Security Evaluation in the NIST Post-Quantum Cryptography Project

Markku-Juhani O. Saarinen
<mjos@mjos.fi>
Lab 1339 / teserakt.io
P.O. Box 1339, CB1 0BZ, Cambridge, UK

VTT CryptoMetrics Seminar
Espoo, Finland – May 21, 2018
History: Cryptographic Standardization and NIST

- **DES** (1975) and **3DES** were long deemed sufficient for commercial applications (mainly in finance). Before 1990s almost all other cryptography was proprietary.

- In the first half of 1990s many important cryptanalytic techniques (especially **differential** and **linear** cryptanalysis) were invented and the use of cryptographic protocols became more widespread due commercial requirements.

- In January 1997 NIST announced the **Advanced Encryption Standard** process (organized as an international “competition”) to replace DES with a block cipher that had a 128-bit block size and 128/192/256 - bit key size.

12.09.1997: Call for algorithms; evaluation criteria, reference code API etc.
09.08.1999: 5 finalists selected by NIST (Rijndael, MARS, RC6, Serpent, Twofish)
02.10.2000: Announcement of Rijndael (from Belgium) as AES.
Lessons learned from the (very public) AES Competition

- Even though the final decision was with NIST and not everyone agreed with it, consensus was that all finalists were **thoroughly evaluated** and that the process was **transparent**. AES has been universally adopted since standardization.

- The AES competition clearly caused a **very positive spur in development** of symmetric cryptography: Lots of new research, PhD graduates, professionalism.
Requirements are fairly clear for block ciphers:

- Security should be exactly commensurate to external bit size parameters.
  - Brute force over $2^k$ trials should be the fastest way of breaking a $k$-bit cipher.
  - Chosen plaintext / ciphertext attacks can use “almost” the entire dictionary.
  - Attack memory is bound only by time ($2^k$).
  - Modes of operation secure up to birthday bound $2^{\frac{b}{2}}$ for $b$-bit block size.

- One should be able give convincing design arguments for security against “standard attacks” such as differential and linear cryptanalysis.

- Algorithms should be universal across a wide range of target platforms, applicable from low-power smartcards to high-throughput core routers.

So anyone can just RTFM and design a cipher now?

- I have professionally evaluated (proprietary) ciphers for well-funded customers in the Middle East. I broke many of the ciphers that they were offered. Some were by well known authors. Laziness, incompetence, or malice? Who knows.
NIST Hash Function (SHA3) Competition (2007-2012)

▶ Similar methodology was adopted. In the end, a “sufficiently different” algorithm from previous SHA designs was selected as a cryptanalytic contingency plan.

▶ Additionally implementation security was considered; Keccak (SHA3) has no S-Boxes and is therefore less vulnerable to (cache timing) side-channel attacks.

CAESAR (2014-2018)

▶ An attempt to find AEAD (Authenticated Encryption with Associated Data) primitives for standardization. Ran by Dan Bernstein, seemingly by himself. Criticized for bad organization and lack of transparency.

National Ciphers

▶ There are many national block cipher standards: ARIA (South Korea), Camellia (Japan), SM4 (China), Kuznyechik (Russia). Standardized after public evaluation. Not carbon copies, but close to AES. Russians didn’t publish design criteria.

NIST Lightweight Cryptography Project (Draft Call for Proposals 2018)
Best Known “Classical” Factoring Algorithm

General Number Field Sieve has $O \left( \exp \left( \frac{64}{9} n \log n \right)^{3/2} \right)$ complexity. This curve matches NIST’s RSA key size recommendations (SP 800-57 Pt. 1 Rev. 4, Jan 2016).
Shor’s factoring algorithm has time complexity $O(n^3 \log n)$ and requires a quantum circuit of $O(n^2 \log n \log \log n)$ gates. It is therefore a polynomial complexity algorithm.
Post-Quantum Cryptography: Background and Terminology

Quantum Computing (QC) uses quantum superpositions, rather than binary digits, to perform computations. This computational model was first considered in 1980s.

Quantum Algorithms are algorithms for Quantum Computers. They often have different performance asymptotics from classical algorithms.

