SOME RESULTS ON A GROUP UNDER WHICH SYMMETRIC REED-MULLER CODES ARE INVARIANT

SİMETRİK REED-MULLER KODLARININ DEĞİŞMEZ OLDUĞU BİR GRUP ÜZERİNE BAZI SONUÇLAR

SİBEL KURT TOPLU

PROF. DR. PINAR AYDOĞDU

Supervisor

DOÇ. DR. OĞUZ YAYLA

2nd Supervisor

Submitted to

Graduate School of Science and Engineering of Hacettepe University

as a Partial Fulfillment to the Requirements

for the Award of the Degree of Doctor of Philosopy

in Mathematics

June 2024

ABSTRACT

SOME RESULTS ON A GROUP UNDER WHICH SYMMETRIC REED-MULLER CODES ARE INVARIANT

Sibel KURT TOPLU

Doctor of Philosopy, Mathematics

Supervisor: Prof. Dr. PINAR AYDOĞDU

2nd Supervisor: Doç. Dr. OĞUZ YAYLA

June 2024, 91 pages

The Reed-Muller codes are a family of error-correcting codes that have been widely studied in coding theory. In 2020, Wei Yan and Sian-Jheng Lin introduced a variant of Reed-Muller codes so called symmetric Reed-Muller codes. We investigate linear maps of the automorphism group of symmetric Reed-Muller codes and show that the set of these linear maps forms a subgroup of the general linear group, which is the automorphism group of punctured Reed-Muller codes. We provide a method to determine all the automorphisms in this subgroup explicitly for some special cases.

Keywords: Reed-Muller codes, Symmetric Reed-Muller codes, Affine invariant, Automorphism groups.

ÖZET

SİMETRİK REED-MULLER KODLARININ DEĞİŞMEZ OLDUĞU BİR GRUP ÜZERİNE BAZI SONUÇLAR

Sibel KURT TOPLU

Doktora, Matematik

Danışman: Prof. Dr. PINAR AYDOĞDU

Eş Danışman: Doç. Dr. OĞUZ YAYLA

Haziran 2024, 91 sayfa

Kodlama teorisinde, yıllardır gönderici ve alıcı arasında veri alışverişini daha iyi hale getirmek için çalışmalar yapılmaktadır. Bu bilgi alışverişi sırasında başa çıkılması gereken en önemli sorun iletişim sırasında oluşacak hatalardır. Bunun için mesajın sonuna eklenen kontrol bitleri gibi çeşitli önlemler alınmıştır. Bu kontrol bitleri sayesinde alıcı aldığı mesajı kontrol edebilir. Aşağıda bu şekilde kontrol edilebilen kodlar için temel kavramlar verilecektir.

$\mathbb{F}_q^n$ vektör uzayı üzerinde n uzunluklu bir C kodu, $\mathbb{F}_q^n$ kümesinin bir alt kümesidir. Özel olarak, lineer kodların alfabesi de sonlu cisim üzerinde tanımlıdır. $\mathbb{F}_q$ üzerinde bir *lineer kod*, $\mathbb{F}_q^n$ vektör uzayının bir alt uzayıdır.

Kod kelimeleri arasındaki *minimum uzaklık* bir kodun önemli parametrelerindendir. C, $\mathbb{F}_q^n$ üzerinde bir lineer kod olsun. C'nin minimum uzaklığı kod kelimelerinin birbirleri arasındaki uzaklıklardan en küçüğüdür, yani

$$d = d(C) = min\{d(x, y) \mid x, y \in C, x \neq y\}.$$

şeklinde tanımlıdır.

Bir lineer C kodu, $\mathbb{F}_q^n$ vektör uzayının k boyutlu alt uzayı ve C'nin minimum uzaklığı d ise C koduna q-lu bir $[n, k, d]$-kod denir (bkz. [1]).

Bir kodu oluşturan tüm elemanlara *kod sözcüğü* denir. C lineer kodu q^k adet kod kelimesine sahiptir.

Bir lineer $[n, k]$-kodunun *oranı* $\frac{k}{n}$'dir. Bu oran her bir kod kelimesinin ne kadar bilgi taşıdığının ölçüsüdür. Bir C kodunun *göreceli uzaklığı* ise $\frac{d(C)}{n}$ olarak tanımlıdır.

Bir $x \in \mathbb{F}_q^n$ vektörünün ağırlığı, x'teki sıfırdan farklı sembollerin sayısıdır ve $w(x)$ ile gösterilir. Minimum uzaklık ve minimum ağırlık tanımları kullanıldığında

$$x, y \in \mathbb{F}_q^n \iff d(x, y) = w(x - y)$$

olduğu görülür. Bundan dolayı, bir C lineer kodunun sıfırdan farklı kod kelimelerinin minimum ağırlığı $w(C)$, minimum uzaklığı $d(C)$ birbirine eşittir.

C bir $[n, k]$-kod olsun. Satırları, C kodu için bir taban oluşturan $k \times n$ boyutlu bir G matrisine, *üreteç matrisi* denir. Bileşenleri $\mathbb{F}_q$'dan alınan ve sıralı k-lilerden oluşan vektör uzayı $V(k, q)$ olmak üzere, C kod uzayı aşağıdaki gibidir :

$$C = \{xG \mid x \in V(k, q)\}.$$

Üreteç matrisi, I birim matris olmak üzere, $G = [I_k \mid A]$ biçimindedir.

C'nin dual kodu $C^\perp = \{v \in V(n, q) \mid v.c = 0, \forall c \in C\}$ olarak tanımlıdır. Bu durumda, C kodunun duali $C^\perp$ bir lineer $[n, n - k]$-koddur. Bir C lineer kodunun dualinin duali de kendisine eşittir, yani $(C^\perp)^\perp = C$'dir.

C bir lineer $[n, k]$-kod olsun. $C^\perp$'nin üreteç matrisi H, C'nin *kontrol matrisi* olarak adlandırılır. Bu durumda,

$$x \in C \iff xH^T = 0$$

sağlanır.

Kodlama teorisinin ana amaçlarından biri en yüksek hata doğrulama kapasitesine, geniş bir göreceli uzaklığa ve yüksek bir bilgi oranına ulaşmaktır. Ancak, bu amaca ulaşmada belirli kısıtlayıcı özellikler mevcuttur. Bu kısıtlardan bir tanesi de *Singleton kısıtıdır*. C bir $[n, k, d]$-kod olmak üzere, bu kısıt aşağıdaki şekilde tanımlanır :

$$d \leq n - k + 1.$$

Reed-Solomon kodları, 1960 yıllarında bulunmuş lineer kodlardır. Reed Solomon kodları çoklu hata tespit eden ve düzeltebilen lineer kodlardır. Bu kod, $\mathbb{F}_q^n$ üzerinde tanımlı, blok uzunluğu n, boyutu k, minimum uzaklığı $d = n - k + d$'dir.

$$\{m_K \mid K \subseteq \{1, 2, \ldots, n\}, |K| \leq r\}$$

olur.

Reed-Solomon kodlarını inşa etmede birçok yöntem vardır. Her bir Reed-Solomon kod kelimesi, derecesi k'den küçük polinom değerlerinin bir dizisidir. Reed-Solomon kod kelimesinin mesaj sembolleri, derecesi k'den küçük, $\mathbb{F}_q$ üzerindeki polinomun katsayılarıdır. Bir mesaj $m = (m_0, m_1, \ldots, m_{k-1})$, bir t polinomu $t(x) = m_0 + m_1 x + \ldots + m_{k-1}x^{k-1}$ içerisine gömülür. α ilkel bir eleman olmak üzere Reed-Solomon kodu

$$C = \{(t(1), t(\alpha), \ldots, t(\alpha^{n-1})) \mid t \in \mathbb{F}_q[x], der(t) \leq k\}$$

şeklinde oluşturulur. Gönderici bu şekilde bir kod kelimesini $\{1, \alpha, \ldots, \alpha^{n-1}\}$ kümesinde değerlendirdiğinde ortaya çıkan $s = (s_1, s_2, \ldots, s_n)$'yi alıcıya gönderir. Alıcı bu s_i noktalarından en az t tanesinin $s_i = t(\alpha^i)$ formatında olduğunu bilir ve buna göre aldığı koddan mesajı tekrar oluşturur.

Reed-Muller (kısaca, RM) *kodları* ise 1950'lerde bulunmuş, kolay deşifre edilebilen lineer blok kodlardır ve birçok oluşturulma yöntemi vardır. Bu kodlar, Boolean fonksiyonlarla ya

da polinomlarla oluşturulur. r-dereceli Reed-Muller $R(r,m)$ kodu derecesi en fazla r olan tüm polinomların kümesidir. RM kodları yinelemeli olarak tanımlanabilir. $R(0,m)$, tabanı 1 olan, 0-dereceli ve kod sözcüklerinin uzunluğu 2^m olan koddur. $R(r,m)$, $1 \leq r < m$ için r-dereceli ve uzunluğu 2^m olan bir kod olsun. Bu kodu yinelemeli olarak

$$R(r,m) = \{(u, u+v) \mid u \in R(r,m-1), v \in R(r-1,m-1)\}.$$

şeklinde oluşturabiliriz. $R(r,m)$'nin üreteç matrisi $G_{r,m}$ olsun. Bu durumda $R(r+1,m+1)$ için üreteç matrisi yinelemeli olarak üretilebilir ve bu durumda

$$G(r+1,m+1) = \begin{pmatrix} G(r+1,m) & G(r+1,m) \\ 0 & G(r,m) \end{pmatrix}.$$

matrisi elde edilir (bkz. [1]).

$0 \leq r \leq m$ olacak şekilde r ve m iki tamsayı olsun. Bu durumda $R(r,m)$ aşağıdakileri sağlar :

1) $R(r-1,m) \subset R(r,m)$.

2) $R(r,m)^{\perp} = R(m-r-1,m)$.

3) $R(r,m)$ kodunun boyutu $\binom{m}{0} + \binom{m}{1} + \ldots + \binom{m}{d}$.

4) $R(r,m)$ kodunun minimum uzaklığı $d = 2^{m-r}$.

$0 \leq r \leq m-1$ için $R(r,m)$ kodunun tüm kod kelimelerinden belli bir koordinatının silinmesiyle elde edilen kod kelimelerinden oluşan koda *delinmiş (punctured) Reed-Muller kod* $R(r,m)^*$ denir. Dolayısıyla, delinmiş Reed-Muller kodlar, Reed-Muller kodlarının özel bir halidir.

Lineer kodların birbirlerine denklikleri ve bu denkliklerin çeşitleri vardır. C_1 ve C_2 lineer kodlarının *permutasyon denk* olması, C_1'den C_2'ye giden bir koordinat permutasyonunun

var olması demektir. $\mathbb{P}_n$ *permütasyon matrisleri kümesi*, her satırı ve sütununda yalnızca bir adet 1 olan ve bunun dışındaki tüm yerlerde 0 olan karesel matrislerden oluşur. Permütasyon denk olan C_1 ve C_2 lineer kodları aynı uzunluğa sahip olmakla birlikte, üreteç matrisleri G_1 ve G_2 arasında, $P \in \mathbb{P}_n$ olmak üzere,

$$G_2 = G_1 P.$$

eşitliği sağlanır (bkz. [1]).

Her bir satırında ve sütununda, sıfırdan farklı sadece bir eleman olan bir matrise *monom* adı verilir. İki lineer kodun *monom denk* olması, iki kodun üreteç matrisleri G_1 ve G_2 arasında

$$G_2 = G_1 M$$

eşitliğinin sağlanması demektir (bkz. [1]).

En genel olarak, iki lineer kodun denk olması şöyle tanımlanmaktadır: C_1 ve C_2 bir F sonlu cismi üzerinde aynı uzunluğa sahip iki lineer kod olsun.

$$C_2 = C_1 M\gamma$$

eşitliğini sağlayan bir M monom matrisi ve F üzerinde bir γ cisim otomorfizması varsa, C_1 ile C_2 kodları *denktir* denir.

Bu üç denklikten yola çıkarak, üç tip otomorfizma grubu oluşturulabilir. $\mathbb{F}_q$ sonlu cismi üzerinde uzunluğu n olan bir C kodunu düşünelim. C kodunu kendisine götüren koordinat permutasyonlarının kümesi bir grup oluşturur ve bu gruba C'nin *permutasyon otomorfizması* denir ve $PAut(C)$ ile gösterilir. $PAut(C)$'nin S_n'nin bir alt grubu olduğu açıkca görülür. Bir C kodunu, monom denk olarak yine bu C koduna götüren tüm monom matrisler bir grup oluşturur. Bu gruba, C'nin *monom otomorfizma grubu* denir ve $MAut(C)$ ile gösterilir. Son olarak, C kodunu kendisine götüren tüm $M\gamma$ elemanları C'nin otomorfizma grubunu oluşturur ve $Aut(C)$ ile gösterilir. Bu otomorfizma grupları arasında

$$PAut(C) \subseteq MAut(C) \subseteq Aut(C)$$

kapsamaları sağlanır (bkz. [1]).

İkili kodlar için $PAut(C) = MAut(C) = Aut(C)$ elde edilir. Bu durumda, söz konusu otomorfizma grupları, S_n simetrik grubunun bir alt grubudur. Eğer $Aut(C)$, genel afin gruba izomorfik bir grup içeriyorsa, C koduna *afin değişmez* denir. Bu yüzden, $R(r, m)$ kodu afin değişmezdir.

Genel afin grup $GA(m, 2)$, r dereceli $R(r, m)$'nin tüm kod kelimelerini yer değiştirir ve $GA(m, 2) \subset AutR(r, m)$ olur [2]. Genel lineer grup, delinmiş Reed-Muller kodun tüm kod kelimelerini yer değiştirir. O halde, $GL(m, 2) \subset Aut(R(r, m))^*$ elde edilir (bkz. [2]). Bu bilgiler ışığında MacWilliam vd. kitabından [2] şu eşitlikleri elde ederiz. $1 \leq r \leq m - 2$ için $Aut(R(r, m)^*) = GL(m, 2)$ ve $Aut(R(r, m)) = GA(m, 2)$.

Kodlama teorisi alanında kod otomorfizmaları önemli bir yer tutmaktadır. Örneğin, lineer kodların otomorfizmaları kodun yapısını belirlemede, ağırlık dağılımını hesaplamada, kodları sınıflandırma ve çözme algoritmalarını geliştirilmesinde çok kullanışlı bir kavramdır. Ayrıca otomorfizmaların incelenmesi, etkili çözme algoritmalarının geliştirilmesine katkıda bulunabilir. Bir kodun içindeki simetriler kullanarak, hataları düzeltmek için hesaplama açısından daha verimli algoritmaların tasarlanabileceği düşünülmektedir. Bu nedenle, kodların otomorfizmaları, kodların hata düzeltme özelliklerini anlama ve karakterize etme konusunda kritik bir rol oynar. Dolayısıyla, bir kodun simetrilerini inceleyerek, hataların kodlanmış bilgiyi nasıl etkilediği ve kodun bu hataları nasıl düzeltebileceği veya tespit edebileceği konusunda fikirler kazanılabilir. Kodlama teorisinde kod otomorfizmaları, kodların yapısını ve özelliklerini anlamak için önemli bir araçtır. Kod otomorfizmaları, iki kodun birbirine eşdeğer olup olmadığını belirlemeye yardımcı olur. Eğer iki kod, bir otomorfizma ile birbirine dönüştürülebiliyorsa, bu kodlar aynı bilgiyi taşıdığı anlamına gelir. Kod otomorfizmaları, kodların yapısal özelliklerini incelemek için de bir yöntem sunar. Bu sayede kodların simetrik özellikleri ve yapıları daha iyi anlaşılabilir. Ayrıca, kod otomorfizmalarını deşifre etme sürecine entegre ederek, şifre çözümleme stratejilerinin

verimliliğini ve etkinliğini artırabilir. Otomorfizma grubunu tanımlamak ve özelliklerinden faydalanmak, hesaplama yükünü azaltarak şifre çözümleme performansını iyileştirebilir.

RM kodları, Irving S. Reed ve Gustave Solomon Muller tarafından 1954 yılında tanıtılan bir hata düzeltme kodları ailesidir. Literatürde birçok RM kodu varyasyonu ve genelleştirmesi tanıtılmıştır (bkz. [2–4]). Bu kodların ikili versiyonları, ikili polinomlarla tanımlanır ve polinomun uygun noktalarda değerlendirilerek oluşturulur. RM kodlarının birçok önemli özelliği vardır ve bu da onları birçok uygulamada kullanışlı hale getirir. Ayrıca, basit kodlama ve çözme algoritmalarına sahiptirler, bu da onları daha verimli hale getirir. Benzer şekilde, RM kodlarının otomorfizmaları, kodların hata düzeltme özelliklerini anlama ve karakterize etme konusunda kritik bir rol oynar. Bu nedenle, kodun simetrilerini inceleyerek, hataların kodlanmış bilgiyi nasıl etkilediği ve kodun bu hataları nasıl düzeltebileceği veya tespit edebileceği konusunda öngörüler kazanılabilir. Reed-Muller kodları, örneğin, $\mathbb{F}_2$ üzerinde tanımlanan ikili RM kodları ve $\mathbb{F}_p$ üzerinde tanımlanan p-ary RM kodları gibi çeşitli varyasyonlara sahiptir. RM kodları, özellikle flaş bellek ve kablosuz iletişim gibi çeşitli uygulamalarda yaygın olarak kullanılan bir türdür. Ayrıca, RM kodları birçok şekilde genelleştirilmiştir, örneğin, bir keyfi sonlu cisimde tanımlanan genelleştirilmiş Reed-Muller kodları (kısaca, GRM) [3], ve belirli bir simetri özelliğine sahip olan simetrik Reed-Muller kodları (kısaca, SRM) [5] vardır.

