Introduction to post-quantum cryptography

Tanja Lange

Eindhoven University of Technology

4 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.

Key-encapsulation mechanisms

- Use public-key crypto to obtain shared key for symmetric crypto, then use that key to encrypt the message (KEM-DEM methodology).
- Encapsulation takes public key and randomness to generate KEM ciphertext c and a secret key k.
  Decapsulation takes private key and c to compute k.
- Example using RSA:
  - Public key is (n, e), private key is (n, d) with $ed \equiv 1 \bmod \phi(n)$.
  - Encapsulation:
    Pick random integer $m < n$, compute $c \equiv m^e \bmod n$, $k = \mathrm{hash}(m)$.
  - Decapsulation: Compute $m \equiv c^d \bmod n$, $k = \mathrm{hash}(m)$.
- For contrast:
  Key exchange takes two public keys and produces shared secret key.
- Can build KEM from KE by using one-time public key at sender side.
- Can not necessarily build KE from KEM, at least not non-interactive.
- Example: Diffie-Hellman key exchange in $\langle g \rangle < \mathbf{F}_p^*$:
  Alice posts $A = g^a$, Bob posts $B = g^b$;
  they share $\mathrm{hash}(A^b) = \mathrm{hash}(B^a)$.

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

	Parameter	Legacy	Future System Use	
			Near Term	Long Term
Symmetric Key Size	k	80	128	256
Hash Function Output Size	m	160	256	512
MAC Output Size*	m	80	128	256
RSA Problem	$\ell(n) \geq$	1024	3072	15360
Finite Field DLP	$\ell(p^n) \geq$	1024	3072	15360
	$\ell(p), \ell(q) \geq$	160	256	512
ECDLP	$\ell(q) \geq$	160	256	512

- 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).
- Source: ECRYPT-CSA "Algorithms, Key Size and Protocols Report"

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.

Sender
"Alice"

"Eve"
with a quantum computer

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 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

- See Mike Mosca's talk for details; this is a quick preview.
- Lots of active development on building quantum computers.
- Quantum computers will have a huge effect on public-key cryptography.
- Shor's algorithm solves:
  - Integer factorization. RSA is dead.
  - The discrete-logarithm problem in finite fields. DSA is dead.
  - The discrete-logarithm problem on elliptic curves. ECDHE is dead.
- This breaks all current public-key cryptography on the Internet!

Effects of large universal quantum computers

- See Mike Mosca's talk for details; this is a quick preview.
- Lots of active development on building quantum computers.
- Quantum computers will have a huge effect on public-key cryptography.
- Shor's algorithm solves:
  - Integer factorization. RSA is dead.
  - The discrete-logarithm problem in finite fields. DSA is dead.
  - The discrete-logarithm problem on elliptic curves. ECDHE 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.

National Academy of Sciences (US)

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."

National Academy of Sciences (US)

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."

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.

NIST round two

30 January 2019: 26 candidates retained for second round.

. BIKE. . Classic McEliece. .
CRYSTALS-DILITHIUM. CRYSTALS-KYBER. .
. . . . .
. FALCON. FrodoKEM. GeMSS. .
. . . HILA5. . . HQC.
. LAC. LAKE. LEDAkem. LEDApkc. . . .
LOCKER. . LUOV. . . MQDSS.
NewHope. NTRUEncrypt. . NTRU-HRSS-KEM. NTRU
Prime. NTS-KEM. . .
Ouroboros-R. Picnic. . . .
. qTESLA. . Rainbow. . .
. Round2. RQC. . SABER. SIKE. SPHINCS+. .
Three Bears. . .

Some less security than claimed; some really broken; some attack scripts.
Merges: HILA5 & Round2; LAKE, LOCKER, & Ouroboros-R; LEDAkem & LEDApkc; NTRUEncrypt &NTRU-HRSS-KEM.

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.

Target of this workshop: figure out what we really can do with a quantum computer.

Short summaries

- Code-based encryption: short ciphertexts and large public keys. More in a moment.
- 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, see talk by Lorenz.
- Lattice-based encryption and signatures: possibility for balanced sizes. Security relies on finding short vectors in some (typically special) lattice; see talk by Elena.
- Multivariate-quadratic signatures: short signatures and large public keys. Systems rely on hardness of solving systems of multi-variate equations over finite fields.

