Introduction to Code-Based Cryptography

Paulo S. L. M. Barreto

Universidade de São Paulo
USP

Objectives

- 1st Part:
  - Basics of coding theory (notation).
  - Panorama of code-based cryptosystems.

- 2nd Part:
  - Security considerations.
  - Choice of codes.
  - Implementation issues.
  - Research problems.

CODING THEORY

Linear Codes

- Let $q = p^m$ for some prime p and $m > 0$.
- A *linear* $[n, k]$-*code* $\mathcal{C}$ over $\mathbb{F}_q$ is a k-dimensional vector subspace of $\mathbb{F}_q^n$.
- Let $d \mid m$ and let $s = p^d$, so that $\mathbb{F}_p \subseteq \mathbb{F}_s \subseteq \mathbb{F}_q$.
- An $\mathbb{F}_s$-*subfield subcode* of a code $\mathcal{C}$ is the subspace of $\mathcal{C}$ consisting of all words with all components in $\mathbb{F}_s$.

Weight and Distance

- The (Hamming) *weight* of $u \in \mathbb{F}_q^n$ is the number of nonzero components of u: $\mathrm{wt}(u) \coloneqq \#\{\, j \mid u_j \neq 0 \,\}$.
- The (Hamming) *distance* between $u, v \in \mathbb{F}_q^n$ is $\mathrm{dist}(u, v) \coloneqq \mathrm{wt}(u - v)$.
- The *minimum distance* of a code $\mathcal{C}$ is $\mathrm{dist}(\mathcal{C}) \coloneqq \min\{\, \mathrm{dist}(u, v) \mid u, v \in \mathcal{C}, u \neq v \,\}$.
- Determining $\mathrm{dist}(\mathcal{C})$ is *NP*-hard.

Generator and Parity-Check

- A *generator matrix* for an $[n, k]$-code $\mathcal{C}$ is a matrix $G_{k\times n} \in \mathbb{F}_q^{k\times n}$ whose rows form a basis of $\mathcal{C}$: $\mathcal{C} = \left\{ uG \in \mathbb{F}_q^n \mid u \in \mathbb{F}_q^k \right\}$.
- A parity-check matrix for the same code is a matrix $H_{r\times n} \in \mathbb{F}_q^{r\times n}$ whose rows form a basis for the orthogonal code, with $n = r + k$: $\mathcal{C} = \left\{ v \in \mathbb{F}_q^n \mid vH^T = 0^r \right\}$.
- Therefore $(uG)H^T = u(GH^T) = 0^r$ for all u, i.e. $GH^T = 0^{k\times r}$.

General & Syndrome Decoding (GDP/SDP)

- **GDP**
- **Input:**
  - positive integers n, k, t;
  - generator matrix $G \in \mathbb{F}_q^{k \times n}$;
  - vector $c \in \mathbb{F}_q^n$.
- **Question:** $\exists ?\ m \in \mathbb{F}_q^k$ such that $e := c - mG$ has weight $\mathrm{wt}(e) \leq t$?

- **SDP**
- **Input:**
  - positive integers n, r, t;
  - parity-check matrix $H \in \mathbb{F}_q^{r \times n}$;
  - vector $s \in \mathbb{F}_q^r$.
- **Question:** $\exists ?\ e \in \mathbb{F}_q^n$ of weight $\mathrm{wt}(e) \leq t$ such that $He^\mathsf{T} = s^\mathsf{T}$?

Both are NP-complete!

Code-Based Cryptography

- There exist codes for which efficient decoders are known.
- Cryptosystems naturally follow if:
  - the decoding trapdoor can be securely hidden;
  - the GDP/SDP remains intractable on average for those codes.
- (Obs.: from now on, *binary* codes)

CODE-BASED CRYPTOSYSTEMS

Chronology

- 1978: McEliece (encryption)
- 1986: Niederreiter (encryption)
- 1993: Stern (identification)
- 2001: CFS (signatures)
- 2009: Cayrel et al. (id-based identification)
- … (other, more arcane schemes)

McEliece Cryptosystem

- Key generation:
  - Choose a secure, uniformly random t-error correcting $[n, k]$-code $\mathcal{C}$ over $\mathbb{F}_2$, equipped with a decoding trapdoor, usually a parity-check matrix $\widehat{H} \in \mathbb{F}_2^{r \times n}$ of some unique form.
  - Compute for $\mathcal{C}$ a systematic generator matrix $G \in \mathbb{F}_2^{k \times n}$.
  - Set $sk = \widehat{H}$, $pk = (G, t)$.

McEliece Cryptosystem

- "Hey, wait, I know McEliece, and this does not look quite like it!"
- Textbook version:
  - computing some (private, highly structured) $\hat{G}$ from $\widehat{H}$
  - hide it as $G = S\hat{G}P$ (with S invertible, P a permutation).
- Does not increase semantic security, is less efficient, and can actually leak side-channel information.
- The description here is simpler, more efficient, and more secure.

McEliece Cryptosystem

- Encryption of a plaintext $m \in \mathbb{F}_2^k$:
  - Choose a uniformly random t-error vector $e \in \mathbb{F}_2^n$ and compute $c \leftarrow mG + e \in \mathbb{F}_2^n$ (IND-CCA2 variant via e.g. Fujisaki-Okamoto).
- Decryption of a ciphertext $c \in \mathbb{F}_2^n$:
  - Compute the (private) syndrome $s \leftarrow c\widehat{H}^T = e\widehat{H}^T$ and decode it to obtain e.
  - Obtain m as the first k components of $c - e$.

McEliece/Fujisaki-Okamoto: Setup

- Random oracles (message authentication code and symmetric cipher) a
  $\mathcal{H} \;:\; \mathbb{F}_2^k \times \{0{,}1\}^* \to \mathbb{Z}/\binom{n}{t}\mathbb{Z}$,
  $\mathcal{E} \;:\; \mathbb{F}_2^k \to \{0{,}1\}^*$.
- (Un)ranking function $\mathcal{U} : \mathbb{Z}/\binom{n}{t}\mathbb{Z} \to \mathcal{B}_t(0^n)$.
- Decoding algorithm $\mathcal{D} :\; \mathbb{F}_2^r \to \mathcal{B}_t(0^n)$ such that $\mathcal{D}\left(e\widehat{H}^T\right) = e$ for all $e \in \mathcal{B}_t(0^n)$.

McEliece/Fujisaki-Okamoto: Encryption

- Input: message $m \in \{0,1\}^*$.
- Output: ciphertext $c \in \mathbb{F}_2^n \times \{0,1\}^*$.
- Algorithm:
  - $z \xleftarrow{\$} \mathbb{F}_2^k$
  - $h \leftarrow \mathcal{H}(z, m)$, $e \leftarrow \mathcal{U}(h)$
  - $w \leftarrow zG + e$
  - $d \leftarrow \mathcal{E}(z) \oplus m$
  - $c \leftarrow (w, d)$

McEliece/Fujisaki-Okamoto: Decryption

- Input: ciphertext $c = (w, d) \in \mathbb{F}_2^n \times \{0,1\}^*$.
- Output: message $m \in \{0,1\}^*$, or rejection.
- Algorithm:
  - $s \leftarrow w\widehat{H}^T, e \leftarrow \mathcal{D}(s), z \leftarrow (w - e)|_k$
  - $m \leftarrow \mathcal{E}(z) \oplus d$
  - $h \leftarrow \mathcal{H}(z, m)$, $v \leftarrow \mathcal{U}(h)$
  - accept $\Leftrightarrow$ $v = e$

Niederreiter Cryptosystem

- Setup:
  - Semantically secure symmetric cipher $\mathcal{E} : \mathcal{B}_t(0^n) \times \{0,1\}^* \to \{0,1\}^* \cup \{\perp\}$.
- Key generation:
  - Choose a secure, uniformly random t-error correcting $[n,k]$-code $\mathcal{C} \subset \mathbb{F}_2^n$, equipped with a decoding-friendly parity-check matrix $\widehat{H} \in \mathbb{F}_2^{r\times n}$ and an efficient decoding algorithm $\mathcal{D} : \mathbb{F}_2^r \to \mathcal{B}_t(0^n)$.
  - Compute the systematic parity-check matrix $H \in \mathbb{F}_2^{r\times n}$ such that $\widehat{H} = \widehat{M}H$ for some nonsingular matrix $\widehat{M} \in \mathbb{F}_2^{r\times r}$.
  - Set $sk = (\widehat{M}, \widehat{H})$, $pk = (H, t)$.

