Code-Based Cryptography

Tanja Lange

Eindhoven University of Technology

14 February 2019
Cryptography

- Motivation #1: Communication channels are spying on our data.
- Motivation #2: Communication channels are modifying our data.
Cryptography

- Motivation #1: Communication channels are spying on our data.
- Motivation #2: Communication channels are modifying our data.
Cryptography

- Motivation #1: Communication channels are spying on our data.
- Motivation #2: Communication channels are modifying our data.

Sender “Alice” → Untrustworthy network “Eve” → Receiver “Bob”

- Literal meaning of cryptography: “secret writing”.
- Security goal #1: **Confidentiality** despite Eve’s espionage.
- Security goal #2: **Integrity**, i.e., recognizing Eve’s sabotage.
Symmetric cryptography

- Alice and Bob share a secret key.
- They use this key for encryption: both parties can encrypt and decrypt.
  - Stream ciphers encrypt streams of bits: Salsa20, ChaCha20, (RC4), ...
  - Block ciphers encrypt messages of fixed length: AES, Serpent, (DES), ...
    Longer messages are encrypted using modes of operations to chain the blocks: CBC, CTR, ...
- They use this key for authentication and integrity protection: each party is convinced that a message comes from the respective other party and that it has not been modified.
  - Message authentication codes (MACs) add such a checksum: GCM, HMAC, Poly1305, ...
- Typically a combination is needed, e.g., AES-GCM, ChaCha20-Poly1305, ...
- Hash functions map strings of arbitrary length to strings of fixed length. Even though there is no secret they are typically considered part of symmetric cryptography.
Public key cryptography

- Alice a pair of keys: her public key and her private key.
- The key parts are linked by some mathematical function so that computing the private key from the public key should be hard.
- Anybody can see and use Alice’s public key (Bob, Charlie, Eve, . . .)
- Only Alice knows her private key.
- Anybody can use Alice’s public key to encrypt to her; only she can decrypt (using the private key).
  - Messages satisfy some mathematical properties, e.g., integer < n.
    - point on an elliptic curve, . . .
    - Examples are RSA, Diffie-Hellman in finite fields, ECDH, . . .
- Alice uses her private key to sign a message; anybody can verify the signature using her public key.
  - Signatures ensure authenticity and integrity: anybody is convinced that the message comes from Alice and that it has not been modified.
  - Examples are RSA, DSA, ECDSA.
Security assumptions

- Hardness assumptions at the basis of all public-key and essentially all symmetric-key systems result from (failed) attempts at breaking systems. Security proofs are built only on top of those assumptions.
- A solid symmetric system is required to be as strong as exhaustive key search.
- For public-key systems the best attacks are faster than exhaustive key search. Parameters are chosen to ensure that the best attack is infeasible.
## Key size recommendations

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Legacy</th>
<th>Near Term</th>
<th>Long Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symmetric Key Size</td>
<td>$k$</td>
<td>80</td>
<td>128</td>
</tr>
<tr>
<td>Hash Function Output Size</td>
<td>$m$</td>
<td>160</td>
<td>256</td>
</tr>
<tr>
<td>MAC Output Size*</td>
<td>$m$</td>
<td>80</td>
<td>128</td>
</tr>
<tr>
<td>RSA Problem</td>
<td>$\ell(n) \geq$</td>
<td>1024</td>
<td>3072</td>
</tr>
<tr>
<td>Finite Field DLP</td>
<td>$\ell(p^n) \geq$</td>
<td>1024</td>
<td>3072</td>
</tr>
<tr>
<td></td>
<td>$\ell(p), \ell(q) \geq$</td>
<td>160</td>
<td>256</td>
</tr>
<tr>
<td>ECDLP</td>
<td>$\ell(q) \geq$</td>
<td>160</td>
<td>256</td>
</tr>
</tbody>
</table>

- Hardness assumptions at the basis of all public-key and essentially all symmetric-key systems result from (failed) attack attempts. Security proofs are built only on top of those assumptions.
- A solid symmetric system is required to be as strong as exhaustive key search.
- For public-key systems the best attacks are faster than exhaustive key search. Parameters are chosen to ensure that the best attack known today is infeasible.
- Attacker power limited to $2^{128}$ operations ($2^{80}$ for legacy).
Algorithms for Quantum Computation: Discrete Logarithms and Factoring