Bu tezde, SRM kodlarının otomorfizma grubunun, lineer dönüşümlerden oluşan bir alt grubunu inceleyeceğiz. Kodlama teorisi literatüründe, klasik kodların çoğunluğunun büyük bir otomorfizm grubu bulunur ve bu genellikle bir doğrusal gruba bağlıdır. Örneğin, GRM kodları, $\mathbb{F}_q$ üzerinde derecesi m olan genel afin grup $\mathrm{GA}(m, q)$ altında değişmezdir (bkz. [3]). Bir kodun afin grup altında değişmez olması için bir gereklilik, Kasami ve arkadaşları tarafından [6]'de verilmiştir. Delsarte, bu kodları belirli doğrusal gruplar altında değişmez olarak tanımlar [7]. $\mathrm{GA}(m, q)$ altında, sadece p-ary RM kodlarının değişmez olduğunu gösterir, ancak tam otomorfizma grubunun belirlenmesi sorunu henüz çözümlenmemiştir. Bir kodun tam otomorfizma grubunun belirlenmesi, sıklıkla basit grupların kategorizasyonuyla ilişkilendirilen zor bir konudur. Dür, Reed-Solomon kodlarının ve uzantılarının otomorfizma grupları üzerine araştırmaları ile tanınır (bkz.

[8]). Berger, GRM kodlarının tam otomorfizma gruplarını [9]'da kanıtlamıştır. 1996'da, Berger $\mathrm{GA}(m,q)$'nın herhangi bir afin değişmez kodun permütasyon grubunu içerdiğini göstermiştir (bkz. [10]). Projektif ve homojen RM kodlarının tam otomorfizma grupları [11]'de bulunmuştur. Bu gruplar, sırasıyla projektif lineer grup ve genel lineer grup ile ilişkilidir.

SRM kodları, $\mathbb{F}_q$ üzerindeki ikili polinomlar kullanılarak ilk kez [5]'da tanıtılmıştır. İki değişkenli SRM kodlarının yerel düzeltilebilirliği [12]'de tartışılmıştır. Adı geçen çalışmada, simetrik yapıların avantajları özetlenmiştir ve SRM kodlarının yerel düzeltilebilirlik açısından GRM kodlarına üstünlüğünü sezgisel olarak ele alınmıştır. SRM kodlarının ve GRM kodlarının aynı sonlu cisimdeki hata tolerans oranları ve kod oranları gösterilmiştir. Daha sonra, yerel olarak düzeltilebilir kodların bir sınıfının, çok değişkenli SRM kodları $\mathrm{SRM}_q[n,r]$'den oluştuğu belirlenmiştir. Ardından, $\mathrm{SRM}_q[n,r]$'nin dualitesi sunulmuştur. Ancak, bu kodları değişmez bırakan dönüşümler, özellikle de otomorfizma grubuna ait olanlar, henüz incelenmemiştir. Araştırmadaki bu boşluktan ilham alarak, odağımızı bu sorunu ele almaya yönlendirdik. Bu çalışmada, simetrik Reed-Muller kodlarının otomorfizma grubunun lineer dönüşümleri incelenmiş ve bu lineer dönüşümler kümesinin, delinmiş Reed-Muller kodlarının otomorfizma grubu olan genel lineer grubun bir alt grubunu oluşturduğu gösterilmiştir. Bazı özel durumlar için, bu alt gruptaki tüm otomorfizmaları açıkça belirlemek için bir yöntem sunulmuştur.

m ve r pozitif tamsayı olmak üzere, r dereceli, q^m uzunluklu genelleştirilmiş Reed-Muller kodu

$$GRM_q(m,r) = \{(f(\alpha))_{\alpha\in\mathbb{F}_q^m} \mid f \in \mathbb{F}_q[x_1,\ldots,x_m], deg(f) \leq r\}.$$

kümesi olarak tanımlanır. qw"qq

$$Aut(GRM_q(m,r)) = \mathrm{GA}(m,q)$$

SRM kodları, kod kelimelerinin belirli simetri özelliklerini sağlayan GRM kodlarının bir alt kod ailesi olarak düşünülebilir (bkz. [5]). SRM'nin tanımını yapmak için önce bazı

notasyonları vermek gerekir :

$x = (x_1, x_2, \ldots, x_n)$, $i = (i_1, i_2, \ldots, i_n)$ olmak üzere

$$E_q(n,r) := \left\{ f(x_1, x_2, \ldots, x_n) = \sum_{\substack{0 \le i_1 < i_2 < \ldots < i_n \le q-1 \\ i_1 + i_2 + \ldots + i_n \le r}} a_{i_1 i_2 \ldots i_n} \det(x,i) \mid a_{i_1 i_2 \ldots i_n} \in \mathbb{F}_q \right\}$$

tanımlansın.

$$\Delta = \{(a_1, a_2, \ldots, a_n) \in \mathbb{F}_q^n \mid a_j \neq a_i, 1 \le j < i \le n\}$$

kümesi üzerinde $\sim$ denklik bağıntısı aşağıdaki gibi tanımlansın:

$$c \sim d \iff c, d \in \mathbb{F}_q^n \text{ olmak üzere } \sigma(c) = d \text{ koşulunu sağlayan bir } \sigma \in S_n \text{ vardır.}$$

$[\alpha]$, α'nın denklik sınıfı olmak üzere, $\Omega_q(n) := \Delta / \sim = \{[\alpha] \mid \alpha \in \Delta\}$ kümesini tanımlayalım.

n ve r pozitif tamsayılar olsun. $\mathbb{F}_q$ üzerinde, r dereceli bir SRM kodu

$$\mathrm{SRM}_q[n,r] = \{(f(\alpha))_{[\alpha] \in \Omega_q(n)} \mid f \in E_q(n,r)\}$$

kümesi ile tanımlıdır. (bkz. [12]).

$n = 2$ ve $n = 3$ için q bir asal olmak üzere $\mathbb{F}_q$ üzerinde bir SRM kodu, genel afin grubunun bir alt grubunda afin değişmezdir. İlk olarak, $n = 2$ durumu için $\mathrm{SRM}_q[2,r]$ kodunun afin değişmez olduğu grubu inceledik.

$E_q(2,r) = \left\{ \sum_{\substack{0 \le i < j \le q-1 \\ i+j \le r}} a_{ij} \begin{vmatrix} {x_1}^i & {x_2}^i \\ {x_1}^j & {x_2}^j \end{vmatrix} \;\middle|\; a_{ij} \in \mathbb{F}_q \right\}$ ele alalım. $f(x_1, x_2) = \begin{vmatrix} {x_1}^i & {x_2}^i \\ {x_1}^j & {x_2}^j \end{vmatrix} \in E_q(2,r)$ ve $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ olsun. Bir matris kullanılarak elde edilen dönüşüme T_f diyelim.

Bu durumda $T_f(A)$ dönüşüm matrisini aşağıdaki gibi elde ederiz :

$$T_f(A) = \begin{vmatrix} (ax_1 + bx_2)^i & (cx_1 + dx_2)^i \\ (ax_1 + bx_2)^j & (cx_1 + dx_2)^j \end{vmatrix}$$
$$= (ax_1 + bx_2)^i \cdot (cx_1 + dx_2)^j - (ax_1 + bx_2)^j \cdot (cx_1 + dx_2)^i.$$

$E_q(2, r)$ kümesini değişmez bırakan dönüşümü bulmak için $a, b, c, d \in \mathbb{F}_q$ katsayılarını bulmak gerekir. Bunun için yardımcı bilgilere ihtiyaç duyulur. Öncelikle bu yardımcı bilgileri verelim ardından bu katsayıları elde etmek için bu bigilerden istenilen kümeye ulaşalım.

$f \in E_q(2, r)$ ve $\alpha, \beta \in \mathbb{F}_q$ ve t pozitif bir tamsayı olsun. Bu durumda

$$(\alpha x_1^2 + \beta x_1 x_2 + \alpha x_2^2)^t \cdot f \in E_q(2, r)$$

olur. Ayrıca, yine aynı a ve b katsayıları için

$$\begin{vmatrix} 1 & 1 \\ (ax_1 + bx_2)^t & (bx_1 + ax_2)^t \end{vmatrix} \in E_q(2, r)$$

elde edilir. En genel anlamda bir afin dönüşüm altındaki $E_q(2, r)$ kümesi, $0 \leq i < j$ için, yine $E_q(2, r)$'dedir, yani

$$\begin{vmatrix} (ax_1 + bx_2)^i & (bx_1 + ax_2)^i \\ (ax_1 + bx_2)^j & (bx_1 + ax_2)^j \end{vmatrix} \in E_q(2, r).$$

elde edilir.

$\mathrm{SRM}_q[2, r]$ kodunun değişmez olduğu grup, $\mathrm{GL}(2, q)$'nin bir alt grubuna izomorftur.

$$M = \left\{ \begin{bmatrix} a & b \\ b & a \end{bmatrix} \mid a, b \in \mathbb{F}_q,\ a \neq \pm b \right\} \subset \mathrm{GL}(2, q).$$

kümesi tanımlansın. $q = r = 3$ dışında $q \geq r > 2$ için, $\mathrm{SRM}_q[2, r]$, kod ailesi M'ye izomorfik bir alt grup kapsar, yani $\mathrm{SRM}_q[2, r]$ kodu M dönüşümü altında değişmezdir.

$q = r = 3$ için $\mathrm{SRM}_3[2, 3]$ kodunun otomorfizma grubu, uzunluğu 3 olan tüm vektörleri kapsar, yani $\mathrm{Aut}(\mathrm{SRM}_3[2, 3]) = S_3$. , $n = 2$ için kullandığımız teknik $n = 3$ için çok kullanışlı değildir ve bu yöntem uzun denklem sistemlerinin çözülmesini zorunlu kılar. Bu nedenle $n = 3$ durumunda, $\mathrm{SRM}_q[3, r]$ kodunun afin değişmez grubunu bulmak için $n = 2$ durumundan farklı bir teknik kullandık. Böylelikle daha verimli ve kullanışlı bir çözüm yöntemi elde etmiş olduk.

q tek olmak üzere, derecesi r'den küçük ya da r'ye eşit

$$f(x_{\pi(1)}, x_{\pi(2)}, x_{\pi(3)}) = \begin{cases} -f(x_1, x_2, x_3), & \pi \text{ tek permütasyon,} \\ f(x_1, x_2, x_3), & \pi \text{ çift permütasyon.} \end{cases}$$

koşulunu sağlayan bir $f(x_1, x_2, x_3) \in \mathbb{F}_q[x_1, x_2, x_3]$ fonksiyonu ve $\pi \in S_3$ verilsin. Bu durumda $f(x_1, x_2, x_3) \in E_q(3, r)$ elde edilir.

Şimdi

$$K = \left\{ P * \begin{bmatrix} b & a & a \\ a & b & a \\ a & a & b \end{bmatrix}, \middle| \ P \in \mathcal{P}_3, a, b \in \mathbb{F}_q, a \neq b,\ b \neq -2a \right\} \subset \mathrm{GL}(3, q)$$

kümesini ele alalım. $A \in K$ ve $f(x_1, x_2, x_3) \in E_q(3, r)$ olsun. g fonksiyonu $g(x_1, x_2, x_3) = T_f(A)$ şeklinde tanımlansın. Bu durumda, $g \in E_q(3, r)$ olur.

Bu yardımcı bilgiler sayesinde, $\mathrm{SRM}_q[3, r]$ kodunu değişmez bırakan lineer dönüşümleri belirledik. $q \geq r > 3$ olmak üzere, $\mathrm{SRM}_q[3, r]$ kodunun otomorfizma grubu, K

kümesine izomorf bir alt grup kapsar, yani $\mathrm{SRM}_q[3, r]$, K'dan elde edilen dönüşümler altında değişmezdir.

Keywords: Reed-Muller kodlar, Simetrik Reed-Muller kodlar, Afin değişmezlik, Otomorfizma grupları.

ACKNOWLEDGEMENTS

First and foremost, I owe a great debt of gratitude to my advisor, Prof. Dr. Pınar Aydoğdu. I consider myself incredibly fortunate to have spent my doctoral journey working with her. She has always provided me with invaluable advice and supported me even on my worst days. Her technical insight, contagious enthusiasm for research, and unique character have inspired and influenced me over the years.

I also want to express my gratitude to my master's advisor, Oğuz Yayla, for guiding me into the academic journey. I extend my thanks for his great guidance and effective recommendations that motivated me throughout my doctoral years.

Furthermore, I am immensely thankful to Dr. Talha Arıkan. This work would not have been possible without his exceptional support. He has continuously supported me throughout this process, patiently assisting me whenever needed. I cannot express enough gratitude for our lengthy discussions, brainstormings and debates during our meetings. His unwavering support, positive attitude, and problem-solving approach, even during my toughest times, enabled me to complete this thesis.

I would like to thank Prof. Dr. Zülfikar Saygı and Prof. Dr. Mesut Şahin for their assistance and encouragement.

Additionally, I am grateful to Damla Acar and Yağmur Çakıroğlu for their close friendship and enjoyable moments during my doctoral journey. Their support during my disappointments, moments of despair, and tough times has been invaluable in helping me overcome these challenges.

I wish to express my deep appreciation and gratitude to Dr. Taner Dursun, my director at TÜBİTAK Bilgem Blockchain Research Laboratory, and my colleagues for providing a pleasant atmosphere, motivation, and a comfortable working environment throughout my professional life.

I extend my thanks to the Scientific and Technological Research Council of Turkey (TÜBİTAK) for the financial support through the 2211/A Graduate Scholarship and to the Council of Higher Education (YÖK) for the 100/2000 PhD Scholarship.

I offer my deepest gratitude to my family for their endless support. Without them, I could not have achieved anything. And Dad, although no longer with us, continues to inspire me by his dedication and devotion to what he believes in. Thank you both for everything and making me who I am right now.

I owe a debt of gratitude to my beloved husband, Tekin, for making me feel peaceful, comfortable, and safe. With his love and support, I was able to complete this thesis.

Lastly, during the thesis writing process, I experienced an indescribable joy and surprise with the wonderful news of my dear son. I am so happy to be completing this work with you. I am filled with indescribable emotions and feel very emotional. I am so glad you came into my life. I believe that with your energy, life will become much more beautiful.

CONTENTS

Page

ABBREVIATIONS

$\mathbb{N}$	:	**N**atural Numbers
$\mathbb{Z}$	:	Integers
1_G	:	**I**dentity element of a group
$\mathbb{F}_q$	:	**Finite** field of order q
S_n	:	The Symmetric group of order n
GL_n	:	General Linear Group of order n
GA_n	:	General Affine Group of order n
$\mathbb{F}_q^n$	:	n dimensional $\mathbb{F}_q$-vector space
$\mathbb{F}_q[x_1, \ldots, x_n]$	:	The polynomial ring with n variables
w_t	:	The set of codes with Hamming weight t
$d(\cdot, \cdot)$	:	The Hamming Distance
I_n	:	The Identity matrix of order n
H^T	:	The **T**ranspose of matrix H
$\mathcal{P}_n$	:	The set of $n \times n$ permutation matrix.
M	:	The **M**onomial Matrix
$\deg(\mathrm{f})$	:	The **d**egree of f function
π	:	Any permutation in S_n
$\sim$	:	Equivalence Relation
RM	:	**R**eed **M**uller
SRM	:	**S**ymmetric **R**eed **M**uller

1. INTRODUCTION

Working with automorphisms of the codes is important for various reasons in the field of coding theory. The study of automorphisms can contribute to developing efficient decoding algorithms. By exploiting the symmetries within a code, it may be possible to design more computationally efficient algorithms to correct errors [5]. Thus, automorphisms of the codes play a crucial role in understanding and characterizing the error-correction properties of the codes. Therefore, by studying the symmetries of a code, one can gain insights into how errors affect the encoded information and how the code can be designed to correct or detect these errors.

Reed-Muller codes (RM codes, for short) are a family of error-correcting codes that were first introduced by Irving S. Reed and Gustave Solomon Muller in 1954 (see [13–15]). A large number of RM codes variations and generalizations were introduced in the literature, for instance, see [2–4]. They have simple encoding and decoding algorithms which make them useful and efficient to implement. There are some variants of RM codes, such as the binary RM codes defined on the prime field $\mathbb{F}_2$ and the p-ary RM codes defined on the prime field $\mathbb{F}_p$, where p is a prime number. For more details on RM codes and variants we refer the reader to [1, 2, 6, 16]. Furthermore, RM codes have been generalized in many ways, such as the generalized Reed-Muller codes (GRM, for short) which are defined over an arbitrary finite field, and the symmetric Reed-Muller codes (SRM, for short) which have a certain symmetry property (see [3] and [5]).

