# Reducing the Key Size of McEliece Cryptosystem from Goppa Codes via Permutations

Zhe Li[1], Chaoping Xing[1], and Sze Ling Yeo[2]

[1]School of Physical and Mathematical Sciences, Nanyang Technological University
[2]Institute for Infocomm Research (I2R), Singapore

October 23, 2018

**Abstract**

In this paper, we propose a new general construction to reduce the public key size of McEliece-based schemes based on Goppa codes. In particular, we generalize the ideas of automorphism-induced Goppa codes by considering nontrivial subsets of automorphism groups to construct Goppa codes with a nice block structure. By considering additive and multiplicative automorphism subgroups, we provide explicit constructions to demonstrate our technique. In addition, we show that our technique can be applied to automorphism-induced Goppa codes to further reduce their key sizes.

## 1 Introduction

Since the introduction of public-key cryptography in the 1970's, all the public-key cryptosystems that have been proposed fall into two broad categories, namely, the classical schemes and the quantum-resistant schemes. The former category comprises most of the schemes used today. They are primarily built up from computational number theoretic problems including integer factoring problem and discrete logarithm problem in different groups. While such schemes are generally believed to be secure against classical computers, their security has been shown to be vulnerable to quantum algorithms such as the Shor algorithm [33].

On the other hand, the class of quantum-resistant schemes, as the name implies, includes schemes whose security is not threatened by existing quantum algorithms. Such schemes are further classified by their underlying mathematical problems into various classes including code-based cryptosystems, lattice-based cryptosystems, hash-based cryptosystems, multivariate cryptosystems or schemes based on elliptic curve isogenies [7]. Among these classes, the code-based cryptography is one of the oldest, dating back to the work of McEliece in 1978 [26].

Essentially, code-based cryptography refers to the class of schemes whose security relies on hard problems in coding theory, such as the general decoding problem. Concretely, the classical code-based encryption scheme works as follows. Let $M$ be a generator matrix of an $[n, k, 2t+1]$-linear code $C$ with a fast decoder. Let $S$ be a $k \times k$ invertible matrix and let $P$ be an $n \times n$ permutation matrix. Let $M' = SMP$. Then, the public key is $M'$ while the secret key comprises the matrices $S, M$ and $P$. To encrypt a message $\mathbf{x}$, one chooses an error $\mathbf{e}$ with Hamming weight $t$ and computes $\mathbf{c} = \mathbf{x}M' + \mathbf{e}$. To recover $\mathbf{x}$, one first computes $\mathbf{c}' = \mathbf{c}P^{-1}$. Use the decoder with respect to $M$ to decode $\mathbf{c}'$ to obtain $\mathbf{c}'' \in C$. Since $\mathbf{c}'' = (\mathbf{x}S)M$, one can compute $\mathbf{x}S$ and hence recover $\mathbf{x}$ using $S^{-1}$.

An alternative but equivalent code-based encryption scheme was proposed by Niederreiter in [30] in which the parity-check matrix instead of the generator matrix was used. In this scheme, a message is converted into a vector of Hamming weight $t$ and encryption is performed by multiplying this vector with the parity check matrix. Once again, decryption is accomplished via a fast decoder while the security is based on the difficulty of the syndrome decoding problem.

The above descriptions only outline the essential ideas of code-based schemes. Variants of these schemes have been proposed to achieve different forms of security such as the CCA2 security [19]. The main advantage of code-based cryptosystems lies in its efficient operations leading to very efficient encryption and decryption. As such, it continues to draw much interest to design new code-based cryptographic primitives. For instance, in the recent NIST submissions, two proposals on code-based key encapsulation mechanisms were proposed [2, 1]. Essentially, any code $C$ used in the schemes must satisfy the main property, namely, it has an efficient decoder using a particular matrix $M$ but multiplying this matrix with a random matrix transforms the corresponding code into a random code. In addition, the code used should not exhibit any structural weakness to recover the private key. The first family of codes suggested by McEliece in [26] is the family of Goppa codes. Subsequently, other families of codes are proposed, including algebraic geometric codes [18], Reed-Muller codes [34], Reed Solomon codes [30], and more recently, MDPC codes [29]. While all these codes have an efficient decoding algorithm, the structures exhibited by some of the codes make them vulnerable to other attacks. Ideally, one hopes to construct public keys with reasonably short lengths such that the underlying structures are properly concealed. Quasi-cyclic MDPC codes are more recent designs that seem promising to achieve these two goals simultaneously. On the other hand, codes such as Reed-Muller codes and Reed Solomon codes, while providing short public keys, have all been broken [27, 22]. In terms of security, Goppa codes seem to be strongest as they have withstood structural attacks since they were first proposed by McEliece. However, they suffer from the disadvantage of having large public key sizes.

As such, it is an interesting problem to consider sub-classes of Goppa codes to better balance the security and public key size requirements. One common approach adopted is to employ quasi-cyclic or quasi-dyadic Goppa codes [28, 6] or more generally, codes with a nontrivial automorphism group [5]. Indeed, in [13], the authors showed that existing quasi-cyclic and quasi-dyadic constructions are induced from nontrivial automorphism subgroups of the rational function field. The main advantage to use a code with a nontrivial automorphism group $\mathcal{G}$ is that there exists a subset of basis codewords such that this subset together with their permutations induced by $\mathcal{G}$ form the whole basis of the code. As such, this allows one to use the subset of codewords as the public key instead of the entire basis, thereby reducing the size of the public key. Nonetheless, the size reduction leads to a trade-off with respect to its resistance to structural attacks. In particular, it was shown in [14, 13, 3] that the public/private key pair of such codes is equivalent to a public/private key pair of another code with smaller parameters. Algebraic cryptanalysis can then be performed on the corresponding codes with smaller parameters and successful attacks following this approach were carried out in [15, 16]. As such, one must select the parameters carefully in order to balance the trade-off and to achieve the desired security level [2].

In this paper, we generalize the approach of automorphism-induced code constructions to seek for Goppa codes with compact public key sizes. Instead of finding codes with nontrivial automorphism groups, we will construct Goppa codes that contain subcodes and their permutations. Specifically, we solve the following problem.

**Problem 1:** Construct Goppa codes $C$ that contain a subset $S$ of linearly independent codewords satisfying the following properties:

- For each $\mathbf{c} \in S$, there exists a set of permutations $\mathcal{P}_{\mathbf{c}}$ such that $\sigma(\mathbf{c}) \in C$ for all $\sigma \in \mathcal{P}_{\mathbf{c}}$;
- $B = \bigcup_{\mathbf{c}\in S} \left( \bigcup_{\sigma\in\mathcal{P}_{\mathbf{c}}} \sigma(\mathbf{c}) \right)$ is a basis of $C$.

By finding such codes $C$, one can then use the set $S$ as its public key. Observe that the automorphism-induced codes including the quasi-cyclic and quasi-dyadic codes are examples of the codes we seek. In these cases, all the $\mathcal{P}_{\mathbf{c}}$ are identical and equal to the automorphism group of the code.

This paper seeks to provide other classes of Goppa codes that satisfy the conditions of Problem 1. In our constructions, the sets $\mathcal{P}_{\mathbf{c}}$ are no longer automorphism groups of the codes. Instead, we consider only subsets of permutations of each codeword. In this way, our codes become resistant to the algebraic attacks of [14, 13, 3] as the folding operation does not lead to an invariant subcode (see 2.3 for the relevant definitions). Moreover, our construction is generic in the sense that it can be applied to reduce the key size of existing code-based schemes from Goppa codes.

In this paper, we first provide conditions for permutations of codewords to lie in the code. We then construct codes with a partial quasi-cyclic and quasi-dyadic structure. We demonstrate that the codes we construct are not vulnerable to the algebraic attacks of [15, 16, 13]. Finally, we apply our technique to automorphism-induced Goppa codes to obtain compact and secure Goppa codes. We show with concrete examples that our technique can reduce the public key sizes in BigQuake [2] by at least half.

