An Exploration of Structural Attacks on the McEliece Public Key Cryptosystem

Filip Stojanovic

University of Ottawa

August 22, 2020

Table of Contents

Prerequisite Coding Theory Knowledge

- Let p be prime and $m \in \mathbb{N}_+$. $\mathbb{F}_{p^m}$ denotes the finite field of size p^m.
  - By its construction, $\mathbb{F}_{p^m} \supseteq \mathbb{F}_p$.

Prerequisite Coding Theory Knowledge

- Let p be prime and $m \in \mathbb{N}_+$. $\mathbb{F}_{p^m}$ denotes the finite field of size p^m.
  - By its construction, $\mathbb{F}_{p^m} \supseteq \mathbb{F}_p$.
- A (n, k) code over $\mathbb{F}_{p^m}$ is a subspace of $\mathbb{F}_{p^m}^n$ of dimension k.
  - n is the length of the code.
  - k is the dimension of the code.

Prerequisite Coding Theory Knowledge

- Let p be prime and $m \in \mathbb{N}_+$. $\mathbb{F}_{p^m}$ denotes the finite field of size p^m.
  - By its construction, $\mathbb{F}_{p^m} \supseteq \mathbb{F}_p$.
- A (n, k) code over $\mathbb{F}_{p^m}$ is a subspace of $\mathbb{F}_{p^m}^n$ of dimension k.
  - n is the length of the code.
  - k is the dimension of the code.
- If C is a (n, k) code over $\mathbb{F}_{p^m}$, then C admits a basis in $\mathbb{F}_{p^m}^n$.

Prerequisite Coding Theory Knowledge

- Let p be prime and $m \in \mathbb{N}_+$. $\mathbb{F}_{p^m}$ denotes the finite field of size p^m.
  - By its construction, $\mathbb{F}_{p^m} \supseteq \mathbb{F}_p$.
- A (n, k) code over $\mathbb{F}_{p^m}$ is a subspace of $\mathbb{F}_{p^m}^n$ of dimension k.
  - n is the length of the code.
  - k is the dimension of the code.
- If C is a (n, k) code over $\mathbb{F}_{p^m}$, then C admits a basis in $\mathbb{F}_{p^m}^n$.

Definition

If C is a (n, k) code over $\mathbb{F}_{p^m}$ and B is a basis for C, then a generator matrix for C is $\mathbf{G} \in \mathcal{M}_{n\times k}(\mathbb{F}_{p^m})$ whose columns are the vectors in B.

Prerequisite Coding Theory Knowledge

- Let p be prime and $m \in \mathbb{N}_+$. $\mathbb{F}_{p^m}$ denotes the finite field of size p^m.
  - By its construction, $\mathbb{F}_{p^m} \supseteq \mathbb{F}_p$.
- A (n, k) code over $\mathbb{F}_{p^m}$ is a subspace of $\mathbb{F}_{p^m}^n$ of dimension k.
  - n is the length of the code.
  - k is the dimension of the code.
- If C is a (n, k) code over $\mathbb{F}_{p^m}$, then C admits a basis in $\mathbb{F}_{p^m}^n$.

Definition

If C is a (n, k) code over $\mathbb{F}_{p^m}$ and B is a basis for C, then a generator matrix for C is $\mathbf{G} \in \mathcal{M}_{n\times k}(\mathbb{F}_{p^m})$ whose columns are the vectors in B.

- Multiplying $\mathbf{G}$ by $m \in \mathbb{F}_{p^m}^k$ will produce a vector in the code C.

Error Correction

- Error to a codeword = entry replaced by a different value in $\mathbb{F}_{p^m}$

Error Correction

- Error to a codeword = entry replaced by a different value in $\mathbb{F}_{p^m}$

Definition

The Hamming distance is a metric d on $\mathbb{F}_{p^m}^n$ s.t. $\forall x, y \in \mathbb{F}_{p^m}^n$,
$d(x, y) := |\{i : x_i \neq y_i\}|$.

Error Correction

- Error to a codeword = entry replaced by a different value in $\mathbb{F}_{p^m}$

Definition

The Hamming distance is a metric d on $\mathbb{F}_{p^m}^n$ s.t. $\forall x, y \in \mathbb{F}_{p^m}^n$, $d(x, y) := |\{i : x_i \neq y_i\}|$.

- To correct an error-ridden codeword, search through the code to find the closest codeword to that vector
  - If there isn't a unique closest codeword, the code can't correct the errors
  - If the closest codeword is unique, the code corrects the error-ridden vector to that codeword

Error Correction

- Error to a codeword = entry replaced by a different value in $\mathbb{F}_{p^m}$

Definition

The Hamming distance is a metric d on $\mathbb{F}_{p^m}^n$ s.t. $\forall x, y \in \mathbb{F}_{p^m}^n$, $d(x, y) := |\{i : x_i \neq y_i\}|$.

- To correct an error-ridden codeword, search through the code to find the closest codeword to that vector
  - If there isn't a unique closest codeword, the code can't correct the errors
  - If the closest codeword is unique, the code corrects the error-ridden vector to that codeword

Definition

A code C can correct t errors if for any vector in $\mathbb{F}_{p^m}^n$ of distance at most t to some codeword of C, there is a unique codeword of distance at most t to that vector.

