Code-based Cryptography

Tung Chou

Academia Sinica

July 13, 2022

Outline

1. Coding theory and code-based cryptography
2. ISD algorithms and other attacks
3. Classic McEliece, a code-based KEM
4. Other code-based KEMs

   (not going to talk about signature schemes)

Finite fields

A finite field F is a finite set

- with two binary operations $(+, \cdot)$
- $\{0, 1\} \subset F$ such that $\alpha + 0 = \alpha$ and $\alpha \cdot 1 = \alpha$ for all $\alpha \in F$.
- for each $\alpha \in F$ there is $-\alpha \in F$ such that $\alpha + (-\alpha) = 0$.
- for each $\alpha \in F \setminus \{0\}$ there is $\alpha^{-1} \in F$ such that $\alpha \cdot \alpha^{-1} = 1$.

We always have $|F| = p^k$ for some prime p.

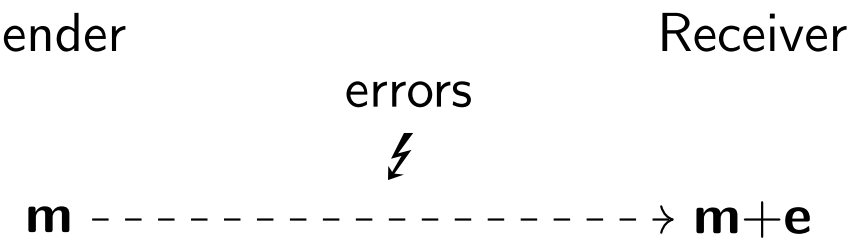
- A field with q elements is denote as $\mathbb{F}_q$.
- if $k = 1$, $F \cong \mathbb{Z}_p$.
- if $k > 1$, $F \cong \mathbb{F}_p[x]/(f(x))$ where $f \in \mathbb{F}_p[x]$ is of degree k and **irreducible**.

We will mainly consider characteristic-2 fields $\mathbb{F}_{2^m}$.

- Important property: for any $\alpha \in \mathbb{F}_{2^m}, \alpha + \alpha = 0$.
- In other words, $-$ is the same as $+$.

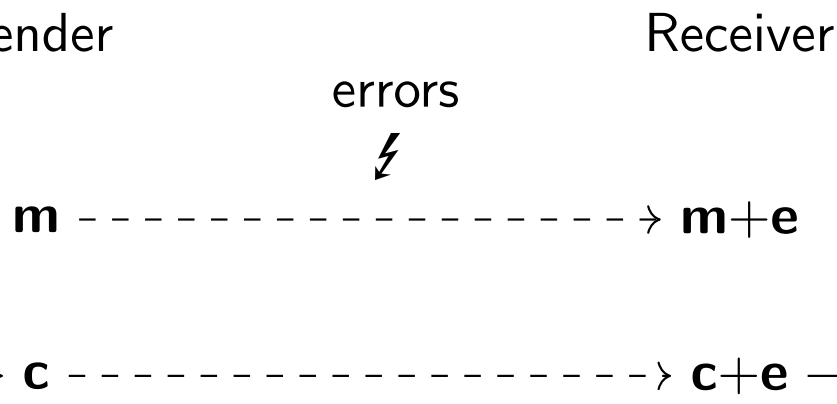
Error correction

- Goal: protect data against errors in a noisy channel.



Error correction

- Goal: protect data against errors in a noisy channel.



- The sender transforms a message $\mathsf{m} \in \mathbb{F}_q^k$ into a **codeword** $\mathsf{c} \in \mathbb{F}_q^n$ ($n > k$) of a **code** C by adding redundancy (**encoding**).

- The channel introduces errors (bitflips), which can be viewed as adding an **error vector** $\mathsf{e} \in \mathbb{F}_q^n$ to the data.

- The receiver uses a **decoding algorithm (decoder)** to correct the errors. This works as long as there are not too many errors.

Linear codes

A **linear code** C of **length** n and **dimension** k is a k-dimensional subspace of $\mathbb{F}_q^n$.

Linear codes

A **linear code** C of **length** n and **dimension** k is a k-dimensional subspace of $\mathbb{F}_q^n$.

C is usually specified as

- the row space of a **generating matrix** $G \in \mathbb{F}_q^{k \times n}$

$$C = \left\{ \mathsf{m}G \mid \mathsf{m} \in \mathbb{F}_q^k \right\}$$

Linear codes

A **linear code** C of **length** n and **dimension** k is a k-dimensional subspace of $\mathbb{F}_q^n$.

C is usually specified as

- the row space of a **generating matrix** $G \in \mathbb{F}_q^{k \times n}$

$$C = \left\{ \mathsf{m}G \mid \mathsf{m} \in \mathbb{F}_q^k \right\}$$

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

$$C = \left\{ \mathsf{c} \mid H\mathsf{c}^\intercal = 0, \ \ \mathsf{c} \in \mathbb{F}_q^n \right\}$$

(will often omit $^\intercal$ from now on)

Linear codes

A **linear code** C of **length** n and **dimension** k is a k-dimensional subspace of $\mathbb{F}_q^n$.

C is usually specified as

- the row space of a **generating matrix** $G \in \mathbb{F}_q^{k \times n}$

$$C = \{\mathsf{m}G \mid \mathsf{m} \in \mathbb{F}_q^k\}$$

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

$$C = \{\mathsf{c} \mid H\mathsf{c}^\intercal = 0, \ \ \mathsf{c} \in \mathbb{F}_q^n\}$$

  (will often omit $^\intercal$ from now on)

- Hv is called the **syndrome** of v w.r.t. to H.

- RowSpace(H) is called the **dual code** of C and denoted as $C^\perp$

Example

- C over $\mathbb{F}_2$ with $n = 7$ and $k = 4$.

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

- $\mathrm{c} = (1001)G = (1001001)$ is a codeword.
- $Hc = 0$.
- Note that $G = (I|Q)$ and $H = (Q^\intercal|I)$.

Hamming weight and distance

- The **Hamming weight** of a word is the number of nonzero coordinates.

$$\mathrm{wt}(1,0,0,1,1) = 3$$

- The **Hamming distance** between two words in $\mathbb{F}_q^n$ is the number of coordinates in which they differ.

$$d((1,1,0,1,1),(1,0,1,1,1)) = 2$$

- The **minimum distance** of a linear code C is
  - the smallest Hamming distance between any two codewords.
  - the smallest Hamming weight of a nonzero codeword in C.

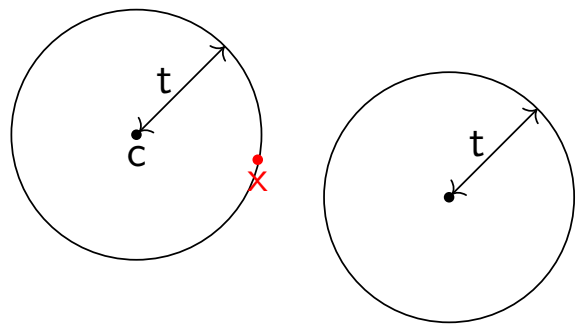
$$d = \min_{0 \neq c \in C} \{\mathrm{wt}(c)\} = \min_{c_1 \neq c_2 \in C} \{d(c_1, c_2)\}$$

The general decoding problem

- Regular decoding: given (G, x), find minimum-weight e (or c) such that $x = c + e$
- Syndrome decoding: given (H, s), find minimum-weight e such that $s = He$
- The two problems are equivalent in the sense that they can be reduced to each other.
  - Given $x = c + e$, can easily compute $Hx = He$.
  - Given He, can easily compute x such that $Hx = He$. Now $x = c + e$ for some c.
- There are variants with $\mathrm{wt}(e) \leq t$ or $\mathrm{wt}(e) = t$.
- Considered as well-studied hard problems when G/H are random.

Minimum distance

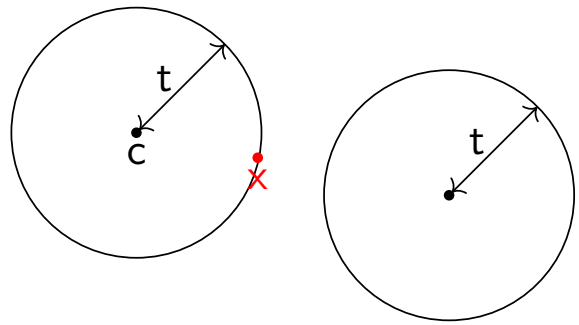
- Minimum distance indicates how many errors can be corrected: any vector $\mathrm{x} = \mathrm{c} + \mathrm{e}$ with $\mathrm{wt}(\mathrm{e}) = t < d/2$ is uniquely decodable to c;



- Equivalently, the code can correct t errors if $d \geq 2t + 1$.
- Equivalently, the code can correct $\lfloor (d-1)/2 \rfloor$ errors.

Minimum distance

- Minimum distance indicates how many errors can be corrected: any vector $\mathrm{x} = \mathrm{c} + \mathrm{e}$ with $\mathrm{wt}(\mathrm{e}) = t < d/2$ is uniquely decodable to c;



- Equivalently, the code can correct t errors if $d \geq 2t + 1$.
- Equivalently, the code can correct $\lfloor (d-1)/2 \rfloor$ errors.

- Having $t < d/2$ does not mean that there is an efficient algorithm for decoding t errors.