In coding theory, the majority of classical codes have a sizable automorphism group that is connected to a linear group. For example, GRM codes are invariant under the general affine group $\mathrm{GA}(m, q)$ of degree m over the finite field $\mathbb{F}_q$ (see [3]). A requirement for a code to be invariant under the affine group is provided by Kasami et al. in [6]. Delsarte describes the codes which are invariant under certain linear groups in [7]. He shows that only p-ary RM codes can be invariant under $\mathrm{GA}(m, q)$. Nonetheless, the issue of fully determining the automorphism group of affine invariant codes has not yet been resolved. Determining the

complete automorphism group of a code is a challenging topic that is frequently connected to the categorization of simple groups. Dür is credited with the research of the automorphism groups of Reed-Solomon codes and their extensions (see [8]). In [9], Berger proved the complete automorphism groups of GRM codes. In 1996, Berger demonstrated that $\mathrm{GA}(m, q)$ contains the permutation group of any affine-invariant code (see [17]) and then he showed how to create a formal expression for every affine-invariant code's permutation group in [10]. The complete automorphism groups of the projective and homogeneous RM codes can be found in [11]. These groups are associated with the projective linear group and the general linear group, respectively. In this work, we aim to investigate a subgroup of the automorphism group of SRM codes whose elements are linear maps.

SRM codes are first introduced by using bivariate polynomials over $\mathbb{F}_q$ in [5]. The local correctability of the bivariate SRM codes is discussed in [12]. The authors begin by outlining the advantages of the symmetric structure and offering intuitions to indicate the superiority of SRM codes over GRM codes in terms of local correctability. The tolerance of error ratios of SRM codes and GRM codes over the same finite field, as well as their code rate, are demonstrated. They establish that a class of locally-correctable codes that is composed of multivariable SRM codes $\mathrm{SRM}_q[n, r]$ in [12]. Furthermore, the dual of $\mathrm{SRM}_q[n, r]$ is also presented. However, transformations preserving these codes, specifically belonging to the automorphism group, have not been studied yet. Taking inspiration from this gap in this research, we have directed our focus toward addressing this problem.

Unless otherwise stated, throughout this dissertation, we will work on the field $\mathbb{F}_q$, where q is a prime number. Main object of the presented work is to determine the set of linear transformations that leaves SRM codes invariant.

In Section 2, some fundamental concepts that will be used throughout this dissertation are provided. In Section 3, the definitions and notations used in constructing SRM codes are given. The original work of this dissertation is presented in Section 4. In this section, we investigate linear transformations that leave SRM codes invariant. The set of these linear transformations forms a subgroup of the general linear group. We indicate a relationship

between the transformations that remain the SRM codes invariant for $n = 2$ and those for $n = 3$. Therefore, we believe that we can form this group for a generic n, despite the challenge of determining all transformations. For any given n, we anticipate the group that leaves the SRM codes invariant, akin to previous cases. This group involving affine transformations is a subgroup of the general linear group $\mathrm{GA}(n, q)$. However, determining whether this group forms a complete set which leaves the SRM codes invariant remains an open problem for the future. Therefore, in Section 5, we state a conjecture on this problem.

2. PRELIMINARIES

In this section, we fix some main notations, and recall some basic definitions and results that will used throughout the dissertation.

2.1 Linear Codes

For many years, coding theory has been utilized to facilitate data transmission between a sender and a receiver. Often, the channel through which messages pass between these two parties is susceptible to errors. In such cases, the receiver may request the sender to resend the message. However, this method proves ineffective as it significantly increases traffic. Instead, by adding redundant information, known as redundant bits, to the message, the sender enables the receiver to accurately check and correct the message. For this process to succeed, the error rate must be within certain constraints. The set of all messages encoded in this manner forms is what we call error correction codes.

The idea here is for the sender to encode fixed length data of length k into codewords of length n, and then transmit these codewords to the receiver. It is necessary to consider the linear algebraic structure on the codes. For linear codes, the alphabet is always a finite field. Construction, encoding and decoding of linear codes are generally easier compared to non-linear codes.

This chapter consists of important concepts from coding theory and field theory that are essential to the reminder of the thesis. Definitions related to coding theory, including binary codes, generator and parity-check matrices, bounds on code parameters, methods for constructing new codes from existing ones, and examples of linear codes, are provided. This section utilizes the book *Fundamentals of Error-Correcting Codes* by Huffman and Pless [1] to provide the essential definitions.

Linear codes are defined over alphabets which are finite fields. We denote by $\mathbb{F}_q$ the finite field with q elements, where q is a prime power.

Definition 2.1.1. [1] *A linear code* C of length n and dimension k over $\mathbb{F}_q$ is a k-dimensional subspace of the vector space $\mathbb{F}_q^n$, where $\mathbb{F}_q$ is the finite field with q elements. Such a code is referred as $[n, k]$ *code* over $\mathbb{F}_q$. All elements of a code are called *codewords*.

Notice that an $[n, k]$ linear code C over $\mathbb{F}_q$ has q^k codewords.

Definition 2.1.2. [18] The *rate* of an $[n, k]$ code is defined as $\frac{k}{n}$.

The rate is a quantity which shows that how much information is being transmitted per codeword. It is associated to the redundancy $r = n - k$, the number of parity symbols in a codeword.

Every codeword in an $[n, k]$ linear code can be represented as a linear combination of the basis vectors. We can present these vectors in a matrix format, where they constitute the columns of an $n \times k$ matrix.

Definition 2.1.3. [1] Let C be an $[n, k]$ linear code over $\mathbb{F}_q$. A matrix $G \in {\mathbb{F}_q}^{n\times k}$ is called a *generator matrix for C* if its k columns span C.

Encoding a message $m \in {\mathbb{F}_q}^k$ is a transformation that sends the message to the codeword $G.m \in C$. In other words, the linear transformation $m \to G.m$ is an encoding map

$$T : \mathbb{F}_q^k \to \mathbb{F}_q^n.$$

Definition 2.1.4. [1] A parity check matrix H of C is an $(n - k) \times n$ matrix over $\mathbb{F}_q$ with rank $n - k$. It is defined by

$$C = \{x \in \mathbb{F}_q^n \mid Hx^T = 0\}.$$

Note that the rows of H will also be independent. In general, there are also several possible parity check matrices for C.

Example 2.1.5. *The generator matrix and corresponding parity check matrix of a* $[5, 3]$ *linear code* C *are* $G = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}$ *and* $H = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix}$*, respectively.*

The standard form of an $[n, k]$ linear code is $(I_k|A)$, where I_k is a $k \times k$ identity matrix, and A is a $k \times (n - k)$ matrix. The corresponding standard form of the parity-check matrix is $(-A^T|I_{n-k})$.

The generator matrix of an $[n, k]$ linear code is a matrix whose rows are linearly independent and generate the code. The rows of the H parity-check matrix are also linearly independent and generate a code known as the dual of C, denoted by $C^\perp$. The code $C^\perp$ is an $[n, n - k]$ code, and the H matrix serves as its generator matrix.

Definition 2.1.6. [18] Let C be a linear $[n, k]$ code. The *dual code* of $C^\perp$, denoted by

$$C^\perp := \{a \in \mathbb{F}_q^n \mid \langle a, b\rangle = 0 \text{ for all } b \in C\},$$

where $\langle a, b\rangle$ is the standard inner product.

It follows from the definition that $C^\perp$ consists of all those vectors $a \in \mathbb{F}_q^n$ such that $a.b^T = 0$ for all $b \in C$, where b^T denotes the transpose of the vector b. This means that the codewords of $C^\perp$ are orthogonal to C.

For $c_1, c_2 \in \mathbb{F}_q^n$, the *(Hamming) distance* $d(c_1, c_2)$ is defined to be the number of coordinates in that c_1 and c_2 differ.

Definition 2.1.7. [1] The minimum distance d of a linear C is the smallest distance between distinct codewords, i.e.,

$$d = d(C) = \min\{d(x, y) : x, y \in C, x \neq y\}.$$

In order to determine the minimum distance of a linear code, it suffices to compute the distance from all zero codewords to their closest codewords. For instance, let v be a codeword, and let $v_1, \ldots, v_n$ be codewords at a distance of d from this codeword. Then, the difference $v - v$ yields all zero code words, and the differences $v_1 - v, \ldots, v_n - v$ provide codewords at a distance d from all zero code words.

A linear code C of length n, dimension k, and minimum distance d over the finite field $\mathbb{F}_q$ is referred as an $[n, k, d]$ code over $\mathbb{F}_q$. Additionally these codes have also some significant features that we would like to incorporate. We aim to measure how much a code differs from another. In coding theory, the parity check matrix of a linear code provides information about the minimum distance of that code.

Theorem 2.1.8. *[1] The minimum distance d of a linear code $[n, k, d]$ is the minimum number of linearly dependent columns of a parity check matrix H of the linear code. Any $d - 1$ columns of H is linearly independent.*

Proof. Given that the parity check matrix H has $n - 1$ columns such as $t_0, t_1, \ldots, t_{n-1}$. For any codeword c, $cH^T = 0$. Then we can write this equality :

$$c.h_0 + c.h_1 + \ldots + c.h_{n-1} = 0.$$

From theorem 2.1.11, any codeword c holds the equality $d = wt(c)$. Let the nonzero coordinates of c be $1, 2, \ldots, d$. So

$$c_1.h_1 + c_2.h_2 + \ldots + c_d.h_d = 0.$$

□

The minimum distance between codewords is a crucial invariant for a code. The minimum distance of a code C is the smallest distance between distinct codewords and is significant in determining the error-correcting capability of the code C. A larger minimum distance allows

for the correction of more errors. The Hamming weight $wt(v)$ of a vector v in F_q^n is the number of non-zero coordinates of x.

Definition 2.1.9. [1] The *Hamming weight* wt of a $v \in \mathbb{F}_q^n$ is defined to be the number of nonzero elements in v.

$$wt(v) = d(v, 0).$$

Similarly, the *weight* of a C code is described as follows:

$$wt(C) = min\{d(v, 0) : 0 \neq v \in C\}.$$

Example 2.1.10. *The linear code C = {0000, 1000, 0100, 1100} over* $\mathbb{F}_2^4$*, then we have:*

$$wt(1000) = 1,$$

$$wt(0100) = 1,$$

$$wt(1100) = 2.$$

Hence $d(C) = 1$.

Theorem 2.1.11. *[1] The minimum distance of a linear code C is the minimum weight of any nonzero codeword.*

One of the primary objectives in coding theory is to achieve high error-correction capability, characterized by a wide relative distance and a high information rate. However, there exist certain constraining properties. For instance, the singleton bound is given as follows:

Theorem 2.1.12. *[1] Let C be an $[n, k, d]$-code. Then the inequality $d \leq n - k + 1$ is satisfied.*

This inequality arises from the properties that generator matrices must satisfy. Specifically, the rank of a code's parity-check matrix is $n - k$. Therefore, this matrix has $n - k + 1$

linearly dependent columns, meaning that any $n - k + 1$ columns are linearly dependent. Consequently, the minimum distance is at most $n - k + 1$. For linear codes, the Singleton bound can also be derived by examining the systematic generator matrix of the code. Each row of this matrix has a Hamming weight of at most $n - k + 1$.

2.2 Reed Solomon Codes

Reed-Solomon codes are error correction codes created by Irving S. Reed and Gustave Solomon in 1960 (see [19]). These codes transform data blocks defined over finite fields into codewords. Reed-Solomon codes have the ability to detect and correct multiple error symbols. By adding $n - k$ parity symbols to the data, they can detect up to $n - k$ errors and correct half of them.

There are multiple methods for constructing Reed-Solomon codes. Each Reed-Solomon codeword is a sequence of values of polynomials of degree less than k. The message symbols of a Reed-Solomon codeword consist of the coefficients of a polynomial of degree less than k over $\mathbb{F}_q$.

Reed-Solomon codes $RS(n, k)$ are codes defined over $\mathbb{F}_q$ with length n and dimension k (i.e. message length). A message $m = (m_0, m_1, \ldots, m_{k-1})$ is embedded into a polynomial $p(x) = m_0 + m_1 x + \ldots + m_{k-1}x^{k-1}$ of degree at most $k - 1$. The coefficients of the polynomial $m_{k-1}, m_{k-2}, \ldots, m_{k-j}$ may be zero for some $j \leq k$. The message is encoded as $c = (p(z_1), p(z_2), \ldots, p(z_n))$ by evaluating the polynomial at distinct points $z_1, z_2, \ldots, z_n \in \mathbb{F}_q$. After the encoding process is completed in this way, the resulting codeword c is transmitted.

$$C = \{(t(z_1), p(z_2), \ldots, p(z^{n-1})) \mid p \in \mathbb{F}_q[x], der(p) \leq k\}.$$

Once the sender produces such a codeword, the receiver obtains the values $s = (s_1, s_2, \ldots, s_n)$, knowing that these points are evaluated on the set $z_1, z_2, \ldots, z^{n-1}$. The

receiver knows that at least k of the points s_i are of the form $s_i = p(z^i)$, and aims to reconstruct the message by attempting to construct the polynomial p.

2.3 Reed-Muller Codes

Reed-Muller codes are linear block codes created by Reed and Muller in the 1950s (see for insatnce, [13, 14]). Initially, binary Reed-Muller codes were developed, and later they were extended to general cases. Although the block length of Reed-Muller codes is lower than that of BCH [20, 21] codes, they are useful error correction codes because they can be easily encoded and decoded. They form the simplest class of geometric codes and generalize Reed-Solomon and Walsh-Hadamard codes. Reed-Muller codes can be constructed in multiple ways. The initial definition of Reed-Muller codes was created using Boolean functions. Since we will consider this definition throughout the dissertation, we will recall the construction using Boolean functions. In this section, the necessary definitions have been derived with reference to the book by MacWilliams and Sloane [2].

We start with recalling the definition of a Boolean polynomial. Consider the polynomial ring $\mathbb{F}_2[x_0, x_1, \ldots, x_{m-1}]$ over $\mathbb{F}_2$ with m variables.

Definition 2.3.1. [2] *A Boolean monomial* t is an element from the polynomial ring $\mathbb{F}_2[x_0, x_1, \ldots, x_{m-1}]$ of the form $t = x_0^{r_0} x_1^{r_1} \ldots x_{m-1}^{r_{m-1}}$, where $r_i \in \mathbb{N}$ for $i = 0, \ldots, m-1$. *A Boolean polynomial* over $\mathbb{F}_2$ consists of a linear combination of Boolean monomials.

Now consider the transformation $\rho : \mathbb{F}_2[x_0, x_1, \ldots, x_{m-1}] \to \mathbb{F}_2^{2^m}$ defined by

$$
\begin{aligned}
\rho(0) &= 00\ldots0 = 0^{2^m} \\
\rho(1) &= 11\ldots1 = 1^{2^m} \\
\rho(x_0) &= 1\ldots10\ldots0 = 1^{2^{m-1}}0^{2^{m-1}} \\
\rho(x_1) &= 1\ldots10\ldots01\ldots10\ldots0 = 1^{2^{m-2}}0^{2^{m-2}}1^{2^{m-2}}0^{2^{m-2}} \\
&\vdots \\
\rho(x_i) &= 1\ldots10\ldots01\ldots10\ldots0 = 1^{2^{m-i-1}}0^{2^{m-i-1}}\ldots1^{2^{m-i-1}}0^{2^{m-i-1}}
\end{aligned}
$$

The transformation ρ possesses a homomorphic property with respect to the addition and the multiplication. Let t be a product of k variables, i.e., $t = x_0x_1 \dots x_k$. Then $\rho(t) = \rho(x_0)\rho(x_1) \dots \rho(x_k)$. Similarly, if h be a sum of k variables, i.e., $h = x_0 + x_1 + \ldots + x_k$, then $\rho(h) = \rho(x_0) + \rho(x_1) + \ldots + \rho(x_k)$. Hence, $\rho : \mathbb{F}_2[x_0, x_1, \dots, x_{m-1}] \to \mathbb{F}_2^{2^m}$ is a ring isomorphism [2].

Now we will recall the definition of a Reed-Muller code.

Definition 2.3.2. [2] *The d-th order Reed-Muller code (RM, for short)* $R(d, m)$ is defined to be the set of all polynomials of degree at most d in the $\mathbb{F}_q[x_0, x_1, \dots, x_{m-1}]$. The set consisting of these monomials is provided below

$$\{\prod_{i \in K} x_i \mid K \subseteq \{1, 2, \dots, m\}, |K| \leq d\}.$$

The ρ isomorphism can be viewed as a subset of $\mathbb{F}_2^{2^m}$, specifically as a binary linear code of length 2^m.

Note that the $R(d, m)$ code can be regarded as a subgroup of $\mathbb{F}_2^{2^m}$ under the ρ transformation. Binary RM codes can be defined recursively. The code $R(0, m)$ is a trivial code. The 0-th order RM code is defined to be the repetition code of length 2^m with basis $\{1\}$. Another trivial code is $R(m, m)$ which is the m-th order RM code $\mathbb{F}_2^{2^m}$.

Theorem 2.3.3. *[2] For $1 \leq d < m$,* the d-th order $R(d, m)$ of length 2^m *is defined recursively as* $R(d, m) = \{(u, u + v) \mid u \in R(d, m - 1), v \in R(d - 1, m - 1)\}$.

Assume that $G(0, m)$ is a generator matrix for $R(1, m)$. Then a generator matrix of the RM code $R(1, m + 1)$ is

$$G(1, m + 1) = \begin{bmatrix} G(0, m) & G(0, m) \\ 0 & 1 \end{bmatrix}.$$

The next result generalizes this idea.