The McEliece PKC

The McEliece PKC

- Private Key
  - $\mathbf{G}$, a $n \times k$ generator matrix for a code C
  - $\mathbf{S} \in GL_k(\mathbb{F}_{p^m})$
  - $\mathbf{P}$, a $n \times n$ permutation matrix
  - D_G, an efficient decryption algorithm for the code C

The McEliece PKC

- Private Key
  - $\mathbf{G}$, a $n \times k$ generator matrix for a code C
  - $\mathbf{S} \in GL_k(\mathbb{F}_{p^m})$
  - $\mathbf{P}$, a $n \times n$ permutation matrix
  - D_G, an efficient decryption algorithm for the code C
- Public Key
  - $\mathbf{M} := \mathbf{PGS}$, a $n \times k$ generator for a permutation of code C.
  - t, the number errors C can correct

The McEliece PKC

- Private Key
  - $\mathbf{G}$, a $n \times k$ generator matrix for a code C
  - $\mathbf{S} \in GL_k(\mathbb{F}_{p^m})$
  - $\mathbf{P}$, a $n \times n$ permutation matrix
  - D_G, an efficient decryption algorithm for the code C
- Public Key
  - $\mathbf{M} := \mathbf{PGS}$, a $n \times k$ generator for a permutation of code C.
  - t, the number errors C can correct
- Encryption
  - For $m \in \mathbb{F}_{p^m}^k$, $m \mapsto \mathbf{M}m + z$ s.t. $d(z, 0) = t$

The McEliece PKC

- Private Key
  - $\mathbf{G}$, a $n \times k$ generator matrix for a code C
  - $\mathbf{S} \in GL_k(\mathbb{F}_{p^m})$
  - $\mathbf{P}$, a $n \times n$ permutation matrix
  - D_G, an efficient decryption algorithm for the code C
- Public Key
  - $\mathbf{M} := \mathbf{PGS}$, a $n \times k$ generator for a permutation of code C.
  - t, the number errors C can correct
- Encryption
  - For $m \in \mathbb{F}_{p^m}^k$, $m \mapsto \mathbf{M}m + z$ s.t. $d(z, 0) = t$
- Decryption
  - Multiply $\mathbf{M}m + z$ by $\mathbf{P}^{-1}$ to get $c' := \mathbf{GS}m + \mathbf{P}^{-1}z$
  - Apply D_G to c' to recover $\mathbf{GS}m$
  - Multiply $\mathbf{GS}m$ by $\mathbf{S}^{-1}\mathbf{G}_{LI}$ to recover m

The McEliece PKC

- Private Key
  - $\mathbf{G}$, a $n \times k$ generator matrix for a code C
  - $\mathbf{S} \in GL_k(\mathbb{F}_{p^m})$
  - $\mathbf{P}$, a $n \times n$ permutation matrix
  - D_G, an efficient decryption algorithm for the code C
- Public Key
  - $\mathbf{M} := \mathbf{PGS}$, a $n \times k$ generator for a permutation of code C.
  - t, the number errors C can correct
- Encryption
  - For $m \in \mathbb{F}_{p^m}^k$, $m \mapsto \mathbf{M}m + z$ s.t. $d(z, 0) = t$
- Decryption
  - Multiply $\mathbf{M}m + z$ by $\mathbf{P}^{-1}$ to get $c' := \mathbf{GS}m + \mathbf{P}^{-1}z$
  - Apply D_G to c' to recover $\mathbf{GS}m$
  - Multiply $\mathbf{GS}m$ by $\mathbf{S}^{-1}\mathbf{G}_{LI}$ to recover m
- Attacking
  - Replace D_G with some generic, efficient decoding algorithm

The McEliece PKC

- Private Key
  - $\mathbf{G}$, a $n \times k$ generator matrix for a code C
  - $\mathbf{S} \in GL_k(\mathbb{F}_{p^m})$
  - $\mathbf{P}$, a $n \times n$ permutation matrix
  - D_G, an efficient decryption algorithm for the code C
- Public Key
  - $\mathbf{M} := \mathbf{PGS}$, a $n \times k$ generator for a permutation of code C.
  - t, the number errors C can correct
- Encryption
  - For $m \in \mathbb{F}_{p^m}^k$, $m \mapsto \mathbf{M}m + z$ s.t. $d(z, 0) = t$
- Decryption
  - Multiply $\mathbf{M}m + z$ by $\mathbf{P}^{-1}$ to get $c' := \mathbf{GS}m + \mathbf{P}^{-1}z$
  - Apply D_G to c' to recover $\mathbf{GS}m$
  - Multiply $\mathbf{GS}m$ by $\mathbf{S}^{-1}\mathbf{G}_{LI}$ to recover m
- Attacking
  - Replace D_G with some generic, efficient decoding algorithm
  - Find the parameters defining D_G from the public key

But What is a Goppa Code?

- Codes coming from algebraic geometry

But What is a Goppa Code?

- Codes coming from algebraic geometry
- They have a messy definition of their own, but we can instead characterize them by their relationships to GRS codes
  - GRS codes are parametrized by a pair of $\mathbb{F}_{p^m}^n$ vectors (α, β)
  - We'll get into that shortly...

But What is a Goppa Code?

- Codes coming from algebraic geometry
- They have a messy definition of their own, but we can instead characterize them by their relationships to GRS codes
    - GRS codes are parametrized by a pair of $\mathbb{F}_{p^m}^n$ vectors (α, β)
    - We'll get into that shortly...

Definition

Let $\alpha, \beta \in \mathbb{F}_{p^m}^n$. The Goppa code defined by (α, β) is $\Gamma(\alpha, \beta) = GRS_{n,k}(\alpha, \beta) \cap \mathbb{F}_p^n$.

But What is a Goppa Code?

- Codes coming from algebraic geometry
- They have a messy definition of their own, but we can instead characterize them by their relationships to GRS codes
  - GRS codes are parametrized by a pair of $\mathbb{F}_{p^m}^n$ vectors (α, β)
  - We'll get into that shortly...