Hamming code

Parity check matrix ($n = 7, k = 4$):

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

- Note that the columns are binary expansions of $\{1, 2, \ldots, 7\}$.
- Minimum distance?

Hamming code

Parity check matrix ($n = 7, k = 4$):

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

- Note that the columns are binary expansions of $\{1, 2, \ldots, 7\}$.
- Minimum distance? $d = 3$.

Hamming code

Parity check matrix ($n = 7, k = 4$):

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

- Note that the columns are binary expansions of $\{1, 2, \ldots, 7\}$.
- Minimum distance? $d = 3$.
- The code can correct

Hamming code

Parity check matrix ($n = 7, k = 4$):

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

- Note that the columns are binary expansions of $\{1, 2, \ldots, 7\}$.
- Minimum distance? $d = 3$.
- The code can correct $\lfloor (d-1)/2 \rfloor = 1$ error.

Hamming code

Parity check matrix ($n = 7, k = 4$):

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

- Note that the columns are binary expansions of $\{1, 2, \ldots, 7\}$.
- Minimum distance? $d = 3$.
- The code can correct $\lfloor (d-1)/2 \rfloor = 1$ error.
- A decoding algorithm: Given $r = c + e$ with $\mathrm{wt}(e) = 1$, compute $s = Hr = Hc + He = He$.

The McEliece cryptosystem (1978)

- The secret key consists of
  - $G \in \mathbb{F}_q^{k \times n}$, a generator matrix for code C.
  - $P \in \mathbb{F}_q^{n \times n}$, a uniform random permutation matrix.
  - $S \in \mathbb{F}_q^{k \times k}$, a uniform random nonsingular matrix.
  - some piece of information for decoding C

- The public key is the $\hat{G} = SGP \in \mathbb{F}_q^{k \times n}$, which can be viewed as a scrambled version of G.

- The (public) key size is kn field elements.

The McEliece cryptosystem (1978)

- The secret key consists of
  - $G \in \mathbb{F}_q^{k \times n}$, a generator matrix for code C.
  - $P \in \mathbb{F}_q^{n \times n}$, a uniform random permutation matrix.
  - $S \in \mathbb{F}_q^{k \times k}$, a uniform random nonsingular matrix.
  - some piece of information for decoding C

- The public key is the $\hat{G} = SGP \in \mathbb{F}_q^{k \times n}$, which can be viewed as a scrambled version of G.

- The (public) key size is kn field elements.

- Main idea: make $\hat{G}$ look like a random matrix, so that the best an attacker can do is to solve the general decoding problem.

The McEliece cryptosystem (cont.)

Encryption

- Given message $\mathsf{m} \in \mathbb{F}_q^k$.
- Generate a uniform random error vector $\mathsf{e} \in \mathbb{F}_q^n$ with $\mathrm{wt}(\mathsf{e}) = t$.
- Return $\mathsf{y} = \mathsf{m}\hat{G} + \mathsf{e} \in \mathbb{F}_q^n$.

Decryption

- Compute $\mathsf{y}P^{-1} = \mathsf{m}\hat{G}P^{-1} + \mathsf{e}P^{-1} = (\mathsf{m}S)G + \mathsf{e}P^{-1}$.
- Note that P is a permutation matrix, so $\mathrm{wt}(\mathsf{e}P^{-1}) = \mathrm{wt}(\mathsf{e}) = t$.
- Use the decoding algorithm to find $(\mathsf{m}S)G$, $\mathsf{m}S$ and m.

The Niederreiter cryptosystem (1986)

- The secret key consists of
  - $H \in \mathbb{F}_q^{(n-k)\times n}$, a parity-check matrix for a code C.
  - $P \in \mathbb{F}_q^{n\times n}$, a uniform random permutation matrix.
  - $S \in \mathbb{F}_q^{(n-k)\times(n-k)}$, a uniform random nonsingular matrix.
  - some piece of information for decoding C
- The public key is $\hat{H} = SHP \in \mathbb{F}_q^{(n-k)\times n}$, a scrambled version of H.
- The (public) key size is $(n-k)n$ field elements.

The Niederreiter cryptosystem (1986)

- The secret key consists of
  - $H \in \mathbb{F}_q^{(n-k)\times n}$, a parity-check matrix for a code C.
  - $P \in \mathbb{F}_q^{n\times n}$, a uniform random permutation matrix.
  - $S \in \mathbb{F}_q^{(n-k)\times(n-k)}$, a uniform random nonsingular matrix.
  - some piece of information for decoding C
- The public key is $\hat{H} = SHP \in \mathbb{F}_q^{(n-k)\times n}$, a scrambled version of H.
- The (public) key size is $(n-k)n$ field elements.
- Main idea: make $\hat{H}$ look like a random matrix, so that the best an attacker can do is to solve the general decoding problem.

The Niederreiter cryptosystem (cont.)

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

$$\mathsf{s} = \hat{H}\mathsf{e}.$$

- Decryption using secret key: Compute

$$\begin{aligned} S^{-1}\mathsf{s} &= S^{-1}\hat{H}\mathsf{e} = S^{-1}(SHP)\mathsf{e} \\ &= H(P\mathsf{e}) \end{aligned}$$

  and observe that $\mathrm{wt}(P\mathsf{e}) = \mathrm{wt}(\mathsf{e}) = t$.

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

- Breaking Niederreiter $\iff$ Breaking McEliece (OW-CPA)

Reducing key size using systematic form

- Idea: choose S such that $\hat{G} = (I_k|Q)$ for McEliece and $\hat{H} = (Q^\intercal|I_{n-k})$ for Niederreiter, where $Q \in \mathbb{F}_q^{k\times(n-k)}$.

Reducing key size using systematic form

- Idea: choose S such that $\hat{G} = (I_k|Q)$ for McEliece and $\hat{H} = (Q^\intercal|I_{n-k})$ for Niederreiter, where $Q \in \mathbb{F}_q^{k\times(n-k)}$.

- Summary on key and ciphertext sizes

	$\lvert\text{key}\rvert$	$\lvert\text{ciphertext}\rvert$
McEliece	$k \times n$	n
Niederreiter	$(n-k) \times n$	$n-k$
Systematic McEliece	$k \times (n-k)$	n
Systematic Niederreiter	$(n-k) \times k$	$n-k$

Reducing key size using systematic form

- Idea: choose S such that $\hat{G} = (I_k|Q)$ for McEliece and $\hat{H} = (Q^\intercal|I_{n-k})$ for Niederreiter, where $Q \in \mathbb{F}_q^{k\times(n-k)}$.

- Summary on key and ciphertext sizes

	|key|	|ciphertext|
McEliece	$k \times n$	n
Niederreiter	$(n-k) \times n$	$n-k$
Systematic McEliece	$k \times (n-k)$	n
Systematic Niederreiter	$(n-k) \times k$	$n-k$

- Does not work for all secret keys. We might need to retry a few times (≈ 3.5 when $q = 2$).

Binary Goppa code

Let $q = 2^m$. A binary Goppa code is defined by

- a list $L = (\alpha_1, \ldots, \alpha_n)$ of n *distinct* elements in $\mathbb{F}_q$, called the **support**.
- a degree-t polynomial $g(x) \in \mathbb{F}_q[x]$ such that $g(\alpha_i) \neq 0$ for all $\alpha_i \in L$. $g(x)$ is called the **Goppa polynomial**.

Binary Goppa code

Let $q = 2^m$. A binary Goppa code is defined by

- a list $L = (\alpha_1, \ldots, \alpha_n)$ of n *distinct* elements in $\mathbb{F}_q$, called the **support**.
- a degree-t polynomial $g(x) \in \mathbb{F}_q[x]$ such that $g(\alpha_i) \neq 0$ for all $\alpha_i \in L$. $g(x)$ is called the **Goppa polynomial**.

The corresponding binary Goppa code $\Gamma(L, g)$ is

$$\left\{ c \in \mathbb{F}_2^n \,\middle|\, \frac{c_1}{x - \alpha_1} + \frac{c_2}{x - \alpha_2} + \cdots + \frac{c_n}{x - \alpha_n} \equiv 0 \bmod g(x) \right\}$$

- $1/(x - \alpha_i)$ is a polynomial that, when multiplied by $x - \alpha_i$, becomes $1 \bmod g(x)$.
- C is a linear code over $\mathbb{F}_2$.

Properties of $\Gamma(L, g)$, $\deg(g) = t$

- Dimension $k \geq n - mt$ (usually equality holds).
- Minimum distance $d \geq t + 1$.
- $\Gamma(L, g) = \Gamma(L, g^2)$ if g is **square-free**.
- $d \geq 2t + 1$ if g is square-free. McEliece suggested to use irreducible g, which implies square-free.
- There exist efficient t-error decoding algorithms when g is square-free. See Patterson algorithm (1975) and the Berlekamp-Massey algorithm (1967, 1969).