Linear codes

A binary linear code C of length n and dimension k is a k-dimensional subspace of $\mathbf{F}_2^n$.
C is usually specified as

- the row space of a generating matrix $G \in \mathbf{F}_2^{k \times n}$

$$C = \{\mathbf{m}G | \mathbf{m} \in \mathbf{F}_2^k\}$$

- the kernel space of a parity-check matrix $H \in \mathbf{F}_2^{(n-k) \times n}$

$$C = \{\mathbf{c} | H\mathbf{c}^\intercal = 0, \ \ \mathbf{c} \in \mathbf{F}_2^n\}$$

  Leaving out the $^\intercal$ from now on.
- A systematic generator matrix is a generator matrix of the form $(I_k|Q)$ where I_k is the $k \times k$ identity matrix and Q is a $k \times (n-k)$ matrix (redundant part).
- Easy to get parity-check matrix from systematic generator matrix, use $H = (Q^\intercal | I_{n-k})$.

Linear codes

A binary linear code C of length n and dimension k is a k-dimensional subspace of $\mathbf{F}_2^n$.
C is usually specified as

- the row space of a generating matrix $G \in \mathbf{F}_2^{k \times n}$

$$C = \{\mathbf{m}G | \mathbf{m} \in \mathbf{F}_2^k\}$$

- the kernel space of a parity-check matrix $H \in \mathbf{F}_2^{(n-k) \times n}$

$$C = \{\mathbf{c} | H\mathbf{c}^\intercal = 0, \ \ \mathbf{c} \in \mathbf{F}_2^n\}$$

  Leaving out the $^\intercal$ from now on.
- A systematic generator matrix is a generator matrix of the form $(I_k|Q)$ where I_k is the $k \times k$ identity matrix and Q is a $k \times (n-k)$ matrix (redundant part).
- Easy to get parity-check matrix from systematic generator matrix, use $H = (Q^\intercal | I_{n-k})$.
  Then

$$H(\mathbf{m}G)^\intercal = HG^\intercal \mathbf{m}^\intercal = (Q^\intercal | I_{n-k})(I_k|Q)^\intercal \mathbf{m}^\intercal = 0.$$

Hamming weight and distance

- The Hamming weight of a word is the number of nonzero coordinates.
- The Hamming distance between two words in $\mathbf{F}_2^n$ is the number of coordinates in which they differ.
  The Hamming distance between $\mathbf{x}$ and $\mathbf{y}$ equals the Hamming weight of $\mathbf{x} + \mathbf{y}$.
- The minimum distance of a linear code C is the smallest Hamming weight of a nonzero codeword in C.

$$d = \min_{0 \neq \mathbf{c} \in C} \{\mathrm{wt}(\mathbf{c})\} = \min_{\mathbf{b} \neq \mathbf{c} \in C} \{d(\mathbf{b}, \mathbf{c})\}$$

- In code with minimum distance $d = 2t + 1$, any vector $\mathbf{x} = \mathbf{c} + \mathbf{e}$ with $\mathrm{wt}(\mathbf{e}) \leq t$ is uniquely decodable to $\mathbf{c}$;
  i. e. there is no closer code word.

Decoding problem

Decoding problem: find the closest codeword $\mathbf{c} \in C$ to a given $\mathbf{x} \in \mathbf{F}_2^n$, assuming that there is a unique closest codeword. Let $\mathbf{x} = \mathbf{c} + \mathbf{e}$. Note that finding $\mathbf{e}$ is an equivalent problem.

- If $\mathbf{c}$ is t errors away from $\mathbf{x}$, i.e., the Hamming weight of $\mathbf{e}$ is t, this is called a t-error correcting problem.
- There are lots of code families with fast decoding algorithms, e.g., Reed–Solomon codes, Goppa codes/alternant codes, etc.
- However, the general decoding problem is hard: Information-set decoding (see later) takes exponential time.