Definition

Let $\alpha, \beta \in \mathbb{F}_{p^m}^n$. The Goppa code defined by (α, β) is $\Gamma(\alpha, \beta) = GRS_{n,k}(\alpha, \beta) \cap \mathbb{F}_p^n$.

- This is a code in $\mathbb{F}_p^n$, not $\mathbb{F}_{p^m}^n$

But What is a Goppa Code?

- Codes coming from algebraic geometry
- They have a messy definition of their own, but we can instead characterize them by their relationships to GRS codes
  - GRS codes are parametrized by a pair of $\mathbb{F}_{p^m}^n$ vectors (α, β)
  - We'll get into that shortly...

Definition

Let $\alpha, \beta \in \mathbb{F}_{p^m}^n$. The Goppa code defined by (α, β) is $\Gamma(\alpha, \beta) = GRS_{n,k}(\alpha, \beta) \cap \mathbb{F}_p^n$.

- This is a code in $\mathbb{F}_p^n$, not $\mathbb{F}_{p^m}^n$

Lemma

Let $\Gamma(\alpha, \beta) = GRS_{n,k}(\alpha, \beta) \cap \mathbb{F}_p^n$. $\dim_{\mathbb{F}_p}(\Gamma(\alpha, \beta)) \leq \dim_{\mathbb{F}_{p^m}}(GRS_{n,k}(\alpha, \beta))$.

GRS Codes

- Parameters
  - $\alpha \in \mathbb{F}_{p^m}^n$ s.t. $\alpha_i \neq \alpha_j \ \forall i \neq j$
  - $\beta \in \mathbb{F}_{p^m}^n$ s.t. $\beta_i \neq 0 \ \forall i$

GRS Codes

- Parameters
  - $\alpha \in \mathbb{F}_{p^m}^n$ s.t. $\alpha_i \neq \alpha_j \;\; \forall i \neq j$
  - $\beta \in \mathbb{F}_{p^m}^n$ s.t. $\beta_i \neq 0 \;\; \forall i$

Definition

The (n, k) GRS code defined by (α, β) is
$GRS_{n,k}(\alpha, \beta) := \{(\beta_1 f(\alpha_1), \ldots, \beta_n f(\alpha_n)) : f \in \mathbb{P}_{k-1}(\mathbb{F}_{p^m})\}$.

GRS Codes

- Parameters
  - $\alpha \in \mathbb{F}_{p^m}^n$ s.t. $\alpha_i \neq \alpha_j \ \forall i \neq j$
  - $\beta \in \mathbb{F}_{p^m}^n$ s.t. $\beta_i \neq 0 \ \forall i$

Definition

The (n, k) GRS code defined by (α, β) is
$GRS_{n,k}(\alpha, \beta) := \{(\beta_1 f(\alpha_1), \ldots, \beta_n f(\alpha_n)) : f \in \mathbb{P}_{k-1}(\mathbb{F}_{p^m})\}$.

- Several different parameters may define the same GRS code.

GRS Codes

- Parameters
  - $\alpha \in \mathbb{F}_{p^m}^n$ s.t. $\alpha_i \neq \alpha_j \;\; \forall i \neq j$
  - $\beta \in \mathbb{F}_{p^m}^n$ s.t. $\beta_i \neq 0 \;\; \forall i$

Definition

The (n, k) GRS code defined by (α, β) is
$GRS_{n,k}(\alpha, \beta) := \{(\beta_1 f(\alpha_1), \ldots, \beta_n f(\alpha_n)) : f \in \mathbb{P}_{k-1}(\mathbb{F}_{p^m})\}$.

- Several different parameters may define the same GRS code.

Proposition

Let $\alpha, \beta \in \mathbb{F}_{p^m}^n$ s.t. $\alpha_i \neq \alpha_j \;\; \forall i \neq j$ and $\beta_i \neq 0 \;\; \forall i$. Let $\mu, \nu, \eta \in \mathbb{F}_{p^m}$ s.t. $\mu, \eta \neq 0$. Define $\alpha', \beta' \in \mathbb{F}_{p^m}^n$ by $\alpha_i' = \mu\alpha_i + \nu$ and $\beta_i' = \eta\beta_i \;\; \forall i = 1, \ldots, n$. In this case, $GRS_{n,k}(\alpha, \beta) = GRS_{n,k}(\alpha', \beta')$.

Setting up the Sidelnikov-Shestakov Attack

- Private Key
  - **G**, a $n \times k$ generator matrix for a code C
  - $\mathbf{S} \in GL_k(\mathbb{F}_{p^m})$
  - **P**, a $n \times n$ permutation matrix
  - D_G, an efficient decryption algorithm for the code C
- Public Key
  - $\mathbf{M} := \mathbf{PGS}$, a $n \times k$ generator for a permutation of code C.
  - t, the number errors C can correct

Setting up the Sidelnikov-Shestakov Attack

- Private Key
  - **G**, the $n \times k$ generator matrix for $GRS_{n,k}(\alpha, \beta)$
  - $\mathbf{S} \in GL_k(\mathbb{F}_{p^m})$
  - ~~**P**, a $n \times n$ permutation matrix~~
  - (α, β), as the decryption algorithm requires only the code parameters
- Public Key
  - $\mathbf{M} := \mathbf{GS}$, a $n \times k$ generator for $GRS_{n,k}(\alpha, \beta)$
  - t, the number errors $GRS_{n,k}(\alpha, \beta)$ can correct