ISD algorithms

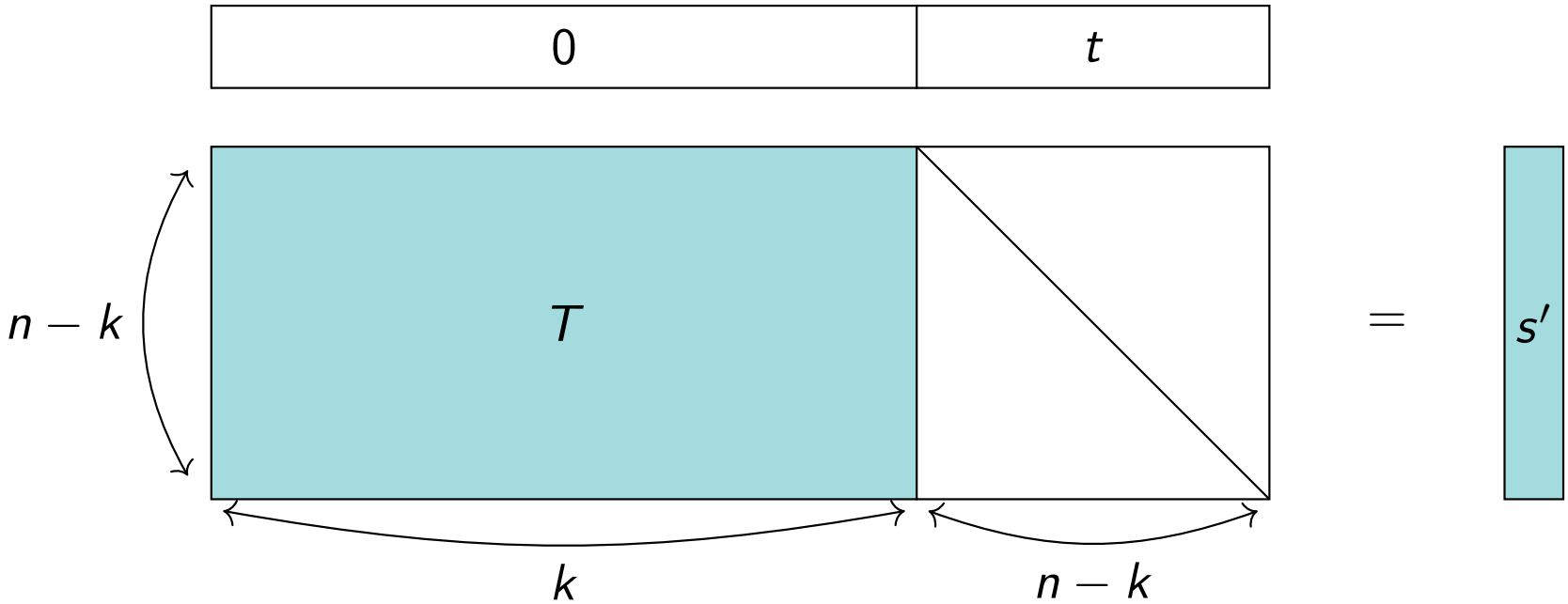
The task:

- Given $H \in \mathbb{F}_2^{(n-k)\times n}$ and $s \in \mathbb{F}_2^n$, find $e \in \mathbb{F}_2^n$ with $\mathrm{wt}(e) = t$ such that $s = He$.
- Can also take the view of generating matrix.

ISD algorithms are iterative. The beginning of an iteration always does the follows.

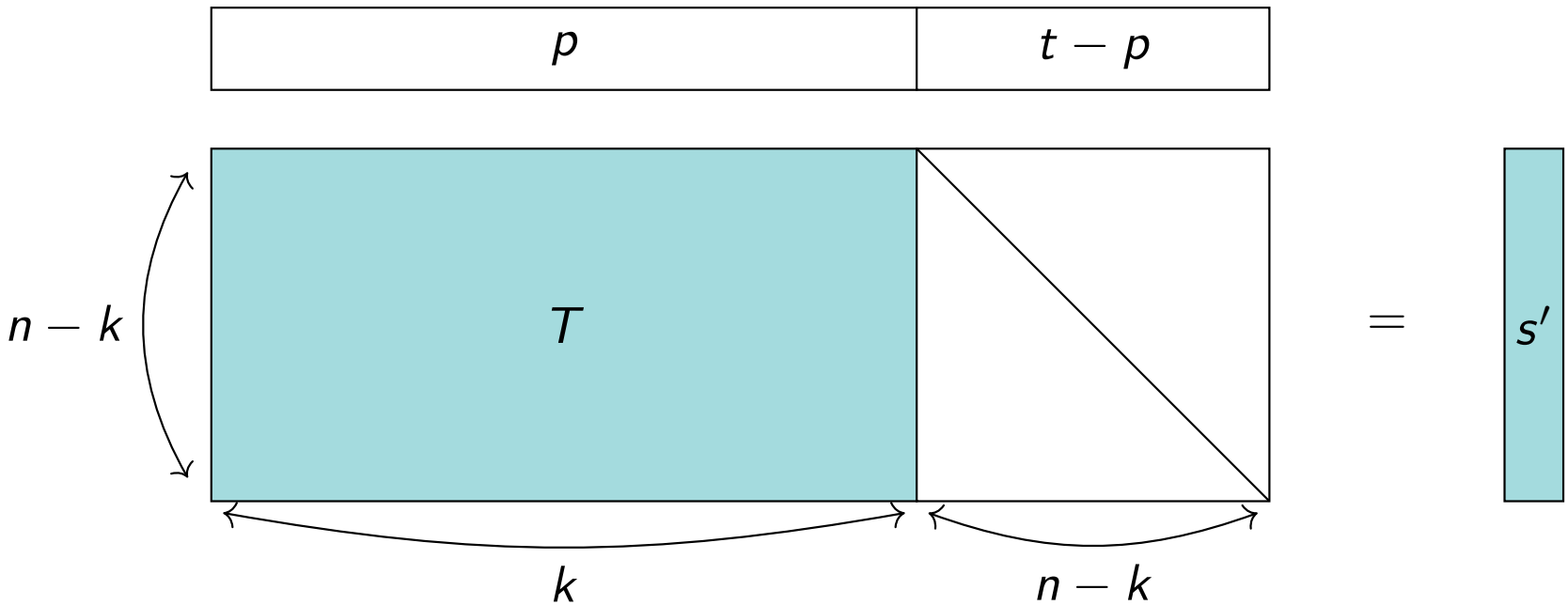
- The columns in H are permuted to obtain a matrix HP.
- Then HP is reduced to systematic form (or something similar) $H' = (T|I) = SHP$. Also $s' = Ss$ is computed.
- If systematic form of HP does not exist, the corresponding iteration is skipped.
- The remainder of the iteration aims to find a solution for $s' = H'e'$.
- We have $Ss = SHPe'$, so $s = H(Pe') = He$.

Prange (1962)



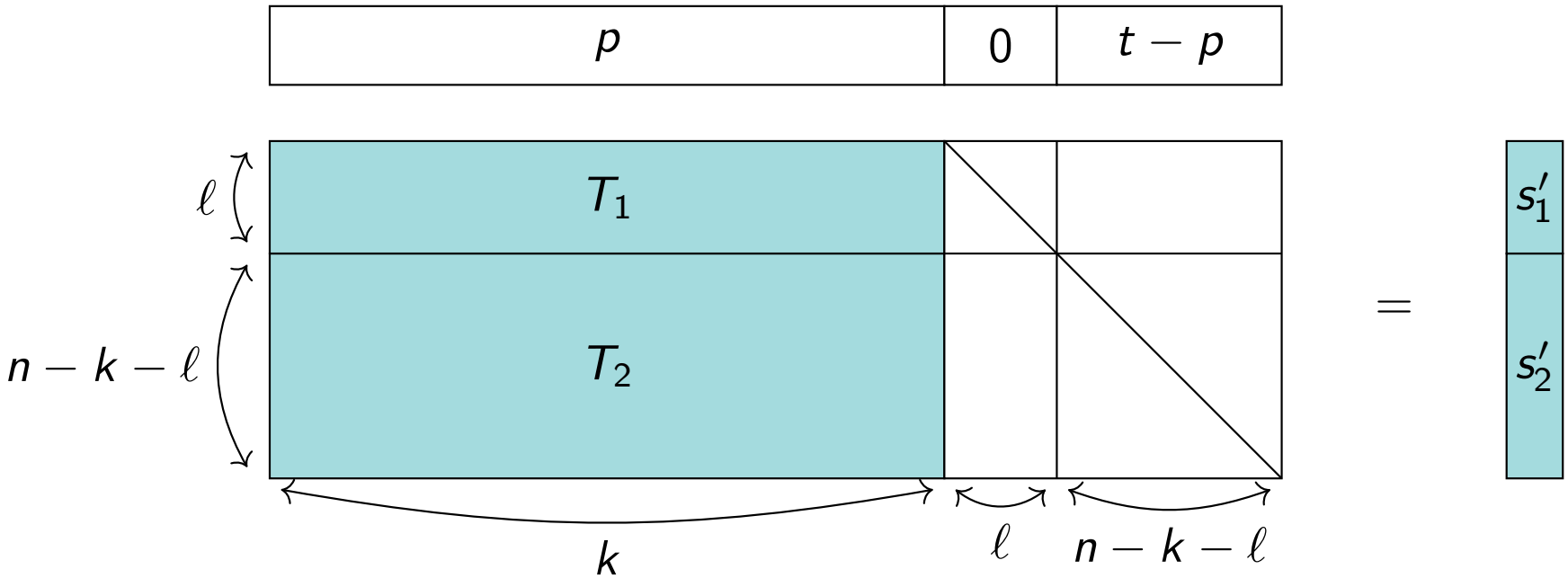
- Guess that there are 0 errors in the first k coordinates and w errors in the last $n - k$ coordinates.
- Prange checks if such a solution exists by checking if $\mathrm{wt}(s') = t$.
- Success probability for 1 iteration is very low.