Theorem 2.3.4. *[2] Let $G(d,m)$ be a generator matrix of $R(d,m)$. Then $R(d,m)$ has a generator matrix*

$$G(d+1,m+1) = \begin{bmatrix} G(d+1,m) & G(d+1,m) \\ 0 & G(d,m) \end{bmatrix}.$$

Reed-Muller codes can also be constructed by using multivariate polynomials. The message bits of RM codes, similar to Reed-Solomon codes, form the coefficients of a multivariate polynomial. This polynomial is evaluated in a vector to obtain the RM codeword. Hence, an alternative definition of the d-th order RM code can be given in this context.

Definition 2.3.5. [2, p. 373] The *r^{th} order binary RM code $R(r,m)$ of length $n = 2^m$* for $0 \leq r \leq m$, is the set of all vectors f, where $f(x_1,\ldots,x_m)$ is a binary multivariable polynomial of degree at most r, i.e.,

$$R(r,m) = \big\{(f(\alpha))_{\alpha\in\mathbb{F}_2^m} \mid f \in \mathbb{F}_2[x_1,\ldots,x_m],\ deg(f) \leq r\big\}.$$

Theorem 2.3.6. *[2] Assume that d and m are integers such that $0 \leq d \leq m$. If $R(d,m)$ is the d-th order RM code then the following statements are satisfied:*

1) $R(k,m) \subseteq R(l,m)$ *holds if* $0 \leq k \leq l \leq m$.

2) $R(d,m)$ *has the dimension* $\binom{m}{0} + \binom{m}{1} + \cdots + \binom{m}{d}$.

3) $R(d,m)$ *has minimum distance* 2^{m-d}.

4) *The dual of* $R(d,m)$ *is the dual code to* $R(m-d-1,m)$ *for* $0 \leq d \leq m-1$.

2.4 Equivalence of Linear Codes

Let $\mathbb{F}_q$ be the finite field of order $q = p^m$ for a prime p and $\mathbb{F}_q^n$ be the n-dimensional vector space over $\mathbb{F}_q$. The measure of dissimilarity between two vectors is established by counting the coordinates in which they differ. Recall that a linear $[n,k,d]$-code is a k-dimensional

linear subspace of $\mathbb{F}_q^n$ with the minimum distance d. A generator matrix G of a linear $[n, k]$-code C is any matrix of row rank k, whose rows come from the code C. We will give the definitions of some types of code equivalences.

Definition 2.4.1. [1] Let $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ be a permutation of $\{1, 2, \ldots, n\}$. The $n \times n$ *permutation matrix* P is defined as follows: (i, π_i)-th entries are 1 for $i = 1, 2, \ldots, n$, and 0 elsewhere. $\mathcal{P}_n$ denotes the set of all $n \times n$ permutation matrices.

Definition 2.4.2. [1] Two linear codes C_1 and C_2 are *permutation equivalent* if there is a permutation of coordinates which sends C_1 to C_2. Hence, two linear codes C_1 and C_2 of the same length are permutation equivalent if there exists a permutation matrix $P \in \mathcal{P}_n$ such that

$$G_2 = G_1 P,$$

where G_1 and G_2 are the generator matrices of the codes C_1 and C_2, respectively.

If we work on a finite field other than $\mathbb{F}_2$, then we may need a more general form of the equivalence.

Definition 2.4.3. [1] *A monomial matrix* is a square matrix with exactly one nonzero entry in each row and column. A monomial matrix M can be written either in the form DP or the form PD_1 , where D and $D1$ are diagonal matrices and P is a permutation matrix.

Example 2.4.4. *Consider the following monomial matrix M:*

$$M = \begin{bmatrix} a & 0 & 0 \\ 0 & 0 & b \\ 0 & c & 0 \end{bmatrix}$$

M satisfies the following equations :

$$DP = \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = PD_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} a & 0 & 0 \\ 0 & c & 0 \\ 0 & 0 & b \end{bmatrix}$$

Definition 2.4.5. Two linear codes C_1 and C_2 of the same length over $\mathbb{F}_q$ are said to be *monomially equivalent* when there exists a monomial matrix M such that

$$G_2 = G_1 M,$$

where G_1 and G_2 are the generator matrices of the codes C_1 and C_2, respectively.

Note that monomial equivalence and permutation equivalence coincide for binary codes.

Let γ be a field automorphism of $\mathbb{F}_q$ and $M = DP$ be a monomial matrix over $\mathbb{F}_q$, where P is a permutation matrix and D is a non-singular diagonal matrix over $\mathbb{F}_q$. Applying the map $M\gamma$ to any codeword is described by the following process: Firstly, the i^{th} component of code is multiplied by the i^{th} diagonal entry of D for $i = 1, 2 \ldots, n$. Then the corresponding permutation associated with the permutation matrix P is applied to the codeword. Finally, the automorphism γ is applied to all components.

Now we will recall the most general form of the equivalence of linear codes.

Definition 2.4.6. Two linear codes C_1 and C_2 of the same length over $\mathbb{F}_q$ are said to be *equivalent* when there is a monomial matrix M and a field automorphism γ of F_q such that

$$C_2 = C_1 M\gamma,$$

where $C_1 M\gamma$ is obtained by applying $M\gamma$ to each codeword of C_1.

Note that all equivalence definitions are the same for the binary codes. Furthermore, monomial equivalence and general equivalence coincide for p-ary codes, where p is a prime.

Since we have three types of equivalences, there exist three possible definitions of the automorphism groups of the code families by considering $C_1 = C_2$ in the above definitions.

Now consider a code C of length n over the field $\mathbb{F}_q$. The set of coordinate permutations that map the code C to itself forms a group, called the *permutation automorphism group of* C and

denoted by $\mathrm{PAut(C)}$. Obviously, $\mathrm{PAut(C)}$ is a subgroup of the symmetric group S_n. The set of monomial matrices, by which C is monomially equivalent to itself, forms the group $\mathrm{MAut}(C)$, which is called the *monomial automorphism group of* C. The set of maps of the form $M\gamma$, where M is a monomial matrix and γ is a field automorphism, that map C to itself forms the group $\mathrm{Aut}(C)$, called *automorphism group of* C.

We always have that $\mathrm{PAut(C)} \subseteq \mathrm{MAut}(C) \subseteq \mathrm{Aut(C)}$. If $q = 2$, then $\mathrm{PAut}(C) = \mathrm{MAut}(C) = \mathrm{Aut}(C)$. If q is a prime, then $\mathrm{MAut}(C) = \mathrm{Aut}(C)$ (see [1, p. 26]).

All these definitions and conclusions are well-known in the literature and can be found in any basic coding theory book, for example in [1].

2.5 The Automorphisms of Reed-Muller Codes

To gain a broader perspective on Reed-Muller automorphism groups, we will first examine the automorphism groups of binary RM codes. Since we are working over $\mathbb{F}_2$, $\mathrm{PAut}(C) = \mathrm{MAut}(C) = \mathrm{Aut}(C)$. Therefore, determining the permutation automorphism or monomial automorphism is sufficient to determine the automorphism of RM codes. Subsequently, we will extend the definitions of automorphisms to fields $\mathbb{F}_q$ distinct from $\mathbb{F}_2$.

RM codes can be defined in terms of multivariable polynomials as follows. Let $x = (x_1, \ldots, x_m)$ range over $\mathbb{F}_2^m$. Any function $f(x) = f(x_1, \ldots, x_m)$ which takes the values 0 and 1 is called *a binary multivariable function*. We recall the following definitions.

Definition 2.5.1. [2, p. 377] For $0 \leq r \leq m-1$, a code which is obtained by puncturing (or deleting) the coordinate corresponding to $x_1 = \cdots = x_m = 0$ from all the codewords of $R(r, m)$ is called *the punctured RM code*, and it is denoted by $R(r, m)^*$.

First we will mention the notion of affine invariance which is crucial for our discussion on automorphism groups. Before going further, we need some basic notions.

Let $A = [a_{ij}]$ be an invertible $m \times m$ binary matrix and $\mathbf{b}$ be a binary $m \times 1$ vector. Consider the transformation T from binary m-tuples to binary m-tuples defined by

$$T : \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \mapsto A \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} + \mathbf{b},$$

which permutes binary m-tuples. T can be also considered as a permutation of multivariate polynomials as follows:

$$T_f(A, \mathbf{b}) : f(x_1, \ldots, x_m) \mapsto f\Big(\sum a_{1j}x_j + b_1, \ldots, \sum a_{mj}x_j + b_m\Big). \tag{1}$$

The set of all such transformations formed by T is a group, which is known as the *general affine group* over $\mathbb{F}_2$ and is denoted by $\mathrm{GA}(m, 2)$ (see [2]). It is obvious that if f is a polynomial of degree r, so is $T_f(A, \mathbf{b})$.

As we mentioned above, for a binary code it is known that $\mathrm{PAut(C)} = \mathrm{Aut}(C)$, and hence it is a subgroup of the symmetric group S_n. Following [22], a code C is said to be *affine invariant* if $\mathrm{Aut}(C)$ includes a subgroup that is isomorphic to the affine linear group.

The following example demonstrates that the RM code families are one of the examples of affine invariant codes. This example is important to see the equivalence between transformations applied to variables of the function to evaluate codeword and transformations applied to the codeword itself.

Example 2.5.2. *[22, p. 7]* $R(r, m)$ *codes are affine invariant.*

Proof. Let A be an $m \times m$ invertible matrix over $\mathbb{F}_2$ and $\mathbf{b} \in \mathbb{F}_2^m$. The affine linear transformation $T : x \mapsto Ax+\mathbf{b}$ yields a permutation on the coordinates of the codeword since the codewords of RM codes are evaluation vectors and are indexed by the vectors $x \in \mathbb{F}_2^m$.

Then such a permutation belongs to $\mathrm{Aut}(R(r,m))$. Let c be a codeword in $R(r,m)$. Then there exists a polynomial $f \in \mathbb{F}_2[x_1,\ldots,x_n]$ with $deg(f) \leq r$ such that $c = (f(\alpha))_{\alpha\in\mathbb{F}_{2^m}}$. Since the result of the transformation $(f \circ T)(x)$ is another polynomial of degree less than or equal to r, we have $c' = (f \circ T)(\alpha)_{\alpha\in\mathbb{F}_2^m} \in R(r,m)$. Thus, $R(r,m)$ codes are affine invariant. □

Thus, the general affine group $\mathrm{GA}(m,2)$ permutes the codewords of the r^{th} order $R(r,m)$ and $\mathrm{GA}(m,2) \subset \mathrm{Aut}R(r,m)$ (see [2]). The subgroup of $\mathrm{GA}(m,2)$ consisting of all transformations

$$T : \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \mapsto A \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

(i.e., for which $\mathbf{b} = 0$) is known as the *general linear group* and is denoted by $\mathrm{GL}(m,2)$. We can consider the transformation T described above in the following way, too:

$$T(A) : (x_1,\ldots,x_m) \mapsto \Big(\sum a_{1j}x_1,\ldots,\sum a_{mj}x_j\Big). \tag{2}$$

For the sake of convenience in usage, the function $T_f(A,b)$ will be denoted as $T_f(A)$ when $\mathbf{b} = 0$:

$$T_f(A) : f(x_1,\ldots,x_m) \mapsto f\Big(\sum a_{1j}x_j,\ldots,\sum a_{mj}x_j\Big). \tag{3}$$

Since the transformation $T(A)$ in (2) fixes the zero m-tuple, the group $\mathrm{GL}(m,2)$ permutes the codewords of the punctured RM code $R(r,m)^*$, i.e., $\mathrm{GL}(m,2) \subset \mathrm{Aut}R(r,m)^*$ (see [2, p. 399]).

We know from [2, p. 400] that

$$\mathrm{Aut}(R(r,m)^*) = S_{2^m-1} \quad \text{for} \quad r = 0 \quad \text{and} \quad m-1,$$
$$\mathrm{Aut}(R(r,m)) = S_{2^m} \quad \text{for} \quad r = 0 \quad \text{and} \quad m.$$

Furthermore, the following result is also given in [2, p. 400] which determines the automorphism group of RM codes of r^{th} order with length 2^m over $\mathbb{F}_2$ completely.

Theorem 2.5.3. *[2, p. 400] For* $1 \leq r \leq m-2$*:*

- $\mathrm{Aut}(R(r,m)^*) = \mathrm{GL}(m,2)$,
- $\mathrm{Aut}(R(r,m)) = \mathrm{GA}(m,2)$.

Note that one may easily adopt the definitions of the general affine group and general linear group on $\mathbb{F}_q$ for any prime q. These groups are denoted by $\mathrm{GA}(m,q)$ and $\mathrm{GL}(m,q)$, respectively. These groups are the subjects of the next subsection.

2.6 Generalized Reed-Muller Codes

GRM codes were first introduced by Kasami et al. [23] as a generalization of RM codewords over any finite field. They also established the fundamental parameters of GRM codes. GRM codes are easily constructed and possess rich structural properties, making them applicable across a wide range of fields, such as homomorphic computation [24] and secret sharing [25]. GRM codes are list-decodable [26], locally testable [27], and locally correctable [28]. These features make GRM codes highly valuable in theoretical computer science [12]. The LCC property allows the correction of a codeword by checking only a few random bits, granting these codes advantages in areas such as circuit lower bounds, data storage and transmission, secure multiparty computation and combinatorics.

Due to these valuable properties, GRM codes are currently the subject of extensive research. Delsarte [3], Assmus and Key [29] studied these codes and their relatives in details.

Delsarte [3] has elucidated the connection between the univariate and multivariate versions of GRM codes and established their code distances. Additionally, various variants of GRM codes have been studied, including Projective Reed-Muller codes [30], Grassmann codes [31], and Quantum Reed-Muller codes [32]. Furthermore, many researchers have been considering GRM codes as significant examples of extended cyclic codes.

GRM codes are obtained by constructing the codes over any finite field $\mathbb{F}_q$, where q is a prime power. The following is a formal definition of GRM codes. For more details see [3, 23, 33, 34]

Definition 2.6.1. [3] Let m and r be positive integers. *The GRM code of order r with block length q^m over $\mathbb{F}_q$* is defined by

$$\mathrm{GRM}_q(m,r) = \big\{(f(\alpha))_{\alpha\in\mathbb{F}_q^m} \mid f \in \mathbb{F}_q[x_1,\ldots,x_m],\ \deg(f) \le r\big\}.$$

The GRM code is a linear code that forms a subspace of the vector space $\mathbb{F}_q^n$. Let the dimension of the $GRM(m,r)$ code be denoted by k. The value of k is as follows:

$$\left|\{(l_0,l_1,\ldots,l_{m-1}) \mid l_i \in \mathbb{Z}, 0 \le l_i \le q-1, \sum_{i=1}^{m-1} l_i \le r\}\right|$$

where l_i is the degree of the monomial with index i.

Recall that for $q=2$, the dimension of the $RM_2(m,r)$ code is $\binom{m}{0}+\binom{m}{1}+\ldots+\binom{m}{r}$. For $q>2$, the minimum distance and dimension are given as follows:

Theorem 2.6.2. *[3] The dimension of $RM_q(m,r)$ is*

$$\sum_{t=0}^{r}\sum_{k=0}^{m}(-1)^k\binom{m}{k}\binom{t-kq+m-1}{t-kq}$$

The minimum distance of $RM_q(m,r)$ is $q^{m-a-1}(q-b)$, where $r=a(q-1)+b$, $0\le b\le q-1$.

Example 2.6.3. *The code $GRM_4(2,4)$ is the linear code over with the generator matrix G*

$$\mathbf{G} = \left[\begin{array}{cccccccccccccccc|c}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & x_0 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & x_1 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & x_2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & x_3 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & x_0x_1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & x_0x_2 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & x_1x_2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & x_0x_3 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & x_1x_3 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & x_2x_3
\end{array}\right]$$

Some codewords belonging to the $GRM_4(2,4)$ code by evaluating are given below:

$$\begin{aligned}
c_1 &= 1 + x_0 + 3x_1 + 2x_3 \\
&= [1201120130233023] \\
c_2 &= 1 + 2x_0 + 2x_1 + 3x_3 \\
&= [1331133102200220]
\end{aligned}$$

It is shown in [3] that for $0 \leq r \leq m(q-1)$, the automorphism group of $GRM_q(m,r)$ codes contains the general affine group $\mathrm{GA}(m,q)$ under the natural action on $V = \mathbb{F}_{q^m}$.

Knorr and Willems in [16] give a complete description of the automorphism group of the p-ary RM codes for any prime p, in which they prove that the automorphism group of the p-ary RM codes equals the general affine group $\mathrm{GA}(m,p)$.

In [9], Berger and Charpin provide a complete description of the automorphism group of a GRM code. They show that the automorphism group of GRM codes is the affine linear

group, i.e.,

$$\mathrm{Aut}(\mathrm{GRM}_q(m,r)) = \mathrm{GA}(m,q).$$

3. SYMMETRIC REED-MULLER CODES

The Symmetric Reed-Muller (SRM, for short) codes, a variant of Reed-Muller codes, are introduced for the first time in this paper [5]. The authors focus on the bivariate version of SRM codes and demonstrate that these codes take the form of locally-correctable codes. They have determined the minimum distance of SRM codes for specific cases. Finally, they introduce the multivariate version of SRM codes. SRM codes which exhibit specific symmetry properties within their codewords may be considered as subcodes of GRM codes. In a subsequent paper [12] by the same authors, it was shown that multivariate Symmetric Reed-Muller codes are locally correctable codes. Additionally, the dual codes of multivariate Symmetric Reed-Muller codes were presented.

