui{184,312,448}
crypto_sign_luov{863256,890351,
    8117404,4849242,6468330,8086399}
crypto_sign_mqdss{48,64}
crypto_sign_picnicl{1,3,5}{fs,ur}
crypto_sign_qtesla{128,192,256}
crypto_sign_rainbow{1a,1b,1c,
    3b,3c,4a,5c,6a,6b}
crypto_sign_sphincs{f,s}{128,192,256}
    {haraka,sha256,shake256}
```

27 encryption systems in libpqcrypto

```
crypto_kem_bigquake{1,3,5}
crypto_kem_mceliece{6960119,8192128}
crypto_kem_kyber{512,768,1024}
crypto_kem_dags{3,5}
crypto_kem_frodokem{640,976}
crypto_kem_kindi{256342,256522,
    512222,512241,512321}
crypto_kem_newhope{512,1024}cca
crypto_kem_ntruhrss701
crypto_kem_{ntrulpr,sntrup}4591761
crypto_kem_ramstakers{216091,756839}
crypto_kem_{lightsaber,saber,firesaber}
```

Python interface for `libpqcrypto`

Generate key pair:

```
pk,sk = pqcrypto.sign.sphincsf128sha256.keypair()
```

Sign message `m`:

```
sm = pqcrypto.sign.sphincsf128sha256.sign(m,sk)
```

Recover message from signed message:

```
m = pqcrypto.sign.sphincsf128sha256.open(sm,pk)
```

If verification fails: exception and **no output**.

A larger Python example

Test script to sign and recover a message under a random key pair:

```
import pqcrypto
sig = pqcrypto.sign.sphincsf128sha256
pk,sk = sig.keypair()
m = b"hello world"
sm = sig.sign(m,sk)
assert m == sig.open(sm,pk)
```

The future

Various `libpqcrypto` goals and ongoing work:

- Eliminate data flow from secrets to array indices and branch conditions. (Stop, e.g., 2016 CacheBleed attack, 2018 OpenSSL RSA keygen attack.) Already done for *some* implementations.

The future

Various `libpqcrypto` goals and ongoing work:

- Eliminate data flow from secrets to array indices and branch conditions. (Stop, e.g., 2016 CacheBleed attack, 2018 OpenSSL RSA keygen attack.) Already done for *some* implementations.
- More tests. (Upstream often fails Valgrind and ASan!)
- More audits.
- Formal verification—eliminating the bugs missed by tests. Some progress: see, e.g., https://sorting.cr.yp.to.

The future

Various `libpqcrypto` goals and ongoing work:

- Eliminate data flow from secrets to array indices and branch conditions. (Stop, e.g., 2016 CacheBleed attack, 2018 OpenSSL RSA keygen attack.) Already done for *some* implementations.
- More tests. (Upstream often fails Valgrind and ASan!)
- More audits.
- Formal verification—eliminating the bugs missed by tests. Some progress: see, e.g., `https://sorting.cr.yp.to`.
- Faster installation.
- Less CPU time. Already many speedups.

The future

Various `libpqcrypto` goals and ongoing work:

- Eliminate data flow from secrets to array indices and branch conditions.
  (Stop, e.g., 2016 CacheBleed attack, 2018 OpenSSL RSA keygen attack.)
  Already done for *some* implementations.
- More tests. (Upstream often fails Valgrind and ASan!)
- More audits.
- Formal verification—eliminating the bugs missed by tests.
  Some progress: see, e.g., https://sorting.cr.yp.to.
- Faster installation.
- Less CPU time. Already many speedups.
- Reducing code volume: e.g., SHA-3 merge.

The future

Various `libpqcrypto` goals and ongoing work:

- Eliminate data flow from secrets to array indices and branch conditions. (Stop, e.g., 2016 CacheBleed attack, 2018 OpenSSL RSA keygen attack.) Already done for *some* implementations.
- More tests. (Upstream often fails Valgrind and ASan!)
- More audits.
- Formal verification—eliminating the bugs missed by tests. Some progress: see, e.g., https://sorting.cr.yp.to.
- Faster installation.
- Less CPU time. Already many speedups.
- Reducing code volume: e.g., SHA-3 merge.
- Long term: **Reduce** number of primitives.