Niederreiter Cryptosystem

- Encryption of plaintext $m \in \{0,1\}^*$:
  - $e \xleftarrow{\$} \mathcal{B}_t(0^n)$
  - $s \leftarrow eH^T$
  - $d \leftarrow \mathcal{E}(e, m)$
  - $c \leftarrow (s, d)$
- Decryption of cryptogram $(s, d) \in \mathbb{F}_2^r \times \{0,1\}^*$:
  - $\hat{s} \leftarrow s\widehat{M}^T$ // NB: $\hat{s} = (eH^T)\widehat{M}^T = e\left(\widehat{M}H\right)^T = e\widehat{H}^T$ (therefore $\hat{s}$ is $\widehat{H}$-decodable to e)
  - $e \leftarrow \mathcal{D}(\hat{s})$
  - $m \leftarrow \mathcal{E}^{-1}(e, d)$
  - accept ⇔ $m \neq \perp$

CFS Signatures

- System setup:
  - Random oracle $\mathcal{H} : \{0,1\}^* \times \mathbb{N} \to \mathbb{F}_2^r$.
- Key generation:
  - Choose a secure, uniformly random t-error correcting $[n, k]$-code $\Gamma \subset \mathbb{F}_2^n$ *with a high density of decodable syndromes*, equipped with a decoding-friendly parity-check matrix $\widehat{H} \in \mathbb{F}_2^{r \times n}$ and an efficient decoding algorithm $\mathcal{D} : \mathbb{F}_2^r \to \mathcal{B}_t(0^n)$.
  - Compute the systematic parity-check matrix $H \in \mathbb{F}_2^{r \times n}$ such that $\widehat{H} = \widehat{M} H$ for some nonsingular matrix $\widehat{M} \in \mathbb{F}_2^{r \times r}$.
  - Set $sk = (\widehat{M}, \widehat{H})$, $pk = (H, t)$.

CFS Signatures

- Signing a message $m \in \{0,1\}^*$:
  - Find $i \in \mathbb{N}$ such that, for $c \leftarrow \mathcal{H}(m,i)$ and $\hat{c} \leftarrow c\widehat{M}^T$, $\hat{c}$ is $\widehat{H}$-decodable.
  - $e \leftarrow \mathcal{D}(\hat{c})$
  - $\sigma \leftarrow (e,i)$ // NB: $c\widehat{M}^T = \hat{c} = e\widehat{H}^T = e\left(\widehat{M}H\right)^T = (eH^T)\widehat{M}^T$, hence $c = eH^T$, i.e. c is the (public) H-syndrome of e.
- Verifying a signature $\sigma = (e,i) \in \mathcal{B}_t(0^n) \times \mathbb{N}$:
  - $c \leftarrow eH^T$
  - accept ⇔ $c = \mathcal{H}(m,i)$.

CFS Signatures

- Best known codes for CFS instantiation: Goppa codes (highest density of decodable syndromes).
- Bad news:
  - number of possible hash values: $2^r \approx n^t$
  - number of decodable syndromes: $\approx \binom{n}{t} \approx \frac{n^t}{t!}$.
  - probability of finding a codeword of weight t: $\approx 1/t!$
  - expected value of steps to sign: $\approx t!$ ☹

CFS Signatures

- If the n-bit error e of weight t is encoded via permutation ranking, the signature length is $\approx \lg(n^t/t!) + \lg(t!) = t \lg n \approx mt$.
- Public key is huge: mtn bits.
- Key sizes for usual sec levels are several MiB long, coupled with very long processing times ☹

CFS Signatures

- Bleichenbacher's attack: Wagner's generalized (3-way) birthday attack $\Rightarrow$ security level lower than expected.
- Larger key sizes, longer signature generation.
- Dyadic keys: shorter by a factor u = largest power of 2 dividing t, but longer signature generation times.

m	t=9	t=10	t=11	t=12
15				**0.7**
16				**1.5**
17		(sizes in MiB)		**3.2**
18				**6.75**
		...		
22	**99**	**110**	**121**	**132**

Stern Identification

- $H \xleftarrow{\$} \mathbb{F}_2^{r \times n}$: uniformly random, systematic binary parity-check matrix (e.g. $n = 2r$).
- Gaborit-Girault improvement: uniformly random *quasi-cyclic* $H = [C \mid I]$, with $C_{ij} := h_{(j-i) \bmod r}$ for some $h \xleftarrow{\$} \mathbb{F}_2^r$.
- Key pair:
  - Private key: $e \xleftarrow{\$} \mathcal{B}_t(0^n)$.
  - Public key: $s \leftarrow eH^T \in \mathbb{F}_2^r$.

Stern Identification

- Commitment:
  - The prover chooses a uniformly random word $u \xleftarrow{\$} \mathbb{F}_2^n$ and a uniformly random permutation $\sigma \xleftarrow{\$} S_n$ on $\{0 \dots n-1\}$.
  - The prover sends to the verifier:
    - $c_0 \leftarrow \mathcal{H}\big(\sigma(u)\big)$,
    - $c_1 \leftarrow \mathcal{H}\big(\sigma(e+u)\big)$, and
    - $c_2 \leftarrow \mathcal{H}(\sigma \,||\, uH^T)$.

Stern Identification

- Challenge & Response:
  - The verifier sends a uniformly random $b \xleftarrow{\$} \{0, 1, 2\}$ to the prover.
  - The prover responds by revealing:
    - $e + u$ and σ if $b = 0$;
    - u and σ if $b = 1$;
    - $\sigma(e)$ and $\sigma(u)$ if $b = 2$.

Stern Identification

- Verification:
  - The verifier verifies that:
    - c_1 and c_2 are correct if $b = 0$ (noticing that $uH^T = (e+u)H^T + eH^T = (e+u)H^T + s$);
    - c_0 and c_2 are correct if $b = 1$;
    - c_0 and c_1 are correct and $\mathrm{wt}\big(\sigma(e)\big) = t$ if $b = 2$ (noticing that $\sigma(e+u) = \sigma(e) + \sigma(u)$).
  - The probability of cheating in this ZKP is $2/3$. Repeating $\lceil(\lg \varepsilon)/(1 - \lg 3)\rceil$ times reduces the cheating probability below ε.