Different views on decoding

- The syndrome of $\mathbf{x} \in \mathbf{F}_2^n$ is $\mathbf{s} = H\mathbf{x}$.
  Note $H\mathbf{x} = H(\mathbf{c} + \mathbf{e}) = H\mathbf{c} + H\mathbf{e} = H\mathbf{e}$ depends only on $\mathbf{e}$.
- The syndrome decoding problem is to compute $\mathbf{e} \in \mathbf{F}_2^n$ given $\mathbf{s} \in \mathbf{F}_2^{n-k}$ so that $H\mathbf{e} = \mathbf{s}$ and $\mathbf{e}$ has minimal weight.
- Syndrome decoding and (regular) decoding are equivalent:

Different views on decoding

- The syndrome of $\mathbf{x} \in \mathbf{F}_2^n$ is $\mathbf{s} = H\mathbf{x}$.
  Note $H\mathbf{x} = H(\mathbf{c} + \mathbf{e}) = H\mathbf{c} + H\mathbf{e} = H\mathbf{e}$ depends only on $\mathbf{e}$.
- The syndrome decoding problem is to compute $\mathbf{e} \in \mathbf{F}_2^n$ given $\mathbf{s} \in \mathbf{F}_2^{n-k}$ so that $H\mathbf{e} = \mathbf{s}$ and $\mathbf{e}$ has minimal weight.
- Syndrome decoding and (regular) decoding are equivalent:
  To decode $\mathbf{x}$ with syndrome decoder, compute $\mathbf{e}$ from $H\mathbf{x}$, then $\mathbf{c} = \mathbf{x} + \mathbf{e}$.
  To expand syndrome, assume $H = (Q^\intercal | I_{n-k})$.

Different views on decoding

- The syndrome of $\mathbf{x} \in \mathbf{F}_2^n$ is $\mathbf{s} = H\mathbf{x}$.
  Note $H\mathbf{x} = H(\mathbf{c} + \mathbf{e}) = H\mathbf{c} + H\mathbf{e} = H\mathbf{e}$ depends only on $\mathbf{e}$.
- The syndrome decoding problem is to compute $\mathbf{e} \in \mathbf{F}_2^n$ given $\mathbf{s} \in \mathbf{F}_2^{n-k}$ so that $H\mathbf{e} = \mathbf{s}$ and $\mathbf{e}$ has minimal weight.
- Syndrome decoding and (regular) decoding are equivalent:
  To decode $\mathbf{x}$ with syndrome decoder, compute $\mathbf{e}$ from $H\mathbf{x}$, then $\mathbf{c} = \mathbf{x} + \mathbf{e}$.
  To expand syndrome, assume $H = (Q^\intercal | I_{n-k})$.
  Then $\mathbf{x} = (00\ldots0)||\mathbf{s}$ satisfies $\mathbf{s} = H\mathbf{x}$.
- Note that this $\mathbf{x}$ is not a solution to the syndrome decoding problem, unless it has very low weight.

