Cryptanalysis techniques in algebraic code–based cryptography

Alain Couvreur[1,2]

[1]INRIA
[2]LIX, École polytechnique

Nutmic 2019

1. History of code–based cryptography

2. Algebraic cryptanalysis in code–based cryptography

3. How to design secure schemes with codes?

Prerequisites on error correcting codes

- A linear code is a vector subspace $\mathscr{C} \subseteq \mathbb{F}_q^n$:
  - n is its *length*;

Prerequisites on error correcting codes

- A linear code is a vector subspace $\mathscr{C} \subseteq \mathbb{F}_q^n$:
  - n is its *length*;
  - k is its *dimension* as an $\mathbb{F}_q$–vector space;

Prerequisites on error correcting codes

- A linear code is a vector subspace $\mathscr{C} \subseteq \mathbb{F}_q^n$:
  - n is its *length*;
  - k is its *dimension* as an $\mathbb{F}_q$–vector space;
  - A *t–decoder* for $\mathscr{C}$ is an algorithm $\mathcal{D}$ taking as input $\boldsymbol{x} \in \mathbb{F}_q^n$ and returning:
    - $\boldsymbol{c} \in \mathscr{C}$ such that $d_H(\boldsymbol{x}, \boldsymbol{c}) \leqslant t$ if exists.
    - "?" else.

Prerequisites on error correcting codes

- A linear code is a vector subspace $\mathscr{C} \subseteq \mathbb{F}_q^n$:
  - n is its *length*;
  - k is its *dimension* as an $\mathbb{F}_q$–vector space;
  - A *t–decoder* for $\mathscr{C}$ is an algorithm $\mathcal{D}$ taking as input $\boldsymbol{x} \in \mathbb{F}_q^n$ and returning:
    - $\boldsymbol{c} \in \mathscr{C}$ such that $d_H(\boldsymbol{x}, \boldsymbol{c}) \leqslant t$ if exists.
    - "?" else.

Definition 1

The Hamming distance on $\mathbb{F}_q^n$ is defined by*:*

$$d_H(\boldsymbol{x}, \boldsymbol{y}) \overset{def}{=} \sharp\{i \in \{1, \dots, n\} \mid x_i \neq y_i\}.$$

A classical operation

Definition 2

Let $\mathscr{C} \subseteq \mathbb{F}_{q^m}^n$ *be a code. Its* subfield subcode *is defined by:*

$$\mathscr{C} \cap \mathbb{F}_q^n.$$

Very classical operation. Many algebraic codes derive from generalised Reed–Solomon codes using this operation: Goppa codes, BCH codes, Srivastava codes, etc...

1. History of code–based cryptography

2. Algebraic cryptanalysis in code–based cryptography

3. How to design secure schemes with codes?

It starts with two articles

[1] E.R. Berlekamp, R.J. McEliece and H.C.A. Van Tilborg. *On the inherent intractability of certain coding problems*. IEEE Trans. Inform. Theory 24(2), 1978.

[2] R.J. McEliece. *A public key cryptosystem based on algebraic coding theory.* DSN Progress Report 44; 1978.

It starts with two articles

[1] E.R. Berlekamp, R.J. McEliece and H.C.A. Van Tilborg. *On the inherent intractability of certain coding problems*. IEEE Trans. Inform. Theory 24(2), 1978.

[2] R.J. McEliece. *A public key cryptosystem based on algebraic coding theory.* DSN Progress Report 44; 1978.

In the article [1]:

Theorem 1

The following problem is NP–complete:

Bounded decoding problem. *Given* $\mathscr{C} \subseteq \mathbb{F}_q^n$, $\boldsymbol{y} \in \mathbb{F}_q^n$ *and* $t \geqslant 0$. *Does there exist* $\boldsymbol{c} \in \mathscr{C}$ *such that*

$$d_H(\boldsymbol{c}, \boldsymbol{y}) \leqslant t?$$

It starts with two articles

[1] E.R. Berlekamp, R.J. McEliece and H.C.A. Van Tilborg. *On the inherent intractability of certain coding problems*. IEEE Trans. Inform. Theory 24(2), 1978.

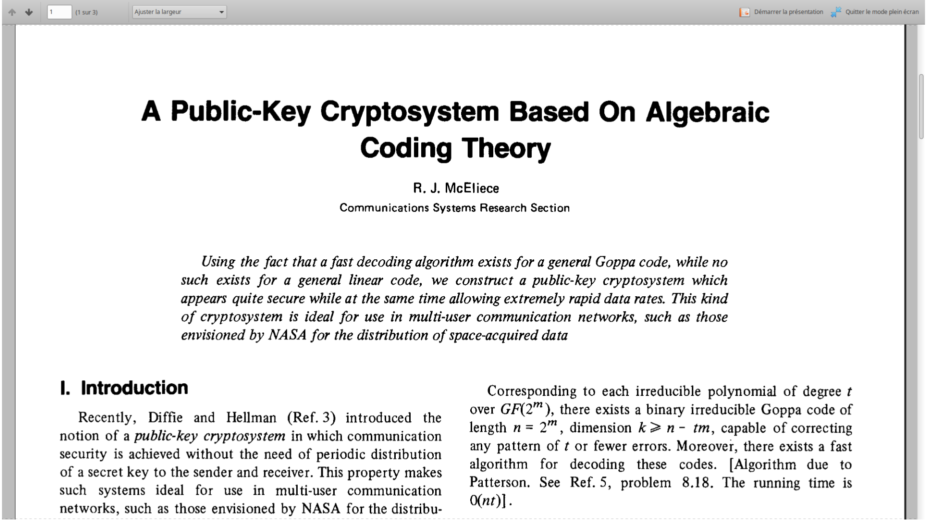
[2] R.J. McEliece. *A public key cryptosystem based on algebraic coding theory.* DSN Progress Report 44; 1978.

In the article [2], McEliece proposes a **new public key encryption scheme**.

McEliece presented in the literature

- **Secret key.**
  - $\boldsymbol{G}$, a structured $k \times n$ matrix whose rows span a code $\mathscr{C}$;
  - $\boldsymbol{S} \in \mathbf{GL}_k$;
  - $\boldsymbol{P} \in \mathfrak{S}_n$.
- **Public key.** $(\boldsymbol{SGP}, t)$;
- **Encryption** $\boldsymbol{m} \mapsto \boldsymbol{mSGP} + \boldsymbol{e}$ for a uniformly random $\boldsymbol{e}$ of weight t;
- **Decryption**
  - Right multiply by $\boldsymbol{P}^{-1}$: $\boldsymbol{mSGP} + \boldsymbol{e} \longmapsto \boldsymbol{mSG} + e\boldsymbol{P}^{-1}$;
  - decode to get $\boldsymbol{mS}$;
  - right multiply it by $\boldsymbol{S}^{-1}$ to get $\boldsymbol{m}$.

This is what McEliece said!



A Public-Key Cryptosystem Based On Algebraic Coding Theory

R. J. McEliece
Communications Systems Research Section

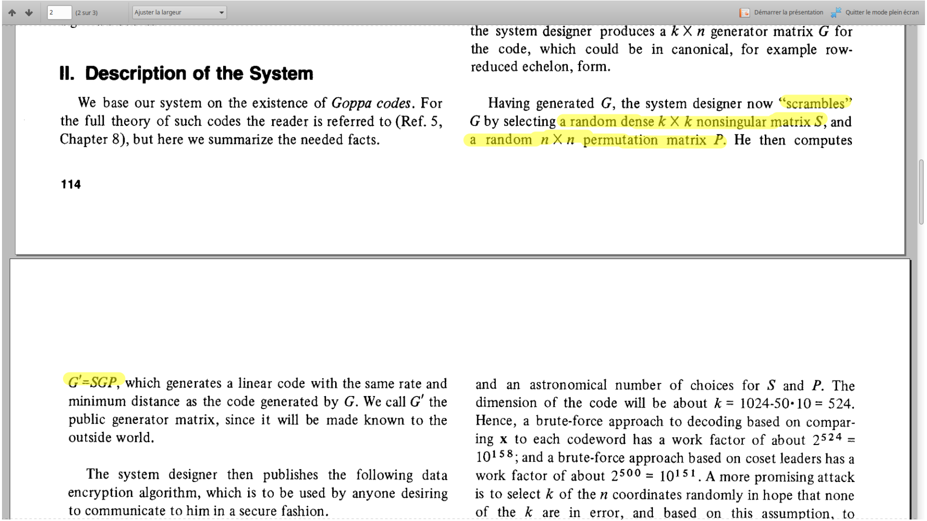
Using the fact that a fast decoding algorithm exists for a general Goppa code, while no such exists for a general linear code, we construct a public-key cryptosystem which appears quite secure while at the same time allowing extremely rapid data rates. This kind of cryptosystem is ideal for use in multi-user communication networks, such as those envisioned by NASA for the distribution of space-acquired data

I. Introduction

Recently, Diffie and Hellman (Ref. 3) introduced the notion of a *public-key cryptosystem* in which communication security is achieved without the need of periodic distribution of a secret key to the sender and receiver. This property makes such systems ideal for use in multi-user communication networks, such as those envisioned by NASA for the distribu-

Corresponding to each irreducible polynomial of degree t over $GF(2^m)$, there exists a binary irreducible Goppa code of length $n = 2^m$, dimension $k \geqslant n - tm$, capable of correcting any pattern of t or fewer errors. Moreover, there exists a fast algorithm for decoding these codes. [Algorithm due to Patterson. See Ref. 5, problem 8.18. The running time is $0(nt)$].

This is what McEliece said!



II. Description of the System

We base our system on the existence of *Goppa codes*. For the full theory of such codes the reader is referred to (Ref. 5, Chapter 8), but here we summarize the needed facts.

114

the system designer produces a $k \times n$ generator matrix G for the code, which could be in canonical, for example row-reduced echelon, form.

Having generated G, the system designer now "scrambles" G by selecting a random dense $k \times k$ nonsingular matrix S, and a random $n \times n$ permutation matrix P. He then computes

$G'=SGP$, which generates a linear code with the same rate and minimum distance as the code generated by G. We call G' the public generator matrix, since it will be made known to the outside world.

The system designer then publishes the following data encryption algorithm, which is to be used by anyone desiring to communicate to him in a secure fashion.

and an astronomical number of choices for S and P. The dimension of the code will be about $k = 1024\text{-}50 \cdot 10 = 524$. Hence, a brute-force approach to decoding based on comparing $\mathbf{x}$ to each codeword has a work factor of about $2^{524} = 10^{158}$; and a brute-force approach based on coset leaders has a work factor of about $2^{500} = 10^{151}$. A more promising attack is to select k of the n coordinates randomly in hope that none of the k are in error, and based on this assumption, to

But... may be we should present it differently

- $\mathcal{F}$ denotes a family of codes of length n and dimension k;
- $\mathcal{S}$ denotes a set "of secrets" with a surjective map $\mathscr{C} : \mathcal{S} \longrightarrow \mathcal{F}$ sending a secret $s \in \mathcal{S}$ into a code $\mathscr{C}(s)$.
- To any $s \in \mathcal{S}$ is associated a decoding algorithm $\mathcal{D}(s)$ for $\mathscr{C}(s)$ correcting up to t errors.