SFS Signatures

- Commitments:

for $i \leftarrow 0 \dots N-1$ **do**

$$u_i \xleftarrow{\$} \mathbb{F}_2^n,\ \sigma_i \xleftarrow{\$} S_n$$

$$c_{i,0} \leftarrow \mathcal{H}(\sigma_i(u_i))$$

$$c_{i,1} \leftarrow \mathcal{H}(\sigma_i(e + u_i))$$

$$c_{i,2} \leftarrow \mathcal{H}(\sigma_i \mid\mid u_i H^T)$$

end

- Challenges:

$$(b_0, \dots, b_{N-1}) \leftarrow \mathcal{H}^*\big(M;\ c_{0,0}||c_{0,1}||c_{0,2};\ \dots;\ c_{N-1,0}||c_{N-1,1}||c_{N-1,2}\big)$$

SFS Signatures

- Responses:

  for $i \leftarrow 0 \ldots N-1$ **do**

  if $b_i = 0$ **then** $\rho_i \leftarrow \left(c_{i,0}; e + u_i\ ; \sigma_i\right)$

  if $b_i = 1$ **then** $\rho_i \leftarrow \left(c_{i,1}; u_i; \sigma_i\right)$

  if $b_i = 2$ **then** $\rho_i \leftarrow \left(c_{i,2}; \sigma_i(u_i); \sigma_i(e)\right)$

  end

- Signature:

  $\Sigma \leftarrow (b_0, \rho_0; \ldots; b_{N-1}, \rho_{N-1})$

SFS Signatures

- Verification:

  for $i \leftarrow 0 \ldots N-1$ **do**

  if $b_i = 0$ **then**

  $c_{i,1} \leftarrow \mathcal{H}(\sigma_i(e+u_i))$, $c_{i,2} \leftarrow \mathcal{H}(\sigma_i \mid\mid (e+u_i)H^T + s)$

  if $b_i = 1$ **then**

  $c_{i,0} \leftarrow \mathcal{H}(\sigma_i(u_i))$, $c_{i,2} \leftarrow \mathcal{H}(\sigma_i \mid\mid u_i H^T)$

  if $b_i = 2$ **then**

  $c_{i,0} \leftarrow \mathcal{H}(\sigma_i(u_i))$, $c_{i,1} \leftarrow \mathcal{H}(\sigma_i(e) + \sigma_i(u_i))$

  if $\mathrm{wt}(\sigma_i(e)) \neq t$ **then** "reject"

  end

SFS Signatures

- Verification:
  $(b'_0, \dots, b'_{N-1}) \leftarrow \mathcal{H}^*\big(M;\ c_{0,0}||c_{0,1}||c_{0,2};\ \dots; c_{N-1,0}||c_{N-1,1}||c_{N-1,2}\big)$
  if $(b'_0, \dots, b'_{N-1}) \neq (b_0, \dots, b_{N-1})$ **then** "reject" **else** "accept"

- Signature size?
- N elements of form $\left[b_i, \left(c_{i,b_i}; v_i\ ; \sigma_i\right)\right] \in \{0 \dots 2\} \times \{0 \dots 2^h - 1\} \times \mathbb{F}_2^n \times S_n$ or $\left[b_i, \left(c_{i,b_i}; v_i\ ; \sigma_i(e)\right)\right] \in \{0 \dots 2\} \times \{0 \dots 2^h - 1\} \times \mathbb{F}_2^n \times \mathcal{B}_t(0^n)$.
- Hence $\approx 1.36h + N \cdot (h + n + ((2n + t)/3) \lg n)$ bits.

AGS Identification

- Aguilar-Gaborit-Schrek: identification in the GDP (rather than SDP) setting.
- $G \stackrel{\$}{\leftarrow} \mathbb{F}_2^{k \times n}$: uniformly random, systematic, quasi-cyclic binary generator matrix (usually $n = 2k$, $G = [I \mid C^T]$).
- Key pair:
  - Private key: $e \stackrel{\$}{\leftarrow} \mathcal{B}_t(0^n)$, $m \stackrel{\$}{\leftarrow} \mathbb{F}_2^k$.
  - Public key: $c \leftarrow mG + e \in \mathbb{F}_2^n$.

AGS Identification

- Commitment 1:
  - The prover chooses a uniformly random word $u \xleftarrow{\$} \mathbb{F}_2^k$ and a uniformly random permutation $\sigma \xleftarrow{\$} S_n$ on $\{0 \ldots n-1\}$.
  - The prover sends to the verifier:
    - $c_0 \leftarrow \mathcal{H}(\sigma)$,
    - $c_1 \leftarrow \mathcal{H}\big(\sigma(uG)\big)$.

AGS Identification

- Challenge 1:
  - The verifier chooses a uniformly random $r \xleftarrow{\$} \{0, \ldots, k-1\}$ and sends it to the prover.
- Commitment 2:
  - The prover sends to the verifier:
    - $c_2 \leftarrow \mathcal{H}\left(\sigma\left(uG + \mathrm{rot}_r(e)\right)\right)$

AGS Identification

- Challenge 2 & Response:
  - The verifier sends a uniformly random $b \xleftarrow{\$} \{0,1\}$ to the prover.
  - The prover responds by revealing:
    - σ and $\mathrm{rot}_r(m) + u$ if $b = 0$;
    - $\sigma(uG)$ and $\sigma(\mathrm{rot}_r(e))$ if $b = 1$.

AGS Identification

- Verification:
  - The verifier verifies that:
    - c_0 and c_2 are correct if $b = 0$, noticing that $(\mathrm{rot}_r(m) + u)G = \mathrm{rot}_r(mG) + uG = \mathrm{rot}_r(mG + e) + \mathrm{rot}_r(e) + uG = \mathrm{rot}_r(c) + uG + \mathrm{rot}_r(e)$, hence $\sigma\big(uG + \mathrm{rot}_r(e)\big) = \sigma\big((\mathrm{rot}_r(m) + u)G + \mathrm{rot}_r(c)\big)$;
    - c_1 and c_2 are correct and $\mathrm{wt}\big(\sigma(\mathrm{rot}_r(e))\big) = t$, if $b = 1$.
  - The probability of cheating in this ZKP is $1/2$. Repeating $\lceil -\lg \varepsilon \rceil$ times reduces the cheating probability below ε.

Stern & AGS Keys

- Gaborit-Girault propose $r = 347$, $t = 76$ to achieve 2^{83} security.
- Modern recommendation would be $r = 449$, $t = 99$ for 2^{80} security, or (better yet) $r = 727$, $t = 160$ for 2^{128} security.
- Private and public keys are very short (respectively $2r$ and r bits long).
- Signatures are possible via the Fiat-Shamir heuristics, but rather large (e.g. ≈ 122 KiB at 2^{80} security).

Identity-Based Identification

- Cayrel *et al.*: Goppa trapdoor for the Stern scheme combined with CFS signatures.
- Stern parameter H is the KGC's CFS public key.
- Stern public key is the user's identity mapped to a decodable syndrome (N.B. necessary to increase weight to cover radius $t + \delta$, otherwise the scheme is *not* id-based).
- Identity-based private key is a CFS signature of the user's identity, i.e. an error vector of weight $t + \delta$ computed by the KGC.

Hashing

- Cryptographic hash functions must be preimage and collision resistant.
- FSB (Fast Syndrome-Based hash) and RFSB (Really Fast Syndrome-Based hash): collision resistance related to the hardness of the SDP.
- In practice, *slow*, and security not particularly impressive ☹

QUESTIONS?

Objectives

- 1st Part:
  - Basics of coding theory (notation).
  - Panorama of code-based cryptosystems.
- 2nd Part:
  - Security considerations.
  - Choice of codes.
  - Implementation issues.
  - Research problems.

INFORMATION SET DECODING

Definition: IS

- Let $\mathcal{C} := \left\{ uG \in \mathbb{F}_2^n \mid u \in \mathbb{F}_2^k \right\}$ be a linear t-error correcting code specified by a generator matrix $G \in \mathbb{F}_2^{k \times n}$, and let $c = uG + e$ be a blurry codeword where $\mathrm{wt}(e) \leq t$.
- An *information set* for the error pattern e is a subset $\mathcal{J} \subseteq \{0, \dots, n-1\}$ such that $e_j = 0$ for all $j \in \mathcal{J}$.
- In other words, c_j is correct at all positions indicated by $\mathcal{J}$.