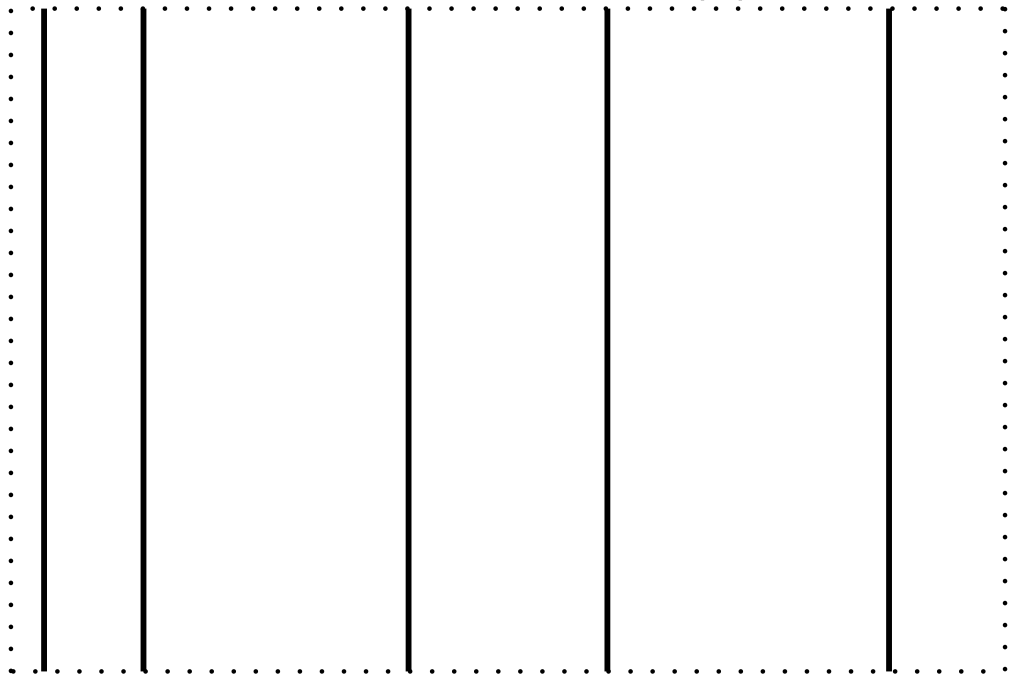
Code-based encryption

Developed in 1978 by Robert McEliece; syndrome view by Harald Niederreiter (1986). This is (mostly) KEM version as in Classic McEliece.

- Private key: decoder for chosen Goppa code $\Gamma(L, g)$.
- Public Key: parity-check matrix in systematic form $H = (Q|I_{n-k})$, represented by Q.
- Encapsulation: Pick random $\mathbf{e} \in \mathbf{F}_2^n$ of weight t.
  Compute $\mathbf{c} = H\mathbf{e}$ and $(C, k) = \text{hash}(\mathbf{e})$. Send $(\mathbf{c}, C)$.
- Decapsulation: Use Goppa decoder on $\mathbf{c}$ to compute $\mathbf{e}$.
  Compute $(C', k') = \text{hash}(\mathbf{e})$. If C matches C', output k'.
  (Else some stuff that makes proofs happy).
- See https://classic.mceliece.org for more details.
- The attacker is facing a t-error correcting problem for the public key.
- Structural attack: find private key from public key.
- Not only Goppa codes, some other constructions look OK (for now). NIST competition has several more entries (QCMDPC, rank metric).
- Many corpses on the way: Reed-Solomon codes, concatenated codes, Reed-Muller codes, several Algebraic Geometry (AG) codes, Gabidulin codes, several LDPC codes, cyclic code.

Generic attack: Brute force

Given H and $\mathbf{c} = H\mathbf{e}$, find $\mathbf{e}$ with $\mathrm{wt}(\mathbf{e}) = t$.

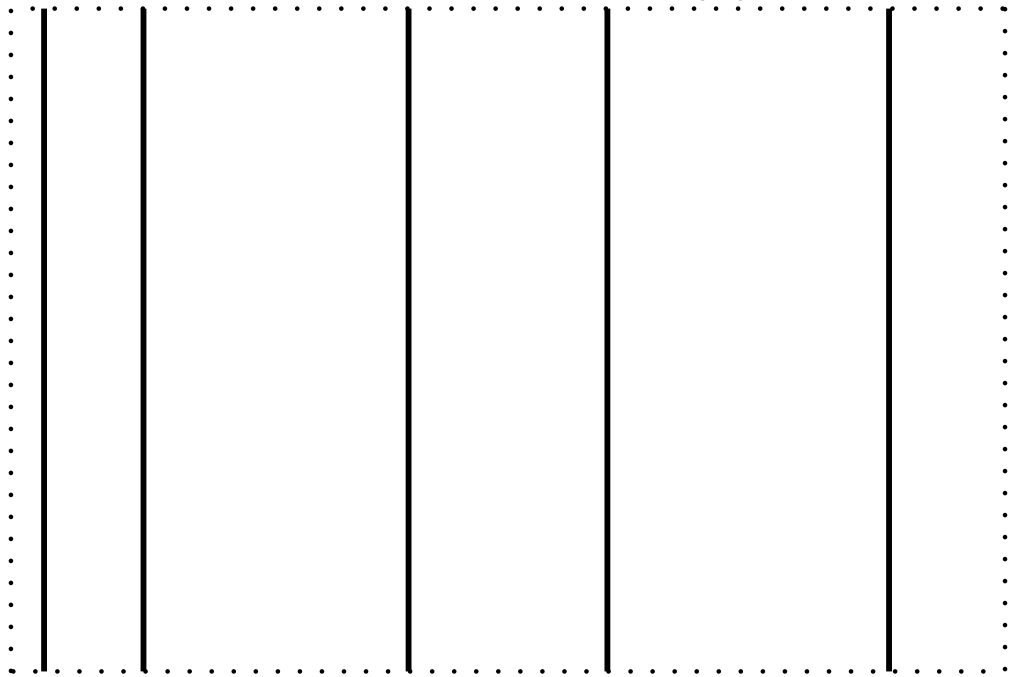
$H =$ 

Pick any group of t columns of H, add them and compare with $\mathbf{s}$.