The subsequent sections are structured as follows. First, we recall Goppa codes and their permutations. We also review automorphism-induced Goppa codes and the associated algebraic attacks. We then present our new construction that exploits subsets of automorphism groups to yield codes with a nice permutation structure. In Section 4, we provide a security discussion of our construction. Finally, we present an algorithm that combines our technique with existing quasi-cyclic constructions to obtain Goppa codes with reduced public key sizes.

# 2 Preliminaries

In this paper, we always assume that $q = 2^m$ for an integer $m \geq 2$. Let $\mathbb{F}_q$ denote the finite field with $q$ elements.

## 2.1 Goppa codes

We first review the definition of Goppa codes [26]. In the following, we present a construction that is relevant for this work.

**Definition 2.1** (Goppa code)**.** *Let $t$ and $n$ be positive integers with $t < n \leq q$. Let $g(x) \in \mathbb{F}_q[x]$ be a polynomial of degree $t$ and let $L = \{\gamma_1, \cdots, \gamma_n\}$ be an ordered set containing $n$ distinct elements of $\mathbb{F}_q$ such that $g(\gamma_i) \neq 0$ for $1 \leq i \leq n$. The Goppa Code $\Gamma(L, g)$ is defined as*

$$\Gamma(L, g) = \left\{ \mathbf{c} = (c_1, \cdots, c_n) \in \mathbb{F}_2^n : \sum_{i=1}^{n} \frac{c_i}{x - \gamma_i} \equiv 0 \pmod{g(x)} \right\}.$$

*The polynomial $g(x)$ is called the Goppa polynomial. When $g(x)$ is irreducible, $\Gamma(L, g)$ is called an irreducible Goppa code. In this paper, we call the ordered set $L$ the Goppa support.*

Some of the main properties of Goppa codes are summarized below.

**Remark 2.2.** (i) The Goppa code $\Gamma(L, g)$ is a subfield subcode of generalized Reed-Solomon codes. In fact, the class of Goppa codes is a special case of alternant codes.

(ii) The Goppa code $\Gamma(L, g)$ is a binary linear code and has dimension at least $n - mt$, and minimum distance at least $t + 1$.

(iii) If $g(x)$ has no multiple roots, then $\Gamma(L, g)$ has minimum distance at least $2t + 1$. In particular, an irreducible Goppa code $\Gamma(L, g)$ has minimum distance at least $2t + 1$.

Given a Goppa support $L$ and a Goppa polynomial $g(x)$, a parity-check matrix of $\Gamma(L, g)$ is given by $H = VD$, where

$$V = \begin{pmatrix} 1 & 1 & \dots & 1 \\ \gamma_1 & \gamma_2 & \dots & \gamma_n \\ \gamma_1^2 & \gamma_2^2 & \dots & \gamma_n^2 \\ \dots & \dots & \dots & \dots \\ \gamma_1^{t-1} & \gamma_2^{t-1} & \dots & \gamma_n^{t-1} \end{pmatrix} \tag{1}$$

and

$$D = \mathrm{diag}(1/g(\gamma_1), 1/g(\gamma_2), \dots, 1/g(\gamma_n)). \tag{2}$$

One of the nice properties of Goppa codes lies in its efficient decodability. In particular, there exists a polynomial-time decoding algorithm for Goppa codes [23] that can decode up to $t$ errors.

Fix $a \in \mathbb{F}_q^*, b \in \mathbb{F}_q, c \in \mathbb{F}_q^*$. For a Goppa support $L \subset \mathbb{F}_q$, let $L' = \{a^{-1}x - b : x \in L\}$. Then, for any $g \in \mathbb{F}_q[x]$, it is easy to check that the Goppa codes $\Gamma(L, g)$ and $\Gamma(L', cg(ax + b))$ are equal. Consequently, for any Goppa code, one can find at least $q(q-1)$ different Goppa supports and monic Goppa polynomials that define the given code.

The following result follows directly from the definition of Goppa codes [23].

**Lemma 2.3.** *Fix $L \subset \mathbb{F}_q$. For any two polynomials $g_1(x)$ and $g_2(x)$, one has:*

$$\Gamma(L, g_1(x)) \cap \Gamma(L, g_2(x)) = \Gamma(L, \mathrm{lcm}(g_1(x), g_2(x))).$$

## 2.2 Permutations of Goppa codes

In this subsection, we take a look at permutations of Goppa codes.

For a positive integer $n$, let $\mathcal{S}_n$ denote the symmetric group on $n$ symbols. We let $\mathcal{S}_n$ act on the vector space $\mathbb{F}_2^n$ via

$$\sigma(\mathbf{c}) = \sigma(c_1, \cdots, c_n) = (c_{\sigma(1)}, \cdots, c_{\sigma(n)})$$

for $\sigma \in \mathcal{S}_n$ and $\mathbf{c} = (c_1, \cdots, c_n) \in \mathbb{F}_2^n$. Then, $\sigma(C) = \{\sigma(\mathbf{c}) : \mathbf{c} \in C\}$ is also a code with the same parameters as the code $C$.

**Definition 2.4.** *The automorphism group of a code $C \subseteq \mathbb{F}_2^n$ is defined as*

$$\mathrm{Aut}(C) = \{\sigma \in \mathcal{S}_n : \sigma(\mathbf{c}) \in C \text{ for all } \mathbf{c} \in C\}.$$

*In particular, $\sigma \in \mathrm{Aut}(C)$ if and only if $\sigma(C) = C$.*

We next show that there exist permutations $\sigma \in \mathcal{S}_n$ such that $\sigma(C)$ is itself a Goppa code.

Let $F = \mathbb{F}_q(x)$ denote the rational function field over $\mathbb{F}_q$. We denote by $\mathrm{Aut}(F/\mathbb{F}_q)$ the automorphism group of $F$ over $\mathbb{F}_q$, i.e.,

$$\mathrm{Aut}(F/\mathbb{F}_q) = \{\sigma : F \to F : \ \sigma \text{ is an } \mathbb{F}_q\text{-automorphism of } F\}. \tag{3}$$

It is clear that an automorphism $\sigma \in \mathrm{Aut}(F/\mathbb{F}_q)$ is uniquely determined by $\sigma(x)$. It is well known that every automorphism $\sigma \in \mathrm{Aut}(F/\mathbb{F}_q)$ is given by

$$\sigma(x) = \frac{ax + b}{cx + d} \tag{4}$$

for some constants $a, b, c, d \in \mathbb{F}_q$ with $ad - bc \neq 0$ (see [10]). Denote by $\mathrm{GL}_2(q)$ the general linear group of $2 \times 2$ invertible matrices over $\mathbb{F}_q$. Thus, every matrix $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathrm{GL}_2(q)$ induces an automorphism of $F$ given by (4). Two matrices of $\mathrm{GL}_2(q)$ induce the same automorphism of

$F$ if and only if they belong to the same coset of $Z(\mathrm{GL}_2(q))$, where $Z(\mathrm{GL}_2(q))$ stands for the center $\{aI_2 : a \in \mathbb{F}_q^*\}$ of $\mathrm{GL}_2(q)$. This implies that $\mathrm{Aut}(F/\mathbb{F}_q)$ is isomorphic to the projective linear group $\mathrm{PGL}_2(q) := \mathrm{GL}_2(q)/Z(\mathrm{GL}_2(q))$. Thus, we can identify $\mathrm{Aut}(F/\mathbb{F}_q)$ with $\mathrm{PGL}_2(q)$.

Consider the subgroup of $\mathrm{PGL}_2(q)$,

$$\mathrm{AGL}_2(q) := \left\{ \begin{pmatrix} a & b \\ 0 & 1 \end{pmatrix} : a \in \mathbb{F}_q^*, b \in \mathbb{F}_q \right\}. \tag{5}$$

$\mathrm{AGL}_2(q)$ is called the affine linear group. Every element $A = \begin{pmatrix} a & b \\ 0 & 1 \end{pmatrix} \in \mathrm{AGL}_2(q)$ defines an affine automorphism $\sigma_{a,b}(x) = ax + b$.

The following lemma is straightforward to verify.