Setting up the Sidelnikov-Shestakov Attack

- Private Key
  - **G**, the $n \times k$ generator matrix for $GRS_{n,k}(\alpha, \beta)$
  - $\mathbf{S} \in GL_k(\mathbb{F}_{p^m})$
  - ~~**P**, a $n \times n$ permutation matrix~~
  - (α, β), as the decryption algorithm requires only the code parameters
- Public Key
  - $\mathbf{M} := \mathbf{GS}$, a $n \times k$ generator for $GRS_{n,k}(\alpha, \beta)$
  - t, the number errors $GRS_{n,k}(\alpha, \beta)$ can correct
- The goal of the attack is to recover the code parameters (α, β) given a scrambled generator matrix

Setting up the Sidelnikov-Shestakov Attack

- Private Key
  - **G**, the $n \times k$ generator matrix for $GRS_{n,k}(\alpha, \beta)$
  - $\mathbf{S} \in GL_k(\mathbb{F}_{p^m})$
  - ~~**P**, a $n \times n$ permutation matrix~~
  - (α, β), as the decryption algorithm requires only the code parameters
- Public Key
  - $\mathbf{M} := \mathbf{GS}$, a $n \times k$ generator for $GRS_{n,k}(\alpha, \beta)$
  - t, the number errors $GRS_{n,k}(\alpha, \beta)$ can correct
- The goal of the attack is to recover the code parameters (α, β) given a scrambled generator matrix
  - These can be <u>equivalent parameters</u> that define the <u>same</u> GRS code

Setting up the Sidelnikov-Shestakov Attack

- Private Key
  - **G**, the $n \times k$ generator matrix for $GRS_{n,k}(\alpha, \beta)$
  - **S** $\in GL_k(\mathbb{F}_{p^m})$
  - ~~**P**, a $n \times n$ permutation matrix~~
  - (α, β), as the decryption algorithm requires only the code parameters
- Public Key
  - **M** $:=$ **GS**, a $n \times k$ generator for $GRS_{n,k}(\alpha, \beta)$
  - t, the number errors $GRS_{n,k}(\alpha, \beta)$ can correct
- The goal of the attack is to recover the code parameters (α, β) given a scrambled generator matrix
  - These can be equivalent parameters that define the same GRS code

Lemma

WLOG, $\alpha_1 = 0, \alpha_2 = 1$, and $\beta_1 = 1$.

Proof:

$\exists \mu, \nu, \eta \in \mathbb{F}_{p^m}$ s.t. $\mu, \eta \neq 0$ and for $\alpha' := \mu\alpha + \vec{\nu}$, $\beta' := \eta\beta$, we have $\alpha'_1 = 0, \alpha'_2 = 1, \beta'_1 = 1$.

Setting up the Sidelnikov-Shestakov Attack

Setting up the Sidelnikov-Shestakov Attack

$$\mathbf{M}^{\mathsf{T}} \sim [\mathbf{I}_k | A] = \begin{bmatrix} R_1 \\ R_2 \\ \vdots \\ R_k \end{bmatrix} \text{ s.t } R_i = (\beta_1 p_{R_i}(\alpha_1), \ldots, \beta_n p_{R_i}(\alpha_n))$$

Setting up the Sidelnikov-Shestakov Attack

$$\mathbf{M}^{\mathsf{T}} \sim [\mathbf{I}_k|A] = \begin{bmatrix} R_1 \\ R_2 \\ \vdots \\ R_k \end{bmatrix} \text{ s.t } R_i = (\beta_1 p_{R_i}(\alpha_1), \ldots, \beta_n p_{R_i}(\alpha_n))$$

- $(R_i)_j = \begin{cases} 1, i = j \\ 0, i \neq j \end{cases} \quad \forall i = 1, \ldots, k, \implies p_{R_i}(\alpha_j) = 0 \;\; \forall j \neq i$

Setting up the Sidelnikov-Shestakov Attack

$$\mathbf{M}^{\mathsf{T}} \sim [\mathbf{I}_k | A] = \begin{bmatrix} R_1 \\ R_2 \\ \vdots \\ R_k \end{bmatrix} \text{ s.t } R_i = (\beta_1 p_{R_i}(\alpha_1), \ldots, \beta_n p_{R_i}(\alpha_n))$$

- $(R_i)_j = \begin{cases} 1, i = j \\ 0, i \neq j \end{cases} \quad \forall i = 1, \ldots, k, \implies p_{R_i}(\alpha_j) = 0 \;\; \forall j \neq i$
- But this means $(x - \alpha_j) \mid p_{R_i}(x) \quad \forall j \in \{1, \ldots, k\} \backslash \{i\}$

Setting up the Sidelnikov-Shestakov Attack

$$\mathbf{M}^{\mathsf{T}} \sim [\mathbf{I}_k | A] = \begin{bmatrix} R_1 \\ R_2 \\ \vdots \\ R_k \end{bmatrix} \text{ s.t } R_i = (\beta_1 p_{R_i}(\alpha_1), \ldots, \beta_n p_{R_i}(\alpha_n))$$

- $(R_i)_j = \begin{cases} 1, i = j \\ 0, i \neq j \end{cases} \quad \forall i = 1, \ldots, k, \implies p_{R_i}(\alpha_j) = 0 \ \forall j \neq i$
- But this means $(x - \alpha_j) \mid p_{R_i}(x) \quad \forall j \in \{1, \ldots, k\} \backslash \{i\}$
- Hence, $\prod_{j \in \{1,\ldots,k\} \backslash \{i\}} (x - \alpha_j) \mid p_{R_i}(x)$

Setting up the Sidelnikov-Shestakov Attack

$$\mathbf{M}^\mathsf{T} \sim [\mathbf{I}_k|A] = \begin{bmatrix} R_1 \\ R_2 \\ \vdots \\ R_k \end{bmatrix} \text{ s.t } R_i = (\beta_1 p_{R_i}(\alpha_1), \ldots, \beta_n p_{R_i}(\alpha_n))$$