Peter W. Shor
AT&T Bell Labs
Room 2D-149
600 Mountain Ave.
Murray Hill, NJ 07974, USA

Abstract

A computer is generally considered to be a universal computational device; i.e., it is believed able to simulate any physical computational device with a cost in computation time of at most a polynomial factor. It is not clear whether this is still true when quantum mechanics is taken into consideration. Several researchers, starting with David Deutsch, have developed models for quantum mechanical computers and have investigated their computational properties. This paper gives Las Vegas algorithms for finding discrete logarithms and factoring integers on a quantum computer that take a number of steps which is polynomial in the input size, e.g., the number of digits of the integer to be factored. These two problems are generally considered hard on a classical computer and have been used as the basis of several proposed cryptosystems. (We thus give the first examples of quantum cryptanalysis.)

[1, 2]. Although he did not ask whether quantum mechanics conferred extra power to computation, he did show that a Turing machine could be simulated by the reversible unitary evolution of a quantum process, which is a necessary prerequisite for quantum computation. Deutsch [9, 10] was the first to give an explicit model of quantum computation. He defined both quantum Turing machines and quantum circuits and investigated some of their properties.

The next part of this paper discusses how quantum computation relates to classical complexity classes. We will thus first give a brief intuitive discussion of complexity classes for those readers who do not have this background. There are generally two resources which limit the ability of computers to solve large problems: time and space (i.e., memory). The field of analysis of algorithms considers the asymptotic demands that algorithms make for these resources as a function of the problem size. Theoretical computer scientists generally classify algorithms as efficient when the number of steps of the algorithms grows as a polynomial in the size of the input. The class of prob-
Cryptography

- Motivation #1: Communication channels are spying on our data.
- Motivation #2: Communication channels are modifying our data.

Sender
“Alice” → Untrustworthy network
“Eve” → Receiver
“Bob”

- Literal meaning of cryptography: “secret writing”.
- Security goal #1: **Confidentiality** despite Eve’s espionage.
- Security goal #2: **Integrity**, i.e., recognizing Eve’s sabotage.
Post-quantum cryptography

- Motivation #1: Communication channels are spying on our data.
- Motivation #2: Communication channels are modifying our data.

- Literal meaning of cryptography: “secret writing”.
- Security goal #1: **Confidentiality** despite Eve’s espionage.
- Security goal #2: **Integrity**, i.e., recognizing Eve’s sabotage.
- Post-quantum cryptography adds to the model that Eve has a quantum computer.
Post-quantum cryptography: Cryptography designed under the assumption that the attacker (not the user!) has a large quantum computer.
Effects of large universal quantum computers

- Massive research effort. Tons of progress summarized in, e.g.,
- Mark Ketchen, IBM Research, 2012, on quantum computing:
  “We’re actually doing things that are making us think like, ‘hey this
  isn’t 50 years off, this is maybe just 10 years off, or 15 years off.’ It’s
  within reach.”
- Fast-forward to 2022, or 2027. Universal quantum computers exist.
- Shor’s algorithm solves in polynomial time:
  - Integer factorization. RSA is dead.
  - The discrete-logarithm problem in finite fields. DH is dead.
  - The discrete-logarithm problem on elliptic curves. ECC is dead.
- This breaks all current public-key cryptography on the Internet!
Effects of large universal quantum computers

- Massive research effort. Tons of progress summarized in, e.g.,
- Mark Ketchen, IBM Research, 2012, on quantum computing:
  “We’re actually doing things that are making us think like, ‘hey this
  isn’t 50 years off, this is maybe just 10 years off, or 15 years off.’ It’s
  within reach.”
- Fast-forward to 2022, or 2027. Universal quantum computers exist.
- Shor’s algorithm solves in polynomial time:
  - Integer factorization.
  - The discrete-logarithm problem in finite fields.
  - The discrete-logarithm problem on elliptic curves.
  - RSA is dead.
  - DH is dead.
  - ECC is dead.
- This breaks all current public-key cryptography on the Internet!
- Also, Grover’s algorithm speeds up brute-force searches.
- Example: Only $2^{64}$ quantum operations to break AES-128;
  $2^{128}$ quantum operations to break AES-256.
4 December 2018: 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.”
4 December 2018: 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.”
Systems expected to survive

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

This list is based on the best known attacks (as always).