**Lemma 2.5.** *Let $b \in \mathbb{F}_q$. Then,*

$$\mathrm{ord}(\sigma_{a,b}) = \begin{cases} \mathrm{ord}(a) & \text{if } a \neq 1, \\ 2 & \text{otherwise} \end{cases}$$

We recall the definition of a group action on a set and some of its basic properties.

**Definition 2.6** ([21])**.** *A group $\mathcal{G}$ action on a set $S$ is a mapping from $\mathcal{G}$ to $\mathcal{G} \times S$ satisfying the following properties:*

*(i)* $g_1 \cdot (g_2 \cdot s) = (g_1 g_2) \cdot s$ *for all* $g_1, g_2 \in G, s \in S$,

*(ii)* $1 \cdot s = s$ *for all* $s \in S$.

**Definition 2.7.** *Let $\mathcal{G}$ be a group acting on a set $S$. The equivalence class $\{g \cdot s : g \in G\}$ is called the orbit of $\mathcal{G}$ containing $s$.*

**Remark 2.8.** *(i) The orbits of two different elements in $S$ are either equal or disjoint. In particular, $S$ is partitioned into a disjoint union of orbits.*

*(ii) For $s \in S$, let $\mathcal{G}_s$ denote the stabilizer subgroup of $\mathcal{G}$ that fixes $s$ under the group action, that is $\mathcal{G}_s = \{g : g \in \mathcal{G} | g \cdot s = s\}$. Then, the orbit containing $s$ has $|\mathcal{G}|/|\mathcal{G}_s|$ different elements.*

For any element $g \in \mathcal{G}$, the action of $g$ induces a permutation on the set $S$. In fact, $g$ restricted to any orbit $\mathcal{O}$ is a permutation of $\mathcal{O}$. In particular, let $L = \{\gamma_1, \ldots, \gamma_n\}$ be a disjoint union of $\mathcal{G}$-orbits. Then, for any $\sigma \in \mathcal{G}$ and $1 \leq i \leq n$, there exists $j$ such that $\sigma \cdot \gamma_i = \gamma_j$. In this case, we will simply denote the corresponding permutation in $\mathcal{S}_n$ as $\sigma(i) = j$.

**Definition 2.9.** *Let $L$ be a subset of $\mathbb{F}_q$ containing $n$ distinct elements. For any $\sigma \in \mathcal{S}_n$, we say that $L$ is invariant under $\sigma$ if $\sigma(L) = L$ as sets. If $L$ is invariant under every $\sigma$ in a subgroup $\mathcal{G}$ of $\mathcal{S}_n$, we say that $L$ is $\mathcal{G}$-invariant. In particular, if $L$ is a disjoint union of $\mathcal{G}$-orbits, $L$ is $\mathcal{G}$-invariant.*

**Remark 2.10.** *Consider a group action of $\mathrm{AGL}_2(q)$ on $\mathbb{F}_q$ defined by $\sigma_{a,b} \cdot \gamma = a^{-1}(\gamma - b)$. One easily checks that this is a group action. Then, one has $\sigma_{a,b}(x - \gamma) = a(x - \sigma_{a,b} \cdot \gamma)$.*

**Proposition 2.11.** *Let $\mathcal{G}$ be a subgroup of $\mathrm{AGL}_2(q)$ and let $L \subset \mathbb{F}_q$ be a disjoint union of $\mathcal{G}$-orbits. Let $g \in \mathbb{F}_q[x]$. For any $\sigma \in G$, one has*

$$\sigma(\Gamma(L, g)) = \Gamma(L, g(\sigma^{-1}(x))).$$

*Proof.* Write $L = \{\gamma_1, \ldots, \gamma_n\}$. Here, it suffices to show that $\sigma(\Gamma(L, g)) \subset \Gamma(L, g(\sigma^{-1}(x)))$. Let $\mathbf{c} = (c_1, c_2, \ldots, c_n) \in \Gamma(L, g)$. By definition, we have:

$$\sum_{i=1}^{n} \frac{c_i}{x - \gamma_i} \equiv 0 \mod g(x).$$

Applying $\sigma^{-1}$ to both sides of the equivalence relation Remark 2.10 yields

$$\sum_{i=1}^{n} \frac{c_i}{x - \sigma^{-1} \cdot \gamma_i} \equiv 0 \mod g(\sigma^{-1}(x)).$$

Since $L$ is a union of $\mathcal{G}$-orbits, for each $i = 1, \ldots, n$, there exists some $j$ such that $\gamma_j \in L$ and $\sigma(\gamma_j) = i$. Rearranging the equation then gives:

$$\sum_{i=1}^{n} \frac{c_{\sigma(i)}}{x - \gamma_i} \equiv 0 \mod g(\sigma^{-1}(x)),$$

and this proves our desired result. □

## 2.3 Automorphism-induced Goppa codes

In Proposition 2.11, suppose further that there exists some $\alpha \in \mathbb{F}_q^*$ such that $g(\sigma^{-1}) = \alpha g$. Then, it follows that $\sigma(\Gamma(L, g)) = \Gamma(L, g)$, that is, $\sigma$ is in the automorphism group of $\Gamma(L, g)$. In other words, for any $\mathbf{c} \in \Gamma(L, g)$, the permuted codeword $\sigma^i(\mathbf{c})$ lies in the code as well for $i = 1, 2, \ldots$. This property enables one to construct generator matrices of the code exhibiting a nice structure, thereby reducing the public key size of schemes built from these codes.

As such, a common approach to construct McEliece-based schemes with shorter keys is to employ Goppa codes induced by subgroups of $\mathrm{AGL}_2(q)$ (or more generally, subgroups of projective semi-linear groups of $\mathrm{Aut}(F/\mathbb{F}_q)$). For instance, in [5], the author described alternant codes that can be constructed from prescribed automorphism subgroups of which, Goppa codes form a special case. These codes resulted in quasi-cyclic, quasi-dyadic and monoidic alternant codes which were subsequently employed to construct McEliece-based schemes with compact keys [28, 6, 2, 1].

We review the main ideas of this approach below. Let $\mathcal{G}$ be a subgroup of $\mathrm{AGL}_2(q)$ of size $r$. Suppose that there are at least $s$ $\mathcal{G}$-orbits of size $r$. Let $L$ be a union of $s$ of these orbits and let $n = rs$. Let $t = dr$ for some positive integer $d$ and let $g(x)$ be a polynomial of degree $t$ that is invariant under the $\mathcal{G}$-action, i.e., for every $\sigma \in \mathcal{G}$, there exists $\alpha \in \mathbb{F}_q^*$ such that $g(\sigma(x)) = \alpha g(x)$. From Proposition 2.11, it follows that for every $\sigma \in \mathcal{G}$, $\sigma(\Gamma(L, g)) = \Gamma(L, g)$. Hence, $\Gamma(L, g)$ is a Goppa code with $\mathcal{G}$ as a subgroup of $\mathrm{Aut}(\Gamma(L, g))$.

One common way to construct the $\mathcal{G}$-invariant polynomial $g(x)$ is as follows. Pick an irreducible polynomial $f(x)$ of degree $d$. Define

$$g(x) = f(\prod_{\sigma \in \mathcal{G}} \sigma(x)).$$

Then, $g(x)$ is invariant under any $\sigma \in \mathcal{G}$. In [5, 13], the authors classified polynomials invariant under some particular subgroups of $\mathrm{AGL}_2(q)$.

Denote the orbits in $L$ by $\mathcal{O}_i$, $i = 1, 2, \ldots, s$. For each orbit $\mathcal{O}_i$, fix a representative $\beta_i$. Fix an order in $\mathcal{G}$, that is $\mathcal{G} = \{\sigma_1, \sigma_2, \ldots, \sigma_r\}$. With this order, write $\mathcal{O}_i = \{\sigma_1 \cdot \beta_i, \sigma_2 \cdot \beta_i, \ldots, \sigma_r \cdot \beta_i\}$. It is clear that for any $\sigma \in \mathcal{G}$, $\sigma$ induces the same permutation for each orbit. We denote this permutation by $\tilde{\sigma}$.