Lee-Brickel (1988)



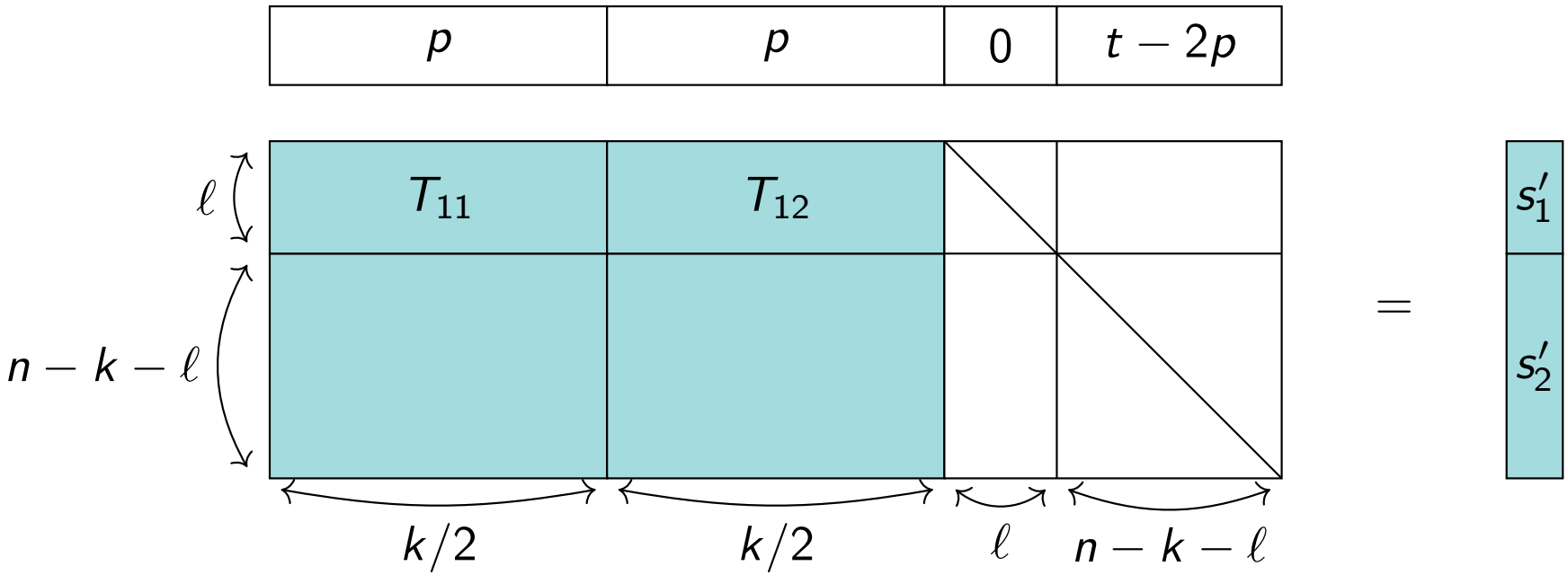
- Guess that there are p errors in the first k coordinates and $t - p$ errors in the last $n - k$ coordinates.
- For each sum v of p columns, Lee-Brickel checks if such a solution exists by checking if $\mathrm{wt}(v + s') = t - p$.

Leon (1988)



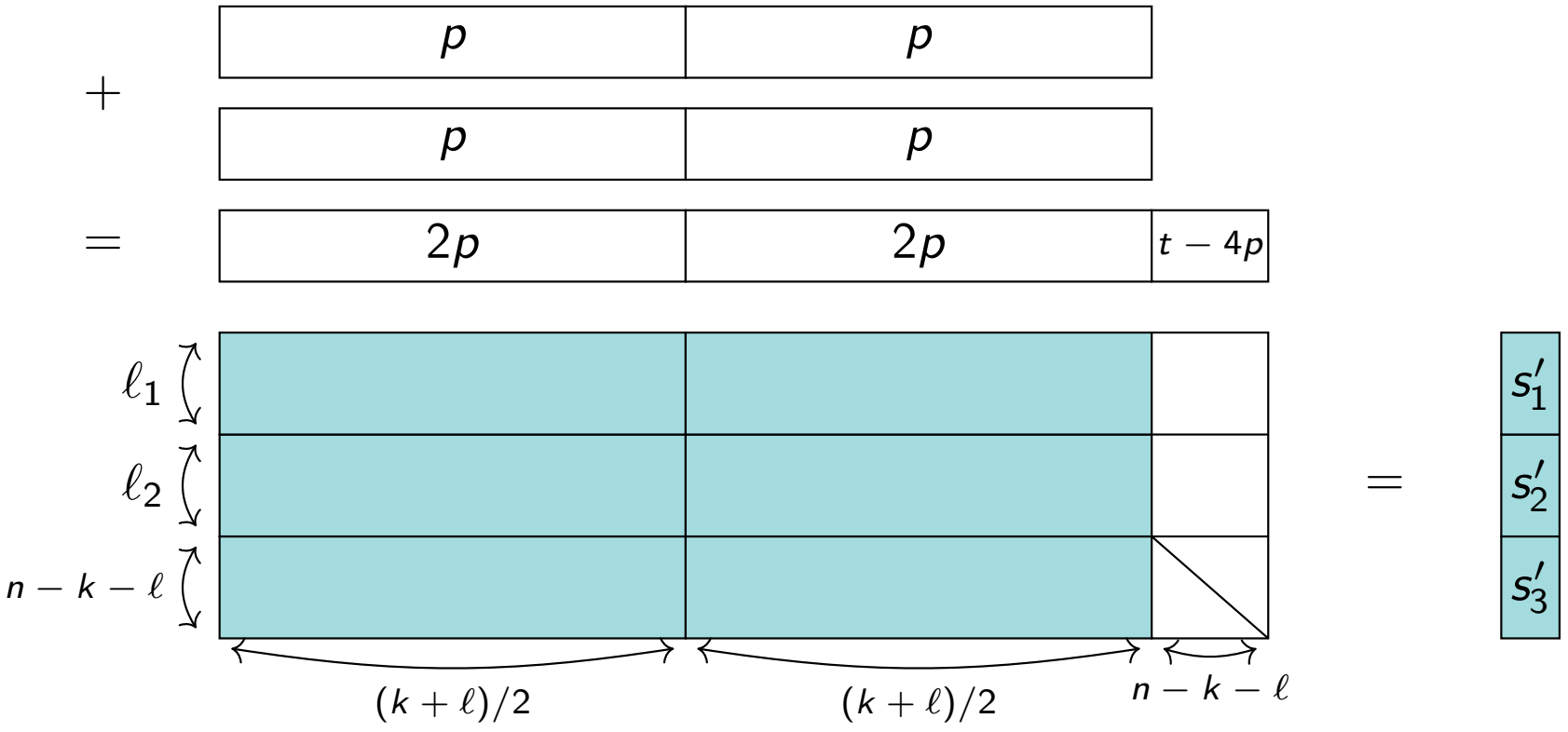
- Guess that there are $p, 0, t-p$ errors in the first $k, \ell, n-k-\ell$ coordinates.
- Each sum v_1 of p columns of T_1 is checked for $v_1 = s'_1$.
- For each valid v_1, the corrsponding v_2 is checked for $\mathrm{wt}(v_2 + s'_2) = t - p$.

Stern (1989)