These are categories of mathematical problems; individual systems may be totally insecure if the problem is not used correctly.
Short summaries

- Hash-based signatures: very solid security and small public keys. Require only a secure hash function (hard to find second preimages).
- Isogeny-based encryption: new kid on the block, promising short keys and ciphertexts and non-interactive key exchange. Systems rely on hardness of finding isogenies between elliptic curves over finite fields.
- Lattice-based encryption and signatures: possibility for balanced sizes. Security relies on finding short vectors in some (typically special) lattice.

1978 McEliece: Use Goppa codes for public-key crypto.
   - Original parameters designed for $2^{64}$ security.
   - 2008 Bernstein–Lange–Peters: broken in $\approx 2^{60}$ cycles.
   - Easily scale up for higher security.

1986 Niederreiter: Simplified and smaller version of McEliece.


The McEliece system (with later key-size optimizations)
uses $(c_0 + o(1))\lambda^2(\lg \lambda)^2$-bit keys as $\lambda \to \infty$
to achieve $2^\lambda$ security against Prange’s attack.
Here $c_0 \approx 0.7418860694$. 

"Code-based encryption"
Security analysis

Some papers studying algorithms for attackers:
Chabanne–Courteau; 1993 Chabaud; 1994 van Tilburg; 1994
Canteaut–Chabanne; 1998 Canteaut–Chabaud; 1998 Canteaut–Sendrier; 2008
Bernstein–Lange–Peters; 2009 Bernstein–Lange–Peters–van Tilborg; 2009
Bernstein (post-quantum); 2009 Finiasz–Sendrier; 2010
Bernstein–Lange–Peters; 2011 May–Meurer–Thomae; 2012
Canto Torres–Sendrier; 2017 Kachigar–Tillich (post-quantum); 2017
Both–May; 2018 Both–May; 2018 Kirshanova (post-quantum).
The McEliece system (with later key-size optimizations) uses $(c_0 + o(1))\lambda^2(\lg \lambda)^2$-bit keys as $\lambda \to \infty$ to achieve $2^\lambda$ security against all these attacks.
Consequence of security analysis

- The McEliece system (with later key-size optimizations) uses \((c_0 + o(1))\lambda^2(\lg \lambda)^2\)-bit keys as \(\lambda \to \infty\) to achieve \(2^\lambda\) security against all these attacks. Here \(c_0 \approx 0.7418860694\).
- 256 KB public key for \(2^{146}\) pre-quantum security.
- 512 KB public key for \(2^{187}\) pre-quantum security.
- 1024 KB public key for \(2^{263}\) pre-quantum security.
The McEliece system (with later key-size optimizations) uses \((c_0 + o(1))\lambda^2(\lg \lambda)^2\)-bit keys as \(\lambda \to \infty\) to achieve \(2^\lambda\) security against all these attacks. Here \(c_0 \approx 0.7418860694\).

- 256 KB public key for \(2^{146}\) pre-quantum security.
- 512 KB public key for \(2^{187}\) pre-quantum security.
- 1024 KB public key for \(2^{263}\) pre-quantum security.
- Post-quantum (Grover): below \(2^{263}\), above \(2^{131}\).
Let $C$ be a length-$n$ binary Goppa code $\Gamma$ of dimension $k$ with minimum distance $2t + 1$ where $t \approx (n - k)/\log_2(n)$; original parameters (1978) $n = 1024$, $k = 524$, $t = 50$.

The McEliece secret key consists of a generator matrix $G$ for $\Gamma$, an efficient $t$-error correcting decoding algorithm for $\Gamma$; an $n \times n$ permutation matrix $P$ and a nonsingular $k \times k$ matrix $S$.

$n, k, t$ are public; but $\Gamma$, $P$, $S$ are randomly generated secrets.

The McEliece public key is the $k \times n$ matrix $G' = SGP$. 