Order the elements in $L$ as $L = \mathcal{O}_1 || \mathcal{O}_2 || \ldots || \mathcal{O}_s = \{(\sigma_1 \cdot \beta_1 \ldots \sigma_r \cdot \beta_1), \ldots, (\sigma_1 \cdot \beta_s \ldots \sigma_r \cdot \beta_s)\}$. Let $\mathbf{c} = (c_1, \ldots, c_n) \in \Gamma(L, g)$. We partition $\mathbf{c}$ as $\mathbf{c} = (\mathbf{c}_{\mathcal{O}_1} || \ldots || \mathbf{c}_{\mathcal{O}_s})$, where each $\mathbf{c}_{\mathcal{O}_i}$ corresponds to the entries indexed by elements in $\mathcal{O}_i$ with $i \in \{1, \ldots, s\}$. Then for each $\sigma \in \mathcal{G}$, $\mathbf{c}_\sigma = (\mathbf{c}_{\tilde{\sigma}(\mathcal{O}_1)} || \ldots || \mathbf{c}_{\tilde{\sigma}(\mathcal{O}_s)}) \in \Gamma(L, g)$.

Next, we seek to construct a generator matrix for $\Gamma(L, g)$ having a nice form. Let $k$ denote the dimension of $\Gamma(L, g)$. Suppose further that $k = n - mt$. Let $k_0 = k/r = (s - md)$.

Consider a generator matrix $M$ of $\Gamma(L, g)$. Without any loss of generality, suppose that $M$ can be put in systematic form (for otherwise, choose other $\beta_i$'s), that is, $M = (I, B)$, where $B$ is

a $k \times (n-k)$ matrix. Label the rows of $M$ by $M_i$. Define $P(L,g) := \{M_1, M_{r+1}, \ldots, M_{(k_0-1)r+1}\}$. For each $\mathbf{c} \in P(L,g)$, form the matrix

$$\mathbf{c}(\mathcal{G}) = \begin{pmatrix} \sigma_1(\mathbf{c}) \\ \vdots \\ \sigma_r(\mathbf{c}) \end{pmatrix}.$$

Finally, form the generator matrix

$$M(L,g) = \begin{pmatrix} \mathbf{c}_1(\mathcal{G}) \\ \vdots \\ \mathbf{c}_{k_0}(\mathcal{G}) \end{pmatrix},$$

for $\mathbf{c}_i = M_{(i-1)r+1}$. Note that since $M$ is in systematic form, each $\mathbf{c}_i = M_{(i-1)r+1}$ is such that the $i$-th block is of the form $(1, 0, \ldots, 0)$ while the other blocks in the first $k_0$ blocks are $(0, 0, \ldots, 0)$.

**Theorem 2.12.** *The matrix $M(L,g)$ constructed above is a generator matrix of $\Gamma(L,g)$.*

*Proof.* First, it is clear that for each $\mathbf{c} \in P(L,g)$ and for each $\sigma \in \mathcal{G}$, $\sigma(\mathbf{c}) \in \Gamma(L,g)$. It thus remains to show that $M(L,g)$ has rank $k$. Observe that for each $\mathbf{c} \in P(L,g)$, $\mathbf{c}(\mathcal{G})$ is an $r \times n$ matrix. Further, $\mathbf{c}(\mathcal{G})$ can be viewed as a concatenation of $s$ $r \times r$ square matrices where each square matrix is in fact $\mathbf{c}_{\mathcal{O}}(\mathcal{G})$ for an orbit $\mathcal{O}$ contained in $L$. Consider the first $k_0$ blocks of $\mathbf{c}_i(\mathcal{G})$. By our choice of $\mathbf{c}_i$, we see that the $i$-th block of $\mathbf{c}_i(\mathcal{G})$ is a permutation matrix while all other blocks are 0. Hence, the first $k$ columns of $M(L,g)$ have the form

$$\begin{pmatrix} A & 0 & \ldots & 0 \\ 0 & A & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & A \end{pmatrix},$$

where $A = \mathbf{x}(\mathcal{G})$ and $\mathbf{x} = (1, 0, \ldots, 0)$. Since the rank of $A$ is $r$, the desired result follows. □

Two particular subgroups of $\mathrm{AGL}_2(q)$ of interest are the multiplicative cyclic subgroup of order $r|(q-1)$ and the additive subgroup of order $2^v$ for $1 \leq v \leq m$. Concretely, the former comprises automorphisms of the form $\sigma_a(x) = ax$ for $a \in \mathbb{F}_q^*$ of order $r$ and gives rise to quasi-cyclic Goppa codes [6, 2]. In this case, the orbits in the Goppa support $L$ are cosets of the subgroup of $\mathbb{F}_q^*$ of order $r$ and the Goppa polynomial is of the form $g(x) = f(x^r)$ for some irreducible polynomial $f(x) \in \mathbb{F}_q[x]$.

On the other hand, the additive subgroup construction results in quasi-dyadic Goppa codes [28, 1]. Let $V$ be any $\mathbb{F}_2$-subspace of $\mathbb{F}_q$ with dimension $v$. The additive automorphism group consists of automorphisms of the form $\sigma_b(x) = x + b$ for all $b \in V$. Thus, the orbits in $L$ are all cosets of $V$ in $\mathbb{F}_q$, each of size $2^v$ and the Goppa polynomial is of the form $f(L(x))$, where $f(x)$ is an irreducible polynomial and $L(x) = \prod_{b \in V}(x + b)$.

Observe from Theorem 2.12 that it suffices to provide the set $P(L,g)$ in order to construct the generator matrix of $\Gamma(L,g)$, that is, these codes are examples of the codes we seek in Problem 1. In other words, when using this code in McEliece-based schemes, the public key size can be reduced. However, matrices with such nice structure also reduce the number of unknowns to mount algebraic attacks, thereby weakening the security with respect to structural attacks. One way to perform algebraic cryptanalysis is to consider the parity-check equations using the parity-check matrix $H = VD$ where $V$ and $D$ are defined in Equations 1 and 2, respectively, and treating the $\gamma_i$'s and $1/g(\gamma_i)$'s as unknowns. One then tries to solve the system of equations

using algebraic solving tools such as Gröbner basis algorithms. In the case of automorphism-induced Goppa codes, the unknowns for the Goppa support are reduced to the elements of the subgroup as well as the orbit representatives. At the same time, since $g(a)$ is identical for all $a$ in the same orbit of $L$, the number of unknowns of the diagonal matrix in Equation 2 is reduced to the number of orbits. Essentially, it was shown in [13, 3] that one can construct Goppa codes with much smaller parameters from these automorphism-induced codes, namely, via folded codes and invariant codes defined as follows.

**Definition 2.13.** *Let $C = \Gamma(L, g)$ be a Goppa code induced by an automorphism subgroup $\mathcal{G}$ so that $L$ is a disjoint union of $\mathcal{G}$-orbits. For $\mathbf{c} \in C$, define* $\mathrm{Punc}(\mathbf{c})$ *as the codeword obtained from $\mathbf{c}$ by puncturing on a set of representatives for the orbits in $L$.*

- *The folded code $\phi(C)$ is the subcode of $C$ defined as:*

$$\phi(C) = \{\mathrm{Punc}(\sum_{\sigma \in \mathcal{G}} \sigma(\mathbf{c})) : \mathbf{c} \in C\}.$$

- *The invariant code $C^{\mathcal{G}}$ is the subcode of $C$ defined by:*

$$C^{\mathcal{G}} = \{\mathrm{Punc}(\mathbf{c}) : \mathbf{c} \in C | \sigma(\mathbf{c}) = \mathbf{c} \text{ for all } \sigma \in \mathcal{G}\}.$$

Clearly, the folded code is a subcode of the invariant code. In [13, 3], it was shown that for the quasi-cyclic case, these two subcodes are identical and equal to $\Gamma(L', f(x))$, where $L'$ is a set of representatives of orbits in $L$ and $g(x) = f(x^r)$ with $r$ being the order of the cyclic subgroup of $\mathbb{F}_q^*$.