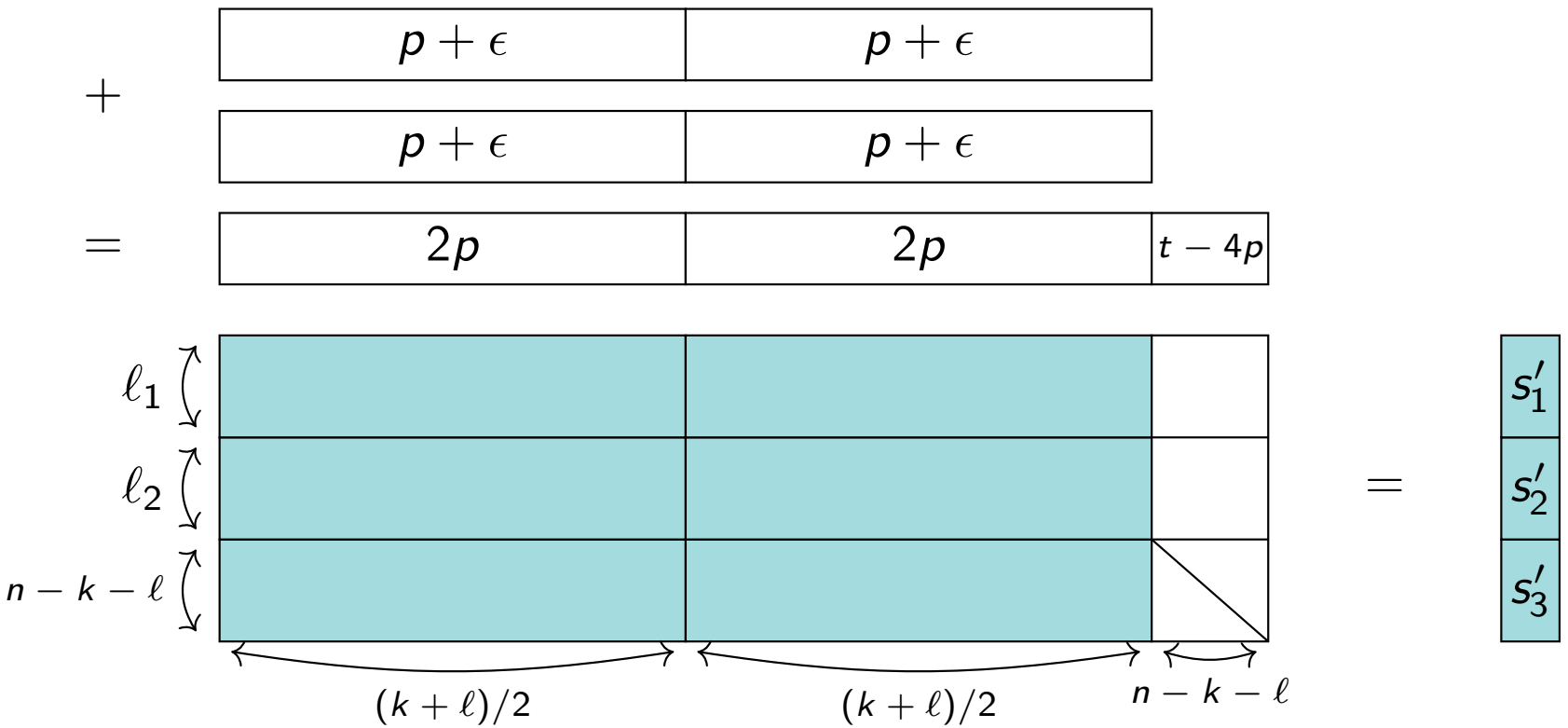
- Making use of **collision search**.
- Let L_1 be the set of sums of p columns of T_{11}.
- Let L_2 be the set of sums of p columns of T_{12} plus s'_1.
- Find collisions (say, using a hash table) between L_1 and L_2.
- Each collision $v_{11} = v_{12} + s'_1$ is checked if $\mathrm{wt}(v_{21} + v_{22} + s'_2) = t - 2p$.

May–Meurer–Thomae (MMT, 2011)



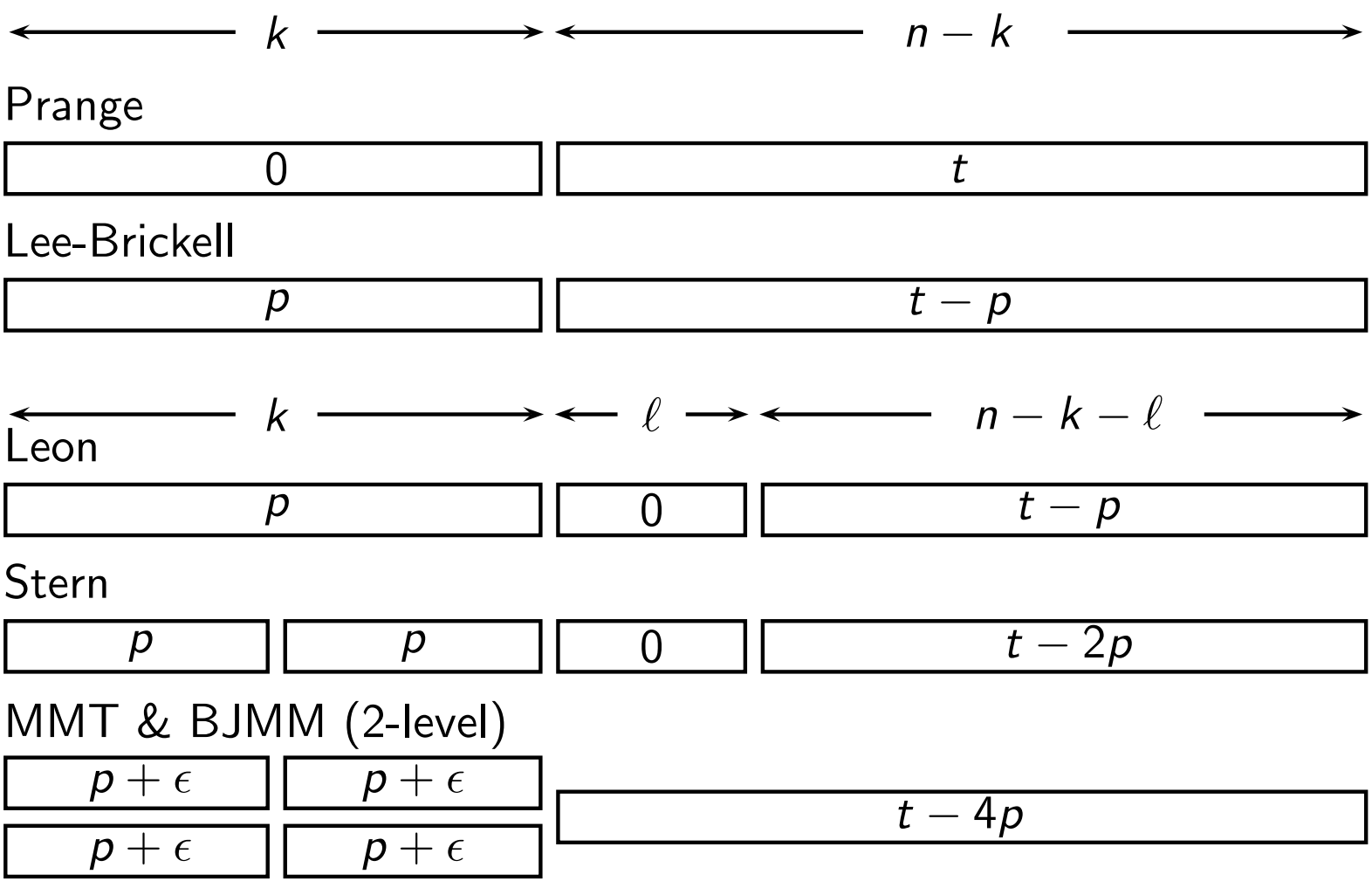
- Multiple-level collision searches, **multiple representations**.
- Build sets of sums of p columns L_1, L_2.
- Find $v_{11} \in L_1$ and $v_{12} \in L_2$ with $v_{11} = v_{12} + \Delta$.
- Find $v'_{11} \in L_1$ and $v'_{12} \in L_2$ with $v'_{11} = v'_{12} + s'_1 + \Delta$.
- For each solution, check if $(v_{21} + v_{22}) = (v'_{21} + v'_{22}) + s'_2$.

Becker–Joux–May–Meurer (BJMM, 2012)



- Similar to MMT but allows 1's to cancel out.

ISD algorithms



Practical attack on McEliece (Bernstein, Lange, Peters, 2008)

- Original McEliece parameters: $(m, n, k, t) = (10, 1024, 524, 50)$.
- Bernstein, Lange, and Peters broke it with their software for Stern.
- About 200 computers involved, with about 300 cores. Computation used about 8000 core-days.
- Lots of optimizations for Stern, e.g.,
  - maintaining systematic form by swapping only a few columns,
  - efficient generation of the base lists,
  - trying several candidates for the error-free coordinates in each iteration.
- Solution: just use larger parameters!

The Verheul–Doumen–Tilborg attack (1998)

- Assume that the decoding algorithm decodes up to t errors, i. e. it decodes $\mathsf{y} = \mathsf{c} + \mathsf{e}$ to c if $\mathrm{wt}(\mathsf{e}) \leq t$. It fails with an overwhelming probability when $\mathrm{wt}(\mathsf{e}) > t$.

The Verheul–Doumen–Tilborg attack (1998)

- Assume that the decoding algorithm decodes up to t errors,
  i. e. it decodes $\mathsf{y} = \mathsf{c} + \mathsf{e}$ to c if $\mathrm{wt}(\mathsf{e}) \leq t$. It fails with an overwhelming probability when $\mathrm{wt}(\mathsf{e}) > t$.

- Eve intercepts ciphertext $\mathsf{y} = \mathsf{m}G' + \mathsf{e}, \mathrm{wt}(\mathsf{e}) = t$.
  Eve poses as Alice towards Bob and sends him tweaks of y.
  She observes whether Bob suceeds or fails to decrypt: this is a **reaction attack**.

The Verheul–Doumen–Tilborg attack (1998)

- Assume that the decoding algorithm decodes up to t errors, i. e. it decodes $\mathrm{y} = \mathrm{c} + \mathrm{e}$ to c if $\mathrm{wt}(\mathrm{e}) \leq t$. It fails with an overwhelming probability when $\mathrm{wt}(\mathrm{e}) > t$.

- Eve intercepts ciphertext $\mathrm{y} = \mathrm{m}G' + \mathrm{e}, \mathrm{wt}(\mathrm{e}) = t$.
  Eve poses as Alice towards Bob and sends him tweaks of y.
  She observes whether Bob suceeds or fails to decrypt: this is a **reaction attack**.

- Eve sends $\mathrm{y}_i = \mathrm{y} + \mathrm{e}_i$ 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.