Decoding with an IS

- Let $\#\mathcal{J} = k$, and let $c|_{\mathcal{J}} \in \mathbb{F}_2^k$ and $G|_{\mathcal{J}} \in \mathbb{F}_2^{k\times k}$ denote the restrictions of c and G to the columns indicated in $\mathcal{J}$. Then $c|_{\mathcal{J}} = uG|_{\mathcal{J}}$.
- If $G|_{\mathcal{J}}$ is invertible, then $u = \left(c|_{\mathcal{J}}\right) \cdot \left(G|_{\mathcal{J}}\right)^{-1}$.
- The process of recovering u from c and G with this method is called *information set decoding*.

Cost Estimate

- Let $\mathcal{J}$ be an IS with $\#\mathcal{J} = s$. The probability that $\mathcal{J} \cup \{j\}$ remains an IS for some uniformly random $j \in \{0, \ldots, n-1\} \setminus \mathcal{J}$ is $1 - t/(n-s)$, since t out of the $n - s$ values in $\{0, \ldots, n-1\} \setminus \mathcal{J}$ correspond to error positions.
- Hence the probability that a uniformly random $\mathcal{J} \subseteq \{0, \ldots, n-1\}$ with $\#\mathcal{J} = k$ is an IS is $\prod_{0 \le s \le k-1} \big(1 - t/(n-s)\big)$.

Cost Estimate

- The decoding cost (or work factor, WF) is slightly increased (by a factor $1/Q_2$ where $Q_2 \approx 0.2887881$) due to the need that $G|_{\mathcal{J}}$ be invertible, i.e.
  - $WF(n,k,t) = (1/Q_2)\prod_{0 \le s \le k-1}\big(1 - t/(n-s)\big)$.

- Examples:
  - $\lg WF(2304, 1280, 64) \approx 78$
  - $\lg WF(4096, 2048, 128) \approx 132.5$
  - $\lg WF(8192, 4096, 256) \approx 263.5$

Cost Estimate

- This estimate assigns unit cost to the whole check that a certain IS leads to a solution.
- Dynamic programming techniques (as well as implementation cleverness) make the amortized cost of each step very light.

Other Attacks

- Message recovery vs. key recovery.
- Finding low-weight codewords in related codes (e.g. in the dual).
- Exploiting the algebraic structure (e.g. properties of the underlying field or mapping to other computational problems like solving MQ systems).
- Exploiting symmetries (e.g. quasi-cyclic).
- Implementation attacks (e.g. timing).

CHOOSING THE CODE

Which Code to Choose?

- Not all codes are suitable for cryptography.
- Needed: code equipped with a trapdoor that can be easily and securely hidden.
- Most popular choice: Goppa codes.
  - … except for a few weak cases, e.g. binary Goppa polynomial (Loidreau-Sendrier 1998).
  - … distinguishing a Goppa code from a random code of the same length can be done in $\tilde{O}(n^2)$ time (Márquez-Corbella, Martínez-Moro and Pellikaan 2013).

Goppa Codes

- Let $g(x) \coloneqq \sum_{i=0}^{t} g_i x^i \in \mathbb{F}_{2^m}[x]$ be a monic $(g_t = 1)$ polynomial.
- Let $L \coloneqq (L_0, \dots, L_{n-1}) \in \mathbb{F}_{2^m}^n$ (all distinct) such that $g(L_j) \neq 0$ for all j. This is called the *support*.
- Properties:
  - Easy to generate and plentiful.
  - Usually $g(x)$ is chosen to be irreducible; if so, $\mathbb{F}_{(2^m)^t} = \mathbb{F}_{2^m}[x]/g(x)$.

Goppa Codes

- The *Goppa syndrome function* is the linear map $S : \mathbb{F}_2^n \to \mathbb{F}_{2^m}[x]/g(x)$:
  $S_c(x) \coloneqq \sum_{i=0}^{n-1} \frac{c_i}{x - L_i} = \sum_{c_i \neq 0} \frac{1}{x - L_i} \pmod{g(x)}$.
- The *Goppa code* $\Gamma(L, g)$ is the kernel of the Goppa syndrome function, i.e.
  $\Gamma(L, g) = \{c \in \mathbb{F}_2^n \mid S_c(x) \equiv 0\}$.

Distance of a Goppa code

- In general the minimum distance of $\Gamma(L, g)$ is only known to be $d \geq t + 1$.
- In the *binary* case when $g(x)$ is *square-free* (e.g. when $g(x)$ is irreducible) the minimum distance becomes $d \geq 2t + 1$.
- How do we correct errors/decode?

Error Locator Polynomial

- Efficient decoding procedure for known g and L via the (Patterson) *error locator polynomial*:

  $$\sigma(x) := \prod_{e_i \neq 0} (x - L_i) \in \mathbb{F}_{2^m}[x]/g(x).$$

- Property: $\sigma(L_i) = 0 \Leftrightarrow e_i = 1$.

The Key Equation

- $\sigma(x) = \prod_i (x - L_i)^{e_i}$.

- $\sigma'(x) = \sum_i e_i (x - L_i)^{e_i - 1} \prod_{j \neq i} \left(x - L_j\right)^{e_j} = \sum_i \frac{e_i}{x - L_i} \prod_j \left(x - L_j\right)^{e_j} = \sum_i \frac{e_i}{x - L_i} \sigma(x)$.

- $\therefore \sigma'(x) = \sigma(x) S_e(x) \bmod g(x)$.

Error Correction

- Let $m \in \Gamma(L, g)$, let $e \in \mathbb{F}_2^n$ be an error vector of weight $wt(e) \leq t$, and $c = m \oplus e$.
- Compute the syndrome of e through the relation $S_e(x) = S_c(x)$.
- Compute the error locator polynomial σ from the syndrome.
- Determine which L_i are zeroes of σ, thus retrieving e and recovering m.

Error Correction

- Let $s(x) \coloneqq S_e(x)$. If $s(x) \equiv 0$, nothing to do (no error), otherwise $s(x)$ is invertible.
  - Property #1: $\sigma(x) = a(x)^2 + xb(x)^2$.
  - Property #2: $\sigma'(x) = b(x)^2$.
  - Property #3: $\sigma'(x) = \sigma(x)s(x)$.
- Thus $b(x)^2 = (a(x)^2 + xb(x)^2)s(x)$, hence
  $$\underbrace{a(x) = b(x)v(x)}_{\text{Extended Euclid!}} \text{ with } v(x) = \sqrt{x + \underbrace{1/s(x)}_{\text{Extended Euclid!}}} \bmod g(x).$$

Decoding a binary Goppa syndrome

- Given: $v(x), g(x) \in \mathbb{K}[x]$
- Find: $a(x), b(x), f(x) \in \mathbb{K}[x]$
- Where: $b(x)v(x) + f(x)g(x) = a(x)$
- Thus $a(x) = b(x)v(x) \bmod g(x)$, i.e. $a(x) = b(x)v(x) \in \mathbb{K}[x]/g(x)$.
- Conditions:
  - $\deg(a) \leq \lfloor t/2 \rfloor$, $\deg(b) \leq \lfloor (t-1)/2 \rfloor$.