# 3 Partial quasi-cyclic and partial quasi-dyadic Goppa codes

In this section, we present a more general construction for codes satisfying the properties in Problem 1, namely, we do not restrict to nontrivial automorphism subgroups of the code. In other words, our codes may have trivial automorphism groups but contain subsets of codewords and sets of permutations where these codewords and their corresponding permutations generate the whole code. These constructions will prevent an attacker from finding folded and invariant subcodes of the code.

First, we give a sufficient condition for a permuted codeword of a code to remain in the code.

**Lemma 3.1.** *Given a linear code $C$ and a permutation $\sigma \in \mathcal{S}_n$, $C' = C \cap \sigma(C)$ is also a linear code. Suppose that $C'$ is nontrivial. For any codeword $\mathbf{c} \in C'$, one has $\sigma^{-1}(\mathbf{c}) \in C$. Further, if $\sigma^{-1}(\mathbf{c}) \in \sigma(C)$, then $\sigma^i(\mathbf{c}) \in C'$ for $i \in \mathbb{Z}$.*

*Proof.* Since $\mathbf{c} \in C' \subseteq \sigma(C)$, it follows that $\sigma^{-1}(\mathbf{c}) \in C$. In the case where $\sigma^{-1}(\mathbf{c}) \in \sigma(C)$, then $\sigma^{-1}(\mathbf{c}) \in C'$. Based on the fact that $\mathbf{c}, \sigma^{-1}(\mathbf{c}) \in C'$ and $\sigma$ is a permutation, one can continue to apply $\sigma^{-1}$ many times and the result is still a codeword of $C'$. □

**Remark 3.2.** *Suppose that $\sigma$ has order 2. Then $\sigma(\mathbf{c}) \in \sigma(C) \cap \sigma(\sigma(C)) = \sigma(C) \cap C$.*

More generally, let $\sigma_0$ be the identity permutation and let $\sigma_1, \ldots, \sigma_l$ be $l$ distinct permutations in $\mathcal{S}_n$. Suppose that for a linear code $C$, the code $C' = \bigcap_{i=0}^{l} \sigma_i(C)$ is nontrivial. Then for $\mathbf{c} \in C'$ and for all $i = 1, 2, \ldots, l$, $\sigma_i^{-1}(\mathbf{c}) \in C$.

In terms of Goppa codes, one has the following result.

**Lemma 3.3.** *Let $L$ be a subset of $n$ distinct elements from $\mathbb{F}_q$. Let $\sigma_1, \ldots, \sigma_l$ be $l$ distinct automorphisms from $\mathrm{AGL}_2(q)$ such that $L$ is invariant under each $\sigma_i$, $i = 1, 2, \ldots, l$. Let $g(x) \in \mathbb{F}_q[x]$ be such that $g(x), g(\sigma_1(x)), \ldots, g(\sigma_l(x))$ are pairwise co-prime. Define $G(x) =$*

$\prod_{i=0}^{l} g(\sigma_i(x))$. *If* $\Gamma(L, G(x))$ *is nontrivial, then for every codeword* $\mathbf{c} \in \Gamma(L, G(x))$, *the permuted codeword* $\sigma_i(\mathbf{c})$ *lies in* $\Gamma(L, g(x))$. *Moreover, if* $\sigma_0 = id, \sigma_1, \ldots, \sigma_l$ *form a subgroup of* $\mathrm{AGL}_2(q)$, *then each* $\sigma_i(\mathbf{c})$ *lies in* $\Gamma(L, G(x))$.

*Proof.* From Lemma 2.3 and Proposition 2.11, one has

$$\begin{aligned}\Gamma(L, G(x)) &= \bigcap_{i=0}^{l} \Gamma(L, g(\sigma_i(x)) \\ &= \bigcap_{i=0}^{l} \sigma_i^{-1}(\Gamma(L, g)).\end{aligned}$$

Hence, it follows from the above argument that for $i \in \{1 \ldots l\}$ and $\mathbf{c} \in \Gamma(L, G(x))$ lies in the code $\Gamma(L, g)$ as well. For the latter part, if $\mathcal{G} = \{\sigma_0, \sigma_1, \ldots, \sigma_l\}$ is a group, then $\Gamma(L, g(x)g(\sigma_1(x)) \ldots g(\sigma_l(x)))$ is a code with $\mathcal{G}$ as its automorphism subgroup. □

There are different ways one can apply Lemma 3.3 to construct Goppa codes with more permuted codewords. Here, we provide two different constructions.

In the following, we assume that $n = lmt$ for some positive integer $l$.

**Proposition 3.4** (Partial quasi-dyadic construction)**.** *Let* $l = 2^v$, *i.e.,* $n = 2^v mt$ *for some positive integer* $v \leq m/2$. *Let* $V$ *be a* $v$*-dimensional subspace of* $\mathbb{F}_q$. *Write* $V$ *as* $V = \{b_0, b_1, b_2, \ldots, b_{2^v-1}\}$ *with* $b_0 = 0$. *Let* $L$ *contain* $mt$ *different cosets of* $V$ *under some ordering. Let* $g(x)$ *be an irreducible polynomial of degree* $t$ *such that the Goppa code* $\Gamma(L, g)$ *has dimension* $k = n - mt = (2^v - 1)mt$. *For* $i = 0, 1, \ldots, 2^v - 1$, *define* $\sigma_i$ *as the automorphism where* $\sigma_i(x) = x + b_i$. *Define* $h(x) = \prod_{i=0}^{2^v-1} g(\sigma_i(x))$. *Suppose that the code* $\Gamma(L, h)$ *is the trivial code. Then, one can construct a subset* $P(L, g)$ *of* $\Gamma(L, g)$ *satisfying the conditions of Problem 1.*

*Proof.* Fix a $\sigma_i$ with $i \neq 0$ and let $h_i(x) = \frac{h(x)}{g(\sigma_i(x))}$. Let $C = \Gamma(L, h_i(x))$. Clearly, $C$ is a subcode of $\Gamma(L, g(x))$. For each codeword $\mathbf{c} \in C$, one has $\sigma_j(\mathbf{c}) \in \Gamma(L, g(x))$ for $j \neq i$. In particular, $\sigma_j(C)$ is a subcode of $\Gamma(L, g(x))$ for all $j \neq i$ (since the order of $\sigma_j$ is 2). We claim that for $j_1, j_2$ such that $i, j_1, j_2$ are all distinct, one has $\sigma_{j_1}(C) \cap \sigma_{j_2}(C)$ is the trivial code. Indeed, $\sigma_j(\Gamma(L, h_i)) = \Gamma(L, \prod_{e \neq i} g(\sigma_e \sigma_j(x)))$. Thus, by the assumption that the $g(\sigma_i(x))$'s are coprime and Lemma 2.3, $\sigma_{j_1}(\Gamma(L, h_i(x))) \cap \sigma_{j_2}(\Gamma(L, h_i(x))) = \Gamma(L, h(x))$ which is trivial by our assumption. In addition, since $\dim(\sigma_j(C)) \geq n - (2^v - 1)mt = mt$, it follows that $\dim(\sigma_j(C)) = mt$ for all $j \neq i$. Let $P(L, g)$ be a basis of $C$. Then, $P(L, g)$ satisfies the conditions of Problem 1 by letting the set of permutations to be $\{\sigma_j : j \neq i\}$. □

**Proposition 3.5** (Partial quasi-cyclic construction)**.** *Let* $r$ *be a positive integer with* $r|(q-1)$. *Let* $n = rmt$. *Let* $H$ *be a subgroup of* $\mathbb{F}_q^*$ *of order* $r$. *Let* $L$ *be a disjoint union of* $mt$ *cosets of* $H$ *in some order. Fix an irreducible polynomial* $g(x)$ *of degree* $t$ *such that the Goppa code* $\Gamma(L, g)$ *has dimension* $(r-1)mt$. *For each* $a \in H$, *let* $\sigma_a$ *be the automorphism such that* $\sigma_a(x) = ax$. *Define* $h(x) = \prod_{a \in H} g(ax)$. *Suppose that the code* $\Gamma(L, h)$ *is trivial. Then, one can find a set* $P(L, g)$ *of* $mt$ *linearly independent vectors that satisfies the conditions of Problem 1.*