The McEliece cryptosystem I
Encrypt: Compute $mG'$ and add a random error vector $e$ of weight $t$ and length $n$. Send $y = mG' + e$.

Decrypt: Compute $yP^{-1} = mG'P^{-1} + eP^{-1} = (mS)G + eP^{-1}$. This works because $eP^{-1}$ has the same weight as $e$. Use fast decoding to find $mS$ and $m$.
The McEliece cryptosystem II

- **Encrypt:** Compute $mG'$ and add a random error vector $e$ of weight $t$ and length $n$. Send $y = mG' + e$.

- **Decrypt:** Compute $yP^{-1} = mG'P^{-1} + eP^{-1} = (mS)G + eP^{-1}$. This works because $eP^{-1}$ has the same weight as $e$ because $P$ is a permutation matrix. Use fast decoding to find $mS$ and $m$.

- Attacker is faced with decoding $y$ to nearest codeword $mG'$ in the code generated by $G'$. This is general decoding if $G'$ does not expose any structure.

- Wrote attack software against original McEliece parameters, decoding 50 errors in a $[1024, 524]$ code.

- Attack on a single computer with a 2.4GHz Intel Core 2 Quad Q6600 CPU would need, on average, 1400 days ($2^{58}$ CPU cycles) to complete the attack.

- Running the software on 200 such computers would reduce the average time to one week.
The Niederreiter cryptosystem I

Developed in 1986 by Harald Niederreiter as a variant of the McEliece cryptosystem. This is the schoolbook version.

- Use parity-check matrix $H$ instead of generator matrix.
- Use $n \times n$ permutation matrix $P$, $n - k \times n - k$ invertible matrix $S$.
- Public Key: a scrambled parity-check matrix $K = SHP \in \mathbb{F}_2^{(n-k)\times n}$.
- Encryption: The plaintext $e$ is an $n$-bit vector of weight $t$. The ciphertext $s$ is the $(n - k)$-bit vector

\[ s = Ke. \]

- Decryption: Find a $n$-bit vector $e$ with $\text{wt}(e) = t$ such that $s = Ke$.
- The passive attacker is facing a $t$-error correcting problem for the public key, which seems to be random.
The Niederreiter cryptosystem II

- **Public Key:** a scrambled parity-check matrix $K = SHP$.

- **Encryption:** The plaintext $e$ is an $n$-bit vector of weight $t$. The ciphertext $s$ is the $(n - k)$-bit vector

  $$ s = Ke. $$

- **Decryption using secret key:** Compute

  $$ S^{-1}s = S^{-1}Ke = S^{-1}(SHP)e = H(Pe) $$

  and observe that $\text{wt}(Pe) = t$, because $P$ permutes.

- **Use efficient syndrome decoder for $H$ to find** $e' = Pe$ **and thus** $e = P^{-1}e'$. 

---

Tanja Lange Code-Based Cryptography
Note on codes

- McEliece proposed to use binary Goppa codes. These are still used today.
- Niederreiter described his scheme using Reed-Solomon codes. These were broken in 1992 by Sidelnikov and Chestakov.
- More corpses on the way: concatenated codes, Reed-Muller codes, several Algebraic Geometry (AG) codes, Gabidulin codes, several LDPC codes, cyclic codes.
- Some other constructions look OK (for now). NIST competition has several entries on QCMDPC and rank-metric codes.
How to hide nice code?

- Let $q = 2^m$. A binary Goppa code is defined by
  - a list $L = (a_1, \ldots, a_n)$ of $n$ distinct elements in $\mathbb{F}_q$, called the support.
  - a square-free polynomial $g(x) \in \mathbb{F}_q[x]$ of degree $t$ such that $g(a) \neq 0$ for all $a \in L$. $g(x)$ is called the Goppa polynomial.
  - E.g. choose $g(x)$ irreducible over $\mathbb{F}_q$. 

E.g. choose $g(x)$ irreducible over $\mathbb{F}_q$. 

Then, the Goppa code is defined by the following properties:

- The code $C$ is a linear code over $\mathbb{F}_q$.
- The generator matrix $G$ of $C$ is given by:
  
  $G = \begin{pmatrix}
  t \cdot \text{Perm}(L) & 0 \\
  \cdot & \cdot \\
  \cdot & \cdot
  \end{pmatrix}
  $ 

  where $\text{Perm}(L)$ is the permutation matrix corresponding to the support $L$.