Cost:

Generic attack: Brute force

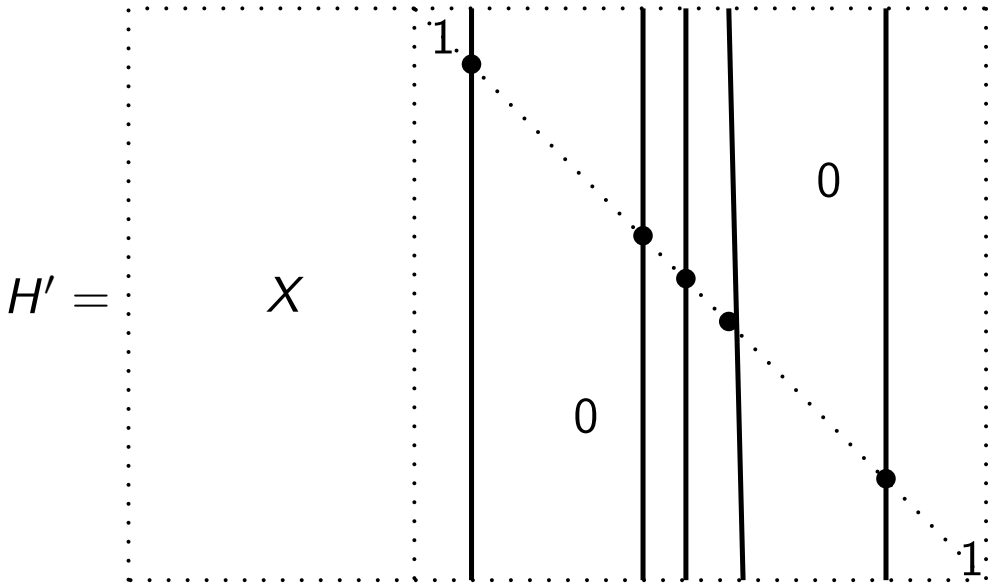
Given H and $\mathbf{c} = H\mathbf{e}$, find $\mathbf{e}$ with $\mathrm{wt}(\mathbf{e}) = t$.

$H =$ 

Pick any group of t columns of H, add them and compare with $\mathbf{s}$.

Cost: $\binom{n}{t}$ sums of t columns.
Can do better so that each try costs only 1 column addition (after some initial additions).
Cost: $O\binom{n}{t}$ additions of 1 column.

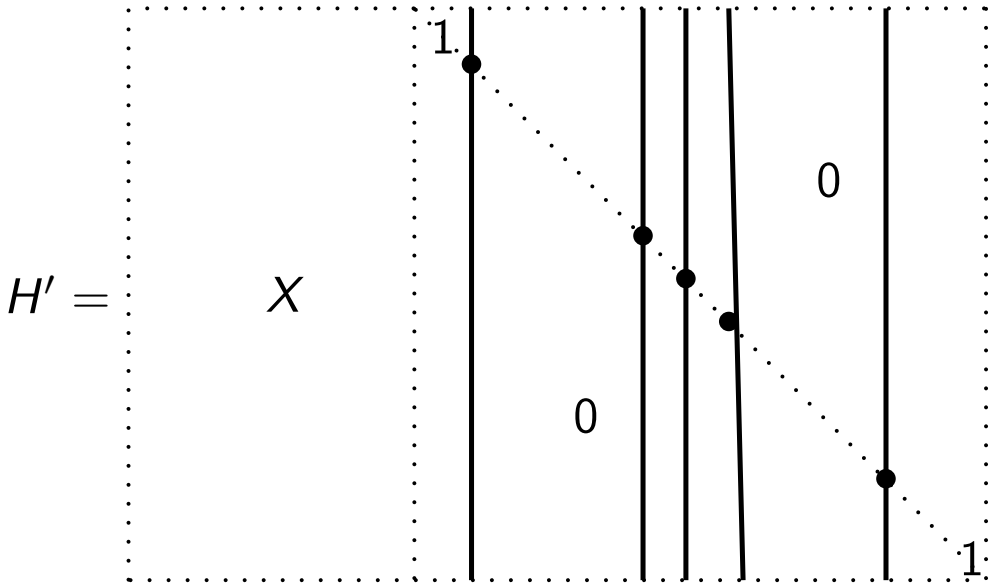
Generic attack: Information-set decoding, 1962 Prange