Shor’s Algorithm (1994) can factor integers and compute discrete logarithms efficiently (polytime). It also applies to the the Elliptic Curve DL Problem. Shor’s algorithms can be devastating to most of current public key cryptography.

Grover’s Search Algorithm (1996) can be used to search for a $k$-bit secret key with $\sqrt{2^k} = 2^{k/2}$ quantum effort. It effectively doubles the required key sizes for ciphers. However, symmetric crypto is safer against quantum attacks than public-key crypto.

Post-Quantum or “Quantum Resistant” cryptography consists of algorithms that run efficiently on classical computers but are hard to break with quantum algorithms.
In August 2015 the Committee on National Security Systems (CNSS) and National Security Agency (NSA) suddenly revised their cryptographic recommendations in CNSSAM 02-15. “Based on analysis of the effect of quantum computing [...] the set of authorized algorithms is [changed] as we anticipate a need to shift to quantum-resistant cryptography in the near future.”

The recommendation killed off shorter key lengths (AES-128, SHA-256, RSA-2048, DL-2048, ECC P-256) allowed in “Suite B”.

The new (interim) set of algorithms is called “Commercial National Security Algorithm Suite” and is approved up to TOP SECRET: RSA 3072, DH 3072, ECDH/DSA P-384, SHA-384, AES-256.

(Only these algorithms, only these key sizes are in CNSA)
NIST Post-Quantum Cryptography Effort

Due to NSA's requirement, and the high risk of quantum attacks within 20-50 year confidentiality period of classified data, a PQC project was started by NIST in 2016. The project focuses on new Public Key Encryption and Digital Signature algorithms.

- **20.12.2016**: Call for proposals released.
- **30.11.2017**: Deadline for submissions.
- **21.12.2017**: Round 1 algorithms announced (69 submissions accepted).
- **11-13.04.2018**: First NIST PQC Standardization Conference.
  - We are here. Candidate algorithms are known, analysis is ongoing.
- **2018 / 2019**: Round 2 begins.
- **August 2019**: Second PQC standardization Conference.
- **2020 / 2021**: Round 3 begins (or select algorithms).
- **2022 / 2024**: Draft standards available.
First Round PQC Candidates

BIG QUAKE, BIKE, **CFPKM**, Classic McEliece, **Compact LWE**, CRYSTALS-DILITHIUM, CRYSTALS-KYBER, **DAGS**, Ding KEX, **DME**, DRS, DualModeMS, Edon-K, EMBLEM, FALCON, FrodoKEM, GeMSS, **Giophantus**, Gravity-SPHINCS, **Guess Again**, Gui, **HILAS**, HiMQ-3, **HK17**, HQC, KINDI, LAC, LAKE, LEDAkm, LEDApkc, Lepton, LIMA, Lizard, LOCKER, LOTUS, LUOV, McNie, Mersenne-756839, MQDSS, NewHope, NTRUEncrypt, NTRU-HRSS-KEMf, NTRU Prime, NTS-KEM, Odd Manhattan, OKCN/AKCN/CNKE, Ouroboros-R, Picnic, pqNTRUSign, 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**

2018.05.21: Of the 69 originally running candidates

▶ 14 have been **broken or withdrawn**.
▶ 7 are being **amended with tweaks**.
Classification of Candidates

- **Lattice based.** Based on problems analogous to quantum shortest vector and other lattice problems, known to be hard. About 26 (40%) of the candidates fall under “lattice cryptography” umbrella. These algorithms are generally fastest.

- **Code-Based.** These algorithms are based on coding theory problem. This is the second-largest set, 18 (28%) of candidates. McEliece (1978) is in this set.

- **Multi-Variate.** Based on systems of equations, often seen as rather *ad hoc*. There’s less than 10 in this set, many which have already been broken.

- **Hash-based.** There are only three hash based signature proposals in the competition, but some additional ones are already being standardized elsewhere. Often seen as having best security assurances for signatures.

- **Isogenies.** One candidate – SIKE – is based on isogeny problem of supersingular curves (ECC people love Isogeny systems because they use elliptic curves).

- **Other.** Some of the proposals were just.. *weird.*
Quantification of Post-Quantum Security

Five categories of NIST: “Any attack that breaks the relevant security definition must require computational resources comparable to or greater than those required for ...