Before presenting the formal definition in [12], we recall some notions.

The set $E_q(n,r) \subseteq \mathbb{F}_q[x_1, x_2, \ldots, x_n]$ is defined by

$$E_q(n,r) := \Big\{ f(x_1, x_2, \ldots, x_n) = \sum_{\substack{0 \le i_1 < i_2 < \ldots < i_n \le q-1 \\ i_1 + i_2 + \ldots + i_n \le r}} a_{i_1 i_2 \ldots i_n} \det(x,i) \mid a_{i_1 i_2 \ldots i_n} \in \mathbb{F}_q \Big\},$$

where $x = (x_1, x_2, \ldots, x_n)$, $i = (i_1, i_2, \ldots, i_n)$ and

$$\det(x,i) := \begin{vmatrix} {x_1}^{i_1} & {x_2}^{i_1} & \ldots & {x_n}^{i_1} \\ {x_1}^{i_2} & {x_2}^{i_2} & \ldots & {x_n}^{i_2} \\ \vdots & \vdots & \ddots & \vdots \\ {x_1}^{i_n} & {x_2}^{i_n} & \ldots & {x_n}^{i_n} \end{vmatrix}.$$

Consider the set

$$\Delta = \{(a_1, a_2, \ldots, a_n) \in \mathbb{F}_q^n \mid a_j \neq a_i, 1 \le j < i \le n\}$$

and an equivalence relation $\sim$ on Δ defined as follows:

$$c \sim d \iff \exists \sigma \in S_n \text{ such that } \sigma(c) = d, \text{ for } c, d \in \mathbb{F}_q^n.$$

Then define the set $\Omega_q(n) := \Delta/\sim = \{[\alpha] \mid \alpha \in \Delta\}$, where $[\alpha]$ denotes the equivalent class of α. In order to get rid of duplications, the definition of SRM codes is given by using this quotient set as follows:

Definition 3.0.1. [12] Let n and r be positive integers, where n denotes the number of variables. *The SRM code of degree r over $\mathbb{F}_q$* is defined by

$$\mathrm{SRM}_q[n, r] = \{(f(\alpha))_{[\alpha]\in\Omega_q(n)} \mid f \in E_q(n, r)\}.$$

Remark 3.0.2. According to the definition, when $q < r$, the possible $\{i_1, i_2, \ldots, i_n\}$ sequence may not cover a partition of r. On the other hand, for the sequence $\{i_1, i_2, \ldots, i_n\} = \{0, 1, 2, \ldots, n-1\}$ the smallest value of r should be $\frac{n(n-1)}{2}$. Thus, the definition of SRM code is correct under the condition $q \geq r \geq \frac{n(n-1)}{2}$, where q is chosen to be large enough.

Note that under the condition mentioned in the remark above, when $n = 1$, $\mathrm{SRM}_q[1, r]$ codes are exactly generalized Reed-Solomon codes with degree parameter r.

When $n = 2$, we have

$$E_q(2, r) := \left\{ \sum_{\substack{0\leq i<j\leq q-1 \\ i+j\leq r}} a_{ij}(x_1^i x_2^j - x_1^j x_2^i) \,\middle|\, a_{ij} \in \mathbb{F}_q \right\} \subseteq \mathbb{F}_q[x_1, x_2].$$

The evaluation of $f(x_1, x_2)$ at $(x_1, x_2) \in \mathbb{F}_q^2$ forms as the following matrix

$$\begin{bmatrix} f(\alpha_0, \alpha_0) & f(\alpha_0, \alpha_1) & \ldots & f(\alpha_0, \alpha_{q-1}) \\ f(\alpha_1, \alpha_0) & f(\alpha_1, \alpha_1) & \ldots & f(\alpha_1, \alpha_{q-1}) \\ \vdots & \vdots & \ddots & \vdots \\ f(\alpha_{q-1}, \alpha_0) & f(\alpha_{q-1}, \alpha_1) & \ldots & f(\alpha_{q-1}, \alpha_{q-1}) \end{bmatrix}.$$

Example 3.0.3. *Let $n = 2$, $r = 2$ and $q = 3$. Then,*

$$E_3(2,2) = \left\{ \sum_{\substack{0\le i<j\le 2 \\ i+j\le 2}} a_{ij} \begin{vmatrix} {x_1}^i & {x_2}^i \\ {x_1}^j & {x_2}^j \end{vmatrix} \;\middle|\; a_{ij} \in \mathbb{F}_3 \right\} = \{a_{01}(Y-X) + a_{02}(Y^2-X^2) \mid a_{01}, a_{02} \in \mathbb{F}_3\}$$

		evaluation points		
functions		01	02	12
	$Y-X$	1	2	1
	$2(Y-X)$	2	1	2
	Y^2-X^2	1	1	0
	$2(Y^2-X^2)$	2	2	0
	$(Y-X)+(Y^2-X^2)$	2	0	1
	$(Y-X)+2(Y^2-X^2)$	0	1	1
	$2(Y-X)+(Y^2-X^2)$	0	2	2
	$2(Y-X)+2(Y^2-X^2)$	1	0	2

Thus, all codewords of the $SRM_3(2,2)$ code are as follows:

$$\{(121),(212),(110),(220),(201),(011),(022),(102)\}$$

Remark 3.0.4. Let $q < r$. Since the set $E_q(n,r)$ has two constraints, $0 \le i < j \le q-1$ and $i+j \le r$, all possible pairs (i,j) correspond to all possible vectors. Therefore, the SRM code loses its distinctiveness. Thus, $q > r$ should be chosen.

Below, an example related to this remark is provided:

Example 3.0.5. *Let $q = r = 3$ and $n = 2$. According to the $E_3(2,3)$, the table of the functions and their evaluation points is provided below :*

		evaluation points		
functions		01	02	12
	$Y-X$	1	2	1
	$2(Y-X)$	2	1	2
	Y^2-X^2	1	1	0
	$2(Y^2-X^2)$	2	2	0
	$(Y-X)+(Y^2-X^2)$	2	0	1
	$(Y-X)+2(Y^2-X^2)$	0	1	1
	$2(Y-X)+(Y^2-X^2)$	0	2	2
	$2(Y-X)+2(Y^2-X^2)$	1	0	2
	XY^2-X^2Y	0	0	2
	$Y-X+XY^2-X^2Y$	1	2	0
	$2(Y-X)+XY^2-X^2Y$	2	1	1
	$Y^2-X^2+XY^2-X^2Y$	1	1	2
	$2(Y^2-X^2)+XY^2-X^2Y$	2	2	2
	$(Y-X)+(Y^2-X^2)+XY^2-X^2Y$	2	0	0
	$(Y-X)+2(Y^2-X^2)+XY^2-X^2Y$	0	1	0
	$2(Y-X)+(Y^2-X^2)+XY^2-X^2Y$	0	2	1
	$2(Y-X)+2(Y^2-X^2)+XY^2-X^2Y$	1	0	1
	$2(XY^2-X^2Y)$	0	0	1
	$Y-X+2(XY^2-X^2Y)$	1	2	2
	$2(Y-X)+2(XY^2-X^2Y)$	2	1	0
	$Y^2-X^2+2(XY^2-X^2Y)$	1	1	1
	$2(Y^2-X^2)+2(XY^2-X^2Y)$	2	2	1
	$(Y-X)+(Y^2-X^2)+2(XY^2-X^2Y)$	2	0	2
	$(Y-X)+2(Y^2-X^2)+2(XY^2-X^2Y)$	0	1	2
	$2(Y-X)+(Y^2-X^2)+2(XY^2-X^2Y)$	0	2	0
	$2(Y-X)+2(Y^2-X^2)+2(XY^2-X^2Y)$	1	0	0

Therefore, the $\mathrm{SRM}_3[2,3]$ *code has no distinctiveness within the vector space* F_3^3. $\mathrm{SRM}_3[2,3] = \{001, 002, 010, 011, 012, 020, 021, 022, 100, 101, 102, 110, 111, 112, 120, 121, 122, 200, 201, 202, 210, 211, 212, 220, 221, 222\}$

Since $f(x_1, x_1) = 0$ and $f(x_1, x_2) = -f(x_2, x_1)$, the above matrix is a skew-symmetric matrix. Thus, it is entirely determined by the entries in the strictly upper triangular part. Therefore, the codeword of bivariate SRM, i.e., when $n = 2$, codes are defined as the strictly upper triangular part of this matrix. Furthermore, $\mathrm{SRM}_q[2, r]$ codes are of length $\frac{q(q-1)}{2}$.

Remark 3.0.6. For each $f(x_1, x_2, \ldots, x_n) \in E_q(n, r)$, the following properties hold:

- $f(x_1, x_2, \ldots, x_n) = 0$, if there exists $1 \leq r \neq s \leq n$ such that $x_r = x_s$.
- Let $\pi = (i, j) \in S_n$ be a transposition. Then

$$f(x_1, x_2, \ldots, x_n) = -f(x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)}).$$

- Let $\tau \in S_n$. Then $f(x_1, x_2, \ldots, x_n) = \mp f(x_{\tau(1)}, x_{\tau(2)}, \ldots, x_{\tau(n)})$. The sign is determined by evenness or oddness of the permutation τ.

For more details about the set $E_q(n, r)$, we refer the reader to [5].

Example 3.0.7. *Let* $n = 3$, $r = 3$ *and* $q = 3$. *Then,*

$$E_3(3,3) = \left\{ \sum_{\substack{0 \leq i<j<k \leq 2 \\ i+j+k \leq 3}} a_{ijk} \begin{vmatrix} {x_1}^i & {x_2}^i & {x_3}^i \\ {x_1}^j & {x_2}^j & {x_3}^j \\ {x_1}^k & {x_2}^k & {x_3}^k \end{vmatrix} \;\middle|\; a_{ijk} \in \mathbb{F}_3 \right\} = \left\{ \begin{vmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ {x_1}^2 & {x_2}^2 & {x_3}^2 \end{vmatrix} \;\middle|\; a_{ijk} \in \mathbb{F}_3 \right\}$$

$$= \{0, (x_2x_3^2 + x_1x_2^2 + x_1^2x_3 - x_2^2x_3 - x_1^2x_2 - x_1x_3^2), 2(x_2x_3^2 + x_1x_2^2 + x_1^2x_3 - x_2^2x_3 - x_1^2x_2 - x_1x_3^2)\}.$$

The coefficients of the monomial pairs $(x_2x_3^2, x_2^2x_3), (x_1x_2^2, x_1^2x_2), (x_1x_3^2, x_1^2x_3)$ *of each element of* $E_3(3,3)$ *have opposite signs pairwise. So the summation of the coefficients of these monomials is zero. Furthermore,* $x_1x_2x_3, x_1^3, x_2^3, x_3^3$ *are not the monomials of any members in* $E_3(3,3)$*, so their coefficients are zero.*

Remark 3.0.8. We would like to note that the observations in Example 3.0.7 are general facts for each element of $E_q(3,r)$. In other words, for any $f \in E_q(3,r)$, the coefficient of the monomial $(x_1^i x_2^j x_3^k)$ is negative of the coefficient of the monomial $(x_{\pi(1)}^i x_{\pi(2)}^j x_{\pi(3)}^k)$, where $\pi \in S_3$ is an odd permutation. Additionally, when any two of i, j, and k are equal, the coefficient of the monomial $(x^i y^j z^k)$ is zero for each $f \in E_q(3,r)$.

4. LINEAR TRANSFORMATIONS UNDER WHICH SRM CODES ARE INVARIANT

In this section, we will derive the invariant groups of SRM for $n = 2$ and $n = 3$ over the field $\mathbb{F}_p$, where p is any prime number. For $n = 2$ and $n = 3$, we determine the exact set formed by transformations that leave SRM codes invariant under a subgroup of the affine linear group. Different methods were employed to determine this set for the values of $n = 2, 3$, separately. The reason for using different methods is that one approach may not be highly suitable for the other. Our main theorems, which identify the group formed by transformations that leave the SRM code invariant, and the necessary lemmas for these theorems are provided for each $n = 2, 3$.

4.1 The case n=2

Recall $\mathrm{SRM}_q[2, r]$ is an evaluated code family whose evaluation polynomials come from the set $E_q(2, r)$,

$$E_q(2,r) = \left\{ \sum_{\substack{0 \le i < j \le q-1 \\ i+j \le r}} a_{ij} \begin{vmatrix} {x_1}^i & {x_2}^i \\ {x_1}^j & {x_2}^j \end{vmatrix} \;\middle|\; a_{ij} \in \mathbb{F}_q \right\}.$$

Let $f(x_1, x_2) = \begin{vmatrix} {x_1}^i & {x_2}^i \\ {x_1}^j & {x_2}^j \end{vmatrix} \in E_q(2, r)$ and $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$. Under the $T_f(A)$ transformation in (1), we get

$$\begin{aligned} T_f(A) &= \begin{vmatrix} (ax_1 + bx_2)^i & (cx_1 + dx_2)^i \\ (ax_1 + bx_2)^j & (cx_1 + dx_2)^j \end{vmatrix} \\ &= (ax_1 + bx_2)^i \cdot (cx_1 + dx_2)^j - (ax_1 + bx_2)^j \cdot (cx_1 + dx_2)^i. \end{aligned}$$

We investigate the coefficients $a, b, c, d \in \mathbb{F}_q$ to find the transformations that keep the set $E_q(2, r)$ invariant. For this purpose, we need the following auxiliary lemmas.

Lemma 4.1.1. *Let $f \in E_q(2, r)$ and $\alpha, \beta \in \mathbb{F}_q$ and t be a positive integer. Then we have*

$$(\alpha x_1^2 + \beta x_1 x_2 + \alpha x_2^2)^t \cdot f \in E_q(2, r).$$

Proof. Without loss of generality, we shall assume that

$$f = \begin{vmatrix} x_1^i & x_2^i \\ x_1^j & x_2^j \end{vmatrix} = x_1^i x_2^j - x_1^j x_2^i,$$

where $0 \leq i < j \leq (q-1)$ such $i + j \leq r$. We use the induction on t to prove the claim. When $t = 1$, we have

$$\begin{aligned}(\alpha x_1^2 + \beta x_1 x_2 + \alpha x_2^2) \cdot f &= \alpha(x_1^2 + x_2^2)(x_1^i x_2^j - x_1^j x_2^i) + \beta(x_1 x_2)(x_1^i x_2^j - x_1^j x_2^i) \\ &= \alpha(f_1 + f_2) + \beta f_3,\end{aligned}$$

where

$$f_1 = x_1^{i+2} x_2^j - x_1^j x_2^{i+2}, \quad f_2 = x_1^i x_2^{j+2} - x_1^{j+2} x_2^i \quad \text{and} \quad f_3 = x_1^{i+1} x_2^{j+1} - x_1^{j+1} x_2^{i+1}.$$

Thus, $f_1, f_2, f_3 \in E_q(2, r)$, which implies that $(\alpha x_1^2 + \beta x_1 x_2 + \alpha x_2^2) \cdot f \in E_q(2, r)$. For the next step, suppose that the claim holds for $t > 1$, i.e., we have that

$$(\alpha x_1^2 + \beta x_1 x_2 + \alpha x_2^2)^t \cdot f \in E_q(2, r). \tag{4}$$

Then it is sufficient to show that the statement holds for $(t + 1)$.

$$(\alpha x_1^2 + \beta x_1 x_2 + \alpha x_2^2)^{t+1} \cdot f = (\alpha x_1^2 + \beta x_1 x_2 + \alpha x_2^2)((\alpha x_1^2 + \beta x_1 x_2 + \alpha x_2^2)^t \cdot f). \tag{5}$$

By the equation (4), we have

$$g = (\alpha x_1^2 + \beta x_1 x_2 + \alpha x_2^2)^t \cdot f \in E_q(2, r).$$

Since $g \in E_q(2, r)$, we may consider the equation (5) as follows:

$$(\alpha x_1^2 + \beta x_1 x_2 + \alpha x_2^2) \cdot g = (\alpha x_1^2 + \beta x_1 x_2 + \alpha x_2^2) \cdot \sum_{\substack{0 \le i < j \le q-1 \\ i+j \le r}} a_{ij} g_{ij},$$

where $a_{ij} \in \mathbb{F}_q$ and

$$g_{ij} = \begin{vmatrix} x_1^i & x_2^i \\ x_1^j & x_2^j \end{vmatrix} = x_1^i x_2^j - x_1^j x_2^i.$$

By the case of $t = 1$, we have all $(\alpha x_1^2 + \beta x_1 x_2 + \alpha x_2^2) \cdot g_{ij} \in E_q(2, r)$. Thus, $(\alpha x_1^2 + \beta x_1 x_2 + \alpha x_2^2) \cdot g \in E_q(2, r)$. In the same manner, this is generalized for any $f \in E_q(2, r)$, which completes the proof. □

Lemma 4.1.2. *Let $a, b \in \mathbb{F}_q$ and t be a positive integer. Then*

$$\begin{vmatrix} 1 & 1 \\ (ax_1 + bx_2)^t & (bx_1 + ax_2)^t \end{vmatrix} \in E_q(2, r).$$

Proof. When t is odd, consider

$$\begin{aligned}
\begin{vmatrix} 1 & 1 \\ (ax_1+bx_2)^t & (bx_1+ax_2)^t \end{vmatrix} &= (bx_1+ax_2)^t - (ax_1+bx_2)^t \\
&= \sum_{k=0}^{t} \binom{t}{k} \left((bx_1)^k (ax_2)^{t-k} - (ax_1)^k (bx_2)^{t-k} \right) \\
&= \sum_{k=0}^{t} \binom{t}{k} \left(b^k a^{t-k} - a^k b^{t-k} \right) x_1^k x_2^{t-k} \\
&= \sum_{k=0}^{(t-1)/2} \binom{t}{k} \left(b^k a^{t-k} - a^k b^{t-k} \right) \left(x_1^k x_2^{t-k} - x_1^{t-k} x_2^k \right) \\
&= \sum_{k=0}^{(t-1)/2} \binom{t}{k} \left(b^k a^{t-k} - a^k b^{t-k} \right) \begin{vmatrix} x_1^k & x_2^k \\ x_1^{t-k} & x_2^{t-k} \end{vmatrix}.
\end{aligned}$$

Clearly, the corresponding determinant is an element of $E_q(2,r)$.