- The Goppa polynomial $g(x)$ is used to construct the code words:
  
  $c_i = g(a_i) \cdot \text{Perm}(L)$

- The systematic form of the code $C$ is given by $K = (K', I)$, where $K'$ is the matrix containing the first $t$ columns of $G$ and $I$ is the identity matrix of size $n - t$.

- The decoding process involves solving the following system of equations:
  
  $G \cdot x = 0$ 

  where $x$ is the error vector.

- The secret key is $(g, L)$, and the secret permutation of the $a_i$ corresponds to a secret permutation of the $n$ positions.

- The public key size is decreased to $(n - k) \times k$.

This approach allows for secure communication by hiding the structure of the code, making it more difficult for an attacker to identify and exploit any patterns.
How to hide nice code?

- Let $q = 2^m$. A binary Goppa code is defined by
  - a list $L = (a_1, \ldots, a_n)$ of $n$ distinct elements in $\mathbb{F}_q$, called the support.
  - a square-free polynomial $g(x) \in \mathbb{F}_q[x]$ of degree $t$ such that $g(a) \neq 0$ for all $a \in L$. $g(x)$ is called the Goppa polynomial.
  - E.g. choose $g(x)$ irreducible over $\mathbb{F}_q$.

- Use secret $g(x)$ and secret permutation of the $a_i$, this corresponds to secret permutation of the $n$ positions; this replaces $P$.

- Secret key is polynomial $g$ and support $L = (a_1, \ldots, a_n)$. Can generate both by expanding a small seed.

- Use systematic form $K = (K'|I)$ for key;
  - This implicitly applies $S$.
  - No need to remember $S$ because decoding does not use $H$.
  - Public key size decreased to $(n - k) \times k$. 

Tanja Lange  Code-Based Cryptography  23
Do not use the schoolbook versions!
Assume that the decoding algorithm decodes up to $t$ errors, i.e., it decodes $y = c + e$ to $c$ if $wt(e) \leq t$.

Eve intercepts ciphertext $y = mG' + e$. Eve poses as Alice towards Bob and sends him tweaks of $y$. She uses Bob’s reactions (success of failure to decrypt) to recover $m$.

Assume $wt(e) = t$. (Else flip more bits till Bob fails).

Eve sends $y_i = y + e_i$ for $e_i$ the $i$-th unit vector. If Bob returns error, position $i$ in $e$ is 0 (so the number of errors has increased to $t + 1$ and Bob fails). Else position $i$ in $e$ is 1.

After $k$ steps Eve knows the first $k$ positions of $mG'$ without error. Invert the $k \times k$ submatrix of $G'$ to get $m$.
Sloppy Alice attacks! 1998 Verheul, Doumen, van Tilborg

- Assume that the decoding algorithm decodes up to \( t \) errors, i.e. it decodes \( \mathbf{y} = \mathbf{c} + \mathbf{e} \) to \( \mathbf{c} \) if \( \text{wt}(\mathbf{e}) \leq t \).
- Eve intercepts ciphertext \( \mathbf{y} = \mathbf{mG'} + \mathbf{e} \).
  Eve poses as Alice towards Bob and sends him tweaks of \( \mathbf{y} \).
  She uses Bob’s reactions (success of failure to decrypt) to recover \( \mathbf{m} \).
- Assume \( \text{wt}(\mathbf{e}) = t \). (Else flip more bits till Bob fails).
- Eve sends \( \mathbf{y}_i = \mathbf{y} + \mathbf{e}_i \) for \( \mathbf{e}_i \) the \( i \)-th unit vector.
  If Bob returns error, position \( i \) in \( \mathbf{e} \) is 0 (so the number of errors has increased to \( t + 1 \) and Bob fails).
  Else position \( i \) in \( \mathbf{e} \) is 1.
- After \( k \) steps Eve knows the first \( k \) positions of \( \mathbf{mG'} \) without error. Invert the \( k \times k \) submatrix of \( G' \) to get \( \mathbf{m} \) assuming it is invertible.
- Proper attack: figure out invertible submatrix of \( G' \) at beginning; recover matching \( k \) coordinates.
- This attack has Eve send Bob variations of the same ciphertext; so Bob will think that Alice is sloppy.
Towards non-schoolbook version

- Attack also works on Niederreiter version:

  Bitflip corresponds to sending $s_i = s_i + K_i$, where $K_i$ is the $i$-th column of $K$. More involved but doable (for McEliece and Niederreiter) if decryption requires exactly $t$ errors.