- $(R_i)_j = \begin{cases} 1, i = j \\ 0, i \neq j \end{cases} \quad \forall i = 1, \ldots, k, \implies p_{R_i}(\alpha_j) = 0 \;\; \forall j \neq i$
- But this means $(x - \alpha_j) \mid p_{R_i}(x) \quad \forall j \in \{1, \ldots, k\} \backslash \{i\}$
- Hence, $\prod_{j \in \{1,\ldots,k\} \backslash \{i\}} (x - \alpha_j) \mid p_{R_i}(x)$
- But since $deg(p_{R_i}) \leq k - 1$, we know p_{R_i} up to scalar multiple

$$p_{R_i}(x) = c_i \cdot \prod_{j \in \{1,\ldots,k\} \backslash \{i\}} (x - \alpha_j) \quad \text{s.t. } c_i \in \mathbb{F}_{p^m}^\times$$

Recovering α

- Divide the non-zero entries of different rows of the RREF of $\mathbf{M}^\intercal$

Recovering α

- Divide the non-zero entries of different rows of the RREF of $\mathbf{M}^{\intercal}$
- $\forall j \geq k+1$, $\frac{(R_1)_j}{(R_2)_j} = \frac{\beta_j p_{R_1}(\alpha_j)}{\beta_j p_{R_2}(\alpha_j)} = \frac{c_1 \prod_{r \in \{1,\ldots,k\} \setminus \{1\}} (\alpha_j - \alpha_r)}{c_2 \prod_{r \in \{1,\ldots,k\} \setminus \{2\}} (\alpha_j - \alpha_r)} = \frac{c_1(\alpha_j - \alpha_2)}{c_2(\alpha_j - \alpha_1)}$

Recovering α

- Divide the non-zero entries of different rows of the RREF of $\mathbf{M}^{\intercal}$
- $\forall j \geq k+1,\ \frac{(R_1)_j}{(R_2)_j} = \frac{\beta_j p_{R_1}(\alpha_j)}{\beta_j p_{R_2}(\alpha_j)} = \frac{c_1 \prod_{r \in \{1,\ldots,k\} \setminus \{1\}} (\alpha_j - \alpha_r)}{c_2 \prod_{r \in \{1,\ldots,k\} \setminus \{2\}} (\alpha_j - \alpha_r)} = \frac{c_1(\alpha_j - \alpha_2)}{c_2(\alpha_j - \alpha_1)}$
- Assuming $\alpha_1 = 0, \alpha_2 = 1,\quad \frac{(R_1)_j}{(R_2)_j} = \frac{c_1(\alpha_j - 1)}{c_2(\alpha_j)}$

Recovering α

- Divide the non-zero entries of different rows of the RREF of $\mathbf{M}^{\intercal}$
- $\forall j \geq k+1$, $\frac{(R_1)_j}{(R_2)_j} = \frac{\beta_j p_{R_1}(\alpha_j)}{\beta_j p_{R_2}(\alpha_j)} = \frac{c_1 \prod_{r \in \{1,\ldots,k\} \setminus \{1\}} (\alpha_j - \alpha_r)}{c_2 \prod_{r \in \{1,\ldots,k\} \setminus \{2\}} (\alpha_j - \alpha_r)} = \frac{c_1(\alpha_j - \alpha_2)}{c_2(\alpha_j - \alpha_1)}$
- Assuming $\alpha_1 = 0, \alpha_2 = 1$, $\frac{(R_1)_j}{(R_2)_j} = \frac{c_1(\alpha_j - 1)}{c_2(\alpha_j)}$
- Guess $\frac{c_1}{c_2}$ and we are left with a system of $n-k$ equations and unknowns

Recovering α

- Divide the non-zero entries of different rows of the RREF of $\mathbf{M}^{\intercal}$
- $\forall j \geq k+1, \ \frac{(R_1)_j}{(R_2)_j} = \frac{\beta_j p_{R_1}(\alpha_j)}{\beta_j p_{R_2}(\alpha_j)} = \frac{c_1 \prod_{r \in \{1,\dots,k\} \setminus \{1\}} (\alpha_j - \alpha_r)}{c_2 \prod_{r \in \{1,\dots,k\} \setminus \{2\}} (\alpha_j - \alpha_r)} = \frac{c_1(\alpha_j - \alpha_2)}{c_2(\alpha_j - \alpha_1)}$
- Assuming $\alpha_1 = 0, \alpha_2 = 1, \quad \frac{(R_1)_j}{(R_2)_j} = \frac{c_1(\alpha_j - 1)}{c_2(\alpha_j)}$
- Guess $\frac{c_1}{c_2}$ and we are left with a system of $n-k$ equations and unknowns
- Rearranging, $\quad \frac{c_2}{c_1} \frac{(R_1)_j}{(R_2)_j} = 1 - \frac{1}{\alpha_j}$ has a unique solution for α_j

Recovering α

- Divide the non-zero entries of different rows of the RREF of $\mathbf{M}^\intercal$
- $\forall j \geq k+1$, $\frac{(R_1)_j}{(R_2)_j} = \frac{\beta_j p_{R_1}(\alpha_j)}{\beta_j p_{R_2}(\alpha_j)} = \frac{c_1 \prod_{r \in \{1,\ldots,k\} \setminus \{1\}} (\alpha_j - \alpha_r)}{c_2 \prod_{r \in \{1,\ldots,k\} \setminus \{2\}} (\alpha_j - \alpha_r)} = \frac{c_1(\alpha_j - \alpha_2)}{c_2(\alpha_j - \alpha_1)}$
- Assuming $\alpha_1 = 0, \alpha_2 = 1$, $\frac{(R_1)_j}{(R_2)_j} = \frac{c_1(\alpha_j - 1)}{c_2(\alpha_j)}$
- Guess $\frac{c_1}{c_2}$ and we are left with a system of $n-k$ equations and unknowns
- Rearranging, $\frac{c_2}{c_1}\frac{(R_1)_j}{(R_2)_j} = 1 - \frac{1}{\alpha_j}$ has a unique solution for α_j
- We recover $\alpha_{k+1}, \ldots, \alpha_n$ in this way