When t is even, similarly, we get that

$$\begin{aligned}
\begin{vmatrix} 1 & 1 \\ (ax_1+bx_2)^t & (bx_1+ax_2)^t \end{vmatrix} &= \sum_{k=0}^{t} \binom{t}{k} \left(b^k a^{t-k} - a^k b^{t-k} \right) x_1^k x_2^{t-k} \\
&= \sum_{k=0}^{t/2-1} \binom{t}{k} \left(b^k a^{t-k} - a^k b^{t-k} \right) \left(x_1^k x_2^{t-k} - x_1^{t-k} x_2^k \right).
\end{aligned}$$

In the above equation, when $k = t/2$, the coefficient of the term $x_1^{t/2} x_2^{t/2}$ is zero. Finally, as in the odd case, the corresponding determinant is an element of $E_q(2,r)$. Thus, the proof is completed. □

Proposition 4.1.3. *Let $a, b \in \mathbb{F}_q$ and i, j be positive integers such that $i < j$. Then*

$$\begin{vmatrix} (ax_1+bx_2)^i & (bx_1+ax_2)^i \\ (ax_1+bx_2)^j & (bx_1+ax_2)^j \end{vmatrix} \in E_q(2,r).$$

Proof. The determinant can be written as:

$$\begin{vmatrix} (ax_1+bx_2)^i & (bx_1+ax_2)^i \\ (ax_1+bx_2)^j & (bx_1+ax_2)^j \end{vmatrix} = (ax_1+bx_2)^i(bx_1+ax_2)^i \begin{vmatrix} 1 & 1 \\ (ax_1+bx_2)^{j-i} & (bx_1+ax_2)^{j-i} \end{vmatrix}$$

$$= \left(abx_1^2+(a^2+b^2)x_1x_2+abx_2^2\right)^i \begin{vmatrix} 1 & 1 \\ (ax_1+bx_2)^t & (bx_1+ax_2)^t \end{vmatrix},$$

where $t = j - i$. By utilizing Lemmas 4.1.1 and 4.1.2, the proof follows. □

The following theorem gives a necessary and sufficient condition for $\mathrm{SRM}_q[2,r]$ to be invariant under which subgroup of $\mathrm{GL}(2,q)$.

Theorem 4.1.4. *Let M be a set defined as*

$$M = \left\{ \begin{bmatrix} a & b \\ b & a \end{bmatrix} \mid a, b \in \mathbb{F}_q,\ a \neq \pm b \right\} \subset \mathrm{GL}(2,q).$$

The automorphism group of the $\mathrm{SRM}_q[2,r]$, where $q \geq r > 2$ except $q = r = 3$, code family contains a subgroup isomorphic to M, i.e., $\mathrm{SRM}_q[2,r]$ is invariant under the transformations of M.

Proof. Let $A \in \mathrm{GL}(2,q)$. Then take the transform T:

$$T : \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \mapsto A \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

In the set $E_q(2,r)$, there exist unique polynomials of degree 1 and 2, which are

$$f(x_1,x_2) = \begin{vmatrix} 1 & 1 \\ x_1 & x_2 \end{vmatrix} = x_2 - x_1$$

and

$$g(x_1, x_2) = \begin{vmatrix} 1 & 1 \\ x_1^2 & x_2^2 \end{vmatrix} = x_2^2 - x_1^2,$$

respectively. If $T_f(A)$ and $T_g(A)$ are elements of $E_q(2, r)$, it is easy to see that $T_f(A)$ and $T_g(A)$ must be scalar multiples of $f(x_1, x_2)$ and $g(x_1, x_2)$, respectively. In the light of this fact, we have

$$\begin{aligned} T_f(A) = f(ax_1 + bx_2, cx_1 + dx_2) &= \begin{vmatrix} 1 & 1 \\ (ax_1 + bx_2) & (cx_1 + dx_2) \end{vmatrix} \\ &= (d - b)x_2 - (a - c)x_1 \end{aligned}$$

and

$$\begin{aligned} T_g(A) = g(ax_1 + bx_2, cx_1 + dx_2) &= \begin{vmatrix} 1 & 1 \\ (ax_1 + bx_2)^2 & (cx_1 + dx_2)^2 \end{vmatrix} \\ &= (cx_1 + dx_2)^2 - (ax_1 + bx_2)^2 \\ &= (d^2 - b^2)x_2^2 - (a^2 - c^2)x_1^2 + (2cd - 2ab)x_1x_2. \end{aligned}$$

From the above, we obtain the following equations

$$\begin{aligned} a - c &= d - b, \\ (a^2 - c^2) &= (b^2 - d^2), \\ (2cd - 2ab) &= 0. \end{aligned}$$

If we solve the equations above together with the fact $ad - bc \neq 0$, then we will get $a = d$ and $b = c$. Combining this with Proposition 4.1.3, we obtain that $\mathrm{SRM}_q[2, r]$ is invariant under the transformations that come from the set M, which completes the proof. □

Note that when $q = r = 3$, the set $\mathrm{SRM}_3[2, 3]$ contains all vectors of length 3 so that $\mathrm{Aut}(\mathrm{SRM}_3[2, 3]) = S_3$.

In the following subsection, we will focus on the $\mathrm{SRM}_q(3,r)$ in the same manner.

4.2 The case n=3

Recall that the code family $\mathrm{SRM}_q[3,r]$ with the length $\sum_{i=1}^{q-2}\frac{(i)(i+1)}{2}$ is an evaluated code family whose evaluation polynomials come from the set $E_q(3,r)$, where

$$E_q(3,r)=\left\{\sum_{\substack{0\le i<j<k\le q-1\\ i+j+k\le r}} a_{ijk}\begin{vmatrix} {x_1}^i & {x_2}^i & {x_3}^i\\ {x_1}^j & {x_2}^j & {x_3}^j\\ {x_1}^k & {x_2}^k & {x_3}^k\end{vmatrix}\ \middle|\ a_{ijk}\in\mathbb{F}_q\right\}.$$

Under the $T_f(A)$ transformation in (1), where

$$A=\begin{bmatrix} a & b & c\\ d & e & f\\ g & h & i\end{bmatrix},$$

we investigate the coefficients a, b, c, d, e, f, g, h, $i\in\mathbb{F}_q$ to determine the transformation that keeps the set $E_q(3,r)$ invariant. We require the auxiliary lemmas for this aim.

The following lemma gives us a different interpretation of the set $E_q(3,r)$.

Lemma 4.2.1. *Let $f(x_1,x_2,x_3)\in\mathbb{F}_q[x_1,x_2,x_3]$ with a degree less than or equal to r, where q is odd, such that for any $\pi\in S_3$,*

$$f(x_{\pi(1)},x_{\pi(2)},x_{\pi(3)})=\begin{cases}-f(x_1,x_2,x_3), & \pi \text{ is an odd permutation},\\ f(x_1,x_2,x_3), & \pi \text{ is an even permutation}.\end{cases}\tag{6}$$

Then $f(x_1,x_2,x_3)\in E_q(3,r)$.

Proof. Firstly, we may assume that we have a homogeneous nonzero multivariate polynomial $f(x_1,x_2,x_3)$ of degree $t\le r$, with the property (6).

Since f is nonzero, we have a monomial term $A_0x_1^ix_2^jx_3^k$ in f. Without loss of generality, we may choose the powers $0 \leq i \leq j \leq k$, where i, j, k are integers such that $i + j + k = t$. So we write f as follows

$$f(x_1, x_2, x_3) = A_0x_1^ix_2^jx_3^k + g_0(x_1, x_2, x_3), \tag{7}$$

where $A_0 \in \mathbb{F}_q$ and $g_0(x_1, x_2, x_3)$ is a homogeneous polynomial of degree t such that the coefficient of the $x_1^ix_2^jx_3^k$ in $g_0(x_1, x_2, x_3)$ is zero.

Consider the case $i = j$. Then by the property (6), we get

$$\begin{aligned} f(x_2, x_1, x_3) &= A_0x_2^ix_1^ix_3^k + g_0(x_2, x_1, x_3) \\ &= -f(x_1, x_2, x_3) = -A_0x_1^ix_2^ix_3^k - g_0(x_1, x_2, x_3). \end{aligned}$$

Equivalently, we have

$$2A_0x_2^ix_1^ix_3^k + g_0(x_2, x_1, x_3) + g_0(x_1, x_2, x_3) = 0.$$

Since the coefficient of the monomial $x_1^ix_2^ix_3^k$ in $g_0(x_1, x_2, x_3)$ is zero, $x_2^ix_1^ix_3^k$ must be a monomial term of $g_0(x_2, x_1, x_3)$, whose coefficient is $-2A_0$. This is the contradiction. Similarly, we get a contradiction for the cases $i = k$ and $j = k$. Thus, there are no monomial terms $x_1^ix_2^jx_3^k$ such that at least two of i, j and k values are the same.

By the cases mentioned above, we may assume that in the monomial term $x_1^ix_2^jx_3^k$, i, j, k are distinct, i.e., $0 \leq i < j < k$. Consider the relation (7). When $\pi = (12)$, we have

$$\begin{aligned} f(x_2, x_1, x_3) &= A_0x_2^ix_1^jx_3^k + g_0(x_2, x_1, x_3) \\ &= -f(x_1, x_2, x_3) = -A_0x_1^ix_2^jx_3^k - g_0(x_1, x_2, x_3). \end{aligned}$$

By the above equation, the monomial term $-A_0x_2^ix_1^jx_3^k$ must be appeared in $g_0(x_1, x_2, x_3)$. Thus, the relation (7) can be rewritten as

$$f(x_1, x_2, x_3) = A_0x_1^ix_2^jx_3^k - A_0x_1^jx_2^ix_3^k + g_1(x_1, x_2, x_3).$$

Applying similar steps for the permutations $\pi = (13)$ and $\pi = (23)$, we get

$$f(x_1, x_2, x_3) = A_0x_1^ix_2^jx_3^k - A_0x_1^jx_2^ix_3^k - A_0x_1^kx_2^jx_3^i - A_0x_1^ix_2^kx_3^j + g_3(x_1, x_2, x_3)$$

Applying the permutation $\pi = (123)$ to $f(x_1, x_2, x_3)$ in the above relation, we get

$$\begin{aligned} f(x_2, x_3, x_1) &= A_0x_2^ix_3^jx_1^k - A_0x_1^jx_2^ix_3^k - A_0x_1^kx_2^jx_3^i - A_0x_1^ix_2^kx_3^j + g_3(x_2, x_3, x_1) \\ &= f(x_1, x_2, x_3) = A_0x_1^ix_2^jx_3^k - A_0x_1^jx_2^ix_3^k - A_0x_1^kx_2^jx_3^i - A_0x_1^ix_2^kx_3^j + g_3(x_1, x_2, x_3), \end{aligned}$$

which implies the monomial term $A_0x_2^ix_3^jx_1^k$ must be appeared in $g_3(x_1, x_2, x_3)$. Thus,

$$f(x_1, x_2, x_3) = A_0x_1^ix_2^jx_3^k + A_0x_1^kx_2^ix_3^j - A_0x_1^jx_2^ix_3^k - A_0x_1^kx_2^jx_3^i - A_0x_1^ix_2^kx_3^j + g_4(x_1, x_2, x_3).$$

Finally, if we apply the permutation $\pi = (132)$, the polynomial $f(x_1, x_2, x_3)$ will be of form

$$\begin{aligned} &A_0x_1^ix_2^jx_3^k + A_0x_1^kx_2^ix_3^j + A_0x_1^jx_2^kx_3^i - A_0x_1^jx_2^ix_3^k - A_0x_1^kx_2^jx_3^i - A_0x_1^ix_2^kx_3^j + g_5(x_1, x_2, x_3) \\ =&A_0 \begin{vmatrix} x_1^i & x_2^i & x_3^i \\ x_1^j & x_2^j & x_3^j \\ x_1^k & x_2^k & x_3^k \end{vmatrix} + g_5(x_1, x_2, x_3), \end{aligned}$$

where $g_5(x_1, x_2, x_3)$ is a homogeneous polynomial of degree t such that the coefficients of the monomials $x_{\pi(1)}^ix_{\pi(2)}^jx_{\pi(3)}^k$, for any $\pi \in S_3$, are zero.

Thereafter, if we apply what we did for $f(x_1, x_2, x_3)$ to $g_5(x_1, x_2, x_3)$ by following the same steps for the other possible triple partition, $t = i_1 + j_1 + k_1$, we will get

$$f(x_1, x_2, x_3) = A_0 \begin{vmatrix} x_1^i & x_2^i & x_3^i \\ x_1^j & x_2^j & x_3^j \\ x_1^k & x_2^k & x_3^k \end{vmatrix} + A_1 \begin{vmatrix} x_1^{i_1} & x_2^{i_1} & x_3^{i_1} \\ x_1^{j_1} & x_2^{j_1} & x_3^{j_1} \\ x_1^{k_1} & x_2^{k_1} & x_3^{k_1} \end{vmatrix} + g_6(x_1, x_2, x_3),$$

where $A_0, A_1 \in \mathbb{F}_q$ and $g_6(x_1, x_2, x_3)$ is a homogeneous polynomial of degree t.

Since the number of the triple partitions of t is finite, we may continue the above procedure until all possible partitions are over. Finally, the polynomial f is of form

$$f(x_1, x_2, x_3) = A_0 \begin{vmatrix} x_1^i & x_2^i & x_3^i \\ x_1^j & x_2^j & x_3^j \\ x_1^k & x_2^k & x_3^k \end{vmatrix} + A_1 \begin{vmatrix} x_1^{i_1} & x_2^{i_1} & x_3^{i_1} \\ x_1^{j_1} & x_2^{j_1} & x_3^{j_1} \\ x_1^{k_1} & x_2^{k_1} & x_3^{k_1} \end{vmatrix} + \cdots + A_d \begin{vmatrix} x_1^{i_d} & x_2^{i_d} & x_3^{i_d} \\ x_1^{j_d} & x_2^{j_d} & x_3^{j_d} \\ x_1^{k_d} & x_2^{k_d} & x_3^{k_d} \end{vmatrix},$$

where A_i's in $\mathbb{F}_q$. Thus, $f(x_1, x_2, x_3) \in E_q(3, r)$ by the definition.

In general, for any polynomial $F(x_1, x_2, x_3)$ satisfying the condition (6), we may write

$$F(x_1, x_2, x_3) = f_3(x_1, x_2, x_3) + f_4(x_1, x_2, x_3) + \cdots + f_r(x_1, x_2, x_3),$$

where f_i's are homogeneous polynomials of degree i for $i \in \{3, 4, \ldots, r\}$. Hence, for any $i \in \{3, 4, \ldots, r\}$, we get that $f_i \in E_q(3, r)$. Thus, $F(x_1, x_2, x_3) \in E_q(3, r)$, which completes the proof. □

Consider the subgroup

$$K = \left\{ P * \begin{bmatrix} b & a & a \\ a & b & a \\ a & a & b \end{bmatrix}, \middle| \, P \in \mathcal{P}_3, a, b \in \mathbb{F}_q, a \neq b, \, b \neq -2a \right\} \tag{8}$$

of $\mathrm{GL}(3,q)$. We will show that K is a subgroup under which $E_q(3,r)$ is invariant. First, we need the following key lemma.

Lemma 4.2.2. *Let $A \in K$ and $f(x_1,x_2,x_3) \in E_q(3,r)$. Then the function g defined as $g(x_1,x_2,x_3) = T_f(A)$ belongs the set $E_q(3,r)$.*

Proof. Firstly, assume that $A = \begin{bmatrix} b & a & a \\ a & b & a \\ a & a & b \end{bmatrix}$. Let $f(x_1,x_2,x_3) \in E_q(3,r)$. Under the transformation $T(A)$ in (2), we have variables $x_1 \mapsto bx_1 + ax_2 + ax_3$, $x_2 \mapsto ax_1 + bx_2 + ax_3$ and $x_3 \mapsto ax_1 + ax_2 + bx_3$.

Let $g = T_f(A)$. Then

$$g(x_1,x_2,x_3) = T_f(A) = f(bx_1 + ax_2 + ax_3, ax_1 + bx_2 + ax_3, ax_1 + ax_2 + bx_3)$$

Now let $\pi = (12) \in S_3$.

$$\begin{aligned} g(x_{\pi(1)}, x_{\pi(2)}, x_{\pi(3)}) = g(x_2,x_1,x_3) &= f(bx_2 + ax_1 + ax_3, ax_2 + bx_1 + ax_3, ax_2 + ax_1 + bx_3) \\ &= -f(bx_1 + ax_2 + ax_3, ax_1 + bx_2 + ax_3, ax_1 + ax_2 + bx_3) \\ &= -g(x_1,x_2,x_3), \end{aligned}$$

where second line comes from the fact that $f(x_1,x_2,x_3) \in E_q(3,r)$. Similarly, when $\pi = (13)$ or $\pi = (23)$, we get that $g(x_{\pi(1)}, x_{\pi(2)}, x_{\pi(3)}) = -g(x_1,x_2,x_3)$.