Secret key $s \in \mathcal{S}$;

Public key $(\boldsymbol{G}, t)$, where $\boldsymbol{G}$ denotes a $k \times n$ generator matrix of $\mathscr{C}(s)$;

Encryption $\boldsymbol{m} \in \mathbb{F}_q^k \longmapsto \boldsymbol{mG} + \boldsymbol{e}$ where $\boldsymbol{e}$ is a uniformly random word of weight t.

Decryption Apply $\mathcal{D}(s)$ to $\boldsymbol{mG} + \boldsymbol{e}$ to recover $\boldsymbol{m}$.

Example – Generalised Reed Solomon codes

Definition 2 (Generalised Reed–Solomon codes)

Let n, k be positive integers $k \leqslant n$. Let $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{F}_q^n$ be a vector with distinct entries and $\boldsymbol{y} = (y_1, \ldots, y_n) \in (\mathbb{F}_q^\times)^n$.

$$\mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y}) \stackrel{\text{def}}{=} \{(y_1 f(x_1), \ldots, y_n f(x_n)) \mid \deg(f) < k\}\,.$$

- $\mathcal{F}$ the set of $[n, k]$ GRS codes;
- $\mathcal{S} = \{(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{F}_q^n \times (\mathbb{F}_q^\times)^n \mid \forall i \neq j, x_i \neq x_j\}$;
- $\mathcal{D}(s)$ is your favorite decoder for GRS, e.g. Berlekamp Welch algorithm, with $t = \lfloor \frac{n-k}{2} \rfloor$.

Example – Alternant codes

Definition 3 (Alternant codes)

Let $\boldsymbol{x} = (x_1, \dots, x_n) \in \mathbb{F}_{q^m}^n$ be a vector with distinct entries and $\boldsymbol{y} = (y_1, \dots, y_n) \in (\mathbb{F}_{q^m}^\times)^n$. An alternant code of degree r is a code of the form

$$\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y}) = \mathbf{GRS}_r(\boldsymbol{x}, \boldsymbol{y})^\perp \cap \mathbb{F}_q^n$$

Example – Alternant codes

Definition 3 (Alternant codes)

Let $\boldsymbol{x} = (x_1, \dots, x_n) \in \mathbb{F}_{q^m}^n$ be a vector with distinct entries and $\boldsymbol{y} = (y_1, \dots, y_n) \in (\mathbb{F}_{q^m}^\times)^n$. An alternant code of degree r is a code of the form

$$\begin{aligned}\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y}) &= \mathbf{GRS}_r(\boldsymbol{x}, \boldsymbol{y})^\perp \cap \mathbb{F}_q^n \\ &= \mathbf{GRS}_{n-r}(\boldsymbol{x}, \boldsymbol{y}^\perp) \cap \mathbb{F}_q^n\end{aligned}$$

- $\mathcal{F}$ the set of alternant codes of length n and degree r;
- $\mathcal{S} = \{(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{F}_q^n \times (\mathbb{F}_q^\times)^n \mid \forall i \neq j, x_i \neq x_j\}$;
- $\mathcal{D}(s)$ is your favorite decoder for alternant codes, e.g. Berlekamp Welch algorithm.

Example – Alternant codes

Definition 3 (Alternant codes)

Let $\boldsymbol{x} = (x_1, \dots, x_n) \in \mathbb{F}_{q^m}^n$ be a vector with distinct entries and $\boldsymbol{y} = (y_1, \dots, y_n) \in (\mathbb{F}_{q^m}^\times)^n$. An alternant code of degree r is a code of the form

$$\begin{aligned}\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y}) &= \mathbf{GRS}_r(\boldsymbol{x}, \boldsymbol{y})^\perp \cap \mathbb{F}_q^n \\ &= \mathbf{GRS}_{n-r}(\boldsymbol{x}, \boldsymbol{y}^\perp) \cap \mathbb{F}_q^n\end{aligned}$$

- $\mathcal{F}$ the set of alternant codes of length n and degree r;
- $\mathcal{S} = \{(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{F}_q^n \times (\mathbb{F}_q^\times)^n \mid \forall i \neq j, x_i \neq x_j\}$;
- $\mathcal{D}(s)$ is your favorite decoder for alternant codes, e.g. Berlekamp Welch algorithm.

Example – Classical Goppa codes – McEliece (1978)

Definition 4 (Classical Goppa codes)

Let $\boldsymbol{x} = (x_1, \dots, x_n) \in \mathbb{F}_{q^m}^n$ be a vector with distinct entries and $g \in \mathbb{F}_{q^m}[x]_{<t}$ be a polynomial such that $\forall i, g(x_i) \neq 0$. The Goppa code associated to $(\boldsymbol{x}, g)$ is defined as

$$\mathscr{G}(\boldsymbol{x}, g) \stackrel{\text{def}}{=} \mathscr{A}_{\deg g}(\boldsymbol{x}, g(\boldsymbol{x})^{-1}) \cap \mathbb{F}_q^n$$

where $g(\boldsymbol{x})^{-1} = (g(x_1)^{-1}, \dots, g(x_n)^{-1})$

- $\mathcal{S} = \{(\boldsymbol{x}, g) \mid \cdots\}$;
- etc...

Example – Classical Goppa codes – McEliece (1978)

Definition 4 (Classical Goppa codes)

Let $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{F}_{q^m}^n$ be a vector with distinct entries and $g \in \mathbb{F}_{q^m}[x]_{<t}$ be a polynomial such that $\forall i, g(x_i) \neq 0$. The Goppa code associated to $(\boldsymbol{x}, g)$ is defined as

$$\mathscr{G}\left(\boldsymbol{x}, g\right) \stackrel{\text{def}}{=} \mathscr{A}_{\deg g}(\boldsymbol{x}, g(\boldsymbol{x})^{-1}) \cap \mathbb{F}_q^n$$

where $g(\boldsymbol{x})^{-1} = (g(x_1)^{-1}, \ldots, g(x_n)^{-1})$

- $\mathcal{S} = \{(\boldsymbol{x}, g) \mid \cdots\}$;
- etc...

Example – MDPC codes

Definition 5 (QC-MDPC codes)

Let n be a positive even integer and $f, g \in \mathbb{F}_2[X]_{<n}$ be two polynomials of weight in $O(\sqrt{n})$. A $[2n, n]$ QC-MDPC code is the kernel of the sparse matrix

$$\left(\begin{array}{cccc|cccc} f_0 & f_1 & \cdots & f_{n-1} & g_0 & g_1 & \cdots & g_{n-1} \\ f_{n-1} & f_0 & \cdots & f_{n-2} & g_{n-1} & g_0 & \cdots & g_{n-2} \\ \ddots & \ddots & \ddots & & \ddots & \ddots & \ddots & \end{array}\right)$$

- $\mathcal{F}$ the set of $[2n, n]$ MDPC, codes
- $\mathcal{S} = \{(f, g) \in \mathbb{F}_q[x]_{<n} \text{ of weight } O(\sqrt{n})\}$;
- $\mathcal{D}(s)$ is your favorite decoder for MDPC codes, e.g. Bit Flipping algorithm.

Example – Algebraic geometry codes

Definition 6 (Algebraic geometry codes)

Let X be a smooth projective geometrically connected curve over $\mathbb{F}_q$, G be a divisor on X and $\mathcal{P} = (P_1, \dots, P_n)$ be a set of $\mathbb{F}_q$–points of X. We define

$$\mathcal{C}_L(X, \mathcal{P}, G) \stackrel{\text{def}}{=} \{(f(P_1), \dots, f(P_n)) \mid f \in L(G)\}.$$

- $\mathcal{F}$ the set of AG codes of length n from X.
- $\mathcal{S} = \{(\mathcal{P}, G) \in X(\mathbb{F}_q)^n \times Div_{\mathbb{F}_q}(X) \mid \forall i \neq j, P_i \neq P_j\}$;
- $\mathcal{D}(s)$ is your favorite decoder for AG codes, e.g. Error Correcting Pairs algorithm.

History – McEliece 1978

- 1978 : McEliece's original proposal based on binary Goppa codes (special case of alternant codes). Public key : 32kB for $\approx$ 65 bits of security[1].
- 2018 : NIST proposals :
  - *Classic McEliece*, public key 1 to 1.3 MByte for $>$ 256 bits security.
  - *NTS KEM*, 319 KBytes for $>$ 128 bits security.

[1] With respect to Prange algorithm

History – McEliece 1978

- 1978 : McEliece's original proposal based on binary Goppa codes (special case of alternant codes). Public key : 32kB for $\approx$ 65 bits of security[1].
- 2018 : NIST proposals :
  - *Classic McEliece*, public key 1 to 1.3 MByte for $>$ 256 bits security.
  - *NTS KEM*, 319 KBytes for $>$ 128 bits security.

During these 40 years many attempts to get shorter keys.

[1] With respect to Prange algorithm

History – McEliece 1978

- 1978 : McEliece's original proposal based on binary Goppa codes (special case of alternant codes). Public key : 32kB for $\approx$ 65 bits of security[1].
- 2018 : NIST proposals :
  - *Classic McEliece*, public key 1 to 1.3 MByte for $>$ 256 bits security.
  - *NTS KEM*, 319 KBytes for $>$ 128 bits security.

During these 40 years many attempts to get shorter keys. **How?**

[1] With respect to Prange algorithm

Idea 1 : Reducing the extension degree

$$\begin{tikzcd}
\mathbb{F}_{q^m} \arrow[d, dash] \arrow[d, dash, bend left, "m"] & & \mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y}) \arrow[d, dash] \\
\mathbb{F}_q & & \mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y}) \cap \mathbb{F}_q^n
\end{tikzcd}$$

Fact. The larger the m the worse the parameters. But:

Idea 1 : Reducing the extension degree

$$\begin{tikzcd}
\mathbb{F}_{q^m} \arrow[d, dash, "m"] & \mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y}) \arrow[d, dash] \\
\mathbb{F}_q & \mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y}) \cap \mathbb{F}_q^n
\end{tikzcd}$$

Fact. The larger the m the worse the parameters. But:

- Case $m = 1$ is broken (Sidelnikov, Shestakov 1992);
- Some specific cases of $m = 2$ and 3 called *wild Goppa codes* are broken too:
  - C., Otmani, Tillich, 2014;
  - Faugère, Perret, de Portzamparc, 2014

Idea 2 : Using codes with a non trivial automorphism group

In 2005, Gaborit proposes to use codes with a non trivial automorphism group $\mathcal{G}$.