Support splitting

- What can an attacker do if 1) the polnomial g and 2) the set $\{\alpha_1, \ldots, \alpha_n\}$ are leaked?
- The only information missing is the order of the support elements.
- Sendrier: compute the "**signature**" of each of the n coordinates.
- The attacker knows the code C and computes a coordinate-permuted code C'.
- **Punctured code**: a code resulted by removing one or more coordinate of the input code.
- For each coordinate i, compute the signature

$$\text{WeightEnumerator}(Punc_i(C) \bigcap Punc_i(C)^{\perp})$$

- Do the same for C'. Try to find matching signatures.

Gibson's attack

What can the attacker do if the support $\alpha_1, \ldots, \alpha_n$ is leaked?

- Recall that each code word $c \in \mathbb{F}_2^n$ satisfies

$$\frac{c_1}{x - \alpha_1} + \cdots + \frac{c_n}{x - \alpha_n} \equiv 0 \mod g$$

- We have

$$\frac{\sum_i c_i \prod_{j \neq i}(x - \alpha_j)}{\prod_i (x - \alpha_i)} \equiv 0 \mod g$$

- So

$$\sum_i c_i \prod_{j \neq i}(x - \alpha_j) \equiv 0 \mod g$$

- For each code word we can derive a multiple of g. Compute GCD of many polynomials.

Classic McEliece highlights

Basics:

- Classic McEliece is a **CCA-secure Key Encapsulation Mechanism (KEM)**.
- Niederreiter + binary Goppa codes.
- NIST 3rd-round finalist and 4th-round candidate. Recommended by BSI.

Classic McEliece highlights

Basics:

- Classic McEliece is a **CCA-secure Key Encapsulation Mechanism (KEM)**.
- Niederreiter + binary Goppa codes.
- NIST 3rd-round finalist and 4th-round candidate. Recommended by BSI.

Pros:

- Confidence-inspiring.
- Very short ciphertexts.
- Fast encapsulation and decapsulation.

Classic McEliece highlights

Basics:

- Classic McEliece is a **CCA-secure Key Encapsulation Mechanism (KEM)**.
- Niederreiter + binary Goppa codes.
- NIST 3rd-round finalist and 4th-round candidate. Recommended by BSI.

Pros:

- Confidence-inspiring.
- Very short ciphertexts.
- Fast encapsulation and decapsulation.

Cons:

- Very big public keys
- Slow key genearation

Classic McEliece: key generation and encapsulation

- Key gen:
  - Secret key (roughly speaking): g, α_i's, n-bit string s
  - Public key: the parity-check matrix H in systematic form

- Encapsulation:
  - Generate e, compute $c_0 = He$ and $c_1 = \mathsf{Hash}_{32}(2, e)$
  - Ciphertext: $c = (c_0, c_1)$
  - Session Key: $\mathsf{Hash}_{32}(1, e, c)$

Classic McEliece: key generation and encapsulation

- Key gen:
  - Secret key (roughly speaking): g, α_i's, n-bit string s
  - Public key: the parity-check matrix H in systematic form
- Encapsulation:
  - Generate e, compute $c_0 = He$ and $c_1 = \mathsf{Hash}_{32}(2, e)$
  - Ciphertext: $c = (c_0, c_1)$
  - Session Key: $\mathsf{Hash}_{32}(1, e, c)$
- Note:
  - The constants are for **domain separation**.
  - The goal is to build a session between two parties where the traffic is encrypted with the session key using **symmetric-key crytography**.

Classic McEliece: decapsulation

- Decapsulation:
  - Perform **decode**(sk, c_0), which returns e when c_0 is a valid syndrome, or $\perp$ otherwise.
  - Check if $c_1' := \mathsf{Hash}_{32}(2, e) = c_1$.
  - If "decode" did not return $\perp$ and $c_1' = c_1$, return $\mathsf{Hash}_{32}(1, e, c)$.
  - Otherwise, return $\mathsf{Hash}_{32}(0, s, c)$.

Classic McEliece: decapsulation

- Decapsulation:
  - Perform **decode**(sk, c_0), which returns e when c_0 is a valid syndrome, or $\perp$ otherwise.
  - Check if $c_1' := \mathsf{Hash}_{32}(2, e) = c_1$.
  - If "decode" did not return $\perp$ and $c_1' = c_1$, return $\mathsf{Hash}_{32}(1, e, c)$.
  - Otherwise, return $\mathsf{Hash}_{32}(0, s, c)$.
- If something goes wrong in (c_0, c_1), return $\mathsf{Hash}_{32}(0, s, c)$.

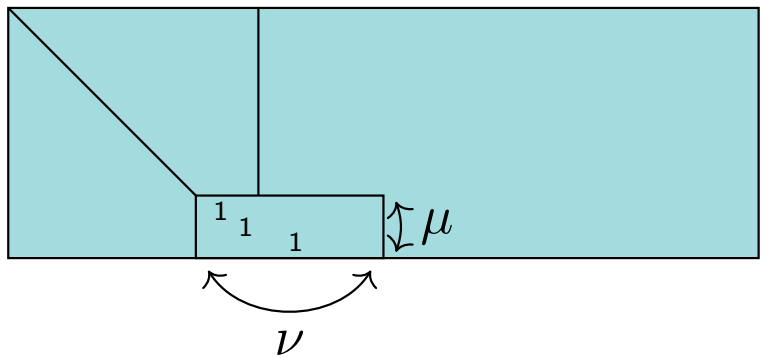
Classic McEliece: faster key generation

- The probability that a random parity-check matrix can be reduced to systematic form is low: need several attempts on average
- NTS-KEM team: reduce H to reduced row echelon form and do column swaps to obtain systematic form.

  - Note that the corresponding α_i's also need to be swapped.
  - The success probability of each attempt of public-key generation is now $\approx 1 - 2^{-k}$.
  - But each attempt is much more expensive for **constant-time** implementations.

Classic McEliece: faster key generation (cont.)

- Chou (2019): reduce H to **semi-systematic form** and do column swaps.



- Setting $\mu = \nu$ means systematic form.
- Setting $(\mu, \nu) = (n - k, n)$ means reduced row echelon form.
- The f parameter sets use $(\mu, \nu) = (32, 64)$. Success probability is $\approx 1 - 2^{-30}$.

Classic McEliece: secret key format

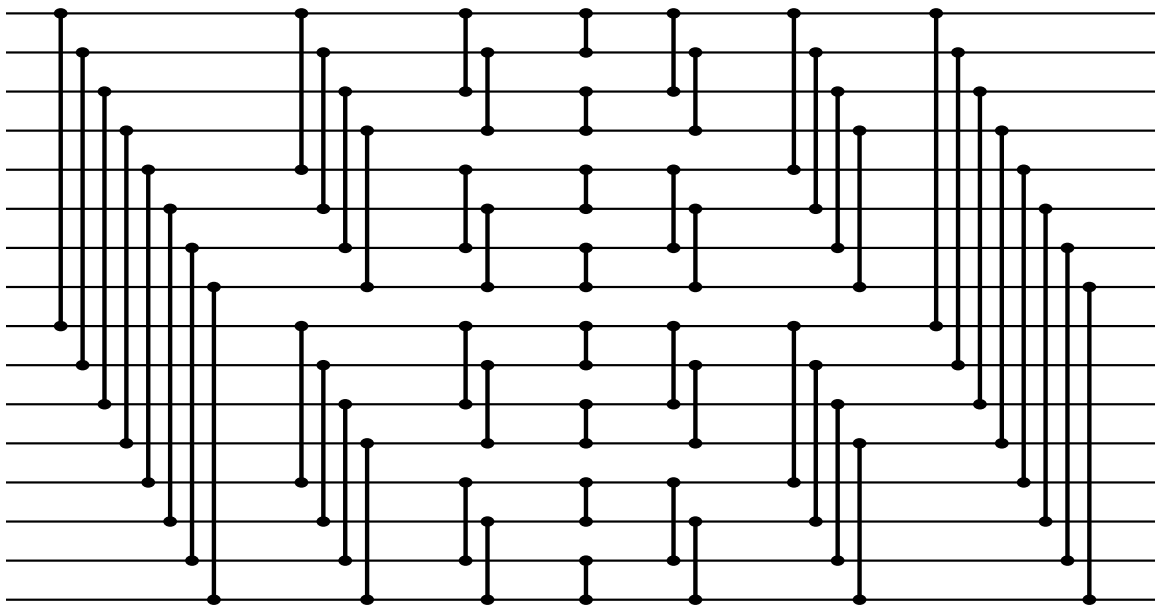
Each secret key consists of 5 components:

- δ: a 256-bit seed for generating other parts
- c: a 64-bit string that indicates the pivot columns in the $\mu \times \nu$ matrix
- g: a $2t$-byte string representing the irreducible polynomial
- α: a $(2m-1)2^{m-1}$-bit string representing the support
- s: the n-bit string

If necessary, can use any truncated format, e.g., (δ, c, g)

Classic McEliece: secret key format (cont.)

- The support (or the permutation matrix P) is represented as **control bits** of a **permutation network** for 2^m elements.
- The support is defined as the first n field elements after the permutation.
- Fast in (constant-time) software.