Recovery of the Remaining Parameters

- We recover the remaining parameters in a similar manner

Recovery of the Remaining Parameters

- We recover the remaining parameters in a similar manner
- Recovering $\alpha_3, \ldots, \alpha_k$
  - $\forall i \in \{3, \ldots, k\}$, pick $j_1, j_2 \in \{k+1, \ldots, n\}$, find $\frac{(R_1)_{j_1}}{(R_i)_{j_1}}$ and $\frac{(R_1)_{j_2}}{(R_i)_{j_2}}$
  - Invert $\frac{(R_1)_{j_1}(R_i)_{j_2}}{(R_1)_{j_2}(R_i)_{j_1}} \frac{\alpha_{j_1}}{\alpha_{j_2}} = \frac{\alpha_{j_1} - \alpha_i}{\alpha_{j_2} - \alpha_i}$ for α_i

Recovery of the Remaining Parameters

- We recover the remaining parameters in a similar manner
- Recovering $\alpha_3, \ldots, \alpha_k$
  - $\forall i \in \{3, \ldots, k\}$, pick $j_1, j_2 \in \{k+1, \ldots, n\}$, find $\frac{(R_1)_{j_1}}{(R_i)_{j_1}}$ and $\frac{(R_1)_{j_2}}{(R_i)_{j_2}}$
  - Invert $\frac{(R_1)_{j_1}(R_i)_{j_2}}{(R_1)_{j_2}(R_i)_{j_1}} \frac{\alpha_{j_1}}{\alpha_{j_2}} = \frac{\alpha_{j_1} - \alpha_i}{\alpha_{j_2} - \alpha_i}$ for α_i
- Recovering $\beta_2, \ldots, \beta_k$
  - Divide diagonal entries of the RREF to get

$$\beta_j = \frac{c_1}{c_j} \frac{\prod_{r \in \{2, \ldots, k\}} (-\alpha_r)}{\prod_{r \in \{1, \ldots, k\} \setminus \{2\}} (\alpha_j - \alpha_r)}$$

Recovery of the Remaining Parameters

- We recover the remaining parameters in a similar manner
- Recovering $\alpha_3, \ldots, \alpha_k$
  - $\forall i \in \{3, \ldots, k\}$, pick $j_1, j_2 \in \{k+1, \ldots, n\}$, find $\frac{(R_1)_{j_1}}{(R_i)_{j_1}}$ and $\frac{(R_1)_{j_2}}{(R_i)_{j_2}}$
  - Invert $\frac{(R_1)_{j_1}(R_i)_{j_2}}{(R_1)_{j_2}(R_i)_{j_1}} \frac{\alpha_{j_1}}{\alpha_{j_2}} = \frac{\alpha_{j_1} - \alpha_i}{\alpha_{j_2} - \alpha_i}$ for α_i
- Recovering $\beta_2, \ldots, \beta_k$
  - Divide diagonal entries of the RREF to get

$$\beta_j = \frac{c_1}{c_j} \frac{\prod_{r \in \{2,\ldots,k\}}(-\alpha_r)}{\prod_{r \in \{1,\ldots,k\} \setminus \{2\}}(\alpha_j - \alpha_r)}$$

- Recovering $\beta_{k+1}, \ldots, \beta_n$
  - Pick $j \in \{k+1, \ldots, n\}$ and divide $(R_1)_1$ by $(R_1)_j$ to get

$$\beta_j = (R_1)_j \prod_{r \in \{2,\ldots,k\}} \frac{-\alpha_r}{\alpha_j - \alpha_r}$$

Complexity of the Sidelnikov-Shestakov Attack

Complexity broken down

Complexity of the Sidelnikov-Shestakov Attack

Complexity broken down

- Row-reducing $\mathbf{M}^\intercal$ is done in $\mathcal{O}(nk^2)$ operations

Complexity of the Sidelnikov-Shestakov Attack

Complexity broken down

- Row-reducing $\mathbf{M}^{\intercal}$ is done in $\mathcal{O}(nk^2)$ operations
- α is recovered in $\mathcal{O}(np^m)$ operations (with guessing $\frac{c_1}{c_2}$)

Complexity of the Sidelnikov-Shestakov Attack

Complexity broken down

- Row-reducing $\mathbf{M}^\intercal$ is done in $\mathcal{O}(nk^2)$ operations
- α is recovered in $\mathcal{O}(np^m)$ operations (with guessing $\frac{c_1}{c_2}$)
- β is recovered in $\mathcal{O}(nk)$ operations

Complexity of the Sidelnikov-Shestakov Attack

Complexity broken down

- Row-reducing $\mathbf{M}^{\intercal}$ is done in $\mathcal{O}(nk^2)$ operations
- α is recovered in $\mathcal{O}(np^m)$ operations (with guessing $\frac{c_1}{c_2}$)
- β is recovered in $\mathcal{O}(nk)$ operations

Lemma [S]

$\frac{c_1}{c_2}$ can be computed from $\mathbf{M}$ in $\mathcal{O}(1)$ operations. Furthermore, this means α can be recovered in $\mathcal{O}(n)$ operations.