- Quasi–cyclic codes (QC–codes) : $\mathcal{G} = \mathbb{Z}/\ell\mathbb{Z}$;
- Quasi–dyadic codes (QD–codes) : $\mathcal{G} = (\mathbb{Z}/2\mathbb{Z})^\gamma$.

- **Advantage.** Permits to reduce the public key size with almost no incidence on the security

Idea 2 : Using codes with a non trivial automorphism group

In 2005, Gaborit proposes to use codes with a non trivial automorphism group $\mathcal{G}$.

- Quasi–cyclic codes (QC–codes) : $\mathcal{G} = \mathbb{Z}/\ell\mathbb{Z}$;
- Quasi–dyadic codes (QD–codes) : $\mathcal{G} = (\mathbb{Z}/2\mathbb{Z})^\gamma$.

- **Advantage.** Permits to reduce the public key size with almost no incidence on the security **w.r.t. message security attacks.**

Idea 2 : Using codes with a non trivial automorphism group

In 2005, Gaborit proposes to use codes with a non trivial automorphism group $\mathcal{G}$.

- Quasi–cyclic codes (QC–codes) : $\mathcal{G} = \mathbb{Z}/\ell\mathbb{Z}$;
- Quasi–dyadic codes (QD–codes) : $\mathcal{G} = (\mathbb{Z}/2\mathbb{Z})^\gamma$.

- **Advantage.** Permits to reduce the public key size with almost no incidence on the security **w.r.t. message security attacks.**
- **But, may affect the security w.r.t. key recovery attacks.**

Idea 2 : Using codes with a non trivial automorphism group

In 2005, Gaborit proposes to use odes with a non trivial automorphism group $\mathcal{G}$.

Caution! Some tempting choices of using large groups lead to key recovery attacks:

- QC–BCH codes: Otmani, Tillich, Dallot (2008);
- QC–altenant codes : Faugère, Otmani, Perret, Tillich (2010);
- QC and QD–alternant codes : Faugère, Otmani, Perret, Tillich, de Portzamparc (2016).
- DAGS (QD–Alternant codes): Barelli, C. (2018).

Further constructions from GRS codes

- **Berger Loidreau, 2001**. Subcodes of GRS codes.
- **Wieschebrink, 2006**. Adds random columns in a GRS code's generator matrix.
- **Baldi, Bianchi, Chiaraluce, Rosenthal, Schipani, 2013**. Right multiply the GRS code by a sparse matrix.
- **Wang's RLCE system, 2016**. Replaces some columns of a GRS's generator matrix by linear combinations of GRS and random columns.

Other families of codes

- **Sidelnikov 1994**. Binary Reed Muller codes.
- **Janwa Moreno 1996**. Algebraic geometry codes and their subfield subcodes.
- **Misoczki, Tillich, Sendrier, Barreto 2012.** QC–MDPC codes.

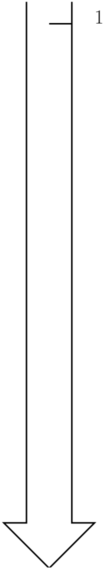
Other families of codes

- **Sidelnikov 1994**. Binary Reed Muller codes.
- **Janwa Moreno 1996**. Algebraic geometry codes and their subfield subcodes.
- **Misoczki, Tillich, Sendrier, Barreto 2012.** QC–MDPC codes.

Remark

Non exhaustive list.

Chronology



1978 : McEliece

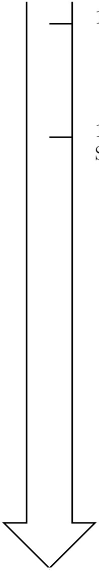
Proposals

Attacks

~~Broken~~

~~Partially Broken~~

Chronology

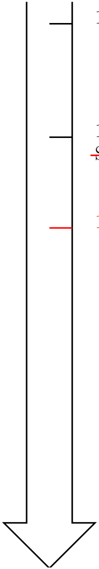


1978 : McEliece

1986 : Niederreiter
Suggests GRS codes

Proposals
Attacks
~~Broken~~
~~Partially Broken~~

Chronology



1978 : McEliece

Proposals

Attacks

~~Broken~~

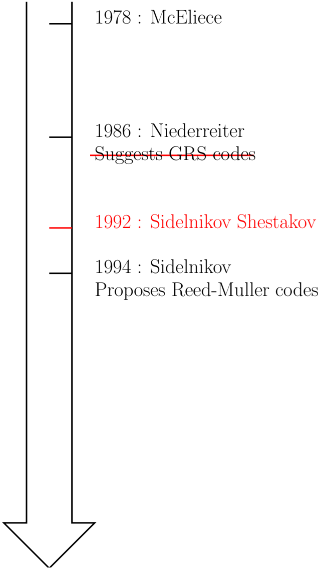
~~Partially Broken~~

1986 : Niederreiter

~~Suggests GRS codes~~

1992 : Sidelnikov Shestakov

Chronology



1978 : McEliece

1986 : Niederreiter
~~Suggests GRS codes~~

1992 : Sidelnikov Shestakov

1994 : Sidelnikov
Proposes Reed-Muller codes

Proposals
Attacks
~~Broken~~
~~Partially Broken~~

Chronology

1978 : McEliece

Proposals

Attacks

~~Broken~~

~~Partially Broken~~

1986 : Niederreiter
~~Suggests GRS codes~~

1992 : Sidelnikov Shestakov

1994 : Sidelnikov
Proposes Reed-Muller codes

1996 : Janwa, Moreno
Propose AG codes and their subfield subcodes

Chronology

1978 : McEliece

Proposals
Attacks
~~Broken~~
~~Partially Broken~~

1986 : Niederreiter
~~Suggests GRS codes~~

1992 : Sidelnikov Shestakov

1994 : Sidelnikov
Proposes Reed-Muller codes

1996 : Janwa, Moreno
Propose AG codes and their subfield subcodes

2001 : Berger Loidreau
Propose subcodes of GRS codes

Chronology

1978 : McEliece

1986 : Niederreiter
~~Suggests GRS codes~~

1992 : Sidelnikov Shestakov

1994 : Sidelnikov
Proposes Reed-Muller codes

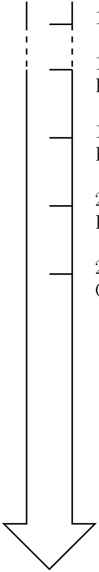
1996 : Janwa, Moreno
Propose AG codes and their subfield subcodes

2001 : Berger Loidreau
Propose subcodes of GRS codes

2005 : Gaborit
Quasi–cyclic BCH codes

Proposals
Attacks
~~Broken~~
~~Partially Broken~~

Chronology



1978 : McEliece

1994 : Sidelnikov
Proposes Reed-Muller codes

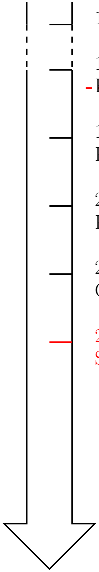
1996 : Janwa, Moreno
Propose AG codes and their subfield subcodes

2001 : Berger Loidreau
Propose subcodes of GRS codes

2005 : Gaborit
Quasi-cyclic subcodes of BCH codes

Proposals
Attacks
~~Broken~~
~~Partially Broken~~

Chronology



1978 : McEliece

1994 : Sidelnikov
~~Proposes Reed-Muller codes~~

1996 : Janwa, Moreno
Propose AG codes and their subfield subcodes

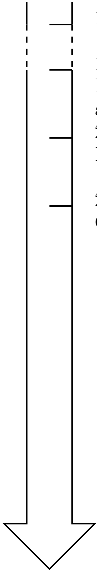
2001 : Berger Loidreau
Propose subcodes of GRS codes

2005 : Gaborit
Quasi-cyclic subcodes of BCH codes

2007 : Minder Shokrollahi
Subexponential time attack on RM codes

Proposals
Attacks
~~Broken~~
~~Partially Broken~~

Chronology



1978 : McEliece

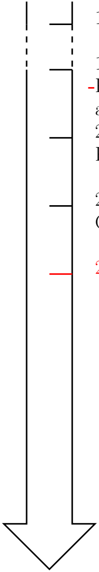
1996 : Janwa, Moreno
Propose AG codes
and their subfield subcodes

2001 : Berger Loidreau
Propose subcodes of GRS codes

2005 : Gaborit
Quasi-cyclic subcodes of BCH codes

Proposals
Attacks
~~Broken~~
~~Partially Broken~~

Chronology



1978 : McEliece

1996 : Janwa, Moreno
~~Propose AG codes~~
and their subfield subcodes

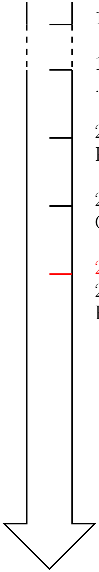
2001 : Berger Loidreau
Propose subcodes of GRS codes

2005 : Gaborit
Quasi-cyclic subcodes of BCH codes

2008 : Faure Minder, Attack on AG codes for genus ≤ 2

Proposals
Attacks
~~Broken~~
~~Partially Broken~~

Chronology



1978 : McEliece

1996 : Janwa, Moreno
... and their subfield subcodes

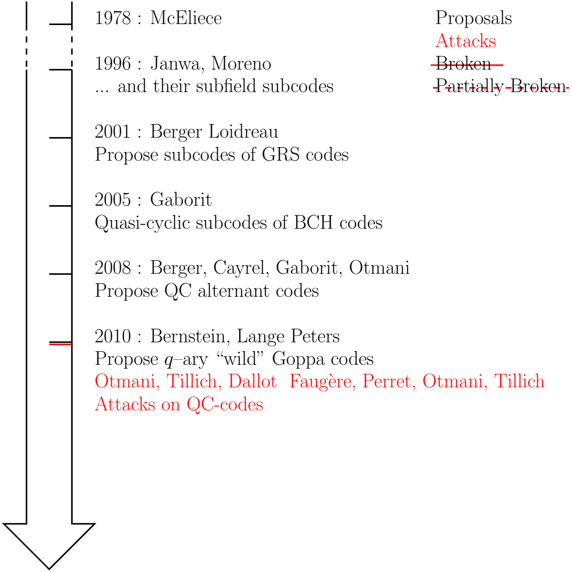
2001 : Berger Loidreau
Propose subcodes of GRS codes

2005 : Gaborit
Quasi-cyclic subcodes of BCH codes

2008 : Faure Minder, Attack on AG codes for genus ≤ 2
2008 : Berger, Cayrel, Gaborit, Otmani
Propose QC alternant codes