*Proof.* Fix a generator $a$ of $H$. Define $g_a(x) = \prod_{i=0}^{r-2} g(a^i x)$. Let $C$ be the code $C = \Gamma(L, g_a)$. By Proposition 3.3, for all $i = 1, \ldots, r-2$, we have $\sigma_{a^i}(\mathbf{c}) \in \Gamma(L, g)$ for all $\mathbf{c} \in C$, that is, $\sigma_{a^i}(C) \subset \Gamma(L, g)$ for $i = 2, 3, \ldots, r$. We claim that $\Gamma(L, g)$ is a disjoint union of $C, \sigma_{a^2}(C), \ldots, \sigma_{a^{r-1}}(C)$. First, we show that for $i \neq j$, $\sigma_{a^i}(C) \cap \sigma_{a^j}(C)$ is trivial. Now, we have $\sigma_{a^i}(C) = \Gamma(L, h_a(\sigma_{a^{-i}}(x))) = \Gamma(L, \prod_{w \neq r-1-i} g(\sigma_{a^w}(x)))$. Similarly, $\sigma_{a^j}(C) = \Gamma(L, \prod_{w \neq r-1-j} g(\sigma_{a^w}(x)))$. From Lemma 2.3, $\sigma_{a^i}(C) \cap \sigma_{a^j}(C) = \Gamma(L, h)$ which is trivial. Since $\dim(C) \geq n - (r-1)mt = mt$, the claim follows. Consequently, one may choose $P(L, g)$ as a basis of $C$. □

**Remark 3.6.**
- *In both Propositions 3.4 and 3.5, we have* $\dim(\Gamma(L,h)) \geq n - \deg(h)m = 0$. *In practice, this code is trivial for most cases.*
- *In fact, one can generalize the results to non-irreducible polynomials* $g$. *In this case, one needs to check that the polynomials* $g(\sigma(x))$*'s are all pairwise coprime for all the automorphisms* $\sigma$ *involved.*

**Remark 3.7.** *Just as in Theorem 2.12, one may transform* $P(L,g)$ *such that* $P(L,g)$ *takes the following form. Let*

$$P(L,g) = (S_1||S_2||\dots S_{mt-1}||S_{mt}),$$

*where each* $S_i$ *is an* $mt \times r$ *matrix and for* $i = 1, 2, \dots, mt-1$, $S_i$ *is* 0 *everywhere except at the* $(i,1)$*-position. Thus, one may use* $S_{mt}$ *as the public key.*

From Propositions 3.4 and 3.5, we see that one can generalize the construction to any subgroup of $\mathrm{AGL}_2(q)$. More precisely, we give the general construction in the next theorem and the proof is similar to the proofs of Propositions 3.4 and 3.5.

**Theorem 3.8.** *Let* $t$ *be a positive integer. Let* $\mathcal{G}$ *be a subgroup of* $\mathrm{AGL}_2(q)$ *and let* $L$ *be a disjoint union of* $mt$ $\mathcal{G}$*-orbits under the action of* $\mathcal{G}$ *on* $\mathbb{F}_q$. *Let* $n = |L| = |\mathcal{G}|mt$. *Let* $g(x)$ *be an irreducible polynomial of degree* $t$ *such that the code* $\Gamma(L, g(x))$ *has dimension* $n - mt$. *Define* $h(x) = \prod_{\sigma\in\mathcal{G}} g(\sigma(x))$. *Assume that* $\Gamma(L,h)$ *is trivial. Fix a* $\sigma \in \mathcal{G}$ *where* $\sigma$ *is not the identity automorphism and let* $\mathcal{G}' = \mathcal{G}\backslash\{\sigma\}$. *Construct* $C = \Gamma(L, \prod_{\sigma'\in\mathcal{G}'} g(\sigma'(x)))$. *Then, we may take a basis of* $C$ *as the set* $P(L,g)$ *in Problem 1 with the set of permutations to be the permutations in* $\mathcal{G}'$.

# 4 Security discussion

In general, there are two classes of attacks on McEliece scheme, namely, information set decoding (ISD) attacks and structural attacks. The former attacks were proposed in [31] as a generic decoding attack. Essentially, such attacks seek to perform the decoding directly from the public matrix given. In its most basic form, an adversary tries to guess $k$ positions such that the submatrix restricted to these positions is invertible and the error vector, restricted to these positions, has very small Hamming weight. Such properties will enable the adversary to brute force the error entries at these positions and subsequently, discover the message. Other improvements and variants were later proposed to improve the attack [20, 35, 11, 9, 17, 8, 24, 4, 25]. So far, the best attack has complexity $2^{0.0967n}$ [25], where $n$ is the code length.

On the other hand, the structural attacks attempt to take advantage of the structure of the specific code used. For random Goppa codes, one can perform a brute force search on the Goppa polynomial and/or the elements in the Goppa support. In [32], an algorithm known as the support splitting algorithm was proposed which can correct permutation of the support elements once the Goppa polynomial and the set of support elements are known.

In addition, one can launch an algebraic attack on a Goppa code $C$ with Goppa polynomial $g$ as follows. Consider the equations:

$$\sum_{i=1}^{n} c_i x_i^j y_i^w = 0 \tag{6}$$

for $\mathbf{c} = (c_1, \dots, c_n) \in C$, $0 \leq j \leq t-1$ when $w = 1$ and $0 \leq j \leq 2t-1$ when $w = 2$, $x_1, x_2, \dots, x_n$ are the unknowns for the Goppa support and $y_i = 1/g(x_i)$. Recall that these equations come from the parity-check equations (refer to Equations 1 and 2). Hence, solving for the unknowns will give the Goppa support and together with the knowledge of the $y_i$'s will enable one to recover $g(x)$. Thus the challenge remains to solve the above system.

Observe that when $j = 0$ yields $k$ linear equations in the unknowns $y_i$'s. Thus, one way to solve the system is to guess the remaining $mt$ $y_i$'s which results in a system in the $x_i$'s. By considering the $k\lfloor 2 \log_2 t \rfloor$ equations which are powers of 2, one can then solve for the unknowns $x_i$'s. The complexity of the approach is asymptotically $O(2^{m^2 t})$. Other approaches exist to solve the system [15, 14] but the complexities of these approaches are more difficult to estimate.

As mentioned in Subsection 2.3, the automorphism-induced Goppa codes are more vulnerable to algebraic attacks as compared to random Goppa codes. This is due to the existence of invariant subcodes which are themselves Goppa codes with smaller supports and lower degree Goppa polynomials. In particular, for an automorphism-induce Goppa code, particularly quasi-cyclic code, such that it has length $n$, degree $t$ Goppa polynomial and automorphism group of order $l$, the invariant code has length $n/l$ with degree $t/l$ Goppa polynomial. A thorough security analysis of quasi-cyclic Goppa codes can be found in [2].

For our construction, only subsets of the automorphism subgroup of $\mathrm{AGL}_2(q)$ are used. In particular, we typically remove an element $\sigma$ from a subgroup $\mathcal{G}$ of $\mathrm{AGL}_2(q)$ (see Theorem 3.8).

Thus, a similar folding operation will be of the form

$$\sum_{\sigma' \in \mathcal{G}, \sigma' \neq \sigma} \sigma'(\mathbf{c})$$

for any $\mathbf{c}$ in the code. Since the coordinates of $\mathbf{c}$ on each orbit are unlikely to be identical, one cannot use the same trick to puncture the code or to construct an invariant code.

However, one may attempt to mount an algebraic attack on our constructions.