  Fix by using CCA2 transformation (e.g. Fujisaki-Okamoto transform) or (easier) KEM/DEM version: pick random $e$ of weight $t$, use hash($e$) as secret key to encrypt and authenticate (for McEliece or Niederreiter).

  Can prove security under the assumption that McEliece/Niederreiter are One-Way Encryption (OWE) schemes.
Towards non-schoolbook version

- Attack also works on Niederreiter version:
  Bitflip corresponds to sending $s_i = s + K_i$, where $K_i$ is the $i$-th column of $K$.

- More involved but doable (for McEliece and Niederreiter) if decryption requires exactly $t$ errors.

- Fix by using CCA2 transformation (e.g. Fujisaki-Okamoto transform) or (easier) KEM/DEM version:
  pick random $e$ of weight $t$, use hash($e$) as secret key to encrypt and authenticate (for McEliece or Niederreiter).

- Can prove security under the assumption that McEliece/Niederreiter are One-Way Encryption (OWE) schemes.
Classic McEliece highlights

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

<table>
<thead>
<tr>
<th>Metric</th>
<th>mceliece6960119</th>
<th>mceliece8192128</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public-key size</td>
<td>1047319 bytes</td>
<td>1357824 bytes</td>
</tr>
<tr>
<td>Secret-key size</td>
<td>13908 bytes</td>
<td>14080 bytes</td>
</tr>
<tr>
<td>Ciphertext size</td>
<td>226 bytes</td>
<td>240 bytes</td>
</tr>
<tr>
<td>Key-generation time</td>
<td>1108833108 cycles</td>
<td>1173074192 cycles</td>
</tr>
<tr>
<td>Encapsulation time</td>
<td>153940 cycles</td>
<td>188520 cycles</td>
</tr>
<tr>
<td>Decapsulation time</td>
<td>318088 cycles</td>
<td>343756 cycles</td>
</tr>
</tbody>
</table>

See [https://classic.mceliece.org](https://classic.mceliece.org) for more details.
March 7-8, 2019 @TUe, Eindhoven

Please join us in Eindhoven next March to celebrate Ruud's contribution to the fields of Coding Theory, Cryptology and Curves (Algebraic Geometry). We encourage you to submit your contribution to this conference within those or related topics. A Special Issue of the journal Designs, Codes and Cryptography is being scheduled also on those topics in Honor of Ruud Pellikaan, please check this link for further information.

http://www.singacom.uva.es/~edgar/CCC/index.html
We’re hiring!

Werken bij

**TT) Assistant Professor Coding Theory (V32.3695)**

The Coding Theory and Cryptology (CC) group of the Discrete Mathematics (DM) section of the Department of Mathematics and Computer Science (M&CS) at Eindhoven University of Technology (TU/e) is looking for a research leader.

Links and more upcoming events

- 18 & 19 May 7th Code-Based Cryptography Workshop
- 1 & 2 July 2019: Executive summer school on post-quantum cryptography in Eindhoven.
- [https://pqcrypto.eu.org](https://pqcrypto.eu.org): PQCRYPTO EU project.
  - Expert recommendations.
  - Free software libraries ([libpqcrypto](https://pqcrypto.eu.org), [pqm4](https://pqcrypto.eu.org), [pqhw](https://pqcrypto.eu.org)).
  - Lots of reports, scientific papers, (overview) presentations.
- PQCrypto 2018 & PQCrypto 2017 conferences.
- PQCrypto 2016 with slides and videos from lectures + school.
- [https://pqcrypto.org](https://pqcrypto.org): Survey site by D.J. Bernstein and me.
  - Many pointers: e.g., PQCrypto conference series.
  - Bibliography for 4 major PQC systems.