Proposals
Attacks
~~Broken~~
~~Partially Broken~~

Chronology



1978 : McEliece

1996 : Janwa, Moreno
... and their subfield subcodes

2001 : Berger Loidreau
Propose subcodes of GRS codes

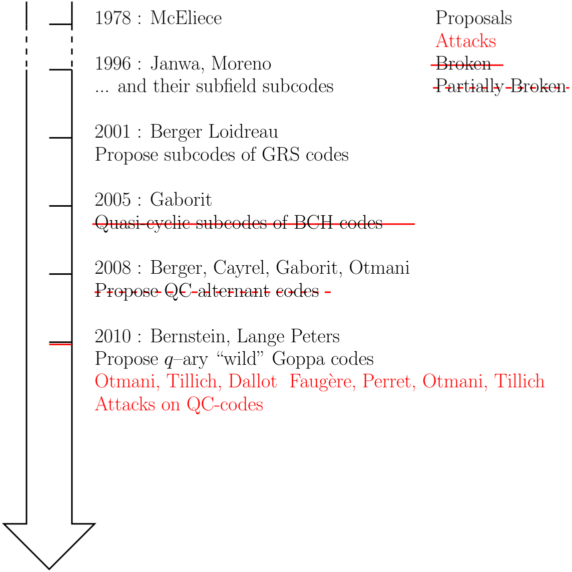
2005 : Gaborit
Quasi-cyclic subcodes of BCH codes

2008 : Berger, Cayrel, Gaborit, Otmani
Propose QC alternant codes

2010 : Bernstein, Lange Peters
Propose q–ary “wild” Goppa codes
Otmani, Tillich, Dallot Faugère, Perret, Otmani, Tillich
Attacks on QC-codes

Proposals
Attacks
~~Broken~~
~~Partially Broken~~

Chronology



1978 : McEliece

1996 : Janwa, Moreno
... and their subfield subcodes

2001 : Berger Loidreau
Propose subcodes of GRS codes

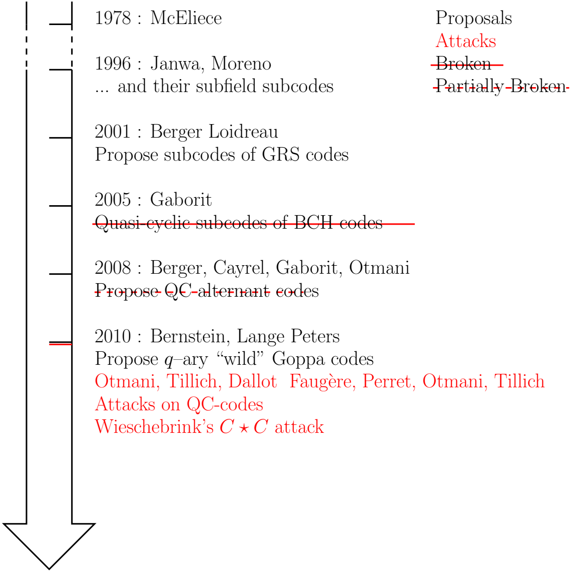
2005 : Gaborit
~~Quasi-cyclic subcodes of BCH codes~~

2008 : Berger, Cayrel, Gaborit, Otmani
~~Propose QC-alternant codes~~

2010 : Bernstein, Lange Peters
Propose q–ary “wild” Goppa codes
Otmani, Tillich, Dallot Faugère, Perret, Otmani, Tillich
Attacks on QC-codes

Proposals
Attacks
~~Broken~~
~~Partially Broken~~

Chronology



1978 : McEliece

1996 : Janwa, Moreno
... and their subfield subcodes

2001 : Berger Loidreau
Propose subcodes of GRS codes

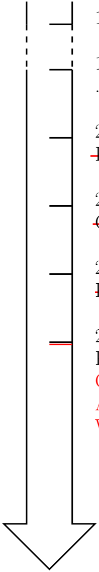
2005 : Gaborit
~~Quasi-cyclic subcodes of BCH codes~~

2008 : Berger, Cayrel, Gaborit, Otmani
~~Propose QC-alternant codes~~

2010 : Bernstein, Lange Peters
Propose q–ary “wild” Goppa codes
Otmani, Tillich, Dallot Faugère, Perret, Otmani, Tillich
Attacks on QC-codes
Wieschebrink’s $C \star C$ attack

Proposals
Attacks
~~Broken~~
~~Partially Broken~~

Chronology



Proposals

Attacks

~~Broken~~

~~Partially Broken~~

1978 : McEliece

1996 : Janwa, Moreno
... and their subfield subcodes

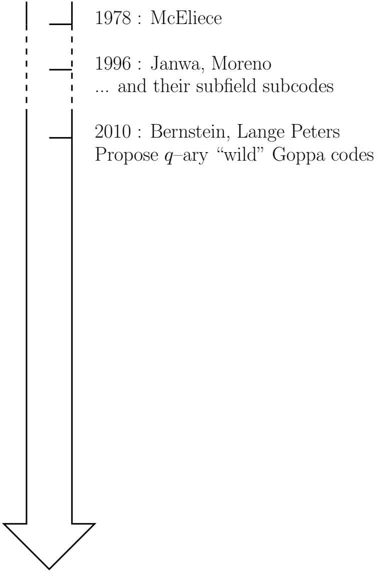
2001 : Berger Loidreau
~~Propose subcodes of GRS codes~~

2005 : Gaborit
~~Quasi-cyclic subcodes of BCH codes~~

2008 : Berger, Cayrel, Gaborit, Otmani
~~Propose QC-alternant codes~~

2010 : Bernstein, Lange Peters
Propose q–ary “wild” Goppa codes
Otmani, Tillich, Dallot Faugère, Perret, Otmani, Tillich
Attacks on QC-codes
Wieschebrink’s $C \star C$ attack

Chronology



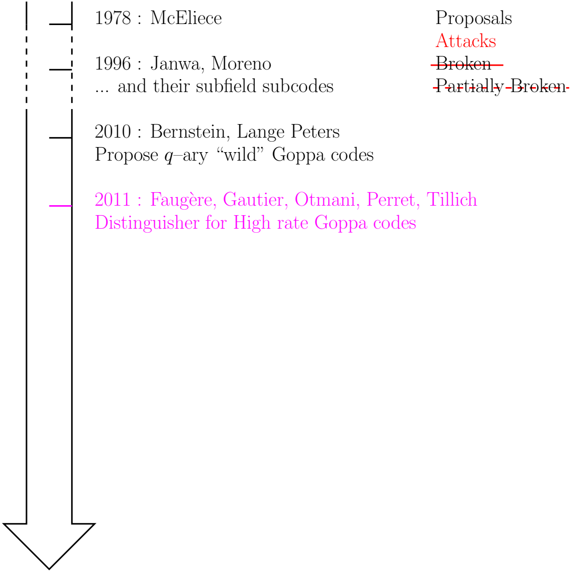
1978 : McEliece

1996 : Janwa, Moreno
... and their subfield subcodes

2010 : Bernstein, Lange Peters
Propose q–ary "wild" Goppa codes

Proposals
Attacks
~~Broken~~
~~Partially Broken~~

Chronology



1978 : McEliece

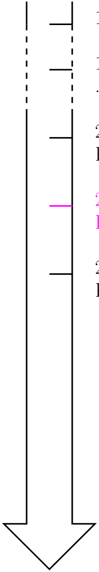
1996 : Janwa, Moreno
... and their subfield subcodes

2010 : Bernstein, Lange Peters
Propose q–ary “wild” Goppa codes

2011 : Faugère, Gautier, Otmani, Perret, Tillich
Distinguisher for High rate Goppa codes

Proposals
Attacks
~~Broken~~
~~Partially Broken~~

Chronology



1978 : McEliece

1996 : Janwa, Moreno
... and their subfield subcodes

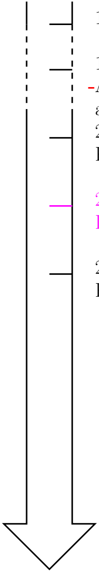
2010 : Bernstein, Lange Peters
Propose q–ary "wild" Goppa codes

2011 : Faugère, Gautier, Otmani, Perret, Tillich
Distinguisher for High rate Goppa codes

2012 : Misoczki, Tillich, Sendrier, Barreto
Propose MDPC codes

Proposals
Attacks
~~Broken~~
~~Partially Broken~~

Chronology



1978 : McEliece

1996 : Janwa, Moreno
~~AG codes~~
and their subfield subcodes

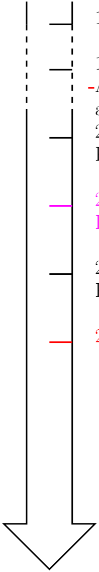
2010 : Bernstein, Lange Peters
Propose q–ary “wild” Goppa codes

2011 : Faugère, Gautier, Otmani, Perret, Tillich
Distinguisher for High rate Goppa codes

2012 : Misoczki, Tillich, Sendrier, Barreto
Propose MDPC codes

Proposals
Attacks
~~Broken~~
~~Partially Broken~~

Chronology



1978 : McEliece

1996 : Janwa, Moreno
~~AG codes~~
and their subfield subcodes

2010 : Bernstein, Lange Peters
Propose q–ary "wild" Goppa codes

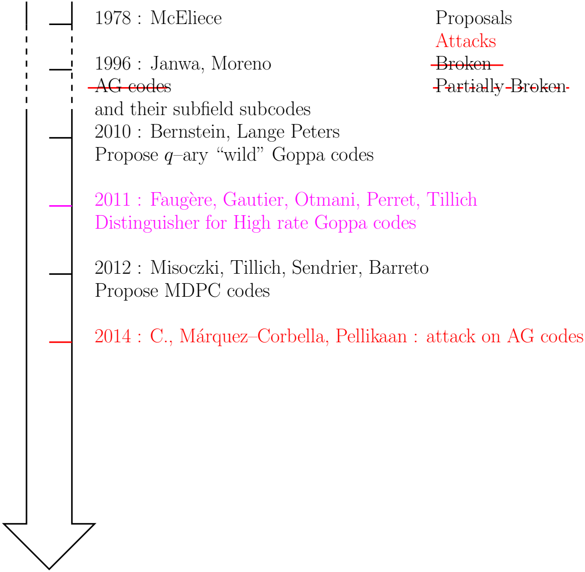
2011 : Faugère, Gautier, Otmani, Perret, Tillich
Distinguisher for High rate Goppa codes