On the other hand, when $\pi = (123) \in S_3$, we have that

$$\begin{aligned} g(x_{\pi(1)}, x_{\pi(2)}, x_{\pi(3)}) = g(x_2,x_3,x_1) &= f(bx_2 + ax_3 + ax_1, ax_2 + bx_3 + ax_1, ax_2 + ax_3 + bx_1) \\ &= -f(ax_2 + bx_3 + ax_1, bx_2 + ax_3 + ax_1, ax_2 + ax_3 + bx_1) \\ &= f(bx_1 + ax_2 + ax_3, ax_1 + bx_2 + ax_3, ax_1 + ax_2 + bx_3) \\ &= g(x_1,x_2,x_3). \end{aligned}$$

Similarly, when $\pi = (132)$, we obtain that $g(x_{\pi(1)}, x_{\pi(2)}, x_{\pi(3)}) = g(x_1, x_2, x_3)$.

Finally from above, we can characterize the multivariate polynomial $g(x_1, x_2, x_3)$ as follows for any $\pi \in S_3$:

$$g(x_{\pi(1)}, x_{\pi(2)}, x_{\pi(3)}) = \begin{cases} -g(x_1, x_2, x_3), & \pi \text{ is an odd permutation,} \\ g(x_1, x_2, x_3), & \pi \text{ is an even permutation.} \end{cases}$$

Hence, by Lemma 4.2.1, $g = T_f(A) \in E_q(3, r)$.

Now let $B \neq A$ be an element of the set K. Then there exists $I_3 \neq P \in \mathcal{P}_3$ such that $B = PA$, where P is the permutation matrix associated with a permutation σ. Under the transformation $T(B) = T(PA)$, we have

$$x_{\sigma(1)} \mapsto bx_1 + ax_2 + ax_3, \quad x_{\sigma(2)} \mapsto ax_1 + bx_2 + ax_3 \quad \text{and} \quad x_{\sigma(3)} \mapsto ax_1 + ax_2 + bx_3.$$

Equivalently, we get

$$x_1 \mapsto bx_{\sigma^{-1}(1)} + ax_{\sigma^{-1}(2)} + ax_{\sigma^{-1}(3)}, \quad x_2 \mapsto ax_{\sigma^{-1}(1)} + bx_{\sigma^{-1}(2)} + ax_{\sigma^{-1}(3)}$$
$$x_3 \mapsto ax_{\sigma^{-1}(1)} + ax_{\sigma^{-1}(2)} + bx_{\sigma^{-1}(3)},$$

where σ^{-1} denotes the inverse of the permutation σ. Thus,

$$g(x_1, x_2, x_3) = T_f(PA) =$$
$$f(bx_{\sigma^{-1}(1)}+ax_{\sigma^{-1}(2)}+ax_{\sigma^{-1}(3)}, ax_{\sigma^{-1}(1)}+bx_{\sigma^{-1}(2)}+ax_{\sigma^{-1}(3)}, ax_{\sigma^{-1}(1)}+ax_{\sigma^{-1}(2)}+bx_{\sigma^{-1}(3)}),$$

which gives

$$h(x_1, x_2, x_3) = g(x_{\sigma(1)}, x_{\sigma(2)}, x_{\sigma(3)}) = f(bx_1+ax_2+ax_3, ax_1+bx_2+ax_3, ax_1+ax_2+bx_3).$$

We obtain $h \in E_q(3, r)$ from the previous steps. Finally, $g = T_f(PA) \in E_q(3, r)$, which completes the proof. □

Remark 4.2.3. Let $\begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 1 \\ 2 & 2 & 0 \end{bmatrix} \in \mathrm{GL}(3,q) \setminus K$, $q > 3$ and

$$f(x_1, x_2, x_3) = -x^3y^2z + x^3yz^2 + x^2y^3z - x^2yz^3 - xy^3z^2 + xy^2z^3 \in E_q(3,r).$$

Consider the function g defined as :

$$\begin{aligned} g(x_1, x_2, x_3) &= T_f(A) \\ &= f(x_1 + 2x_2, x_2 + x_3, 2x_1 + 2x_2) \\ &= -4x^5y - 4x^5z - 10x^4y^2 - 12x^4yz - 2x^4z^2 - 26x^3y^2z - 8x^3yz^2 + 18x^3z^3 \\ &+ 10x^2z^4 - 14x^2y^3z - 18x^2y^2z^2 + 14x^2yz^3 + 8x^2z^4 + 4xy^5 + 12xy^4z - 28xy^3z^2 \\ &- 4xy^2z^3 + 24xyz^4 - 8xz^5 + 8y^5z - 16y^4z^2 + 16y^2z^4 - 8yz^5. \end{aligned}$$

By the definition of $E_q(3,r)$ and the Remark 3.0.6, it is easily seen that g does not belong to the set $E_q(3,r)$.

As seen above, Lemma 4.2.2 is not valid for any element of $\mathrm{GL}(3,q)$. However, Lemma 4.2.2 guarantees that the function $g = T_f(A)$, where $A \in K$ and $f \in E_q(3,r)$, remains in the set $E_q(3,r)$. The following theorem also ensures that the set K is an exact set in $\mathrm{GL}(3,r)$.

Theorem 4.2.4. *The automorphism group of the* $\mathrm{SRM}_q[3,r]$*, where* $q \geq r > 3$*, code family contains a subgroup isomorphic to* K *in the equation (8), i.e.,* $\mathrm{SRM}_q[3,r]$ *is invariant under the transformations that come from* K*.*

Proof. By the definition of $E_q(3,r)$, the members of degrees 3 and 4 in $E_q(3,r)$ are the sets

$$N = \left\{ a_{012} \begin{vmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ x_1^2 & x_2^2 & x_3^2 \end{vmatrix} \middle| \, a_{012} \in \mathbb{F}_q \right\}$$

and

$$Q = \left\{ a_{013} \begin{vmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ x_1^3 & x_2^3 & x_3^3 \end{vmatrix} \middle| \, a_{013} \in \mathbb{F}_q \right\},$$

respectively. Let $B \in \mathbb{F}_q^{3\times 3}$ be an invertible matrix such that $B = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$. It is clear that when $f \in N$ and $g \in Q$, the following conditions must hold:

$$T_f(B) \in N \qquad \text{and} \qquad T_g(B) \in Q.$$

By Equation 3, the transformations $T_f(B)$ and $T_g(B)$ are obtained as below:

$$T_f(B) = \begin{vmatrix} 1 & 1 & 1 \\ (ax_1 + bx_2 + cx_3) & (dx_1 + ex_2 + fx_3) & (gx_1 + hx_2 + ix_3) \\ (ax_1 + bx_2 + cx_3)^2 & (dx_1 + ex_2 + fx_3)^2 & (gx_1 + hx_2 + ix_3)^2 \end{vmatrix}$$

and

$$T_g(B) = \begin{vmatrix} 1 & 1 & 1 \\ (ax_1 + bx_2 + cx_3) & (dx_1 + ex_2 + fx_3) & (gx_1 + hx_2 + ix_3) \\ (ax_1 + bx_2 + cx_3)^3 & (dx_1 + ex_2 + fx_3)^3 & (gx_1 + hx_2 + ix_3)^3 \end{vmatrix}.$$

If $T_f(B), T_g(B) \in E_q(3, r)$, then we will have $T_f(B) \in N$ and $T_g(B) \in Q$. By Remark 3.0.8, we have the following 19 equations for the corresponding monomials:

$$
\begin{aligned}
[x_1x_2x_3]T_f(B) &= 0 & [x_1x_2x_3^2]T_g(B) &= 0\\
[x_1^3]T_f(B) &= 0 & [x_1x_2^2x_3]T_g(B) &= 0\\
[x_2^3]T_f(B) &= 0 & [x_1^2x_2x_3]T_g(B) &= 0\\
[x_3^3]T_f(B) &= 0 & [x_1^2x_2^2]T_g(B) &= 0\\
[x_1x_2^2]T_f(B) + [x_1^2x_2]T_f(B) &= 0 & [x_1^2x_3^2]T_g(B) &= 0\\
[x_2x_3^2]T_f(B) + [x_2^2x_3]T_f(B) &= 0 & [x_2^2x_3^2]T_g(B) &= 0\\
[x_1x_3^2]T_f(B) + [x_1^2x_3]T_f(B) &= 0 & [x_1^4]T_g(B) &= 0\\
& & [x_2^4]T_g(B) &= 0\\
& & [x_3^4]T_g(B) &= 0\\
& & [x_1x_2^3]T_g(B) + [x_1^3x_2]T_g(B) &= 0\\
& & [x_2x_3^3]T_g(B) + [x_2^3x_3]T_g(B) &= 0\\
& & [x_1x_3^3]T_g(B) + [x_1^3x_3]T_g(B) &= 0
\end{aligned}
$$

Here, $[x_1^i x_2^j x_3^k]h(x_1, x_2, x_3)$ denotes the coefficient of the monomial $x_1^i x_2^j x_3^k$ of the polynomial h. The explicit equations that come from the above 19 constraints are provided below in sequence:

$$-2bcd-2(ac-cd)e-2(ab-bd-ae)f+2(bc-ef)g+2(ac-df-(c-f)g)h+2(ab-de-(b-e)g-(a-d)h)i = 0$$

$$-a^2d + ad^2 - (a-d)g^2 + (a^2-d^2)g = 0$$

$$-b^2e + be^2 - (b-e)h^2 + (b^2-e^2)h = 0$$

$$-c^2f + cf^2 - (c-f)i^2 + (c^2-f^2)i = 0$$

$$-2abd-b^2d+bd^2+ae^2-(b-e)g^2-(a-d)h^2-(a^2-2ad)e-2(ab-bd)e$$
$$+2(ab-de)g+(b^2-e^2)g+(a^2-d^2-2(a-d)g)h+2(ab-de-(b-e)g)h=0$$

$$-2bce-c^2e+ce^2+bf^2-(c-f)h^2-(b-e)i^2-(b^2-2be)f-2(bc-ce)f+2(bc-ef)h+(c^2-f^2)h+(b^2-e^2-2(b-e)h)i$$
$$+2(bc-ef-(c-f)h)i=0$$

$$-2acd-c^2d+cd^2+af^2-(c-f)g^2-(a-d)i^2-(a^2-2ad)f-2(ac-cd)f-2(ac-df)g+(c^2-f^2)g+$$
$$(a^2-d^2-2(a-d)g)i+2(ac-df-(c-f)g)i=0$$

$$-3bc^2d-3ac^2e-3(bd+ae)f^2-3((b-e)g+(a-d)h)i^2-6(abc-cde)f+3(bc^2-ef^2)g+3(ac^2-df^2)h$$
$$+6(abc-def-(c-f)gh)i=0$$

$$-3b^2cd-6abce+3cde^2-3(c-f)gh^2-3(ab^2-2bde-ae^2)f$$
$$+3(b^2c-e^2f)g+6(abc-def)h+3(ab^2-de^2-2(b-e)gh-(a-d)h^2)i=0$$

$$-6abcd-3(a^2c-cd^2)e-3(a^2b-bd^2-2ade)f+6(abc-def)g+3(a^2c-d^2f-(c-f)g^2)h+3(a^2b-d^2e$$
$$-2(b-e)gh-(a-d)h^2)i=0$$

$$-3ab^2d+3ade^2-3(a-d)gh^2-3(a^2b-bd^2)e+3(ab^2-de^2)g+3(a^2b-d^2e-(b-e)g^2)h=0$$

$$-3ac^2d+3adf^2-3(a-d)gi^2-3(a^2c-cd^2)f+3(ac^2-df^2)g+3(a^2c-d^2f-(c-f)g^2)i=0$$

$$-3bc^2e+3bef^2-3(b-ed)hi^2-3(b^2c-ce^2)f+3(bc^2-ef^2)h+3(b^2c-e^2f-(c-f)h^2)i=0$$
$$-a^3d+ad^3-(a-d)g^3+(a^3-d^3)g=0$$
$$-b^3e+be^3-(b-e)h^3+(b^3-e^3)h=0$$
$$-c^3f+cf^3-(c-f)i^3+(c^3-f^3)i=0$$

$$-3a^2bd-b^3d+bd^3-3ab^2e+3bde^2+ae^3-(b-e)g^3-3(b-e)gh^2-(a-d)h^3-(a^3-3ad^2)e+3(a^2b-d^2e)g$$
$$+(b^3-e^3)g+(a^3-d^3-3(a-d)g^2)h+3(ab^2-de^2)h=0$$

$$-3b^2ce-c^3e+ce^3-3bc^2f+3cef^2+bf^3-(c-f)h^3-3(c-f)hi^2-(b-e)i^3-(b^3-3be^2)f+3(b^c-e^2f)h$$
$$+(c^3-f^3)h+(b^3-e^3-3(b-e)h^2)i+3(bc^2-ef^2)i=0$$

$$-3a^2cd-c^3d+cd^3-3ac^2f+3cdf^2+af^3-(c-f)g^3-3(c-f)gi^2-(a-d)i^3-(a^3-3ad^2)f+3(a^2c-d^2f)g$$
$$+(c^3-f^3)g+(a^3-d^3-3(a-d)g^2)i+3(ac^2-df^2)i=0$$

When these equations are solved by SageMath, we obtain the set of matrices K in (8).

If the transformations with coefficient matrices from the set K are applied to functions taken from the sets N and Q, respectively, then the resulting new functions will remain within the sets N and Q.

We solve these equations for the unknowns $a, b, c, d, e, f, g, h, i$ with the help of the computer algebra system SageMath [35]. We refer the reader to the Appendix for a detailed examination of the code, see Section 7.

We obtain the solution set S as follows, where a and $b \in \mathbb{F}_q$:

$$\left\{ \begin{bmatrix} a & b & b \\ b & a & b \\ b & b & a \end{bmatrix}, \begin{bmatrix} a & b & b \\ b & a & b \\ b & b & a \end{bmatrix}, \begin{bmatrix} a & b & b \\ b & b & a \\ b & a & b \end{bmatrix}, \begin{bmatrix} b & a & b \\ a & b & b \\ b & b & a \end{bmatrix}, \begin{bmatrix} b & a & b \\ b & b & a \\ a & b & b \end{bmatrix}, \begin{bmatrix} b & b & a \\ b & a & b \\ a & b & b \end{bmatrix}, \begin{bmatrix} b & b & a \\ a & b & b \\ b & b & a \end{bmatrix} \right\}.$$

In order all the matrices in S to be invertible, $a, b \in \mathbb{F}_q$ must satisfy the conditions $a \neq b$ and $a \neq -2b$. Hence, the solution set will be the set K in (8). By combining this with Lemma 4.2.2, we obtain that the set K is the maximal set in $\mathrm{GL}(3, q)$ such that $E_q(3, r)$ is invariant under the transformations that come from $\mathrm{GL}(3, q)$. □

We are not able to apply the same technique used in the case $n = 2$ when $n = 3$, because we have some cumbersome identities to simplify and overcome. Instead, we prefer different approaches for the proofs when $n = 2$ and $n = 3$. Furthermore, we believe that the approach used for $n = 3$ may be adapted to the general n. Nevertheless, finding the exact set is still challenging for the general n.

We give examples of $\mathrm{SRM}_q[2, r]$ and $\mathrm{SRM}_q[3, r]$ for some q, r values, respectively.

Example 4.2.5. *Let* $q = 5$, $n = 2$, $r = 4$ *and* $(i_1, i_2) \in \{(0,1),(0,2),(0,3),(0,4),(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)\}$. *For a matrix* $\begin{bmatrix} a & c \\ b & d \end{bmatrix} \in \left\{ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 2 \\ 2 & 0 \end{bmatrix}, \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}, \begin{bmatrix} 0 & 3 \\ 3 & 0 \end{bmatrix}, \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix}, \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix}, \begin{bmatrix} 0 & 4 \\ 4 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}, \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}, \begin{bmatrix} 1 & 3 \\ 3 & 1 \end{bmatrix}, \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}, \begin{bmatrix} 2 & 4 \\ 4 & 2 \end{bmatrix}, \begin{bmatrix} 4 & 2 \\ 2 & 4 \end{bmatrix}, \begin{bmatrix} 3 & 4 \\ 4 & 3 \end{bmatrix}, \begin{bmatrix} 4 & 3 \\ 3 & 4 \end{bmatrix} \right\}$ *and* $\alpha \in \mathbb{F}_5$, *the following equation holds:*

$$\alpha\Big[(x_1^{i_1}x_2^{i_2}) - (x_1^{i_2}x_2^{i_1})\Big] = (ax_1 + bx_2)^{i_1}(cx_1 + dx_2)^{i_2} - (ax_1 + bx_2)^{i_2}(cx_1 + dx_2)^{i_1}$$

Example 4.2.6. *Let* $q = 7$, $n = 3$ *and* $r = 5$. *Then* $(i_1, i_2, i_3) \in \{(0,1,2),(0,1,3),(0,1,4),(0,2,3)\}$. *Let*

$$g_1(x_1,x_2,x_3) = x_1x_2^2 + x_2x_3^2 + x_1^2x_3 - x_1^2x_2 - x_2^2x_3 - x_1x_3^2,$$
$$g_2(x_1,x_2,x_3) = x_1x_2^3 + x_2x_3^3 + x_1^3x_3 - x_1^3x_2 - x_2^3x_3 - x_1x_3^3,$$
$$g_3(x_1,x_2,x_3) = x_1x_2^4 + x_2x_3^4 + x_1^4x_3 - x_1^4x_2 - x_2^4x_3 - x_1x_3^4,$$
$$g_4(x_1,x_2,x_3) = x_1^2x_2^3 + x_2^2x_3^3 + x_1^3x_3^2 - x_1^3x_2^2 - x_2^3x_3^2 - x_1^2x_3^3,$$

and

$$\mathcal{A} = \left\{ P \begin{bmatrix} a & b & b \\ b & a & b \\ b & b & a \end{bmatrix} \;\middle|\; P \in \mathcal{P}_3, a, b \in \mathbb{F}_7, a \neq b,\ a \neq -2b \right\}.$$

Then

$$E_7(3,5) = \{a_1g_1 + a_2g_2 + a_3g_3 + a_4g_4 \mid a_1, a_2, a_3, a_4 \in \mathbb{F}_7\}.$$

For the matrix $K \in \mathcal{A}$ *and* $g \in E_7(3,5)$, *the polynomial* $T_g(K) \in E_7(3,5)$ *by Theorem 4.2.4. Since the* $\mathrm{SRM}_7[3,5]$ *code is a type of polynomial evaluation codes, as in Example 2.5.2,* $\mathrm{SRM}_7[3,5]$ *is invariant under the corresponding transformations in* $\mathcal{A}$.

