Overview of Code-Based Crypto Assumptions

Tanja Lange
with some slides by Tung Chou and Christiane Peters

Eindhoven University of Technology

Quantum Cryptanalysis of Post-Quantum Cryptography

Hamming code

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

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

An error-free string of 7 bits $\mathbf{b} = (b_0, b_1, b_2, b_3, b_4, b_5, b_6)$ satisfies these three equations:

$$\begin{array}{ccccccccc} b_0 & +b_1 & & +b_3 & +b_4 & & & = & 0 \\ b_0 & & +b_2 & +b_3 & & +b_5 & & = & 0 \\ & b_1 & +b_2 & +b_3 & & & +b_6 & = & 0 \end{array}$$

If one error occurred, at least one of these equations will not hold.
Failure pattern uniquely identifies the error location,
e.g., $1, 0, 1$ means

Hamming code

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

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

An error-free string of 7 bits $\mathbf{b} = (b_0, b_1, b_2, b_3, b_4, b_5, b_6)$ satisfies these three equations:

$$\begin{array}{cccccccccc} b_0 & +b_1 & & +b_3 & +b_4 & & & = & 0 \\ b_0 & & +b_2 & +b_3 & & +b_5 & & = & 0 \\ & b_1 & +b_2 & +b_3 & & & +b_6 & = & 0 \end{array}$$

If one error occurred, at least one of these equations will not hold.
Failure pattern uniquely identifies the error location,
e.g., $1, 0, 1$ means b_1 flipped.

Coding theory

- Names: code word $\mathbf{c}$, error vector $\mathbf{e}$, received word $\mathbf{b} = \mathbf{c} + \mathbf{e}$.
  length n, 2^k code words, $(n-k) \times n$ parity-check matrix H.
- Very common to transform the matrix so that the right part has just 1 on the diagonal (no need to store that).

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

- Many special constructions discovered in 65 years of coding theory:
  Fast decoding algorithm to find $\mathbf{e}$ given $\mathbf{s} = H \cdot (\mathbf{c} + \mathbf{e})$,
  whenever $\mathbf{e}$ does not have too many bits set.
- 1978 Berlekamp–McEliece–Van Tilborg:
  decoding problem is NP hard for random codes (random H).
- Use this difference in complexities for encryption.

Code-based encryption

- 1971 Goppa: Fast decoders for many matrices H.
- 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.
- 1962 Prange: simple attack idea guiding sizes in 1978 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^λ security against Prange's attack.
  Here $c_0 \approx 0.7418860694$.

Security analysis

Some papers studying algorithms for attackers:
1962 Prange; 1981 Clark–Cain, crediting Omura; 1988 Lee–Brickell; 1988 Leon; 1989 Krouk; 1989 Stern; 1989 Dumer; 1990 Coffey–Goodman; 1990 van Tilburg; 1991 Dumer; 1991 Coffey–Goodman–Farrell; 1993 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 Becker–Joux–May–Meurer; 2013 Hamdaoui–Sendrier; 2015 May–Ozerov; 2016 Canto Torres–Sendrier; 2017 Kachigar–Tillich (**post-quantum**); 2017 Both–May; 2018 Both–May; 2018 Kirshanova (**post-quantum**).

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^λ 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^λ 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.

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^λ 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}.

Decoding problem

Decoding problem: find the closest code word $\mathbf{c} \in C$ to a given $\mathbf{x} \in \mathbb{F}_2^n$, assuming that there is a unique closest code word. 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 (1978 Berlekamp–McEliece–Van Tilborg).
- Information-set decoding (see later) takes exponential time.

Different views on decoding

- The syndrome of $\mathbf{x} \in \mathbb{F}_2^n$ 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 $\mathbf{e} \in \mathbb{F}_2^n$, given $\mathbf{s} \in \mathbb{F}_2^{n-k}$, 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 $\mathbf{x} \in \mathbb{F}_2^n$ 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 $\mathbf{e} \in \mathbb{F}_2^n$, given $\mathbf{s} \in \mathbb{F}_2^{n-k}$, 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 $\mathbf{x} \in \mathbb{F}_2^n$ 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 $\mathbf{e} \in \mathbb{F}_2^n$, given $\mathbf{s} \in \mathbb{F}_2^{n-k}$, 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 $\mathbf{x} = (00 \ldots 0) || \mathbf{s}$ 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.