2012 : Misoczki, Tillich, Sendrier, Barreto
Propose MDPC codes

2014 : C., Márquez–Corbella, Pellikaan : attack on AG codes

Proposals
Attacks
~~Broken~~
~~Partially Broken~~

Chronology



1978 : McEliece

1996 : Janwa, Moreno
~~AG codes~~
and their subfield subcodes

2010 : Bernstein, Lange Peters
Propose q–ary “wild” Goppa codes

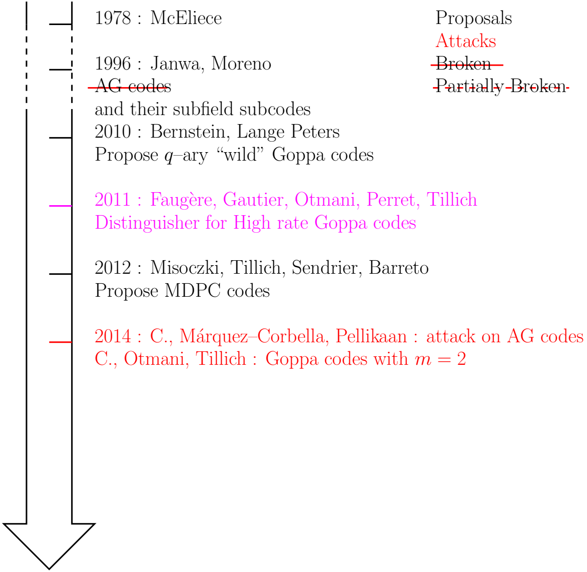
2011 : Faugère, Gautier, Otmani, Perret, Tillich
Distinguisher for High rate Goppa codes

2012 : Misoczki, Tillich, Sendrier, Barreto
Propose MDPC codes

2014 : C., Márquez–Corbella, Pellikaan : attack on AG codes

Proposals
Attacks
~~Broken~~
~~Partially Broken~~

Chronology



1978 : McEliece

1996 : Janwa, Moreno
~~AG codes~~
and their subfield subcodes

2010 : Bernstein, Lange Peters
Propose q–ary “wild” Goppa codes

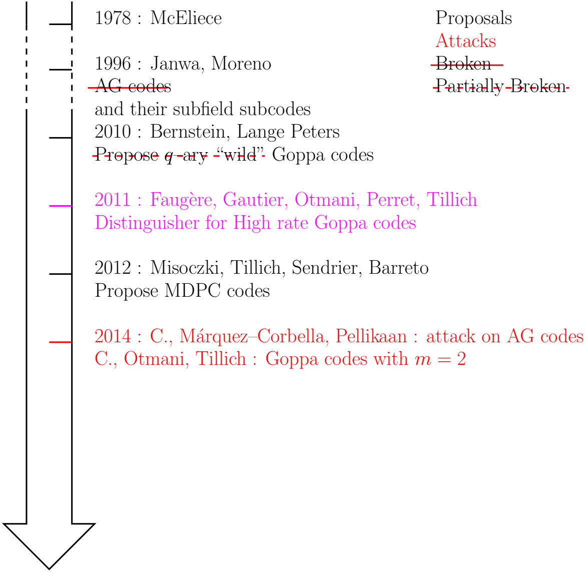
2011 : Faugère, Gautier, Otmani, Perret, Tillich
Distinguisher for High rate Goppa codes

2012 : Misoczki, Tillich, Sendrier, Barreto
Propose MDPC codes

2014 : C., Márquez–Corbella, Pellikaan : attack on AG codes
C., Otmani, Tillich : Goppa codes with $m = 2$

Proposals
Attacks
~~Broken~~
~~Partially Broken~~

Chronology



1978 : McEliece

1996 : Janwa, Moreno
~~AG codes~~
and their subfield subcodes

2010 : Bernstein, Lange Peters
~~Propose q-ary "wild"~~ Goppa codes

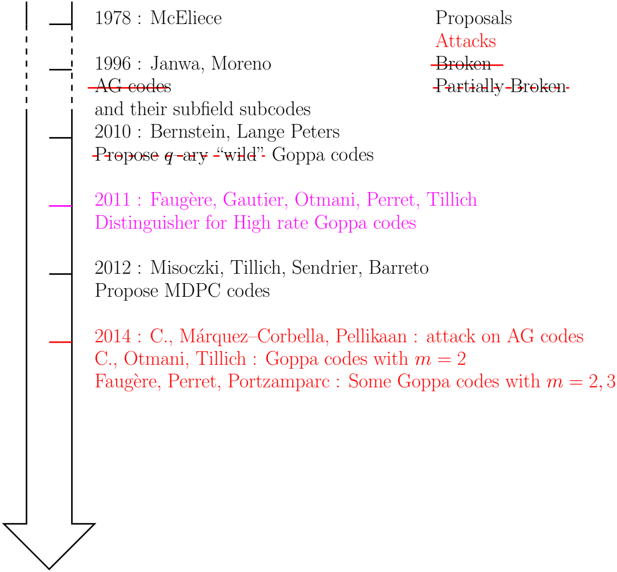
2011 : Faugère, Gautier, Otmani, Perret, Tillich
Distinguisher for High rate Goppa codes

2012 : Misoczki, Tillich, Sendrier, Barreto
Propose MDPC codes

2014 : C., Márquez–Corbella, Pellikaan : attack on AG codes
C., Otmani, Tillich : Goppa codes with $m = 2$

Proposals
Attacks
~~Broken~~
~~Partially Broken~~

Chronology



1978 : McEliece

1996 : Janwa, Moreno
~~AG codes~~
and their subfield subcodes

2010 : Bernstein, Lange Peters
~~Propose q-ary "wild"~~ Goppa codes

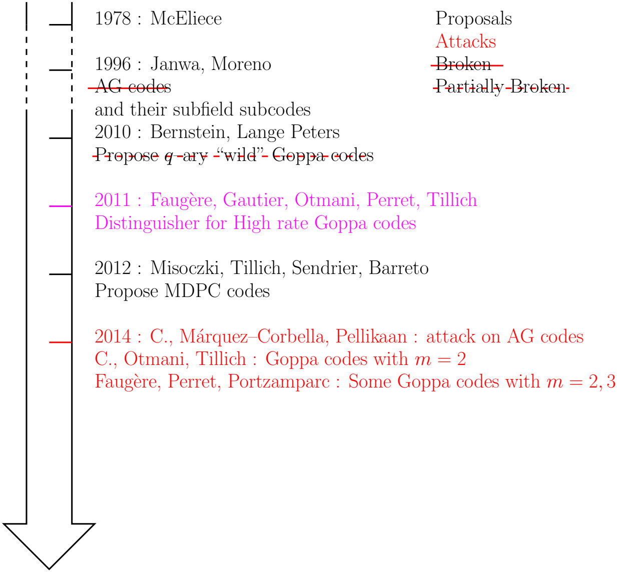
2011 : Faugère, Gautier, Otmani, Perret, Tillich
Distinguisher for High rate Goppa codes

2012 : Misoczki, Tillich, Sendrier, Barreto
Propose MDPC codes

2014 : C., Márquez–Corbella, Pellikaan : attack on AG codes
C., Otmani, Tillich : Goppa codes with $m = 2$
Faugère, Perret, Portzamparc : Some Goppa codes with $m = 2, 3$

Proposals
Attacks
~~Broken~~
~~Partially Broken~~

Chronology



1978 : McEliece

1996 : Janwa, Moreno
~~AG codes~~
and their subfield subcodes

2010 : Bernstein, Lange Peters
~~Propose q-ary "wild" Goppa codes~~

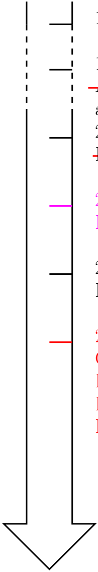
2011 : Faugère, Gautier, Otmani, Perret, Tillich
Distinguisher for High rate Goppa codes

2012 : Misoczki, Tillich, Sendrier, Barreto
Propose MDPC codes

2014 : C., Márquez–Corbella, Pellikaan : attack on AG codes
C., Otmani, Tillich : Goppa codes with $m = 2$
Faugère, Perret, Portzamparc : Some Goppa codes with $m = 2, 3$

Proposals
Attacks
~~Broken~~
~~Partially Broken~~

Chronology



1978 : McEliece

1996 : Janwa, Moreno
~~AG codes~~
and their subfield subcodes

2010 : Bernstein, Lange Peters
~~Propose q-ary "wild" Goppa codes~~

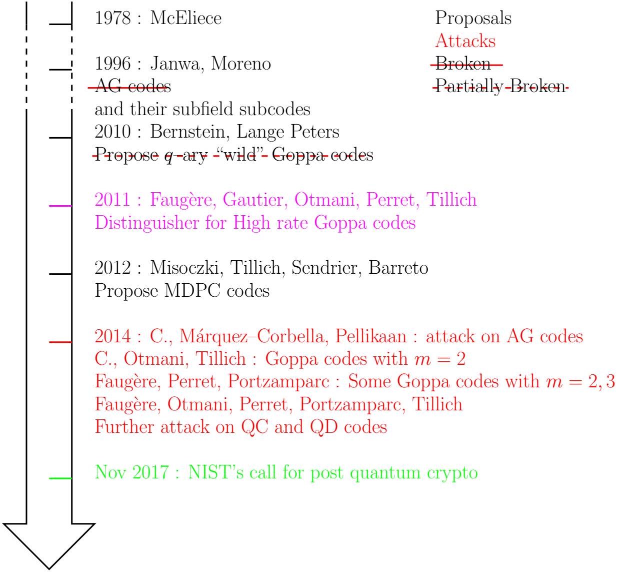
2011 : Faugère, Gautier, Otmani, Perret, Tillich
Distinguisher for High rate Goppa codes

2012 : Misoczki, Tillich, Sendrier, Barreto
Propose MDPC codes

2014 : C., Márquez–Corbella, Pellikaan : attack on AG codes
C., Otmani, Tillich : Goppa codes with $m = 2$
Faugère, Perret, Portzamparc : Some Goppa codes with $m = 2, 3$
Faugère, Otmani, Perret, Portzamparc, Tillich
Further attack on QC and QD codes

Proposals
Attacks
~~Broken~~
~~Partially Broken~~

Chronology



1978 : McEliece

1996 : Janwa, Moreno
~~AG codes~~
and their subfield subcodes

2010 : Bernstein, Lange Peters
~~Propose q-ary "wild" Goppa codes~~

2011 : Faugère, Gautier, Otmani, Perret, Tillich
Distinguisher for High rate Goppa codes