1. Permute columns of H and bring to systematic form $H' = (X|I_{n-k})$. (If this fails, repeat with other permutation).
2. Then $H' = UHP$ for some permutation matrix P and U the matrix that produces systematic form.
3. This updates $\mathbf{s}$ to $U\mathbf{s}$.
4. If $\mathrm{wt}(U\mathbf{s}) = t$ then $\mathbf{e}' = (00\ldots0)||U\mathbf{s}$. Output unpermuted version of $\mathbf{e}'$.
5. Else return to 1 to rerandomize.

Cost:

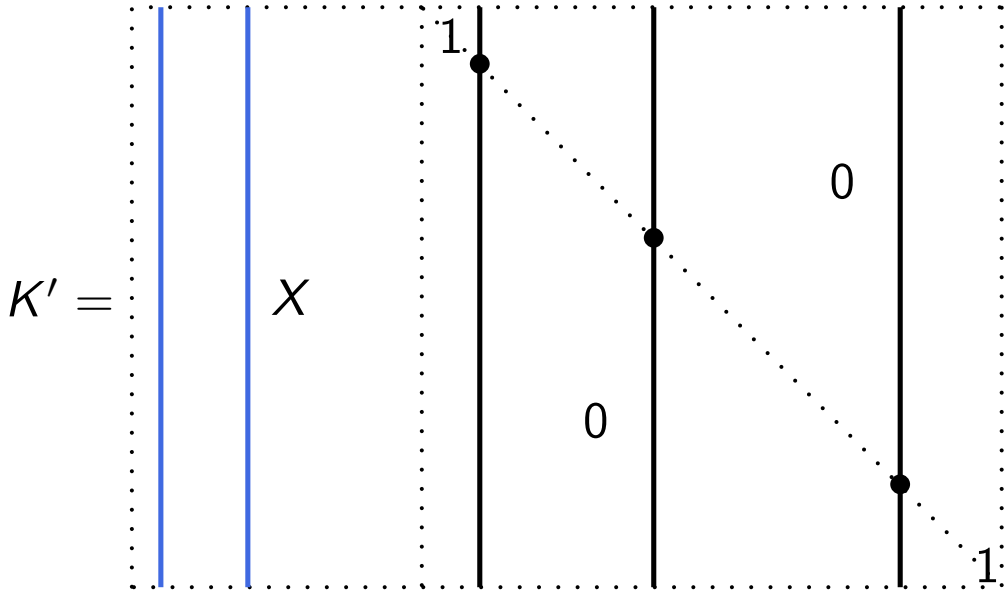
Generic attack: Information-set decoding, 1962 Prange



1. Permute columns of H and bring to systematic form $H' = (X|I_{n-k})$. (If this fails, repeat with other permutation).
2. Then $H' = UHP$ for some permutation matrix P and U the matrix that produces systematic form.
3. This updates $\mathbf{s}$ to $U\mathbf{s}$.
4. If $\mathrm{wt}(U\mathbf{s}) = t$ then $\mathbf{e}' = (00\ldots0)||U\mathbf{s}$. Output unpermuted version of $\mathbf{e}'$.
5. Else return to 1 to rerandomize.

Cost: $O(\binom{n}{t}/\binom{n-k}{t})$ matrix operations.