The Niederreiter cryptosystem I

Developed in 1986 by Niederreiter as a variant of the 1978 McEliece cryptosystem. This is the schoolbook version.

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

$$\mathbf{s} = K\mathbf{e}.$$

- Decryption: Find a n-bit vector $\mathbf{e}$ with $\mathrm{wt}(\mathbf{e}) = t$ such that $\mathbf{s} = K\mathbf{e}$.
- 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 $\mathbf{e}$ is an n-bit vector of weight t. The ciphertext $\mathbf{s}$ is the $(n-k)$-bit vector
$$\mathbf{s} = K\mathbf{e}.$$
- Decryption using secret key: Compute
$$\begin{aligned} S^{-1}\mathbf{s} &= S^{-1}K\mathbf{e} = S^{-1}(SHP)\mathbf{e} \\ &= H(P\mathbf{e}) \end{aligned}$$

  and observe that $\mathrm{wt}(P\mathbf{e}) = t$, because P permutes.
  Use efficient syndrome decoder for H to find $\mathbf{e}' = P\mathbf{e}$ and thus $\mathbf{e} = P^{-1}\mathbf{e}'$.

Note on codes

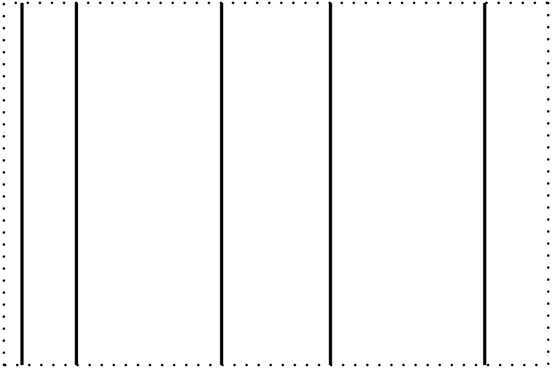
- McEliece proposed to use binary Goppa codes.
  These are still used today.
- Niederreiter described his scheme using Reed-Solomon codes.
  These were broken in 1992 by Sidelnikov and Chestakov.
- More corpses on the way: concatenated codes, Reed-Muller codes, several Algebraic Geometry (AG) codes, Gabidulin codes, several LDPC codes, cyclic codes.
- Some other constructions look OK (for now).
  NIST competition has several entries on QCMDPC codes.
- Rank-metric codes in NIST competition got some scratches
  (2020 Bardet, Briaud, Bros, Gaborit, Neiger, Ruatta, Tillich).

Security notions and codes

- McEliece/Niederreiter are One-Way Encryption (OWE) schemes.
- The schemes as presented are not CCA–II secure. Fix by using CCA–II transformation (e.g. Fujisaki-Okamoto transform) and turn into KEM by picking random $\mathbf{e}$ of weight t, use hash($\mathbf{e}$) as secret key to encrypt and authenticate (for McEliece or Niederreiter).
- Breaking OWE implies distinguishing key from random or breaking one-wayness for random key.
- We distinguish between generic attacks (such as information-set decoding) and structural attacks (that use the structure of the code).
- Gröbner basis computation is a generally powerful tool for structural attacks.

Generic attack: Brute force

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

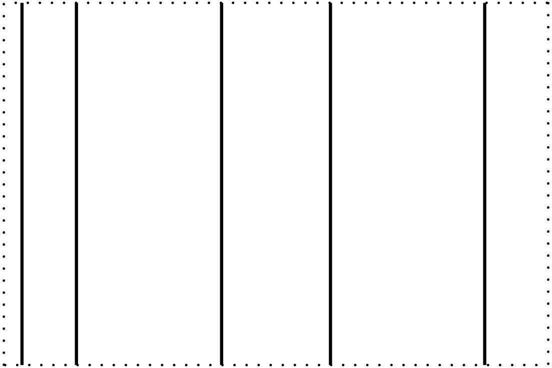
$K =$ 

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

Cost:

Generic attack: Brute force

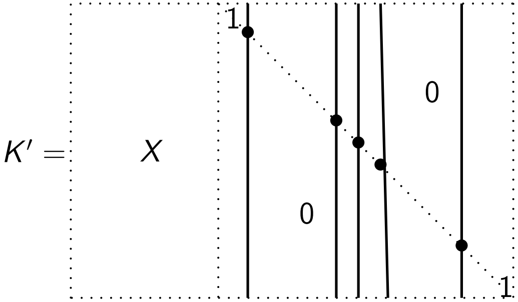
Given K and $\mathbf{s} = K\mathbf{e}$, find $\mathbf{e}$ with $\mathrm{wt}(\mathbf{e}) = t$.

$K =$ 

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

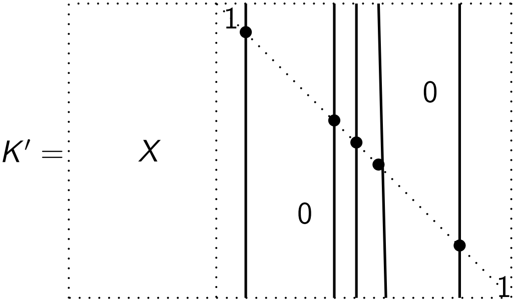
Cost: $\binom{n}{t}$ sums of t columns.
Can do better so that each try costs only 1 column addition
(after some initial additions).

Generic attack: Information-set decoding, 1962 Prange



1. Permute K and bring to systematic form $K' = (X|I_{n-k})$.
   (If this fails, repeat with other permutation).
2. Then $K' = UKP$ 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 $\mathbf{e}'$.
   Else return to 1 to rerandomize.

Generic attack: Information-set decoding, 1962 Prange

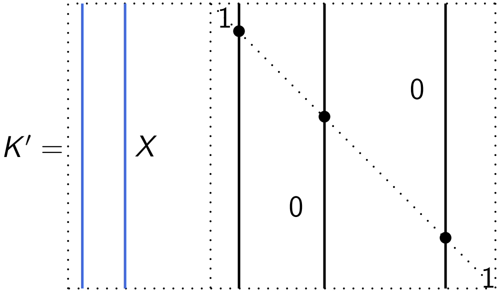


Cost: $O\left(\binom{n}{t} / \binom{n-k}{t}\right)$ matrix operations.

2010 Bernstein: Grover speedup to $O\left(\sqrt{\binom{n}{t} / \binom{n-k}{t}}\right)$

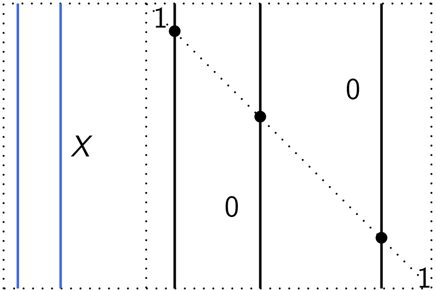
1. Permute K and bring to systematic form $K' = (X|I_{n-k})$. (If this fails, repeat with other permutation).
2. Then $K' = UKP$ for some permutation matrix P and U the matrix that produces systematic form.
3. This updates $\mathbf{s}$ to $U\mathbf{s}$.
4. If $\mathrm{wt}(U\mathbf{s}) = t$ then $\mathbf{e}'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

Lee–Brickell attack



1. Permute K and bring to systematic form $K' = (X|I_{n-k})$.
   (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 $\mathbf{e}'$.
   Else return to 2 or return to 1 to rerandomize.

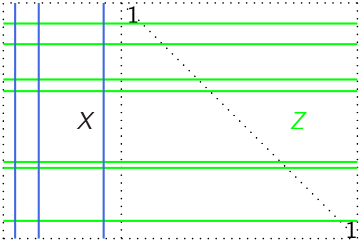
Lee–Brickell attack

$K' =$ 

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

1. Permute K and bring to systematic form $K' = (X|I_{n-k})$.
   (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 $\mathbf{e}'$.
   Else return to 2 or return to 1 to rerandomize.