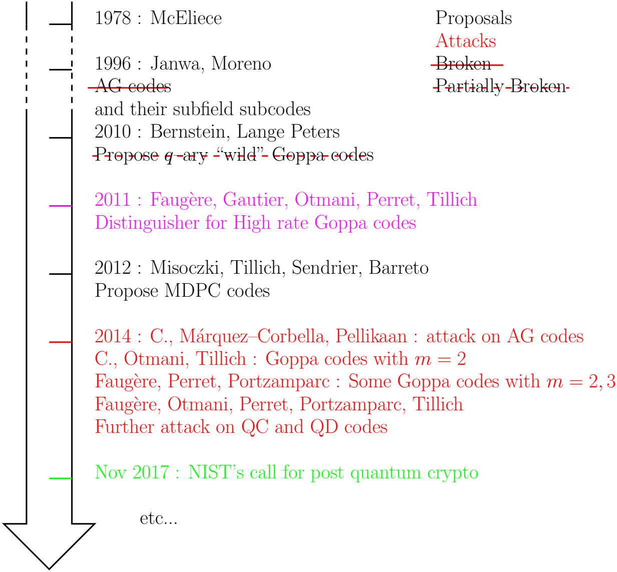
2012 : Misoczki, Tillich, Sendrier, Barreto
Propose MDPC codes

2014 : C., Márquez–Corbella, Pellikaan : attack on AG codes
C., Otmani, Tillich : Goppa codes with $m = 2$
Faugère, Perret, Portzamparc : Some Goppa codes with $m = 2, 3$
Faugère, Otmani, Perret, Portzamparc, Tillich
Further attack on QC and QD codes

Nov 2017 : NIST's call for post quantum crypto

Proposals
Attacks
~~Broken~~
~~Partially Broken~~

Chronology



1978 : McEliece

Proposals

Attacks

~~Broken~~

~~Partially Broken~~

1996 : Janwa, Moreno
~~AG codes~~
and their subfield subcodes

2010 : Bernstein, Lange Peters
~~Propose q-ary “wild” Goppa codes~~

2011 : Faugère, Gautier, Otmani, Perret, Tillich
Distinguisher for High rate Goppa codes

2012 : Misoczki, Tillich, Sendrier, Barreto
Propose MDPC codes

2014 : C., Márquez–Corbella, Pellikaan : attack on AG codes
C., Otmani, Tillich : Goppa codes with $m = 2$
Faugère, Perret, Portzamparc : Some Goppa codes with $m = 2, 3$
Faugère, Otmani, Perret, Portzamparc, Tillich
Further attack on QC and QD codes

Nov 2017 : NIST’s call for post quantum crypto

etc...

1 History of code–based cryptography

2 Algebraic cryptanalysis in code–based cryptography

3 How to design secure schemes with codes?

Theoretical security analysis of McEliece encryption

Security proofs consist in reducing to the **Bounded decoding problem** under the following assumption:

> **Assumption.** *The uniform distribution on the public $[n, k]$ codes in family $\mathcal{F}$ is computationally indistinguishable from the uniform distribution on the whole family of $[n, k]$ codes.*

Two types of attacks

In algebraic code–based cryptography, there are two major types of attacks:

- **Message recovery attacks** based on generic decoding algorithms. Exponential time if $t = \Theta(n)$.
- **Key recovery attacks :** ad hoc methods to recover $s \in \mathcal{S}$ such that the public key $\mathscr{C}_{\text{pub}} = \mathscr{C}(s)$.

Two types of attacks

In algebraic code–based cryptography, there are two major types of attacks:

- **Message recovery attacks** based on generic decoding algorithms. Exponential time if $t = \Theta(n)$.
- **Key recovery attacks :** ad hoc methods to recover $s \in \mathcal{S}$ such that the public key $\mathscr{C}_{\text{pub}} = \mathscr{C}(s)$.

We focus on **key recovery attacks** in the present talk.

Sidelnikov Shestakov, 1992

Efficient key recovery attack on GRS codes.

Idea.

- From a generator matrix $\boldsymbol{G}$ of a code $\mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y})$,
- compute two minimum weight codewords whose supports are close,
- they correspond to split polynomials with many common roots. The ratio of these polynomial is a homography. This provides information on $\boldsymbol{x}$.

Sidelnikov Shestakov, 1992

Efficient key recovery attack on GRS codes.

Idea.

- From a generator matrix $\boldsymbol{G}$ of a code $\mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y})$,
- compute two minimum weight codewords whose supports are close,
- they correspond to split polynomials with many common roots. The ratio of these polynomial is a homography. This provides information on $\boldsymbol{x}$.

Note.

- Computing minimum weight codewords is hard but...
- is only Gaussian elimination for GRS codes!

Sidelnikov Shestakov, 1992

Efficient key recovery attack on GRS codes.

Idea.

- From a generator matrix $\boldsymbol{G}$ of a code $\mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y})$,
- compute two minimum weight codewords whose supports are close,
- they correspond to split polynomials with many common roots. The ratio of these polynomial is a homography. This provides information on $\boldsymbol{x}$.

Note.

- Computing minimum weight codewords is hard but...
- is only Gaussian elimination for GRS codes!

This is a **polynomial time distinguisher!**

Some attacks deriving from Sidelnikov Shestakov

- Minder Shokrollahi 2007. Broke Sidelnikov's proposal based on binary Reed Muller codes. Subexponential time attack;
- Faure Minder, Broke AG codes from hyperelliptic curves. The cost of the attack is exponential in the curve's genus.

Some attacks deriving from Sidelnikov Shestakov

- Minder Shokrollahi 2007. Broke Sidelnikov's proposal based on binary Reed Muller codes. Subexponential time attack;
- Faure Minder, Broke AG codes from hyperelliptic curves. The cost of the attack is exponential in the curve's genus.

In red: due to the cost of computing minimum weight codewords.

Algebraic attacks by polynomial system solving

Idea. A code $\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})$ code is contained in the kernel of a matrix of the form:

$$\boldsymbol{H} = \begin{pmatrix} y_1 & \cdots & y_n \\ x_1 y_1 & \cdots & x_n y_n \\ \vdots & & \vdots \\ x_1^{r-1} y_1 & \cdots & x_n^{r-1} y_n \end{pmatrix}$$

Put x_i, y_i as formal variables X_i, Y_i and solve the polynomial system:

$$\boldsymbol{H}(X_i, Y_i) \cdot {}^t\boldsymbol{G} = 0$$

Algebraic attacks by polynomial system solving

Idea. A code $\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})$ code is contained in the kernel of a matrix of the form:

$$\boldsymbol{H} = \begin{pmatrix} y_1 & \cdots & y_n \\ x_1 y_1 & \cdots & x_n y_n \\ \vdots & & \vdots \\ x_1^{r-1} y_1 & \cdots & x_n^{r-1} y_n \end{pmatrix}$$

Put x_i, y_i as formal variables X_i, Y_i and solve the polynomial system:

$$\boldsymbol{H}(X_i, Y_i) \cdot {}^t\boldsymbol{G} = 0$$

- For usual McEliece parameters, the resolution of such a polynomial system is out of reach. But...

Algebraic attacks by polynomial system solving

Idea. A code $\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})$ code is contained in the kernel of a matrix of the form:

$$\boldsymbol{H} = \begin{pmatrix} y_1 & \cdots & y_n \\ x_1 y_1 & \cdots & x_n y_n \\ \vdots & & \vdots \\ x_1^{r-1} y_1 & \cdots & x_n^{r-1} y_n \end{pmatrix}$$

Put x_i, y_i as formal variables X_i, Y_i and solve the polynomial system:

$$\boldsymbol{H}(X_i, Y_i) \cdot {}^t\boldsymbol{G} = 0$$

- For usual McEliece parameters, the resolution of such a polynomial system is out of reach. But... if you use alternant codes with automorphisms...

Algebraic attacks on alternant codes with automorphisms

Given a code $\mathscr{C} \subseteq \mathbb{F}_q^n$ with a group action $\mathcal{G}$, one can define:

- The *invariant code*

$$\mathscr{C}^{\mathcal{G}} \stackrel{\text{def}}{=} \{\boldsymbol{x} \in \mathscr{C} \mid \forall \sigma \in \mathcal{G},\ \sigma(\boldsymbol{x}) = \boldsymbol{x}\}.$$

Algebraic attacks on alternant codes with automorphisms

Given a code $\mathscr{C} \subseteq \mathbb{F}_q^n$ with a group action $\mathcal{G}$, one can define:

- The *invariant code*

$$\mathscr{C}^{\mathcal{G}} \stackrel{\text{def}}{=} \{\boldsymbol{x} \in \mathscr{C} \mid \forall \sigma \in \mathcal{G},\ \sigma(\boldsymbol{x}) = \boldsymbol{x}\}.$$

If the action of $\mathcal{G}$ is public, then $\mathscr{C}^{\mathcal{G}}$ is computable in polynomial time. Moreover,

Theorem 1 (Faugère, Otmani, Perret, Portzamparc, Tillich 2014)

If $\mathscr{C} = \mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})$ then $\mathscr{C}^{\mathcal{G}} = \mathscr{A}_{r'}(\boldsymbol{x}^{\mathcal{G}}, \boldsymbol{y}^{\mathcal{G}})$ for $r' \approx \frac{r}{|\mathcal{G}|}$ and for some $\boldsymbol{x}^{\mathcal{G}}$, $\boldsymbol{y}^{\mathcal{G}}$ of lengths $\approx \frac{n}{|\mathcal{G}|}$.

Theorem 2 (Barelli, 2018)

If $\mathscr{C} = \mathcal{C}_L(X, \mathcal{P}, G)$ then $\mathscr{C}^{\mathcal{G}} = \mathcal{C}_L(X/\mathcal{G}, \mathcal{P}^{\mathcal{G}}, G^{\mathcal{G}})$ where $|\mathcal{P}^{\mathcal{G}}| \approx \frac{|\mathcal{P}|}{|\mathcal{G}|}$ and $\deg G^{\mathcal{G}} \approx \frac{\deg G}{|\mathcal{G}|}$*. (+ This results extends to subfield subcodes).*

Algebraics attacks on the invariant code

- The algebraic attack can be performed on the invariant code and is easier (less variables, equations of smaller degree).
  - Attacks on quasi–cyclic and quasi–dyadic Goppa/alternant codes, (Faugère, Otmani, Perret, Portzamparc, Tillich 2010, 2014)
- Deducing the secret on the original code from the structure of the invariant code can be done in polynomial time (Barelli, WCC 2017).

$\star$–product and square codes

In $\mathbb{F}_q^n$ we denote by $\star$ the component wise product:

$$\boldsymbol{u} \star \boldsymbol{v} \stackrel{\text{def}}{=} (u_1 v_1, \dots, u_n v_n).$$

Then, the star product of two codes $\mathscr{A}, \mathscr{B} \subseteq \mathbb{F}_q^n$:

$$\mathscr{A} \star \mathscr{B} \stackrel{\text{def}}{=} \mathbf{Span}\{\boldsymbol{a} \star \boldsymbol{b} \mid \boldsymbol{a} \in \mathscr{A},\ \boldsymbol{b} \in \mathscr{B}\}$$

If $\mathscr{A} = \mathscr{B}$, then we denote by $\mathscr{A}^2 \stackrel{\text{def}}{=} \mathscr{A} \star \mathscr{A}$.