1. ... key search on a block cipher with a 128-bit key (e.g. AES-128)"
2. ... collision search on a 256-bit hash function (e.g. SHA-256 / SHA3-256)"
3. ... key search on a block cipher with a 192-bit key (e.g. AES-192)"
4. ... collision search on a 384-bit hash function (e.g. SHA-384/ SHA3-384)"
5. ... key search on a block cipher with a 256-bit key (e.g. AES-256)"

Furthermore NIST suggests considering attacks “limited by MAXDEPTH” (analogous to constant runtime of attack < $2^{96}$) and estimating the number of quantum gates.

**Which gates? What is MAXDEPTH? What is time? What about reliability?**
So how hard is AES-256 to break with Quantum Computing?

Grover’s complexity is $O(2^{n/2})$ so rule of thumb estimates “128 quantum bit security”.
The cipher itself needs to implemented as an oracle; NIST states an estimate of $2^{298}/\text{MAXDEPTH}$ quantum gates for $2^{272}$ classical gate operations.

Most relevant attack technique against candidate X may depend on completely different features of quantum computing than a quantum circuit for breaking AES.
Visualization: Toy Grover on IBM’s 5-Qubit Quantum Computer

“Quantum score”:

![Quantum Circuit Diagram]

Standard gates and symbols:

**Identity gate (I)**: The identity gate performs an idle operation on the qubit for a time equal to the single-qubit gate duration.

**Pauli X gate (X)**: The Pauli X gate is a π rotation around the X axis and has the property that $X \rightarrow X$, $Z \rightarrow -Z$. Also referred to as a bit-flip.

**Pauli Z gate (Z)**: The Pauli Z gate is a π rotation around the Z axis and has the property that $X \rightarrow -X$, $Z \rightarrow Z$. Also referred to as a phase-flip.

**Pauli Y gate (Y)**: The Pauli Y gate is a π rotation around the Y axis and has the property that $X \rightarrow -X$, $Z \rightarrow -Z$. This is both a bit-flip and a phase-flip, and satisfies $Y = XZ$.

**Controlled-NOT gate (CNOT)**: A two-qubit gate that flips the target qubit (i.e., applies Pauli X) if the control is in state 1. This gate is required to generate entanglement.

**Hadamard gate (H)**: The Hadamard gate has the property that it maps $X \rightarrow Z$, and $Z \rightarrow X$. This gate is required to make superpositions.

**Phase gate (S)**: The Phase gate that is $\sqrt{Z}$ and has the property that it maps $X \rightarrow Y$ and $Z \rightarrow Z$. This gate extends $H$ to make complex superpositions.

**Phase gate (S†)**: The Phase gate that is the transposed conjugate of $S$. This gate extends $H$ to make complex superpositions.

**Phase gate (T)**: The Phase gate that is $\sqrt{S}$, which is a $\pi/4$ rotation around the Z axis. This gate is required for universal control.

**Phase gate (T†)**: The Phase gate that is the transposed conjugate of $T$.

**Measurement (†)**: Measurement in the computational (standard) basis ($Z$).

**Bloch measurement**: Tomography of the individual qubits.

Images from [CEP+18]. IBM Q has dev tools and up to 20 Qubit systems available on-line.
Quantum Complexity Estimates

If we’re so sure that RSA and Elliptic Curve cryptography can be broken, how come it is difficult to estimate Post-Quantum security?

- We know that Shor’s algorithm is polynomial on modulus size (or elliptic curve group size) \( n \). We want the best attacking algorithm to be superpolynomial.
- Proving optimality of algorithm is very, very difficult. Most submitters simply state that “no polynomial algorithm is known” for their underlying problem.
- Not all hard problems can be characterized by a single variable \( n \), making attack asymptotics for parameter selection little bit more difficult.

We note that many (or most) of the current breaks have been attacks with feasible classical complexity, or based on other gross design errors.

As in previous competitions, we expect new attack techniques to be developed.
PART 2: LWE and RLWE Algorithms

The largest single category of algorithms are those based on (Ring-)LWE Problem.