Paterson's decoding algorithm

$F \leftarrow v,\;\; G \leftarrow g,\;\; B \leftarrow 1,\;\; C \leftarrow 0,\;\; t \leftarrow \deg(g)$
while $\deg(G) > \lfloor t/2 \rfloor$ **do**
    $F \leftrightarrow G,\;\; B \leftrightarrow C$
    while $\deg(F) \geq \deg(G)$ **do**
        $j \leftarrow \deg(F) - \deg(G),\;\; h \leftarrow F_{\deg(F)}/G_{\deg(G)}$
        $F \leftarrow F - h\,x^j\,G,\;\; B \leftarrow B - h\,x^j\,C$
    end
end
$\sigma(x) \leftarrow G(x)^2 + xC(x)^2$
return σ // error locator polynomial

The Key Size Problem

- Using systematic Goppa codes, key size is only $k \times (n - k)$ bits. And yet…

level	m	n	k	t	key size
2^{80}	11	1893	1431	42	661122
2^{128}	12	3307	2515	66	1991880
2^{256}	13	7150	5447	131	9276241

Compact Goppa Codes?

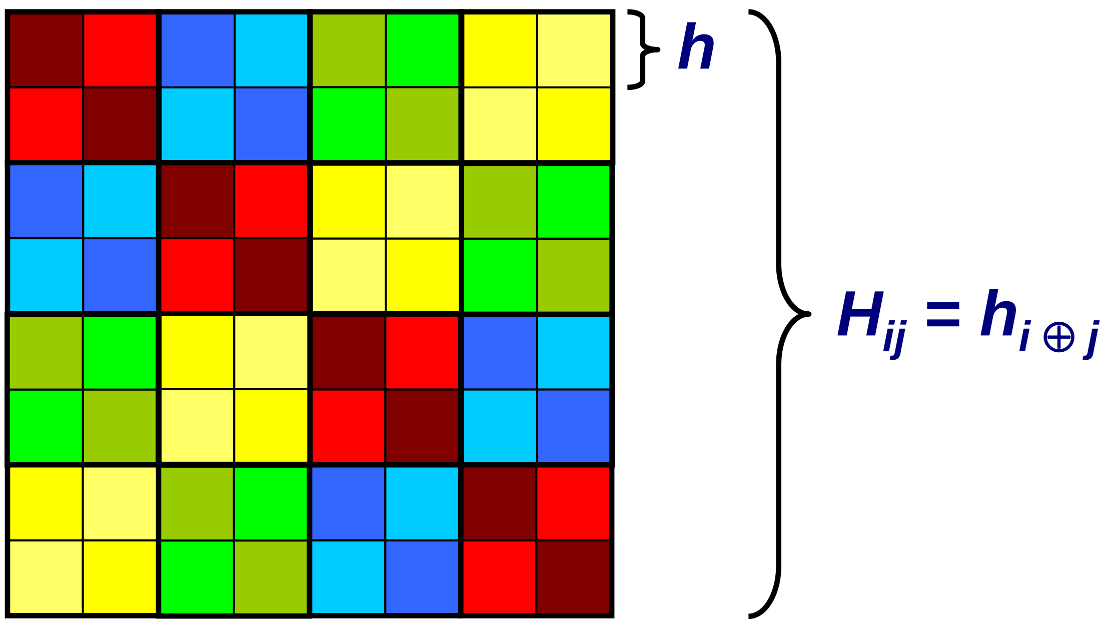
- Recap: a *Goppa code* is entirely defined by:
  - a monic polynomial $g(x) \in \mathbb{F}_q[x]$ of degree t with $q = 2^m$,
  - a sequence $L \in \mathbb{F}_q^n$ of distinct elements with $g(L) \neq 0$.
- Features:
  - good error correction capability (all t design errors in the binary case).
  - withstood cryptanalysis quite well.
- Goal: replace the large $O(n^2)$-bit representation by a compact one (like above!).

Cauchy Matrices

- A matrix $M \in \mathbb{K}^{t \times n}$ over a field $\mathbb{K}$ is called a *Cauchy matrix* iff $M_{ij} = 1/(z_i - L_j)$ for disjoint sequences $z \in \mathbb{K}^t$ and $L \in \mathbb{K}^n$ of distinct elements.
- Property: any Goppa code where $g(x)$ is square-free admits a parity-check matrix in Cauchy form [TZ 1975].
- Compact representation, but:
  - code structure is apparent,
  - usual tricks to hide it destroy the Cauchy structure.

Dyadic Matrices

- Let r be a power of 2. A matrix $H \in \mathcal{R}^{r \times r}$ over a ring $\mathcal{R}$ is called *dyadic* iff $H_{ij} = h_{i \oplus j}$ for some vector $h \in \mathcal{R}^r$.



Dyadic Matrices

- Dyadic matrices form a subring of $\mathcal{R}^{r \times r}$ (commutative if $\mathcal{R}$ is commutative).
- Compact representation: $O(r)$ rather than $O(r^2)$ space.
- Efficient arithmetic: multiplication in time $O(r \lg r)$ time via fast Walsh-Hadamard transform, inversion in time $O(r)$ in characteristic 2.
- Idea: find a dyadic Cauchy matrix.

Quasi-Dyadic Codes

- **Theorem:** a dyadic Cauchy matrix is only possible over fields of characteristic 2, and any suitable $h \in \mathbb{F}_q^n$ satisfies

$$\frac{1}{h_{i \oplus j}} = \frac{1}{h_i} + \frac{1}{h_j} + \frac{1}{h_0}$$

with $z_i = 1/h_i$, $L_j = 1/h_j - 1/h_0$, and $H_{ij} = h_{i \oplus j} = 1/(z_i - L_j)$.

Quasi-Dyadic Codes

- Complexity: key generation $O(n \lg^3 n)$, encoding/decoding $O(n \lg n)$.
- Reasonably short keys (11968 bits for security 2^{80}, 19968 bits for security 2^{128}).
- Caveat: security still under scrutiny (e.g. folding attacks FOPPT 2014).

Alternatives?

- Most alternant codes have been shown to contain weaknesses.
- Goppa codes are popular but not quite friendly to key size reduction.
- Recent trend: graph-based codes, specifically Gallager (LDPC) codes.