Bit security estimation

	Category 1 ($n = 3488$)		Category 3 ($n = 4608$)		Category 5 (n=6688)		Category 5 ($n = 6960$)		Category 5 ($n = 8192$)	
	T	M	T	M	T	M	T	M	T	M
PRANGE	173	22	217	23	296	24	297	24	334	24
STERN	151	50	193	60	268	80	268	90	303	109
BOTH-MAY	143	88	182	101	250	136	249	137	281	141
MAY-OZEROV	141	89	180	113	246	165	246	160	276	194
BJMM	142	97	183	121	248	160	248	163	278	189
BJMM-P-DW	143	86	183	100	249	160	248	161	279	166
BJMM-DW	144	97	183	100	250	130	250	160	282	164
$M \leq 60$	145	60	187	60	262	58	263	60	298	59
$M \leq 80$	143	74	183	77	258	76	258	74	293	77
$\log M$ access	147	89	187	113	253	165	253	160	283	194
$\sqrt[3]{M}$ access	156	25	199	26	275	36	276	36	312	47

Table 2: Bit security estimates for the suggested parameter sets of the Classic McEliece scheme.

- The unit for "T" is vector operation.
- Classic McEliece covers a wide range of security levels.

Classic McEliece: cycle counts

	operation	quartile	median	average	quartile
mceliece348864	keypair	35492688	46526112	58034411	68561244
mceliece348864f	keypair	36612936	36627388	36641040	36645716
mceliece460896	keypair	119530424	158155696	215785433	236056616
mceliece460896f	keypair	116896272	116914656	117067765	116938560
mceliece6688128	keypair	364883936	458561448	556495649	738637632
mceliece6688128f	keypair	284363176	284468140	284584602	284551052
mceliece6960119	keypair	249975756	330214944	438217685	490873872
mceliece6960119f	keypair	246252520	246291008	246508730	246336840
mceliece8192128	keypair	316082088	409854088	514489441	594965680
mceliece8192128f	keypair	316118712	316166640	316202817	316221040
mceliece348864	enc	42812	43832	44350	44872
mceliece460896	enc	111140	115540	117782	121460
mceliece6688128	enc	143540	149080	151721	154632
mceliece6960119	enc	156212	159116	161224	163412
mceliece8192128	enc	175840	177480	178093	178688
mceliece348864	dec	134056	134184	134745	134324
mceliece460896	dec	270444	270856	271694	271048
mceliece6688128	dec	322756	322988	323957	323148
mceliece6960119	dec	300448	300688	301480	301072
mceliece8192128	dec	325624	325744	326531	325904

Figure: Cycle counts of the three operations on Intel Haswell.

- Encapsulation and decapsulation are pretty fast.
- Key generation is pretty slow.
- "f parameter sets" are much faster in key generation.
- The corresponding software is **constant-time**.

Is key generation time a problem?

- This depends on how often you have to generate a key.
- Assuming that secret keys are never leaked, we can keep reusing the same key pair.
- But we also care about "**forward secrecy**": when machines are compromised, previous session keys are not compromised.
- Standard way to achieve forward secrecy: use short-term or even one-time keys in addition to the long-term keys.

- Can use schemes with fast key generation for short-term keys.
- Can generate a CM short-term key every hour, for example.

Is key size a problem?

	Public key	Private key	Ciphertext	Session key
mceliece348864	261120	6492	128	32
mceliece348864f	261120	6492	128	32
mceliece460896	524160	13608	188	32
mceliece460896f	524160	13608	188	32
mceliece6688128	1044992	13932	240	32
mceliece6688128f	1044992	13932	240	32
mceliece6960119	1047319	13948	226	32
mceliece6960119f	1047319	13948	226	32
mceliece8192128	1357824	14120	240	32
mceliece8192128f	1357824	14120	240	32

Figure: Key and ciphertext sizes in bytes.

Is key size a problem?

	Public key	Private key	Ciphertext	Session key
mceliece348864	261120	6492	128	32
mceliece348864f	261120	6492	128	32
mceliece460896	524160	13608	188	32
mceliece460896f	524160	13608	188	32
mceliece6688128	1044992	13932	240	32
mceliece6688128f	1044992	13932	240	32
mceliece6960119	1047319	13948	226	32
mceliece6960119f	1047319	13948	226	32
mceliece8192128	1357824	14120	240	32
mceliece8192128f	1357824	14120	240	32

Figure: Key and ciphertext sizes in bytes.

- Depends on how much data you expect to communicate in a session.
- Average webpage size is over 2MB now according to `httparchive.org`.
- Growth rate: $\approx 50\%$ from 2017 to 2021.

Is key size a problem?

	Public key	Private key	Ciphertext	Session key
mceliece348864	261120	6492	128	32
mceliece348864f	261120	6492	128	32
mceliece460896	524160	13608	188	32
mceliece460896f	524160	13608	188	32
mceliece6688128	1044992	13932	240	32
mceliece6688128f	1044992	13932	240	32
mceliece6960119	1047319	13948	226	32
mceliece6960119f	1047319	13948	226	32
mceliece8192128	1357824	14120	240	32
mceliece8192128f	1357824	14120	240	32

Figure: Key and ciphertext sizes in bytes.

- Depends on how much data you expect to communicate in a session.
- Average webpage size is over 2MB now according to `httparchive.org`.
- Growth rate: $\approx 50\%$ from 2017 to 2021.
- Classic McEliece keys are secure against reuse:
  - Clients can store frequently-used static server keys locally.
  - Servers can distribute their static keys.

Is key size a problem? (cont.)

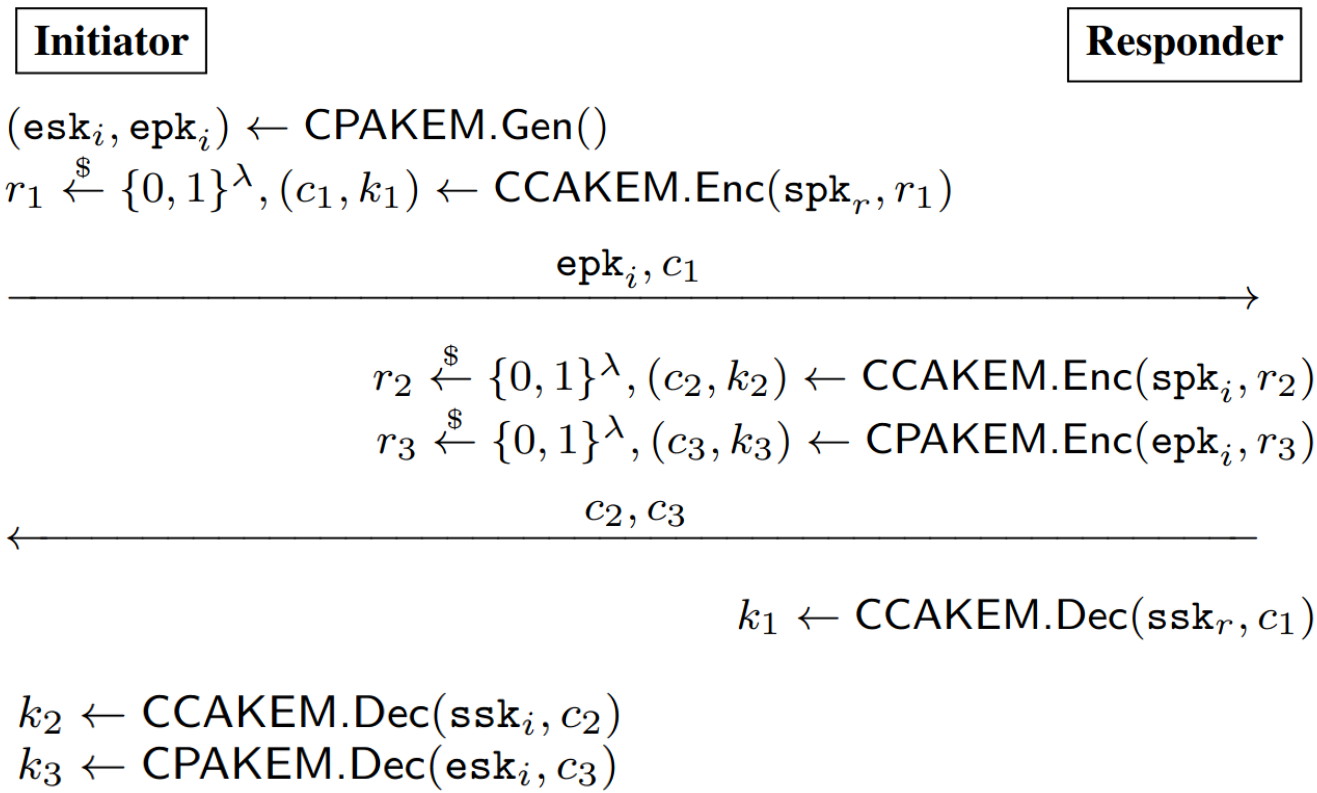
- Large public keys can cause trouble on devices with small RAMs.
- But we can make use of **streaming**.

- Encapsulation is basically doing a matrix-vector product.
- Can operate on submatrices one by one.
- The size of the submatrices can be very small.

- Public-key generation is basically computing systematic form $(I \mid A^{-1}B)$ of a matrix $(A \mid B)$.
- Can compute $A \to A^{-1} \to A^{-1}B_i$ (still need the space for A).

Post-quantum WireGuard (2021)

- A VPN protocol developed by Hulsing, Ning, Schwabe, Weber, and Zimmermann.
- Uses Classic McEliece for static keys.