We will focus on these algorithms in the rest of this talk. I’ll use my proposal, HILA5, as an example on how they are actually designed and implemented.
General and Ideal Lattices

Lattice basis:

\[ \mathcal{L}(b_1, \ldots, b_n) = \sum_{i=1}^{n} x_i b_i \mid x_i \in \mathbb{Z} \]

If we replace \( b_2 \) with the shorter green vector, same lattice is generated. This is a **reduced basis**.

Shortest Vector Problem (**SVP**) is task of finding the shortest vector in a lattice. Related hard problems: Bounded Distance Decoding (**BDD**), Short Integer Solution (**SIS**), Learning With Errors (**LWE**)

In order to go from \( n^2 \) variables to \( n \) variables, one may use **ideal lattices**, which are defined in a finite cyclotomic rings. (Hence, “Ring-LWE”).
Let $\mathcal{R}$ be a ring with elements $v \in \mathbb{Z}_q^n$. We use fast NTT arithmetic in $\mathbb{Z}_q[x]/(x^n + 1)$.

**Definition (Informal)**
With all distributions and computations in ring $\mathcal{R}$, let $s, e$ be elements randomly chosen from some non-uniform distribution $\chi$, and $g$ be a uniformly random public value. Determining $s$ from $(g, g \ast s + e)$ in ring $\mathcal{R}$ is the (Normal Form Search) Ring Learning With Errors (RLWE $\mathcal{R}, \chi$) problem.

Typically $\chi$ is chosen so that each coefficient is a Discrete Gaussian or from some other “Bell-Shaped” distribution that is relatively tightly concentrated around zero.

The hardness of the problem is a function of $n, q,$ and $\chi$. HILA5 uses very fast and well-studied “New Hope” parameters: $n = 1024, q = 3 \times 2^{12} + 1 = 12289, \chi = \Psi_{16}$.
Quantum Complexity Estimation

The LWE problem was invented by Oded Regev [Re05,Re09] who showed its connection to worst case shortest vector problems in a quantum setting.

These ideas were extended to ring setting (RLWE) starting with [LPR10]. The connection between a uniform secret $s$ and a secret chosen from $\chi$ is provided by Applebaum et al. [ACP+09] for LWE case, and for the ring setting in [LPR13].

(R)LWE and NTRU parameter selection is typically based on estimating the complexity of the Schnorr-Euchner BKZ [SE94] algorithm in classical and quantum setting under a number of different cost models. I use Albrecht’s estimation scripts [ACD+18]: [https://estimate-all-the-lwe-ntru-schemes.github.io/docs/](https://estimate-all-the-lwe-ntru-schemes.github.io/docs/)

**WARNING:** There are a lot of cost models, some of which are purely hypothetical (i.e. algorithms that have not been invented yet), which has led to some angry exchanges in the mailing lists between Dan Bernstein and others.
Discrete Gaussian $D_{\sqrt{8}}$ vs. Binomial “bitcount” Distribution $\chi = \Psi_{16}$

“Sampling algorithm”: Count the 1 bits in a random 32-bit word and subtract 16.

Green bars are the probability mass of binomial distribution $P(X = x) = 2^{-32} \binom{32}{x+16}$.

Blue line is the discrete Gaussian distribution $D_\sigma$ with deviation parameter $\sigma = \sqrt{8}$.

$$\rho_\sigma(x) \propto \exp\left(-\frac{x^2}{2\sigma^2}\right).$$

Very good approximation: $\rho_\sigma(x) \approx \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$. 
### Noisy Diffie-Hellman in a Ring

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a \xleftarrow{} \chi$</td>
<td>$b \xleftarrow{} \chi$</td>
</tr>
<tr>
<td>$e \xleftarrow{} \chi$</td>
<td>$e' \xleftarrow{} \chi$</td>
</tr>
<tr>
<td>$A = g \ast a + e$</td>
<td>$B = g \ast b + e'$</td>
</tr>
<tr>
<td>$x = B \ast a$</td>
<td>$y = A \ast b$</td>
</tr>
</tbody>
</table>

Here $g$ is a uniform, public generator. By substituting variables in $A$ and $B$ we get

\[
\begin{align*}
    x &= (g \ast b + e') \ast a = g \ast a \ast b + e' \ast a \\
    y &= (g \ast a + e) \ast b = g \ast a \ast b + e \ast b.
\end{align*}
\]

Because error terms are much smaller than the common term $g \ast a \ast b$ we have $x \approx y$. 