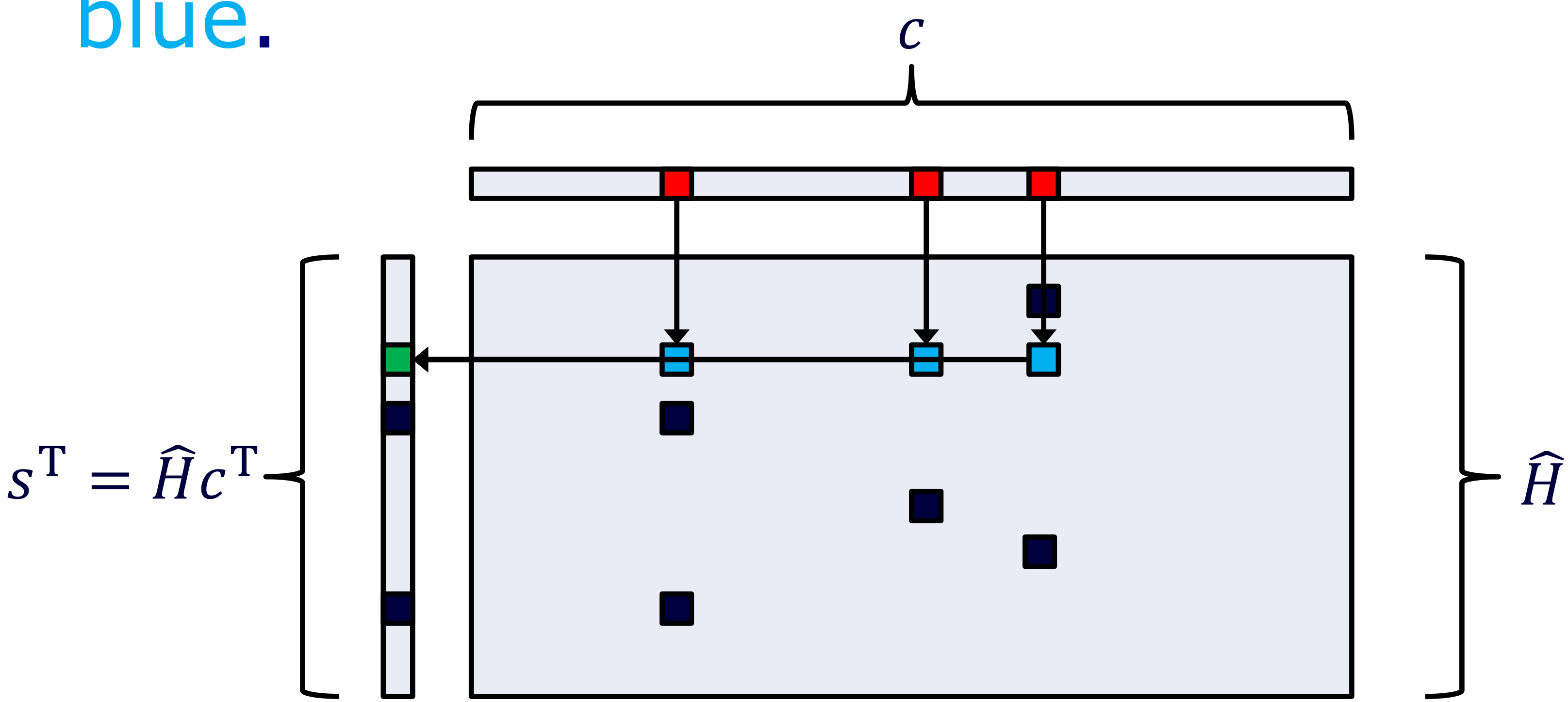
Gallager (LDPC) Codes

- Extremely sparse parity-check matrices, e.g. $\widehat{H} \in \mathbb{F}_2^{10000\times 20000}$ with ~ 3 nonzero components at randomly chosen positions on each column.

- Higher error-correction capability than Goppa codes (almost 3 times in the above example).

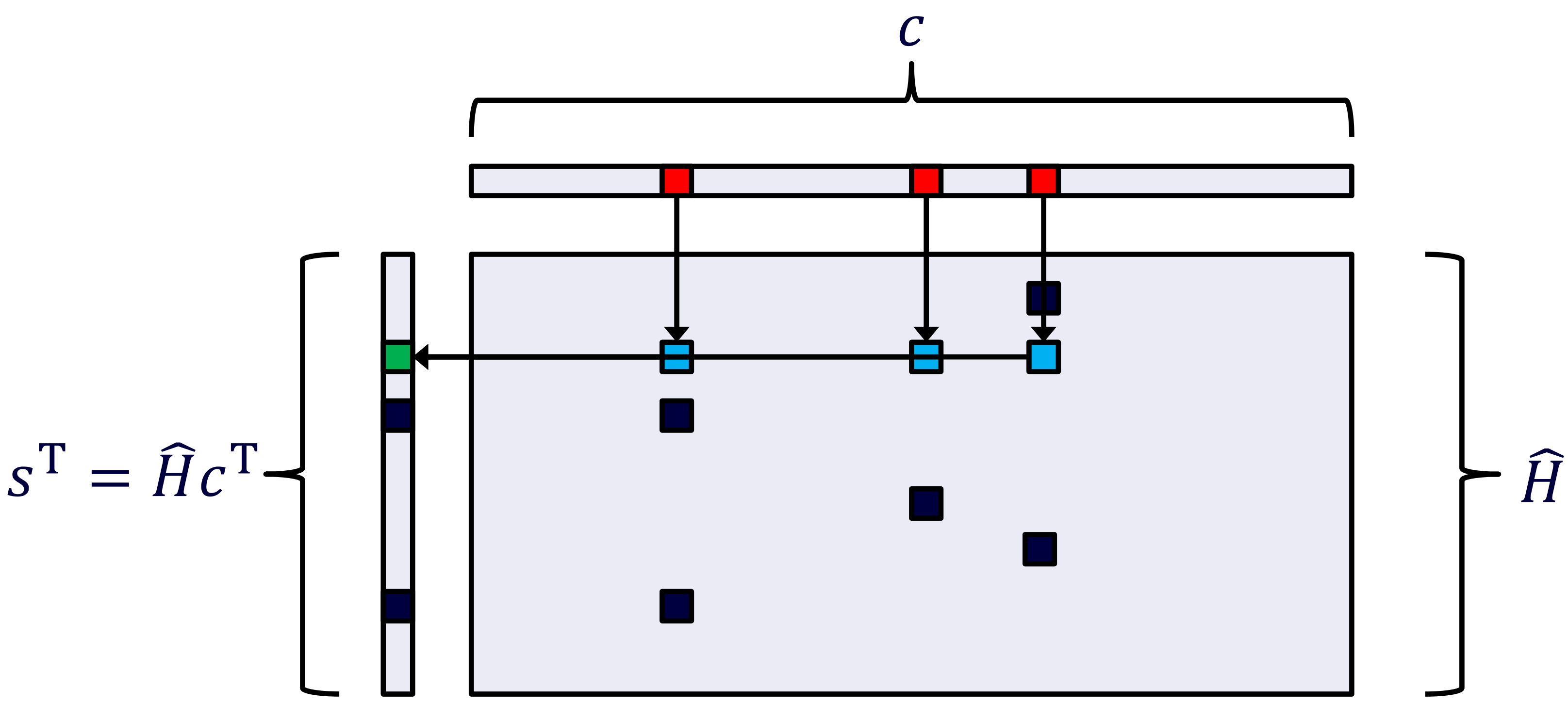
Gallager (LDPC) Codes

- Symbols in red affect parity bit in green through the parity-checks in blue.