The why of $\star$–product

Algebraic codes are *evaluation codes* from an algebra

- $\mathbb{F}_q[X]$ (GRS, alternant codes),
- $\mathbb{F}_q[X_1, \dots, X_n]$ (Reed–Muller codes)
- Ring $\mathcal{O}_S$ of regular functions on an open subset of a curve (AG codes and their subcodes)

The why of $\star$–product

Algebraic codes are *evaluation codes* from an algebra

- $\mathbb{F}_q[X]$ (GRS, alternant codes),
- $\mathbb{F}_q[X_1, \dots, X_n]$ (Reed–Muller codes)
- Ring $\mathcal{O}_S$ of regular functions on an open subset of a curve (AG codes and their subcodes)

Idea. Import the ring structure at the level of codes to get further information on the public key.

A wonderful distinguisher

Theorem 3 (Cascudo, Cramer, Mirandola, Zémor 2013)

Let $\mathscr{R}$ be a random $[n, k]$–code then

$$\mathbf{Prob}\left(\dim \mathscr{R}^2 < \min\left(n, \binom{k+1}{2}\right)\right) \longrightarrow 0. \qquad (n, k \to \infty)$$

A wonderful distinguisher

Theorem 3 (Cascudo, Cramer, Mirandola, Zémor 2013)

Let $\mathscr{R}$ be a random $[n,k]$–code then

$$\mathbf{Prob}\left(\dim \mathscr{R}^2 < \min\left(n, \binom{k+1}{2}\right)\right) \longrightarrow 0. \qquad (n, k \to \infty)$$

Theorem 4

For $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{F}_q^n \times (\mathbb{F}_q^\times)^n$,

$$\mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y})^2 = \mathbf{GRS}_{2k-1}(\boldsymbol{x}, \boldsymbol{y} \star \boldsymbol{y}).$$

Remark

Similar result for AG codes $\mathcal{C}_L(X, \mathcal{P}, G)^2 = \mathcal{C}_L(X, \mathcal{P}, 2G)$ under some conditions on $\deg G$.

First use of $\star$ Wieschebrink 2010

On Berger Loidreau system:

Public key $\mathscr{C} \subseteq \mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y})$ of codimension $\ell \approx 5$;

Secret key $s = (\boldsymbol{x}, \boldsymbol{y})$.

First use of $\star$ Wieschebrink 2010

On Berger Loidreau system:

Public key $\mathscr{C} \subseteq \mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y})$ of codimension $\ell \approx 5$;

Secret key $s = (\boldsymbol{x}, \boldsymbol{y})$.

Fact.

$$\mathscr{C}^2 = \mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y})^2 \text{ with a high probability.}$$

First use of $\star$ Wieschebrink 2010

On Berger Loidreau system:

Public key $\mathscr{C} \subseteq \mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y})$ of codimension $\ell \approx 5$;

Secret key $s = (\boldsymbol{x}, \boldsymbol{y})$.

Fact.

$$\mathscr{C}^2 = \mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y})^2 \text{ with a high probability.}$$

Wieschebrink's attack.

- Compute $\mathscr{C}^2$;
- Perform Sidelnikov Shestakov attack on $\mathscr{C}^2$ to recover $(\boldsymbol{x}, \boldsymbol{y} \star \boldsymbol{y})$.
- Deduce $(\boldsymbol{x}, \boldsymbol{y})$.

Other attacks based on the raw $\star$–product distinguisher

- Wieschebrink's scheme (C., Gautier, Gaborit, Otmani, Tillich, 2015);
- BBCRS scheme (C., Gautier, Otmani, Tillich, 2015);
- RLCE scheme (C. Lequesne, Tillich, 2019)

Distinguisher and filtration attack

Illustrative example on GRS codes. Suppose you know the codes

- $\mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y})$ $(\mathbb{F}_q[X]_{\leqslant k-1})$
- $\mathbf{GRS}_{k-1}(\boldsymbol{x}, \boldsymbol{y})$ $(\mathbb{F}_q[X]_{\leqslant k-2})$

Distinguisher and filtration attack

Illustrative example on GRS codes. Suppose you know the codes

- $\mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y})$ $(\mathbb{F}_q[X]_{\leqslant k-1})$
- $\mathbf{GRS}_{k-1}(\boldsymbol{x}, \boldsymbol{y})$ $(\mathbb{F}_q[X]_{\leqslant k-2})$

You'd like to compute

- $\mathbf{GRS}_{k-2}(\boldsymbol{x}, \boldsymbol{y})$ $(\mathbb{F}_q[X]_{\leqslant k-3})$

Distinguisher and filtration attack

Illustrative example on GRS codes. Suppose you know the codes

- $\mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y})$ $(\mathbb{F}_q[X]_{\leqslant k-1})$
- $\mathbf{GRS}_{k-1}(\boldsymbol{x}, \boldsymbol{y})$ $(\mathbb{F}_q[X]_{\leqslant k-2})$

You'd like to compute

- $\mathbf{GRS}_{k-2}(\boldsymbol{x}, \boldsymbol{y})$ $(\mathbb{F}_q[X]_{\leqslant k-3})$

Then note that

$$\mathbf{GRS}_{k-2}(\boldsymbol{x}, \boldsymbol{y}) \star \mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y}) \subseteq \mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y})^2$$

Indeed :

$$(k-3) + (k-1) = 2(k-2).$$

Distinguisher and filtration attack

$\mathbf{GRS}_{k-2}(\boldsymbol{x}, \boldsymbol{y})$ can be computed as the set

$$\mathbf{Cond}(\mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y}), \mathbf{GRS}_{k-1}(\boldsymbol{x}, \boldsymbol{y})^2) \stackrel{\text{def}}{=} \left\{ \boldsymbol{z} \in \mathbb{F}_q^n \mid \boldsymbol{z} \star \mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y}) \subseteq \mathbf{GRS}_{k-1}(\boldsymbol{x}, \boldsymbol{y})^2 \right\}$$

Distinguisher and filtration attack

$\mathbf{GRS}_{k-2}(\boldsymbol{x}, \boldsymbol{y})$ can be computed as the set

$$\mathbf{Cond}(\mathbf{GRS}_{k}(\boldsymbol{x}, \boldsymbol{y}), \mathbf{GRS}_{k-1}(\boldsymbol{x}, \boldsymbol{y})^2) \stackrel{\text{def}}{=} \left\{\boldsymbol{z} \in \mathbb{F}_q^n \mid \boldsymbol{z} \star \mathbf{GRS}_{k}(\boldsymbol{x}, \boldsymbol{y}) \subseteq \mathbf{GRS}_{k-1}(\boldsymbol{x}, \boldsymbol{y})^2\right\}$$

Then reiterate the process to deduce the filtration

$$\mathbf{GRS}_{k}(\boldsymbol{x}, \boldsymbol{y}) \supseteq \mathbf{GRS}_{k}(\boldsymbol{x}, \boldsymbol{y}) \supseteq \cdots \supseteq \mathbf{GRS}_{r}(\boldsymbol{x}, \boldsymbol{y}) \supseteq \cdots$$

Distinguisher and filtration attack

$\mathbf{GRS}_{k-2}(\boldsymbol{x}, \boldsymbol{y})$ can be computed as the set

$$\mathbf{Cond}(\mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y}), \mathbf{GRS}_{k-1}(\boldsymbol{x}, \boldsymbol{y})^2) \stackrel{\text{def}}{=} \left\{\boldsymbol{z} \in \mathbb{F}_q^n \mid \boldsymbol{z} \star \mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y}) \subseteq \mathbf{GRS}_{k-1}(\boldsymbol{x}, \boldsymbol{y})^2\right\}$$

Then reiterate the process to deduce the filtration

$$\mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y}) \supseteq \mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y}) \supseteq \cdots \supseteq \mathbf{GRS}_r(\boldsymbol{x}, \boldsymbol{y}) \supseteq \cdots$$

Remark

There is no reason to know both $\mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y})$ *and* $\mathbf{GRS}_{k-1}(\boldsymbol{x}, \boldsymbol{y})$ *but* $\mathbf{GRS}_{k-1}(\boldsymbol{x}, \boldsymbol{y})$ *can be replaced by a shortening of* $\mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y})$ *at one position.*

Applications

- Alternative attack on GRS codes (C., Gautier, Gaborit, Otmani, Tillich, 2015);
- AG codes and their subcodes (C., Márquez–Corbella, Pellikaan, 2014–17);
- Wild Goppa codes for $m = 2$ (C. Otmani, Tillich, 2014–17);

Applications

- Alternative attack on GRS codes (C., Gautier, Gaborit, Otmani, Tillich, 2015);
- AG codes and their subcodes (C., Márquez–Corbella, Pellikaan, 2014–17);
- Wild Goppa codes for $m = 2$ (C. Otmani, Tillich, 2014–17);

Remark

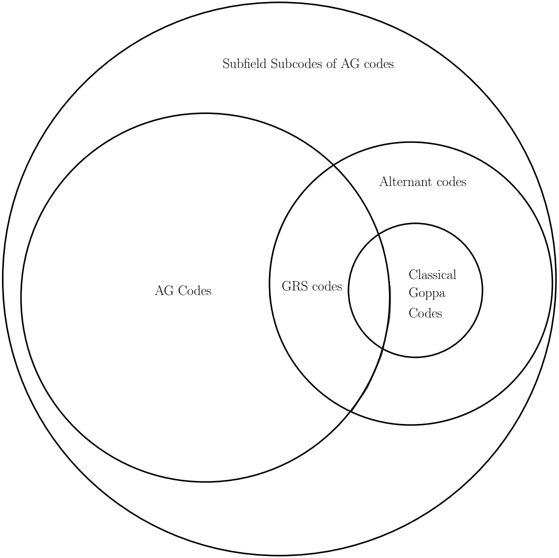
No more need to compute minimum weight codewords. Succeeds where Sidelnikov Shestakov fails!