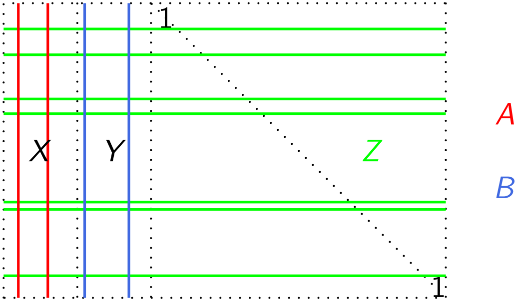
Leon's attack



$\underbrace{\hspace{3em}}_{(n-k)\times(n-k)}$ identity matrix

- 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 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. Else compute $X\mathbf{p}$.
  4. If $\mathrm{wt}(\mathbf{s} + X\mathbf{p}) = t - p$ output unpermuted version of $\mathbf{e}' = \mathbf{p}||(\mathbf{s} + X\mathbf{p})$. Else return to 1 or rerandomize K.
- 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 K' into two disjoint subsets X and Y.
  - Let $A = \{\mathbf{a} \in \mathbb{F}_2^{k/2} | \mathrm{wt}(\mathbf{a}) = p\}$, $B = \{\mathbf{b} \in \mathbb{F}_2^{k/2} | \mathrm{wt}(\mathbf{b}) = p\}$.
  - 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.

Binary Goppa code

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

- a list $L = (a_1, \dots, a_n)$ 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 $\mathbb{F}_q$.

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

$$\left\{ \mathbf{c} \in \mathbb{F}_2^n \,\middle|\, S(\mathbf{c}) = \frac{c_1}{x - a_1} + \frac{c_2}{x - a_2} + \cdots + \frac{c_n}{x - a_n} \equiv 0 \bmod g(x) \right\}$$

- This code is linear $S(\mathbf{b} + \mathbf{c}) = S(\mathbf{b}) + S(\mathbf{c})$ and has length n.
- Bounds on dimension $k \geq n - mt$ and minimum distance $d \geq 2t + 1$.

How to hide nice code?

- Do not reveal matrix H related to nice-to-decode code.
- Pick a random invertible $(n-k) \times (n-k)$ matrix S and random $n \times n$ *permutation matrix* P. Put
$$K = SHP.$$
- K is the public key and S and P together with a decoding algorithm for H form the private key.
- For suitable codes K looks like random matrix.
- How to decode syndrome $\mathbf{s} = K\mathbf{e}$?

How to hide nice code?

- Do not reveal matrix H related to nice-to-decode code.
- Pick a random invertible $(n-k) \times (n-k)$ matrix S and random $n \times n$ *permutation matrix* P. Put
$$K = SHP.$$
- K is the public key and S and P together with a decoding algorithm for H form the private key.
- For suitable codes K looks like random matrix.
- How to decode syndrome $\mathbf{s} = K\mathbf{e}$?
- Computes $S^{-1}\mathbf{s} = S^{-1}(SHP)\mathbf{e} = H(P\mathbf{e})$.
- P permutes, thus $P\mathbf{e}$ has same weight as $\mathbf{e}$.
- Decode to recover $P\mathbf{e}$, then multiply by P^{-1}.

How to hide nice code?

- For Goppa code use secret polynomial $g(x)$.
- Use secret permutation of the a_i, this corresponds to secret permutation of the n positions; this replaces P.
- 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 $(n-k) \times k$.
- Secret key is polynomial g and support $L = (a_1, \ldots, a_n)$.

How to hide nice code?

- For Goppa code use secret polynomial $g(x)$.
- Use secret permutation of the a_i, this corresponds to secret permutation of the n positions; this replaces P.
- 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 $(n-k) \times k$.
- Secret key is polynomial g and support $L = (a_1, \ldots, a_n)$.
- 2000 Sendrier (support splitting) computes code equivalence in polynomial time, but there are **many** codes.

NIST submission Classic McEliece

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

Metric	**mceliece6960119**	**mceliece8192128**
Public-key size	1047319 bytes	1357824 bytes
Secret-key size	13908 bytes	14080 bytes
Ciphertext size	226 bytes	240 bytes
Key-generation time	813812960 cycles	898881136 cycles
Encapsulation time	156624 cycles	172576 cycles
Decapsulation time	298472 cycles	316888 cycles

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