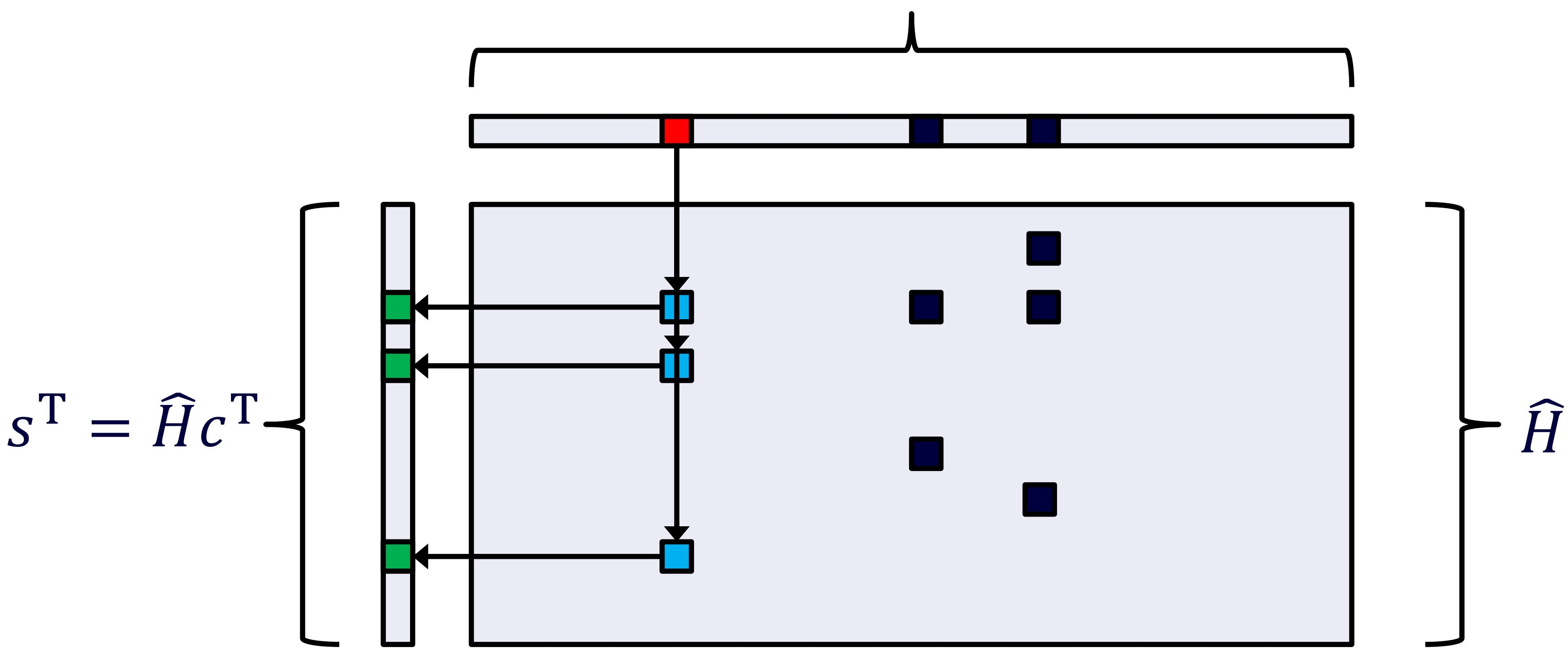
Gallager (LDPC) Codes

- If the green parity bit is 1, at least one of the red bits is wrong.



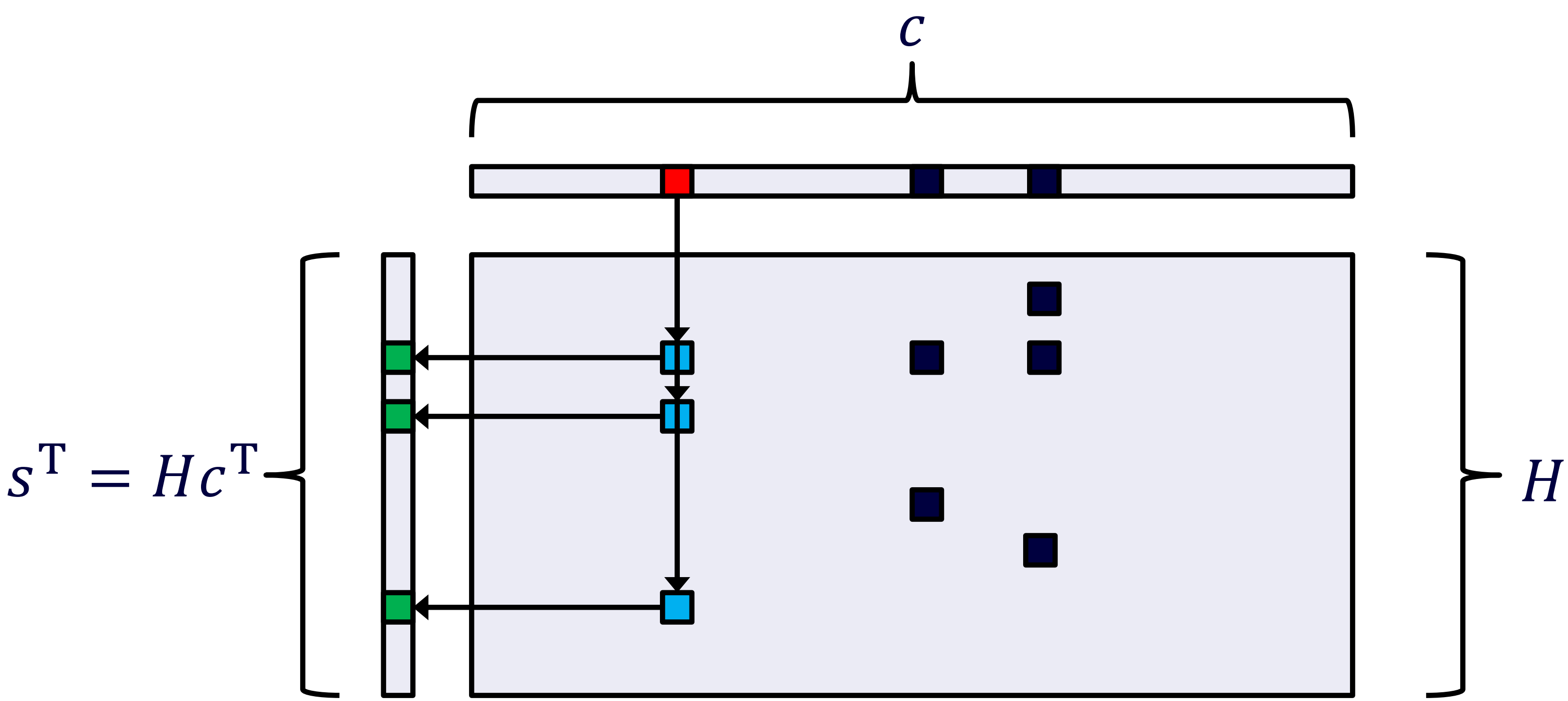
Gallager (LDPC) Codes

- Symbol in red affects parity bits in green through the parity-checks in blue.



Gallager (LDPC) Codes

- If the red bit is wrong, some of the the green parity bits will likely reveal it.



Gallager (LDPC) Codes

- Bit flipping:
  - Determine which symbol bits are the most suspect (i.e. influence the largest number of parity bits in error) by counting how many parity errors it influences via the parity-check matrix.
  - Flip those bits ($0 \leftrightarrow 1$).
  - Repeat until no parity error is left (or max number of attempts is exceeded).

Bit-flipping

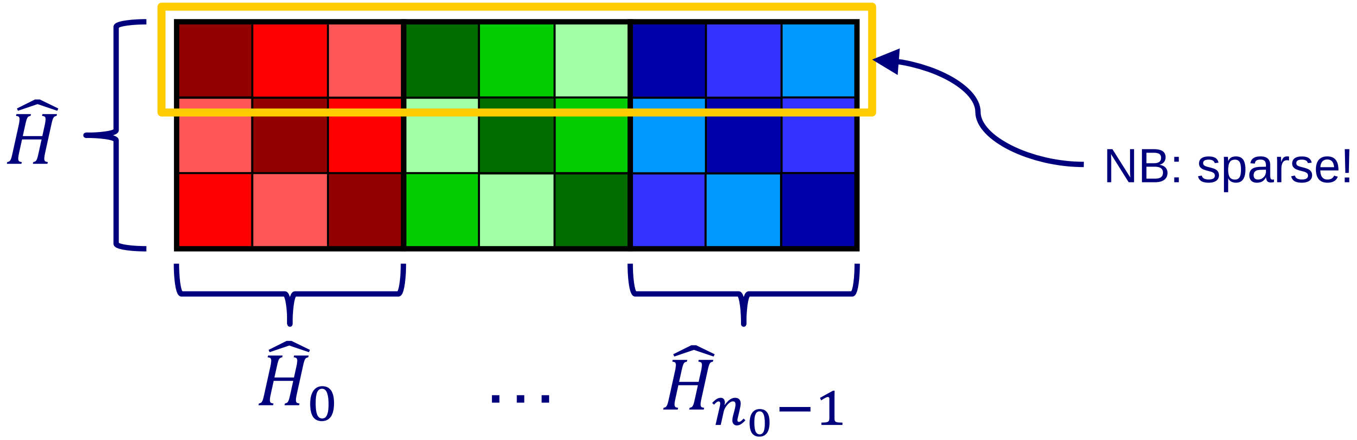
- Trouble: n symbol bits $\Rightarrow$ n counters.
- More trouble: one pass to count and find the maximum count value, another pass to flip most suspect bits and recompute affected parity-check bits.
- Memory-consuming and slow.