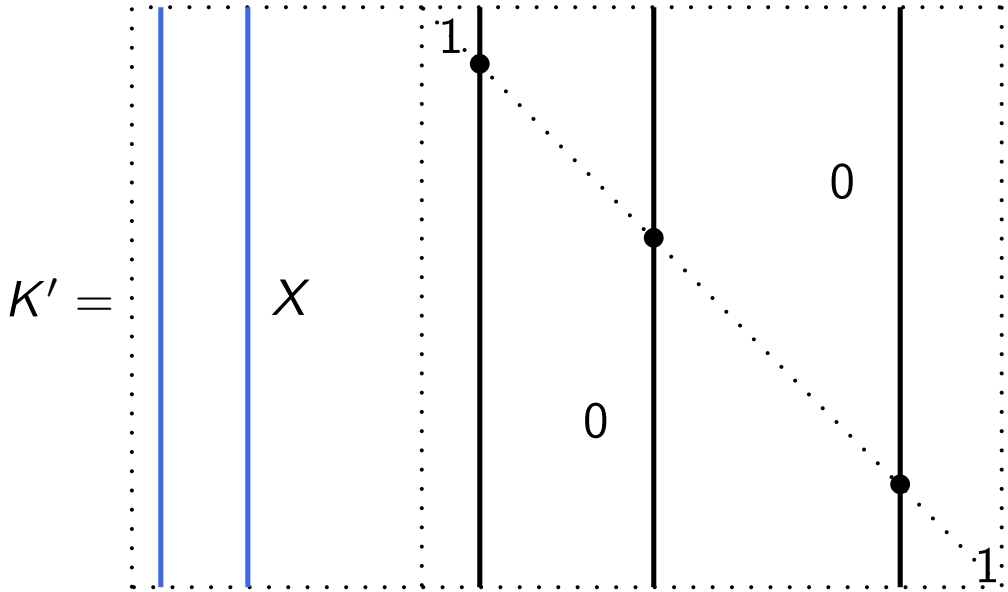
Lee–Brickell attack



1. Permute columns of H and bring to systematic form $H' = (X|I_{n-k})$. (If this fails, repeat with other permutation). $\mathbf{s}$ is updated.
2. For small p, pick p of the k columns on the left, compute their sum $X\mathbf{p}$. ($\mathbf{p}$ is the vector of weight p).
3. If $\mathrm{wt}(\mathbf{s} + X\mathbf{p}) = t - p$ then put $\mathbf{e}' = \mathbf{p}||(\mathbf{s} + X\mathbf{p})$. Output unpermuted version of $\mathbf{e}'$.
4. Else return to 2 or return to 1 to rerandomize.

Cost:

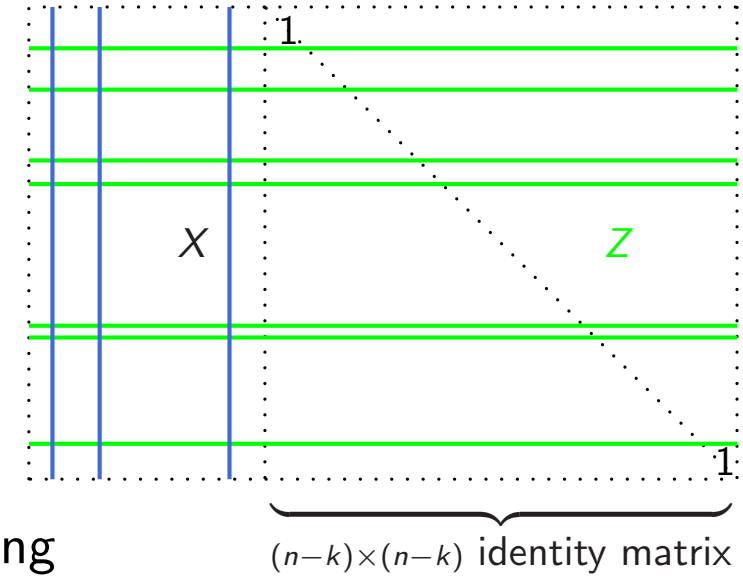
Lee–Brickell attack



1. Permute columns of H and bring to systematic form $H' = (X|I_{n-k})$. (If this fails, repeat with other permutation). $\mathbf{s}$ is updated.
2. For small p, pick p of the k columns on the left, compute their sum $X\mathbf{p}$. ($\mathbf{p}$ is the vector of weight p).
3. If $\mathrm{wt}(\mathbf{s} + X\mathbf{p}) = t - p$ then put $\mathbf{e}' = \mathbf{p}||(\mathbf{s} + X\mathbf{p})$. Output unpermuted version of $\mathbf{e}'$.
4. Else return to 2 or return to 1 to rerandomize.