In Propositions 3.4 and 3.5, suppose that $V$ and $H$ are known, respectively (this assumption is valid as one can exhaustively search for them). Referring to the system of equations in 6 and the structure of the Goppa support as a disjoint union of cosets, the unknowns for the Goppa support are reduced to finding $mt$ representatives in the orbits. However, since our choice of $g(x)$ is such that it is not invariant under any automorphism used in the construction, there is no clear relationship between the $y_i$'s that we can exploit. Consequently, one has to solve a system of equations in $mt$ unknowns in the $x_i$'s and $n$ unknowns in the $y_i$'s. Once again, by exploiting the $k$ linear relationships among the $y_i$'s, one way to solve the system is to perform a brute force search on the remaining free $mt$ $y_i$'s. Alternatively, one can search through all possible $mt$ $x_i$'s and solve the resulting linear equations in the $y_i$'s. In either case, the time complexity is $O(2^{m^2 t})$. Hence, unlike the quasi-cyclic or quasi-dyadic constructions, we see that our constructions do not weaken the security of the codes with respect to algebraic attacks.

# 5 Practical implementation considerations

## 5.1 Practical implementation scheme

Observe that our constructions place some restrictions on the choice of the parameters. For instance, we require $n$ to be an integral multiple of $mt$. Moreover, the reduction factor is at most $(n - mt)/mt = k/(n - k)$. Thus, to achieve a large reduction size, we need $t$ to be small. However, $t$ needs to be sufficiently large to prevent an exhaustive search on the error vector, or more generally, to guard against the state-of-the-art information set-decoding attacks. In addition, for fixed $m$ and $t$, $n$ cannot be to large to prevent the distinguishing attack [12]. On the other hand, even though the automorphism-induced codes are more vulnerable to algebraic attacks, the relatively large gap between the complexity of these attacks and the information set-decoding attacks give some room to choose the parameters appropriately for any security level. As such, we propose combining the two constructions to achieve a greater key reduction for a desired security level. The following algorithm gives a possible construction meeting a given security level. Here, we only consider the quasi-cyclic construction as the dyadic construction is similar.

1. Fix a security parameter $\lambda$.
2. Fix $m$ such that $q-1=2^m-1$ has small factors.
3. For any two distinct integers $l_1$ and $l_2$ with $l_1 l_2 | (q-1)$ and $\gcd(l_1, l_2) = 1$, do the following:
   - Pick a $t_0$ satisfying the following two conditions:
     - $\binom{q}{t_0} \geq 2^\lambda$;
     - A $[l_1 l_2 m t_0, (l_1 l_2 - l_1) m t_0, 2 l_1 t_0]$ Goppa code achieves the security level with respect to the information set-decoding attacks.
4. Let $t = l_1 t_0$ and $n = l_1 l_2 m t_0$.
5. Let $H$ be a subgroup of $\mathbb{F}_q^*$ of order $l_1 l_2$ and pick a generator $a$ of $H$.
6. Let $L$ be a disjoint union of $mt_0$ different cosets of $H$ and randomly choose an ordering of the cosets.
7. Randomly pick a polynomial $g_0(x)$ of degree $t_0$. Let $g(x) = g_0(x^{l_1})$.
8. Let $h(x) = \prod_{i=0}^{l_2-2} g(a^{il_1}x)$ and $f(x) = \prod_{i=0}^{l_2-1} g(a^{il_1}x)$
9. Construct the code $\Gamma(L, g), \Gamma(L, h)$ and $\Gamma(L, f)$.
10. If $\dim(\Gamma(L, f)) \neq 0$ or $\dim(\Gamma(L, h)) \neq mt$ or $\dim(\Gamma(L, g)) \neq (l_2 - 1)mt$, choose another $g_0(x)$.
11. Otherwise, return $\Gamma(L, g)$.

By Proposition 3.5, for all $\mathbf{c} \in \Gamma(L, h)$ and for $i = 0, 1, \ldots, l_2 - 2$, we have $\sigma_{a^{l_1 i}}(\mathbf{c}) \in \Gamma(L, g)$. Moreover, $\Gamma(L, h)$ is a quasi-cyclic code induced by a subgroup of $H$ of order $l_1$. Consequently, one can find a set $P(L, g)$ of cardinality $mt_0$ such that for all $i = 0, 1, \ldots, l_2 - 2$, $j = 0, 1, \ldots, l_1 - 1$, $\sigma_{a^{l_1 i + l_2 j}}(\mathbf{c}) \in \Gamma(L, g)$ whenever $\mathbf{c} \in P(L, g)$. Similar to Theorem 2.12 and Remark 3.7, one can construct an $mt_0 \times mt$ matrix that can be extended to a generator matrix of $\Gamma(L, g)$. Consequently, the public key size of such a code is $l_1 m^2 t_0^2$.

In our construction, by our choice of $t_0$, we ensure that it is secure against a brute force search on all possible $g_0(x)$ and in fact, it is resistant to the existing known structural attacks (see Section 4). In addition, we have ensured that the code $\Gamma(L, g)$ is secure against message recovery attacks.

**Remark 5.1.** *By employing only the quasi-cyclic construction, one gets a public key size of* $(l_2 - 1)l_1 m^2 t_0^2$. *Thus, we see that our technique can be further applied to quasi-cyclic constructions to yield more compact public keys.*

## 5.2 Concrete parameters

We apply our constructions to the parameters in BigQuake [2] to reduce the public key size and to keep the desired security level. We give 3 groups of parameters corresponding to 3 different security levels and each group comprises two code parameters in Table 1. The first is the code parameters from BigQuake [2] and the second is the reduced code parameters we get after applying our scheme to the first code. Now we explain the notations of each column of Table 1 as follows.

- $\lambda$: security level of the parameters.
- $m$: extension degree of the finite field.
- $[n, k, t]$ denotes the resulting Goppa code parameters, $n$ for the code length, $k$ for the code dimension and $t$ for the error correcting ability.
- $l_1$: the size of the quasi-cyclic group which acts on $x$.

- $l_2$: the size of the partial quasi-cyclic group which acts on the Goppa polynomial $g(x)$. This column is not applicable to the BigQuake [2] code parameters.
- $\omega_{msg}$: the logarithm of the work load of ISD, which is computed using the **CaWoF** [36] library.
- $Size(bytes)$: the resulting public key size.

From Table 1, we can see that our scheme reduces the public key size of BigQuake by at least half. We remark that the parameters given in Table 1 may be vulnerable to other attacks. Here we show the parameters solely to illustrate the power our construction which, to our best knowledge, are secure against current known attacks. In fact, our construction can be applied to most of the existing Goppa codes based constructions to reduce the public key size without compromising the security level.

Table 1: More compact public key parameters by applying our construction to BigQuake

| $\lambda$ | $m$ | $[n, k, t]$ | $l_1$ | $l_2$ | $\omega_{msg}$ | Size (bytes) |
|---|---|---|---|---|---|---|
| AES128 | 12 | $[3510, 2418, 91]$ | 13 | NA | 132 | 25389 |
| | 16 | $[12240, 11520, 45]$ | 5 | 17 | 143 | 12960 |
| AES192 | 18 | $[7410, 4674, 152]$ | 19 | NA | 195 | 84132 |
| | 18 | $[8208, 5472, 152]$ | 19 | 3 | 213 | 49248 |
| AES256 | 18 | $[10070, 6650, 190]$ | 19 | NA | 263 | 149625 |
| | 18 | $[10260, 6840, 190]$ | 19 | 3 | 267 | 76950 |

**Remark 5.2.** *The public key size of the foregoing scheme does not involve the parameter $l_2$. Therefore one could select a bigger $l_2$ to keep the public key size and ensure that the increased code parameters $n$ and $k$ will still make the code resistant to, even by a large margin improved and powerful ISD attacks in the future. Alternatively, one is able to exploit this property of our scheme to minimize the public key size by selecting very small $l_1$ and very big $l_2$, and increasing $t_0$ to some reasonable value. Note that this only applies to big enough $m$, for otherwise it will suffer from some algebraic attacks. In conclusion, one can vary the parameters $l_1, l_2, t_0, m, n$ appropriately to get a compact public key achieving the desired security level with respect to all known attacks.*