---

Numbers: 22 / 29
Reconciliation: Traditionally Needs Random Numbers

In reconciliation, we wish the holders of \( x \) and \( y \) (Alice and Bob, respectively) to arrive at exactly the same shared secret \( k \) with minimal communication \( c \).

In Peikert’s reconciliation [Pe14] Bob sends 1 “phase bit” \( c \) for each vector element. Since \( q \) is odd and cannot be evenly divided in half, a fresh random bit is needed to “smoothen” the divide. New Hope’s reconciliation of also needs random numbers.
As we don't need $n = 1024$ bits, we can select “Safe Bits” away from the decision boundary in order to get unbiased secrets without using additional randomness.

We designed error correction codes to push the failure probability well under $2^{-128}$. 
Hey students! Pay attention in the coding theory classes!

I designed a linear block code, XE5, specifically for HILA5.

**Security Requirement**: Fast, constant-time implementable.

After various considerations (SafeBits), ended up with a block size of 496 bits (256 message bits + 240 redundancy bits).

Always corrects 5 random bit flips, more with high probability.

I first described similar constant-time error correction techniques (for TRUNC8) in:


[https://eprint.iacr.org/2016/1058](https://eprint.iacr.org/2016/1058) (Original uploaded November 15, 2016)
Pindakaas: HILA5 is IND-CPA, not IND-CCA


There is a single point on p. 17 of the HILA5 specification which erroneously claims IND-CCA security. With (too) much speculation this was shown not to be correct in [BBLP17]. The original SAC 2017 academic paper never even mentions IND-CCA.

Furthermore even [BBLP17] itself clearly states that:

“We emphasize that our attack does not break the IND-CPA security of HILA5. If HILA5 were clearly labeled as aiming merely for IND-CPA security then our attack would merely be a cautionary note, showing the importance of not reusing keys.”

Creating an IND-CCA variant via Fujisaki–Okamoto transform is straightforward. I will propose such variant, probably not dissimilar to “HILA5FO” from [BBLP17].
What Distinguishes HILA5 from the Rest?

+ **It’s Very Fast and can do KEM and Public Key Encryption.** Only about 5% slower than fastest New Hope (CPA) implementation (Matching Ring-LWE).

+ **Less randomness required.** Reconciliation method produces unbiased secrets without randomized smoothing; much less randomness is therefore required.

+ **HILA5 decryption doesn’t fail.** HILA5 has a failure rate well under $2^{-128}$. Non-negligible decryption failure rate is needed in public key encryption.

+ **Non-malleable.** Computation of the final shared secret in HILA5 KEM uses the full public key and ciphertext messages, thereby reinforcing non-malleability and making a class of adaptive attacks infeasible.

+ **Shorter messages.** Ciphertext messages are slightly smaller than New Hope’s.

+ **Patent free.** As the sender can “choose the message” (as in NewHope-Simple), Ding’s Ring-LWE key exchange patents less likely to be applicable.
Algorithm Purpose: Key Encapsulation and Public Key Encryption.
Underlying problem: Ring-LWE (New Hope: \( n = 1024, q = 12289, \Psi_{16} \))
Public key size: 1824 Bytes (+32 Byte private key hash).
Private key size: 1792 Bytes (640 Bytes compressed).
Ciphertext size: 2012 Byte expansion (KEM) + payload + MAC.
Failure rate: \(< 2^{-128}\), consistent with security level.
Classical security: \(2^{256}\) (Category 5 – Equivalent to AES-256).
Quantum security: \(2^{128}\) (Category 5 – Equivalent to AES-256).

Questions?

NIST Post-Quantum Cryptography Project:  
https://csrc.nist.gov/Projects/Post-Quantum-Cryptography

PQC Forum Mailing list:  
https://groups.google.com/a/list.nist.gov/forum/#!forum/pqc-forum

HILA5 Stuff:  