Application of Sidelnikov-Shestakov to Goppa Codes

- The $(n, k_\Gamma \leq k)$ Goppa code $\Gamma(\alpha, \beta)$ is a subcode of $GRS_{n,k}(\alpha, \beta)$
  - $\Gamma(\alpha, \beta) = GRS_{n,k}(\alpha, \beta) \cap \mathbb{F}_p^n$

$$GRS_{n,k}(\alpha, \beta) = \{(\beta_1 f(\alpha_1), \ldots, \beta_n f(\alpha_n)) : f \in \mathbb{P}_{k-1}(\mathbb{F}_{p^m})\}$$

Application of Sidelnikov-Shestakov to Goppa Codes

- The $(n, k_\Gamma \leq k)$ Goppa code $\Gamma(\alpha, \beta)$ is a subcode of $GRS_{n,k}(\alpha, \beta)$
  - $\Gamma(\alpha, \beta) = GRS_{n,k}(\alpha, \beta) \cap \mathbb{F}_p^n$

$$\Gamma(\alpha, \beta) = \{(\beta_1 q(\alpha_1), \ldots, \beta_n q(\alpha_n)) : q \in \mathcal{P}\}$$

s.t. $\mathcal{P} \subseteq \mathbb{P}_{k-1}(\mathbb{F}_{p^m})$ is a subspace linear over $\mathbb{F}_p$ of dimension k_Γ

Application of Sidelnikov-Shestakov to Goppa Codes

- The $(n, k_\Gamma \leq k)$ Goppa code $\Gamma(\alpha, \beta)$ is a subcode of $GRS_{n,k}(\alpha, \beta)$
  - $\Gamma(\alpha, \beta) = GRS_{n,k}(\alpha, \beta) \cap \mathbb{F}_p^n$

$$\Gamma(\alpha, \beta) = \{(\beta_1 q(\alpha_1), \ldots, \beta_n q(\alpha_n)) : q \in \mathcal{P}\}$$

s.t. $\mathcal{P} \subseteq \mathbb{P}_{k-1}(\mathbb{F}_{p^m})$ is a subspace linear over $\mathbb{F}_p$ of dimension k_Γ

- Public matrix: $\mathbf{M} = \mathbf{G}_\Gamma \mathbf{S}$ s.t. $\mathbf{G}_\Gamma$ is a generator matrix for $\Gamma(\alpha, \beta)$

$$\mathbf{M}^\intercal \sim \left[\mathbf{I}_{k_\Gamma} | \mathbf{A}\right] = \begin{bmatrix} R_1 \\ \vdots \\ R_{k_\Gamma} \end{bmatrix} \quad \text{s.t. } R_i = (\beta_1 q_{R_i}(\alpha_1), \ldots, \beta_n q_{R_i}(\alpha_n))$$

Application of Sidelnikov-Shestakov to Goppa Codes

- The $(n, k_\Gamma \leq k)$ Goppa code $\Gamma(\alpha, \beta)$ is a subcode of $GRS_{n,k}(\alpha, \beta)$
  - $\Gamma(\alpha, \beta) = GRS_{n,k}(\alpha, \beta) \cap \mathbb{F}_p^n$

$$\Gamma(\alpha, \beta) = \{(\beta_1 q(\alpha_1), \ldots, \beta_n q(\alpha_n)) : q \in \mathcal{P}\}$$

s.t. $\mathcal{P} \subseteq \mathbb{P}_{k-1}(\mathbb{F}_{p^m})$ is a subspace linear over $\mathbb{F}_p$ of dimension k_Γ

- Public matrix: $\mathbf{M} = \mathbf{G}_\Gamma \mathbf{S}$ s.t. $\mathbf{G}_\Gamma$ is a generator matrix for $\Gamma(\alpha, \beta)$

$$\mathbf{M}^\intercal \sim \left[\mathbf{I}_{k_\Gamma} | \mathbf{A}\right] = \begin{bmatrix} R_1 \\ \vdots \\ R_{k_\Gamma} \end{bmatrix} \quad \text{s.t. } R_i = (\beta_1 q_{R_i}(\alpha_1), \ldots, \beta_n q_{R_i}(\alpha_n))$$

- $(R_i)_j = \begin{cases} 1, i = j \\ 0, i \neq j \end{cases} \quad \forall i, j \in \{1, \ldots, k_\Gamma\}$, which means

$$\prod_{r \in \{1,\ldots,k_\Gamma\} \setminus \{i\}} (x - \alpha_r) \mid q_{R_i}(\alpha_i)$$

Application of Sidelnikov-Shestakov to Goppa Codes

- Hence, $\exists \rho_i \in \mathbb{P}_{k-k_\Gamma}(\mathbb{F}_{p^m})$ and $q_{R_i}(x) = \rho_i(x) \prod_{r \in \{1,\ldots,k_\Gamma\} \setminus \{i\}} (x - \alpha_r)$

Application of Sidelnikov-Shestakov to Goppa Codes

- Hence, $\exists \rho_i \in \mathbb{P}_{k-k_\Gamma}(\mathbb{F}_{p^m})$ and $q_{R_i}(x) = \rho_i(x) \prod_{r \in \{1,\ldots,k_\Gamma\} \setminus \{i\}} (x - \alpha_r)$

$$\frac{(R_1)_j}{(R_2)_j} = \frac{\beta_j p_{R_1}(\alpha_j)}{\beta_j p_{R_2}(\alpha_j)} = \frac{c_1 \prod_{r \in \{1,\ldots,k\} \setminus \{1\}} (\alpha_j - \alpha_r)}{c_2 \prod_{r \in \{1,\ldots,k\} \setminus \{2\}} (\alpha_j - \alpha_r)} = \frac{c_1(\alpha_j - 1)}{c_2(\alpha_j)}$$