# References


[1] Gustavo Banegas, Paulo S. L. M. Barreto, Brice Odilon Boidje, Pierre-Louis Cayrel, Gilbert Ndollane Dione, Kris Gaj, Cheikh Thecoumba Gueye, Richard Haeussler, Jean Belo Klamti, Ousmane Ndiaye, Duc Tri Nguyen, and Edoardo Persichettiand Jefferson E. Ricardini. Dags: Key encapsulation from dyadic gs codes, Jun 2018. https://www.dags-project.org/pdf/DAGS_spec_v2.pdf. 2, 6, 7

[2] Magali Bardet, Elise Barelli, Olivier Blazy, Rodolfo CantoTorres, Alain Couvreur, Philippe Gaborit, Ayoub Otmani, Nicolas Sendrier, and Jean-Pierre Tillich. Big quake: Binary goppa quasicyclic key encapsulation, Apr 2018. https://bigquake.inria.fr/files/2018/04/corrected_proposal.pdf. 2, 3, 6, 7, 11, 12, 13

[3] Elise Barelli. On the security of some compact keys for mceliece scheme. *CoRR*, abs/1803.05289, 2018. 2, 3, 8

[4] Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. Decoding random binary linear codes in 2 n/20: How 1 + 1 = 0 improves information set decoding. In *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 520–536. Springer, 2012. 10

[5] Thierry P. Berger. Goppa and related codes invariant under a prescribed permutation. *IEEE Trans. Information Theory*, 46(7):2628–2633, 2000. 2, 6

[6] Thierry P. Berger, Pierre-Louis Cayrel, Philippe Gaborit, and Ayoub Otmani. Reducing key length of the mceliece cryptosystem. In *AFRICACRYPT*, volume 5580 of *Lecture Notes in Computer Science*, pages 77–97. Springer, 2009. 2, 6, 7

[7] Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen. *Post Quantum Cryptography*. Springer Publishing Company, Incorporated, 1st edition, 2008. 1

[8] Daniel J. Bernstein, Tanja Lange, and Christiane Peters. Smaller decoding exponents: Ball-collision decoding. In *CRYPTO*, volume 6841 of *Lecture Notes in Computer Science*, pages 743–760. Springer, 2011. 10

[9] Anne Canteaut and Florent Chabaud. A new algorithm for finding minimum-weight words in a linear code: Application to mceliece's cryptosystem and to narrow-sense BCH codes of length 511. *IEEE Trans. Information Theory*, 44(1):367–378, 1998. 10

[10] Claude Chevalley. *Introduction to the theory of algebraic functions of one variable*. Number 6. American Mathematical Soc., 1951. 4

[11] Ilya Dumer. On minimum distance decoding of linear codes. In *Proc. 5th Joint Soviet-Swedish Int. Workshop Inform. Theory*, pages 50–52, 1991. 10

[12] Jean-Charles Faugère, Valérie Gauthier-Umaña, Ayoub Otmani, Ludovic Perret, and Jean-Pierre Tillich. A distinguisher for high-rate mceliece cryptosystems. *IEEE Trans. Information Theory*, 59(10):6830–6844, 2013. 11

[13] Jean-Charles Faugère, Ayoub Otmani, Ludovic Perret, Frédéric de Portzamparc, and Jean-Pierre Tillich. Folding alternant and goppa codes with non-trivial automorphism groups. *IEEE Trans. Information Theory*, 62(1):184–198, 2016. 2, 3, 6, 8

[14] Jean-Charles Faugère, Ayoub Otmani, Ludovic Perret, Frédéric de Portzamparc, and Jean-Pierre Tillich. Structural cryptanalysis of mceliece schemes with compact keys. *Des. Codes Cryptography*, 79(1):87–112, 2016. 2, 3, 11

[15] Jean-Charles Faugère, Ayoub Otmani, Ludovic Perret, and Jean-Pierre Tillich. Algebraic cryptanalysis of mceliece variants with compact keys. In *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings*, pages 279–298, 2010. 2, 3, 11

[16] Jean-Charles Faugre, Ludovic Perret, and Frdric de Portzamparc. Algebraic attack against variants of mceliece with goppa polynomial of a special form. In *ASIACRYPT (1)*, pages 21–41. Springer, 2014. 2, 3

[17] Matthieu Finiasz and Nicolas Sendrier. Security bounds for the design of code-based cryptosystems. In *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 88–105. Springer, 2009. 10

[18] Heeralal Janwa and Oscar Moreno. Mceliece public key cryptosystems using algebraic-geometric codes. *Des. Codes Cryptography*, 8(3):293–307, 1996. 2

[19] Kazukuni Kobara and Hideki Imai. Semantically secure mceliece public-key cryptosystems-conversions for mceliece PKC. In *Public Key Cryptography, 4th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2001, Cheju Island, Korea, February 13-15, 2001, Proceedings*, pages 19–35, 2001. 2

[20] Pil Joong Lee and Ernest F. Brickell. An observation on the security of mceliece's public-key cryptosystem. In *Advances in Cryptology - EUROCRYPT '88, Workshop on the Theory and Application of of Cryptographic Techniques, Davos, Switzerland, May 25-27, 1988, Proceedings*, pages 275–280, 1988. 10

[21] Rudolf Lidl and Harald Niederreiter. Finite fields: Encyclopedia of mathematics and its applications. *Computers & Mathematics with Applications*, 33(7):136–136, 1997. 5

[22] V M. SIDELNIKOV and S O. SHESTAKOV. On insecurity of cryptosystems based on generalized reed-solomon codes. 2:439–444, 01 1992. 2

[23] F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error Correcting Codes*. Number v. 2, pt. 2 in Mathematical Studies. North-Holland Publishing Company, 1978. 4

[24] Alexander May, Alexander Meurer, and Enrico Thomae. Decoding random linear codes in $\tilde{\mathcal{O}}(2^{0.054n})$. In *ASIACRYPT*, volume 7073 of *Lecture Notes in Computer Science*, pages 107–124. Springer, 2011. 10

[25] Alexander May and Ilya Ozerov. On computing nearest neighbors with applications to decoding of binary linear codes. In *EUROCRYPT (1)*, volume 9056 of *Lecture Notes in Computer Science*, pages 203–228. Springer, 2015. 10

[26] Robert J McEliece. A Public-Key Cryptosystem Based On Algebraic Coding Theory. *Deep Space Network Progress Report*, 44:114–116, Jan 1978. 1, 2, 3

[27] Lorenz Minder and Amin Shokrollahi. Cryptanalysis of the sidelnikov cryptosystem. In *EUROCRYPT*, volume 4515 of *Lecture Notes in Computer Science*, pages 347–360. Springer, 2007. 2

[28] Rafael Misoczki and Paulo S. L. M. Barreto. Compact mceliece keys from goppa codes. In *Selected Areas in Cryptography*, volume 5867 of *Lecture Notes in Computer Science*, pages 376–392. Springer, 2009. 2, 6, 7

[29] Rafael Misoczki, Jean-Pierre Tillich, Nicolas Sendrier, and Paulo S. L. M. Barreto. Mdpc-mceliece: New mceliece variants from moderate density parity-check codes. In *ISIT*, pages 2069–2073. IEEE, 2013. 2

[30] H Niederreiter. Knapsack type cryptosystems and algebraic coding theory. 15(2):159–166, 01 1986. 2

[31] Eugene Prange. The use of information sets in decoding cyclic codes. *IRE Trans. Information Theory*, 8(5):5–9, 1962. 10

[32] Nicolas Sendrier. Finding the permutation between equivalent linear codes: The support splitting algorithm. *IEEE Trans. Information Theory*, 46(4):1193–1203, 2000. 10

[33] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 124–134, 1994. 1

[34] Vladimir Michilovich Sidelnikov. A public-key cryptosystem based on binary reed-muller codes. *Discrete Mathematics and Applications*, 4(3):191–208, 1994. 2

[35] Jacques Stern. A method for finding codewords of small weight. In *Coding Theory and Applications, 3rd International Colloquium, Toulon, France, November 2-4, 1988, Proceedings*, pages 106–113, 1988. 10

[36] Rodolfo Canto Torres. Cawof, c library for computing asymptotic exponents of generic decoding work factors, Jan 2017. https://gforge.inria.fr/projects/cawof/. 13