MDPC Codes

- LDPC codes are susceptible to key recovery attacks: by definition, dual codes contain too sparse words of small $O(1)$ weight.
- Idea: set density w and number of errors t near the decodability threshold $O(\sqrt{n})$ for security, but still within the range of bit-flipping or belief-propagation.
- Moderate-density parity-check (MDPC) codes (Misoczki et al. 2013).

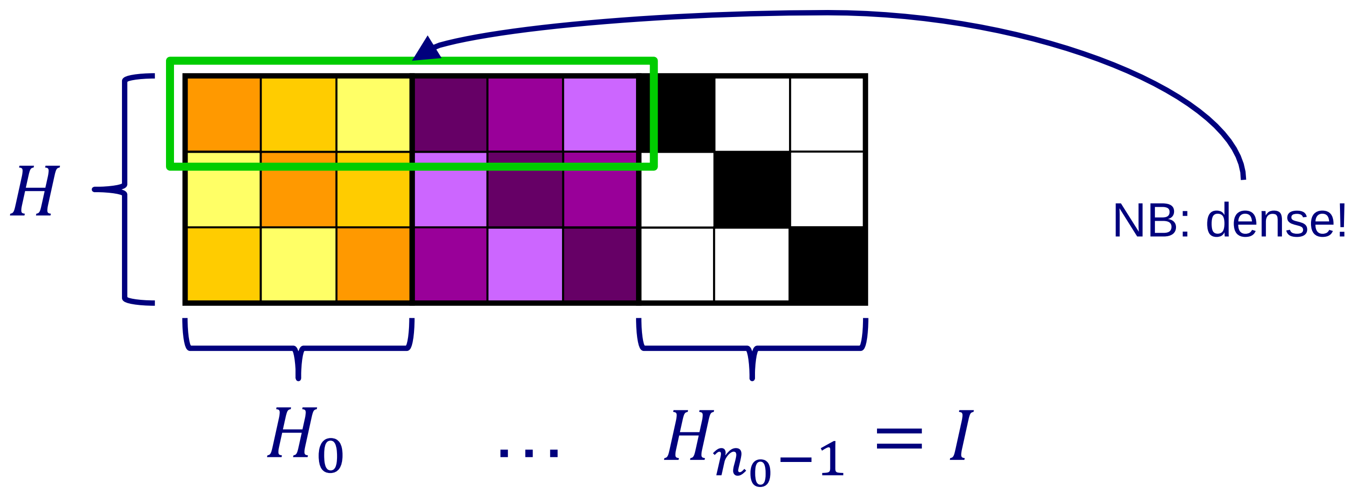
Short Keys

- Quasi-cyclic MDPC codes (QC-MDPC)
- The trapdoor (private) $r \times n$ parity-check matrix consists of n_0 blocks of sparse circulant $r \times r$ matrices, $\widehat{H} = [\widehat{H}_0 \mid \cdots \mid \widehat{H}_{n_0-1}]$, with $n = n_0 r$:



Short Keys

- The systematic (public) parity-check matrix consists of n_0 blocks of dense circulant matrices, $H = [H_0 \mid \cdots \mid H_{n_0-1}]$, with $H_i = H_{n_0-1}^{-1} H_i^*$, $0 \le i < n_0 - 1$:



Short Keys

- Far shorter public (and private) keys than previous proposals:

level	*n*	*k*	*t*	*w*	QC-MDPC	Goppa	shrink	RSA
2^{80}	9602	4801	84	90	***4801***	661122	138×	1024
2^{128}	19714	9857	134	142	***9857***	1991880	202×	3072
2^{256}	65542	32771	264	274	***32771***	9276241	283×	15360

NB: key size is not the only metric (e.g. RSA implementation may just run out of ROM space on many IoT platforms)

Implementation

- On-the-fly counter update $\Rightarrow$ Storage: $O(n) \to O(1)$
- Onset threshold estimation $\Rightarrow$ Counter update passes: $2 \to 1$
- Threshold fine tuning & decoding failure handling $\Rightarrow$ Overall failure rate: $2^{-20} \to \approx 0$
- Simple supporting algorithms e.g. sparse convolution $\Rightarrow$ Small executable footprint & still good performance
- Space-efficient convolution inverse $\Rightarrow$ Folklore(?) algorithm, useful for constrained platforms

Implementation

- Portable C
- Niederreiter code/key generation, encoding/encryption and decoding/decryption (including permutation ranking/unranking)
- PIC24FJ32GA002-I/SP:
  - 8 KiB RAM
  - 32 KiB ROM
  - TCP/IP (hence, IoT-ready)

Implementation

- All-in-one (full functionality):
  - 5.6 KiB ROM, 2.6 KiB RAM
  - key generation ~0.9 s,
    encoding ~25 ms,
    decoding ~2.8 s.
- Split implementation:
  - 1.5 KiB RAM (key generation)
  - <1 KiB RAM (encryption/decryption)

WHAT NEXT?

Limitations and trends

- Codes are fine for encryption 👍
- …but notoriously troublesome for most other applications ☹
- Very recent research trend: other notions of distance, e.g. rank metric.
  - NB: the distance notion is exactly what distinguishes between codes and lattices!
- Advanced functionalities (blind signatures, identity-based encryption)?

QUESTIONS?

APPENDIX

Ranking and Unranking Permutations

- Some SDP-based cryptosystems represent messages as t-error vectors, i.e. vectors (of length n) with Hamming weight t.

- Mapping messages between error vector and normal form involves permutation *ranking* and *unranking*.

Ranking and Unranking Permutations

- Let $\mathcal{B}_t(0^n) = \{u \in \mathbb{F}_2^n \mid \mathrm{wt}(u) = t\}$, with cardinality $r = \binom{n}{t} \approx \frac{n^t}{t!}$
- A *ranking function* is a mapping *rank*: $\mathcal{B}_t(0^n) \to \{0 \ldots r-1\}$ which associates a unique index in $\{0 \ldots r-1\}$ to each element in $\mathcal{B}_t(0^n)$. Its inverse is called the *unranking function*.
- Rank size: $\lg r \approx t(\lg n - \lg t + 1)$ bits.

Ranking and Unranking Permutations

- Ranking and unranking can be done in $O(n)$ time (Ruskey 2003, algorithm 4.10).

- Computationally simplest ordering: colex.

- Definition: $a_1a_2\ldots a_n < b_1b_2\ldots b_m$ in colex order iff $a_n\ldots a_2a_1 < b_m\ldots b_2b_1$ in lex order.

Colex Ranking

- Let $\mathcal{A}_t(n) \coloneqq \{a_1 \ldots a_t \mid 0 \leq a_1 \leq \cdots \leq a_t < n\}$.
- Sum of binomial coefficients:
  $\operatorname{rank}(a_1 \ldots a_t) \coloneqq \sum_{j=1}^{t} \binom{a_j}{j}$
- Implementation strategy: precompute a table of binomial coefficients.

Colex Unranking

input: r // permutation rank
for $j \leftarrow t$ **downto** 1 {
  // find largest $p \geq j - 1$ such that $\binom{p}{j} \leq r$
  $p \leftarrow j - 1$
  while $\binom{p+1}{j} \leq r$ {
    $p \leftarrow p + 1$
  }
  $r \leftarrow r - \binom{p}{j}$
  $a_j \leftarrow p$
}
return $a_1 a_2 \ldots a_t$

Better Methods

- Sendrier: $O(n)$ encoding of words with fixed weight.

- Under certain circumstances (e.g. Niederreiter key encapsulation) ranking/unranking may not even be necessary.