Application of Sidelnikov-Shestakov to Goppa Codes

- Hence, $\exists \rho_i \in \mathbb{P}_{k-k_\Gamma}(\mathbb{F}_{p^m})$ and $q_{R_i}(x) = \rho_i(x) \prod_{r\in\{1,\dots,k_\Gamma\}\setminus\{i\}}(x-\alpha_r)$

$$\frac{(R_1)_j}{(R_2)_j} = \frac{\beta_j q_{R_1}(\alpha_j)}{\beta_j q_{R_2}(\alpha_j)} = \frac{\rho_1(\alpha_j)\prod_{r\in\{1,\dots,k\}\setminus\{1\}}(\alpha_j-\alpha_r)}{\rho_2(\alpha_j)\prod_{r\in\{1,\dots,k\}\setminus\{2\}}(\alpha_j-\alpha_r)} = \frac{\rho_1(\alpha_j)(\alpha_j-1)}{\rho_2(\alpha_j)(\alpha_j)}$$

Application of Sidelnikov-Shestakov to Goppa Codes

- Hence, $\exists \rho_i \in \mathbb{P}_{k-k_\Gamma}(\mathbb{F}_{p^m})$ and $q_{R_i}(x) = \rho_i(x) \prod_{r \in \{1,\ldots,k_\Gamma\} \setminus \{i\}} (x - \alpha_r)$

$$\frac{(R_1)_j}{(R_2)_j} = \frac{\beta_j q_{R_1}(\alpha_j)}{\beta_j q_{R_2}(\alpha_j)} = \frac{\rho_1(\alpha_j) \prod_{r \in \{1,\ldots,k\} \setminus \{1\}} (\alpha_j - \alpha_r)}{\rho_2(\alpha_j) \prod_{r \in \{1,\ldots,k\} \setminus \{2\}} (\alpha_j - \alpha_r)} = \frac{\rho_1(\alpha_j)(\alpha_j - 1)}{\rho_2(\alpha_j)(\alpha_j)}$$

- Solving this amounts to inverting a degree-$(k - k_\Gamma + 1)$ rational function, which is impossible to do if the degree is greater than 1

Vulnerability of Full-Rank Goppa Codes

- Hard to attack $\Gamma(\alpha, \beta)$ if $k - k_\Gamma + 1 > 1$. What if $k = k_\Gamma$?

Vulnerability of Full-Rank Goppa Codes

- Hard to attack $\Gamma(\alpha, \beta)$ if $k - k_\Gamma + 1 > 1$. What if $k = k_\Gamma$?

Lemma

Let D be a code in $\mathbb{F}_p^n$. A basis for D is also a basis for $\text{span}_{\mathbb{F}_{p^m}}(D)$.

Vulnerability of Full-Rank Goppa Codes

- Hard to attack $\Gamma(\alpha, \beta)$ if $k - k_\Gamma + 1 > 1$. What if $k = k_\Gamma$?

Lemma

Let D be a code in $\mathbb{F}_p^n$. A basis for D is also a basis for $\text{span}_{\mathbb{F}_{p^m}}(D)$.

- This means that if $\Gamma(\alpha, \beta)$ is of maximal dimension, a basis for $\Gamma(\alpha, \beta)$ is a basis for $GRS_{n,k}(\alpha, \beta)$

Vulnerability of Full-Rank Goppa Codes

- Hard to attack $\Gamma(\alpha, \beta)$ if $k - k_\Gamma + 1 > 1$. What if $k = k_\Gamma$?

Lemma

Let D be a code in $\mathbb{F}_p^n$. A basis for D is also a basis for $\text{span}_{\mathbb{F}_{p^m}}(D)$.

- This means that if $\Gamma(\alpha, \beta)$ is of maximal dimension, a basis for $\Gamma(\alpha, \beta)$ is a basis for $GRS_{n,k}(\alpha, \beta)$
- A generator matrix for $\Gamma(\alpha, \beta)$ will also be a generator matrix for $GRS_{n,k}(\alpha, \beta)$

Vulnerability of Full-Rank Goppa Codes

- Hard to attack $\Gamma(\alpha, \beta)$ if $k - k_\Gamma + 1 > 1$. What if $k = k_\Gamma$?

Lemma

Let D be a code in $\mathbb{F}_p^n$. A basis for D is also a basis for $\text{span}_{\mathbb{F}_{p^m}}(D)$.

- This means that if $\Gamma(\alpha, \beta)$ is of maximal dimension, a basis for $\Gamma(\alpha, \beta)$ is a basis for $GRS_{n,k}(\alpha, \beta)$
- A generator matrix for $\Gamma(\alpha, \beta)$ will also be a generator matrix for $GRS_{n,k}(\alpha, \beta)$
- The S-S attack applies exactly the same to these codes

Conclusion and Future Work

- We presented the McEliece PKC as well as an efficient attack that renders it insecure when GRS codes are used to form the cryptographic primitive

Conclusion and Future Work

- We presented the McEliece PKC as well as an efficient attack that renders it insecure when GRS codes are used to form the cryptographic primitive
- We also outlined that for certain Goppa codes, the McEliece scheme based on these codes will be insecure

Conclusion and Future Work

- We presented the McEliece PKC as well as an efficient attack that renders it insecure when GRS codes are used to form the cryptographic primitive
- We also outlined that for certain Goppa codes, the McEliece scheme based on these codes will be insecure
- Next steps: see if we can exploit the relationship between Goppa codes and GRS codes to find other special cases that are vulnerable to a S-S-like attack

Acknowledgements

- Thank you to my supervisor, Dr. Monica Nevins, for overseeing my work this summer
- Research funded by an NSERC USRA