BIKE: code-based small-key KEM

- Niederreiter-like
- Makes use of "QC-MDPC" codes
  - PC: parity-check
  - QC: quasi-cyclic, which means two cyclic matrix
  - MD: moderate-density = low-density (LD) = low-weight
- Example of secret key:

$$H = \left[H^{(1)} \mid H^{(2)} \right] = \left[\begin{array}{ccccc|ccccc} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{array}\right]$$

- We only need to store the first row/column of $H^{(i)}$.
- Public key H': systematic form of H (compressed)
- BIKE encryption: $c = H'e$

BIKE in the view of ring operations

- Let $\mathcal{R} = \mathbb{F}_2/(x^n - 1)$.
- A secret key is a low-weight element $(h_1, h_2) \in \mathcal{R}^2$.
- A public key is an element $h = h_2/h_1 \in \mathcal{R}$.
- The vector e can be consider as a low-weight element $(e_1, e_2) \in \mathcal{R}^2$.
- Encryption: $c = e_1 + h \cdot e_2 \in \mathcal{R}$.

Decoding LDPC codes

- BIKE decapsulation: compute $s = H^{(1)}c = H^{(1)}H'e = He$
- Consider the ith row of H: $H_i = (0, 1, 0, 0, 0, 0, 1, 0, 0, 1)$
- Main idea: if $s_i = 1$, there must exist $e_j = 1$ with $(H_i)_j = 1$.
- An LDPC decoder:
  - Set $v = (0, \ldots, 0) \in \mathbb{Z}^n$.
  - For each row $H_i \in \mathbb{F}_2^n$ of H, lift H_i to $\mathbb{Z}^n$ and add H_i to v.
  - If v_j exceeds some threshold, assume $e_j = 1$.
- The decoder can fail with some probability.
- Usually several iterations are required. There are many variants.

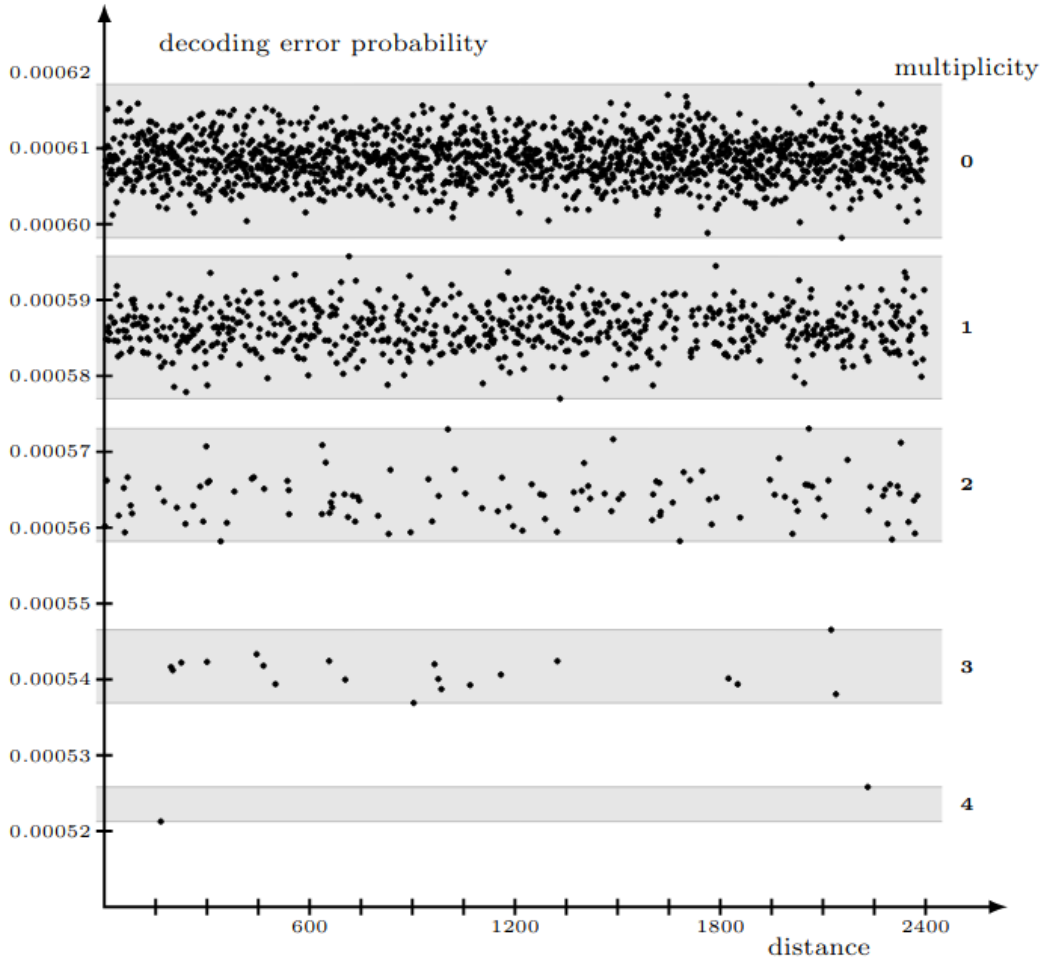
The Guo–Johansson–Stankovski attack (2016)

- Idea: send ciphertexts of special error vectors, observe whether they are successfully decrypted or not.
- One can generate many error vectors where many pairs of 1's are of distance d.
- For example, when $d = 2$, we can use

$$e = (0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0).$$

- Failure rate gets higher when there are fewer 1's of distance d in the secret key.
- GJS designed an algorithm that, given the "multiplicities" of the distances, generates the vector.
- Why the attack works is not well explained.

The Guo–Johansson–Stankovski attack (cont.)



BIKE as a submission to NISTPQC

- 3rd-round alternate candidate. Just entered the 4th round.
- Based on "*MDPC-McEliece: New McEliece variants from moderate density parity-check codes*", ISIT 2013
- Public keys are of size a few kilobytes.
- Not particularly fast, especially on embedded systems.
- The latest spec seems a bit ambiguous regarding CCA security.

> "*However, at the moment, the current analysis gives only an estimation of the DFR, and not a proven upper bound. Consequently, the BIKE instantiation with the BGF decoder does not make a formal claim for IND-CCA security, although by any practical considerations, this is probably the case.*"

ROLLO: another code-based small-key KEM

- A "rank-metric" scheme: "low-weight" now means "of low **rank weight**".
- Rank weight of a vector/matrix over $\mathbb{F}_{2^m}$: the dimension of the $\mathbb{F}_2$-subspace in $\mathbb{F}_{2^m}$ generated by the entries.

- Secret key: parity-check matrix $H \in \mathbb{F}_{2^m}^{n \times 2n}$, $H_{i,j} \in S_H \subset \mathbb{F}_{2^m}$.
- Public key: systematic form of H.
- Error vector: $e \in \mathbb{F}_{2^m}^{2n}$, $e_i \in S_e \subset \mathbb{F}_{2^m}$.

- The corresponding linear code over $\mathbb{F}_{2^m}$ is called an **LRPC code**.
- As in BIKE, some ring structure is used to reduce public key size.

Decoding LRPC code

The goal is to find S_e given $(H, s = He)$

- With S_e one can usually easily compute e
- Let $\alpha_1, \ldots, \alpha_d$ be a basis of S_H.
- Let $\beta_1, \ldots, \beta_r$ be a basis of S_e.
- Each entry of syndrome

$$s_i \in S_H S_e := \left\{ \sum_{i=1}^{d} \sum_{j=1}^{r} \alpha_i \beta_j \right\}$$

- With a high probability $S := \langle s_1, \ldots, s_n \rangle_{\mathbb{F}_2} = S_H S_e$
- A **probabilistic** decoding algorithm: return $\bigcap \alpha_i^{-1} S$
- Rationale: $R_e \subseteq \bigcap \alpha_i^{-1} S$, equality holds with a high probability

ROLLO as a submission to NISTPQC

- Entered the 2nd round.
- Did not enter the 3rd round because of new **algebraic attacks**.
  - See "*Improvements of algebraic attacks for solving the rank decoding and minrank problem*", Asiacrypt 2020
- Algebraic attacks: reducing the problem of breaking a scheme to the problem of solving multivariate (usually non-linear) equations.
- Latest parameter sets:

Instance	q	n	m	r	d	P	security	DFR
ROLLO-II-128	2	189	83	7	8	$X^{189}+X^6+X^5+X^2+1$	128	2^{-134}
ROLLO-II-192	2	193	97	8	8	$X^{193}+X^{15}+1$	192	2^{-130}
ROLLO-II-256	2	211	97	8	9	$X^{211}+X^{11}+X^{10}+X^8+1$	256	2^{-136}

Instance	pk size	sk size	ct size	Security
ROLLO-II-128	1941	40	2089	128
ROLLO-II-192	2341	40	2469	192
ROLLO-II-256	2559	40	2687	256

Exercises

- What is the probability that a random $m \times n$ matrix over $\mathbb{F}_q$ is full rank?
- Prove that the dimension of $\Gamma(L, g)$ with $\deg(g) = t$ is at least $n - mt$.
- Implement one of the ISD algorithms and figure out how many iterations it takes.
- How can we perform the Verheul–Doumen–Tilborg attack against Niederreiter? What if the decoder only works when $\mathrm{wt}(e) = t$?
- Convince yourself that the decapsulation algorithm of Classic McEliece returns $\mathrm{Hash}_{32}(0, s, c)$ with an overwhelming probability if the ciphertext is modified.
- Suppose we have a decoder that, given He with $\mathrm{wt}(e) = t$, always returns e. The output is undefined for other inputs. How can we tell whether the input is valid or not?