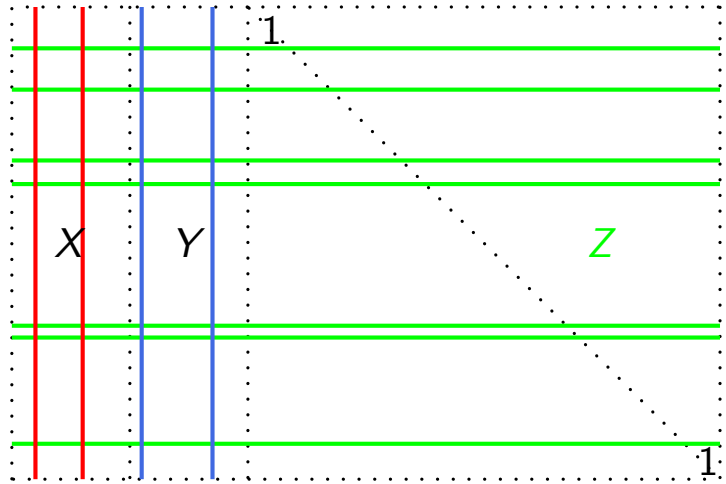
Cost: $O(\binom{n}{t}/(\binom{k}{p}\binom{n-k}{t-p}))$ [matrix operations$+\binom{k}{p}$ column additions].

Leon's attack



- Setup similar to Lee-Brickell's attack.
- Random combinations of p vectors will be dense, so have $\mathrm{wt}(\mathbf{s} + X\mathbf{p}) \sim k/2$.
- Idea: Introduce early abort by checking only ℓ positions (selected by set Z, green lines in the picture). This forms $\ell \times k$ matrix X_Z, length-ℓ vector $\mathbf{s}_Z$.
- Inner loop becomes:
  1. Pick $\mathbf{p}$ with $\mathrm{wt}(\mathbf{p}) = p$.
  2. Compute $X_Z\mathbf{p}$.
  3. If $\mathbf{s}_Z + X_Z\mathbf{p} \neq 0$ goto 1.
  4. Else compute $X\mathbf{p}$.
     - 4.1 If $\mathrm{wt}(\mathbf{s} + X\mathbf{p}) = t - p$ then put $\mathbf{e}' = \mathbf{p}||(\mathbf{s} + X\mathbf{p})$. Output unpermuted version of $\mathbf{e}'$.
     - 4.2 Else return to 1 or rerandomize H.
- Note that $\mathbf{s}_Z + X_Z\mathbf{p} = 0$ means that there are no ones in the positions specified by Z. Small loss in success, big speedup.

Stern's attack



- Setup similar to Leon's and Lee-Brickell's attacks.
- Use the early abort trick, so specify set Z.
- Improve chances of finding $\mathbf{p}$ with $\mathbf{s} + X_Z\mathbf{p} = 0$:
  - Split left part of H' into two disjoint subsets X and Y.
  - Let $A = \{\mathbf{a} \in \mathbf{F}_2^{k/2} | \mathrm{wt}(\mathbf{a}) = p\}$, $B = \{\mathbf{b} \in \mathbf{F}_2^{k/2} | \mathrm{wt}(\mathbf{b}) = p\}$.
  - Search for words having exactly p ones in X and p ones in Y and exactly $w - 2p$ ones in the remaining columns.
  - Do the latter part as a collision search:
    Compute $\mathbf{s}_Z + X_Z\mathbf{a}$ for all (many) $\mathbf{a} \in A$, sort.
    Then compute $Y_Z\mathbf{b}$ for $\mathbf{b} \in B$ and look for collisions; expand.
  - Iterate until word with $\mathrm{wt}(\mathbf{s} + X\mathbf{a} + Y\mathbf{b}) = 2p$ is found for some X, Y, Z.
- Select p, ℓ, and the subset of A to minimize overall work.