1 History of code–based cryptography

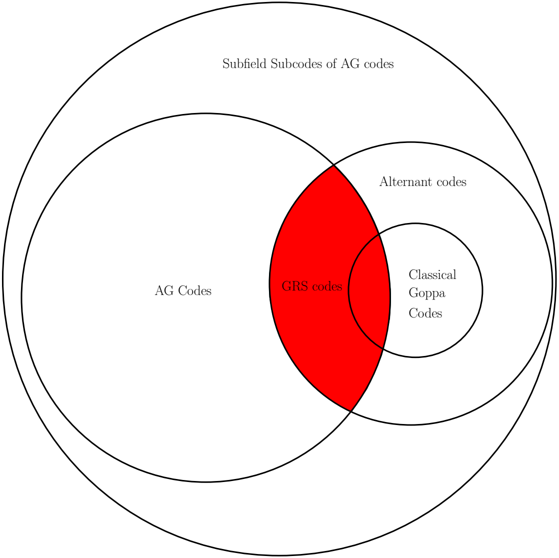
2 Algebraic cryptanalysis in code–based cryptography

3 How to design secure schemes with codes?

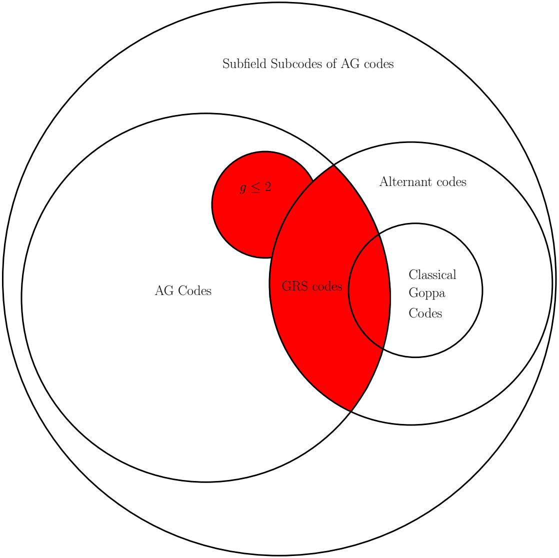
Algebraic codes



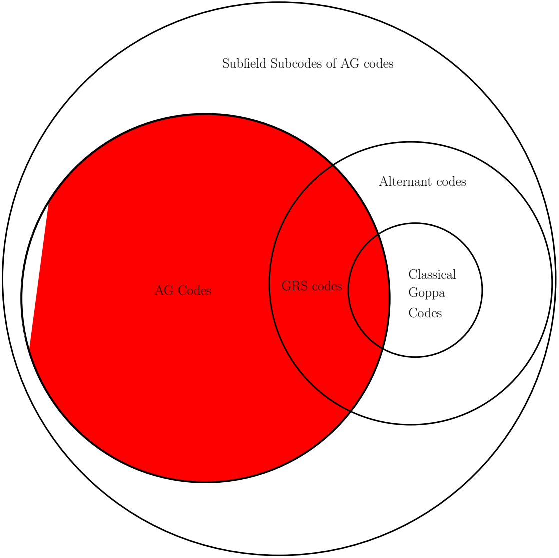
Sidelnikov Shestakov 1992



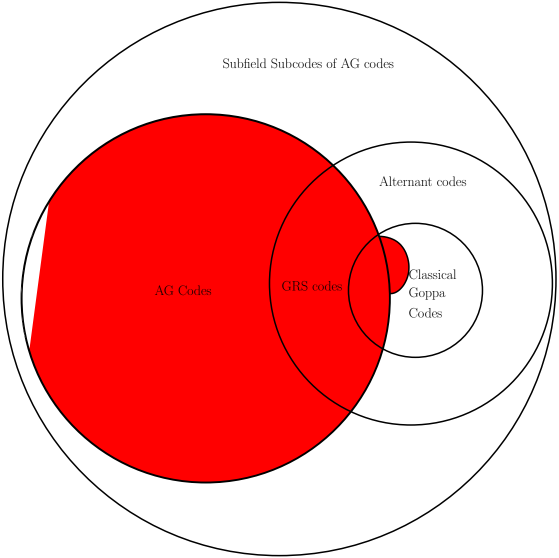
Faure Minder 2008



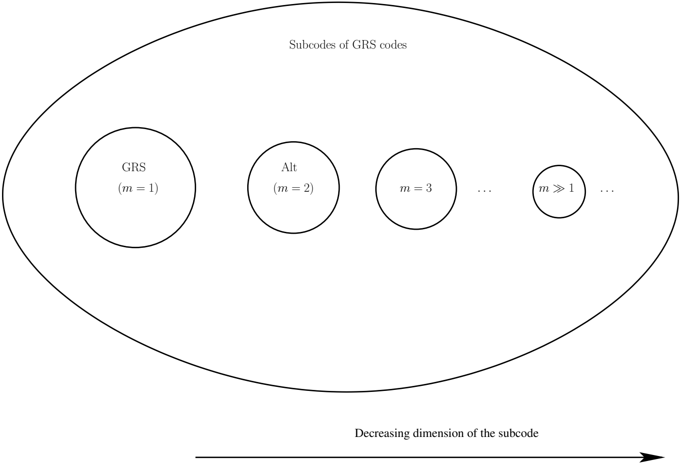
C. Márquez–Corbella, Pellikaan, 2014



C. Otmani, Tillich & Faugère, Perret, Portzamparc 2014

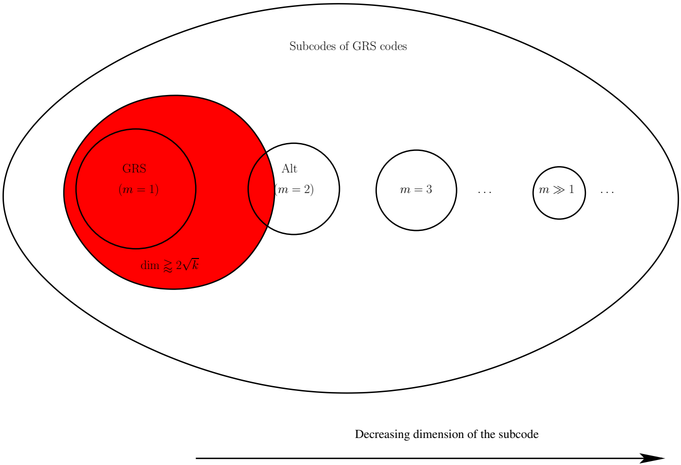


Other point of view : subcodes of GRS codes.



Other point of view : subcodes of GRS codes.

🔴 What filtration attacks can break



Subcodes of GRS codes

GRS

$(m = 1)$

$\dim \gtrapprox 2\sqrt{k}$

Alt

$(m = 2)$

$m = 3$

…

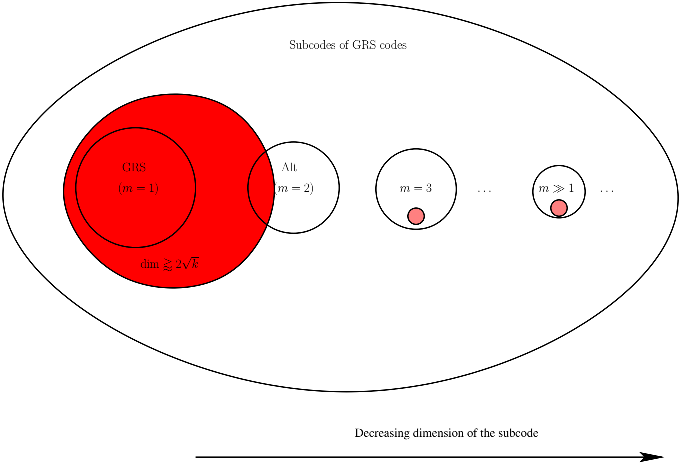
$m \gg 1$

…

Decreasing dimension of the subcode

Other point of view : subcodes of GRS codes.

● What filtration attacks can break ● Faugère, Otmani, Perret, Tillich distinguisher



How to evaluate the security of algebraic codes?

Security analysis framework

- **Sufficiently many codes in the family**, even up to permutation (Sendrier's support splitting algorithm);

How to evaluate the security of algebraic codes?

Security analysis framework

- **Sufficiently many codes in the family**, even up to permutation (Sendrier's support splitting algorithm);
- **Low weight codewords are hard to compute** (avoid, Sidelnikov–Shestakov like attacks).

How to evaluate the security of algebraic codes?

Security analysis framework

- **Sufficiently many codes in the family**, even up to permutation (Sendrier's support splitting algorithm);
- **Low weight codewords are hard to compute** (avoid, Sidelnikov–Shestakov like attacks).
- **No square code distinguisher.**
  - $\mathscr{C}^2, (\mathscr{C}^\perp)^2$ **and their shortenings** should behave like random codes.

How to evaluate the security of algebraic codes?

Security analysis framework

- **Sufficiently many codes in the family**, even up to permutation (Sendrier's support splitting algorithm);
- **Low weight codewords are hard to compute** (avoid, Sidelnikov–Shestakov like attacks).
- **No square code distinguisher.**
  - $\mathscr{C}^2, (\mathscr{C}^\perp)^2$ **and their shortenings** should behave like random codes.
- If you use some automorphis group, check the above properties for both $\mathscr{C}$ and $\mathscr{C}^{\mathcal{G}}$.

How to evaluate the security of algebraic codes?

Security analysis framework

- **Sufficiently many codes in the family**, even up to permutation (Sendrier's support splitting algorithm);
- **Low weight codewords are hard to compute** (avoid, Sidelnikov–Shestakov like attacks).
- **No square code distinguisher.**
  - $\mathscr{C}^2, (\mathscr{C}^\perp)^2$ **and their shortenings** should behave like random codes.
- If you use some automorphis group, check the above properties for both $\mathscr{C}$ and $\mathscr{C}^{\mathcal{G}}$.
- How to resist to attacks by algebraic systems solving? Difficult question...

What is still surviving?

Algebraic world

- Binary Goppa codes (NIST's *Classic McEliece* and *NTS KEM*)
- Goppa codes for $m \gg 2$.
- Goppa codes with a “small” automorphism group
- Subfield subcodes of AG codes

Probabilistic world

- Quasi–cyclic MDPC codes;

What is still surviving?

Algebraic world

- Binary Goppa codes (NIST's *Classic McEliece* and *NTS KEM*)
- Goppa codes for $m \gg 2$.
- Goppa codes with a "small" automorphism group
- Subfield subcodes of AG codes

Advantages : Short ciphertexts, no decoding failure.

Probabilistic world

- Quasi–cyclic MDPC codes;

What is still surviving?

Algebraic world

- Binary Goppa codes (NIST's *Classic McEliece* and *NTS KEM*)
- Goppa codes for $m \gg 2$.
- Goppa codes with a "small" automorphism group
- Subfield subcodes of AG codes

Advantages : Short ciphertexts, no decoding failure.

Probabilistic world

- Quasi–cyclic MDPC codes;

Advantages : short keys, especially designed for cryptography, somehow simpler security analysis.

What is still surviving?

Algebraic world

- Binary Goppa codes (NIST's *Classic McEliece* and *NTS KEM*)
- Goppa codes for $m \gg 2$.
- Goppa codes with a "small" automorphism group
- Subfield subcodes of AG codes

Advantages : Short ciphertexts, no decoding failure.

Probabilistic world

- Quasi–cyclic MDPC codes;

Advantages : short keys, especially designed for cryptography, somehow simpler security analysis.

Other paradigms HQC, RQC : do **not** rely on indistinguishability assumption: promising application of algebraic codes!

Thanks for your attention!

Questions?