Note that the determinants of matrices in solution set must be nonzero. For example, the solution set on $\mathbb{F}_5$ for $n = 2$ does not include the element $\begin{bmatrix} 1 & 4 \\ 4 & 1 \end{bmatrix}$ because its determinant is zero. Moreover, as the parameter n increases, the values of q and r should be adjusted accordingly.

4.3 The general case

Determining all transformations under $\mathrm{GL}(n,q)$ that leaves the SRM code invariant for a general n is quite challenging. For this, there needs to be a general method to identify such transformation. Nevertheless, we can predict the solution set that leaves the SRM code

invariant for a general n and is a subgroup of the affine linear group. However, we have not established that this set may include all possible linear transformations under which SRM codes are invariant. We leave the task of finding a generalized method for this problem for future work.

5. CODES AND THEIR APPLICATIONS IN INFORMATION SECURITY

Reed-Muller codes have been highly influential in computational theory, playing a central role in several key developments across various fields. For example, in cryptography, they are utilized in computational integrity in zero-knowledge proofs, secret sharing scheme, homomorphic encryption etc. Reed-Muller codes have also been applied in error-resilient design, particularly in distributed computation, where they are employed in information dissemination algorithms for networks. Importantly, they play a crucial role in theoretical perspectives such as interpolation, linearity, partial derivatives, self-reducibility of low-degree polynomials, as well as in list decoding, local testing, and decoding. Finally, polynomials, which are fundamental objects in many computational aspects, interact closely with the study of coding-theoretic questions related to RM codes.

We are aware that numerous studies have explored the use of Reed-Muller codes in cryptography, with the volume of research in this area increasing steadily. We believe that by leveraging the advantages of SRM codes over RM codes, more efficient and practical models will emerge. Therefore, we aim to expand the scope of research in these areas and identify new open problems for further investigation.

These cryptographic applications of coding theory, in particular RM codes, have become indispensable for ensuring security in modern communication systems. The issues mentioned above will be discussed in detail below.

5.1 Zero-knowledge Proofs in Industry

In cryptography, zero-knowledge (ZK) proofs enable a prover to demonstrate the truth of a statement to a verifier without revealing any information about the statement itself [36]. A person who possesses confidential information about a situation should be able to produce

evidence related to it with ease. On the other hand, a verifier, even after being convinced of the truth of the situation, should remain unable to provide proof of it to other third parties.

Zero-knowledge proofs require interaction between the prover and the verifier [37]. This interaction involves the verifier selecting one or more random challenges. Despite this randomness, the prover's successful responses to these challenges convince the verifier that the prover possesses the claimed knowledge. If there is no interaction, the verifier could replay the protocol's execution transcript to a third party, thus convincing the third party that the verifier also possesses the confidential information. However, using the Fiat-Shamir heuristic method, non-interactive zero-knowledge proofs can be constructed [38].

Zero-knowledge proofs have three important properties:

- Completeness: If the statement in question is true, an honest verifier will be convinced by an honest prover.
- Soundness: If the statement is false, no dishonest prover can convince an honest verifier that the statement is true, except with a very small probability.
- Zero-Knowledge: If the statement is true, the verifier learns nothing other than the fact that the statement is true. This can be formalized as follows: every verifier has some simulators such that, given only the statement being proved, the simulator can produce a transcript that is indistinguishable from an interaction between an honest prover and an honest verifier.

With zero-knowledge proofs, the concept of computational integrity arises, which means the output of a specific computation is correct. This necessitates the use of a proof mechanism. Zero-knowledge proofs are a cryptographic solution that ensures both privacy and the computational integrity of the data.

Privacy is the ability of an individual to maintain control over their personal information. With the development of services such as e-voting, e-tax, and e-cash, and the widespread use of social media, the need for personal privacy and confidentiality has increased. In this

context, techniques exist to demonstrate that a secret element belongs to a public set. These techniques are known as cryptographic primitives called set membership and range proofs. Set membership proofs allow users to prove that their committed secrets belong to a public set without revealing the secrets themselves. For instance, in an e-voting system, voters can prove they have cast a valid vote without disclosing their choice. Range proofs, on the other hand, are a special case of set membership where the public set is a large range of integers. Typically, range proofs are used in conjunction with electronic identities to enforce age restrictions [39].

KYC procedures can be performed using zk-set membership without revealing user information. For example, an individual can prove that their country of residence is within the European Union without disclosing the specific country. It will also be possible to verify whether the individual is on the white or black lists established for anti-money laundering (AML) purposes between countries.

In a resource-constrained environment, one of the use cases of zero-knowledge proofs is to outsource computations to an external, powerful, but untrusted source. At this point, one of the emerging technologies is layer 2 solutions in blockchain technology. Transactions that take place on layer 2, outside the main chain, are proven on the main chain using zero-knowledge proofs. This process is called zk-rollup, and the validity proof created after off-chain transactions, along with the resulting state change, significantly reduces transaction costs on the main chain. In these transactions, zero-knowledge proofs are used to ensure that the transactions are performed correctly and are valid, rather than to ensure privacy [40].

Zero-knowledge proofs are used to address privacy concerns in distributed systems [41–43]. For example, Zerocash was designed to prevent the traceability of transactions [43]. In this system, both user and transaction information are anonymous. zk-SNARKs are used to ensure anonymity. It is a privacy-focused cryptocurrency inspired by the 2014 work Cryptonote [44]. In Monero [45], users are untraceable, and both the sender and receiver are anonymous. To ensure the anonymity of transaction amounts, range proofs are used, and or-proofs are employed to verify the correctness of the committed values.

In blockchain ecosystem, smart contracts are pieces of code executed by all participants in the Ethereum network. All information in smart contracts can be viewed by other parties. The lack of a privacy mechanism in the system is a major issue. This issue can be resolved by using zk-SNARK or zk-STARK. Additionally, there is a special smart contract called Hawk. However, each contract requires a new setup and a trusted manager who can access the user's information. In Bulletproofs, there is no need for a trusted setup, and the proof sizes are smaller, making them suitable for use in private smart contracts. Zether [46] is a decentralized, privacy-based payment mechanism compatible with Ethereum and other smart contract platforms. For efficiency and usability, an account-based approach similar to Ethereum has been followed. In this mechanism, which keeps account balances encrypted and controls deposit, transfer, and withdrawal operations based on encrypted balances, Σ-protocols are used for zero-knowledge proofs. Ciphertexts encrypted with ElGamal encryption are proven to be within a certain range using Bulletproofs.

Post-quantum zero-knowledge proofs are also being researched today. Benoît Libert et.al. [47] worked on a lattice-based zk-range proof structure; however, the proof size is very large. Even if the secret is small, the proof does not reduce in size. Therefore, optimization efforts have begun.

Stark FRI AG codes in FRI information security

Public trust demands transparency from ZK systems, meaning these systems must be established without relying on any trusted party and ensure that powerful entities cannot provide false testimony. For ZK systems to be used with big data, the public verification process must scale sub-linearly with respect to the size of the data. In the 1990s, transparent ZK proofs that could be verified exponentially faster than the size of the data were defined, but no practical implementations were achieved. To date, no ZK system implemented in code has managed to achieve both transparency and exponential verification speedup for general computations simultaneously. In their work, Eli Ben-Sasson et al. [48] claim to have realized for the first time a transparent ZK system (ZK-STARK) where the verification scales

exponentially faster than the size of the data and exhibits exponential verification speed-up for certain computations. Their system utilizes developments in interactive oracle proofs (IOP), akin to fast (linear-time) IOP systems for error-correcting codes.

In their work, Eli Ben-Sasson et al. presented a scalable and transparent ZK system within the IOP model. They improved the verification time and ensured that the communication complexity is smaller than the witness size. The main source of innovation and performance improvement in this system is the fast Reed-Solomon IOP (FRI) protocol. However, it has been understood that some of the main components of such systems require long verification times and work efficiently only asymptotically for large computations. In the original ZK-STARK algorithm, Reed-Solomon error-correcting codes were used as the error-correcting codes for the verifier. In their work, Eli Ben-Sasson et al. [49] suggested using AG (algebraic geometry) codes instead of these codes.

Computational-integrity (CI) is a crucial aspect to be considered in terms of accountability. However, scalability and privacy emerge as two major problems. This is because scaling performance to meet continuously increasing demand will inevitably prevent some participants from verifying integrity due to limited computational resources. On the other hand, without cryptographically blinding information, making all blockchain transactions publicly available is unacceptable for both businesses and individuals. Methods to address these two problems have been studied for a long time. From the 1980s to the 1990s, theoretical work on interactive proofs, zero-knowledge proofs (ZKP), and probabilistically-checkable-proofs (PCP) has addressed how to tackle these two problems.

The first theoretical work on zero-knowledge proofs began in the early 1990s with discussions on PCP. The PCP theorem establishes a balance between the prover's time to generate the proof and the verifier's time to verify it. Zero-knowledge proofs based on PCP have fundamental advantages regarding computational integrity.

The PCP theorem is one of the most important topics in complexity theory. PCP (probabilistically-checkable-proof) is a proof system that allows the validity of a claim to be checked by querying only a small portion of the proof [50]. The PCP theorem, in

computational complexity theory, is a type of probabilistically checkable proof. It checks the proof by using a limited amount of randomness and reading a limited number of its bits. The algorithm must accept correct proofs and reject incorrect proofs with very high probability.

What makes PCP interesting is the existence of probabilistically checkable proofs, which can be verified by reading only a few random bits of the proof. This theorem has provided lower bounds for constraint satisfaction problems [51].

The PCP theorem states that every NP-proof can be encoded into another proof, known as a probabilistically checkable proof (PCP), which can be tested by a verifier querying only a small portion of it. However, a significant issue is the size of the overhead introduced by this encoding. Specifically, the problem is comparing the length of the encoded PCP with the original NP-proof. Even with the additional overhead in PCP protocols, good performance has been achieved using Reed-Solomon error-correcting codes [50].

In their work, Eli Ben-Sasson et al. [49] explored the idea of replacing low-degree polynomials in PCP structures with tensors of transitive algebraic geometry (AG) codes, where AG codes generalize Reed-Solomon and Reed-Muller codes. They demonstrated that the automorphisms of an AG code can be used to simulate the role of affine transformations in earlier algebraic PCP constructions. Using this observation, they concluded that any family of transitive AG codes that is asymptotically good over a fixed-size alphabet produces a family of constant-rate PCPs with low query complexity in terms of polynomial degree.

In this study [49], AG codes were utilized because they share the multiplication property used in the PCP arithmetization of Reed-Solomon and Reed-Muller codes. Although many error-correcting codes exist, most of them lack other properties of Reed-Solomon codes, such as systematicity and polynomial closure. Most importantly, the key feature of Reed-Solomon and Reed-Muller codes is that they possess a transitive automorphism group. Therefore, the idea of replacing Reed-Solomon codes with AG codes that have the same desirable properties while also achieving a constant rate was considered. When integrating AG codes into the

PCP structure, it is sufficient to work with AG codes that have a transitive automorphism group.

Here, we need to provide the definition of the doubly transitive property, which is a degree of transitivity. A code C has an automorphism group $\mathrm{Aut}(C)$, which is the group of permutations $\pi \in \mathrm{Aut}(C)$ that keeps the code invariant, i.e., for $f \in C$, it holds that $(f \circ \pi) \in C$. For each $i \in D$, the codeword $(f \circ \pi)$ is defined by $w(i) \stackrel{\Delta}{=} w(\pi(i))$. If for any two distinct pairs of elements $(i, j), (i', j') \in D^2$, there exists a $\pi \in G$ that maps i to i' and j to j', then G acts doubly transitively on D.

RM codes are doubly transitive codes, since RM codes are affine-invariant. The RM code family remains fixed under any invertible affine transformation, and the degree of the functions generating the codewords are also invariant under such transformations. One motivation for studying the doubly transitive property is its impact on the code rate under fixed alphabet size and query complexity, which is relatively low. To date, there are no known doubly transitive codes other than RM codes. Although certain code families have highly rich automorphism groups, they do not exhibit this property. However, these codes can be characterized by giving the degree to which they approach the doubly transitive property. The query complexity of AG codes with a nearly doubly transitive automorphism group can be reduced from $|D|^2$ to $|D|$. In fact, the degree of the doubly transitive property of a code family also indicates the degree of its locally correctable property. Doubly transitive AG codes are also locally correctable. Locally correctable codes are error-correcting codes where any symbol in the codeword can be verified using a few other randomly chosen symbols. The doubly transitive property is used to demonstrate the local correctability of tensor codes [52].

At this point, we focused on families of AG codes that possess a transitive automorphism group or doubly transitive automorphism group. Specifically, we studied symmetric Reed-Muller codes within the AG code family and attempted to increase the knowledge on the automorphism group of symmetric Reed-Muller codes.

5.2 Homomorphic Encryption

Fully Homomorphic Encryption (FHE) is considered a solution for preserving privacy and security in computations on encrypted data. In 2009, Gentry proposed the first fully homomorphic encryption (FHE) system [53]. Following this, extensive research has been conducted on homomorphic encryption schemes based on lattice-based hard problems. One of the most notable works is the Cheon-Kim-Kim-Song (CKKS) scheme, which allows approximate homomorphic encryption. However, even in this scheme, the computational complexity remains very high, such that performing homomorphic operations can take days to complete. This emphasizes the need for further advancements to make FHE more practical in real-world applications. Current FHE schemes are predominantly constructed using lattice-based cryptography. However, homomorphic multiplications and refreshing the ciphertext require a significant amount of computational resources. Therefore, there is a need for new methods that can reduce computational complexity in practical applications of FHE. Various solutions have been proposed in the literature to address this challenge. One of these approaches is to tackle the problem through a code-based homomorphic operation scheme.

While linear codes are closed under addition, they are not closed under homomorphic multiplication. The proposed solution addresses this limitation by using Reed-Muller (RM) codes, which support both addition and multiplication. With this method, it is crucial to preserve the rank of RM codes after addition or multiplication operations. To achieve this, a bootstrapping technique is suggested, which creates a one-to-one correspondence between computations on messages and computations on RM codewords. This technique ensures that the integrity of the RM codes is maintained during the homomorphic operations. The potential of symmetric Reed-Muller codes to reduce noise in homomorphic encryption will be explored in future studies.

5.3 Secret Sharing

For a protocol to be secure, the sensitive information it contains must be stored reliably and confidentially. Examples of such sensitive information include private keys in public

key systems. One of the primary motivations for secret sharing is the reconstruction of a cryptographic key in case it is lost. In real-life scenarios, there are situations where a group of people needs access to confidential information. In such cases, only trusted members of the group should have access to this secret information. Additionally, when storing highly sensitive data, it may be necessary to split it into parts and store these parts in different locations. Each piece of data should provide as little information as possible about the entire set of data, and only when a minimum number of pieces are combined should the original data be recoverable. This significantly increases the effort required for an adversary to obtain the full confidential information.

A secret sharing scheme is a method for securely sharing a secret among a group of participants. For a given secret, the so-called dealer calculates appropriate shares and distributes them to the participants. The shares will only allow pre-defined subsets of participants to reconstruct the secret from their shares. These subsets are called authorized, and the set of all authorized subsets is referred to as the access structure. The remaining subsets are unauthorized and should learn as little as possible about the secret from their shares.

In 1979, Shamir [54] introduced a threshold signature scheme in which all participants receive shares that represent parts of a secret as a polynomial. In this scheme, there is no hierarchy among participants, meaning each participant's share has equal importance in reconstructing the secret. Independently of Shamir's work, Blakley [55] conducted similar research in the same year. The schemes developed in both works are referred to as threshold access structures, where any subset of participants that meets the threshold value is capable of reconstructing the secret. In 1991, Simon [56] employed a monotone access structure in his study. In these access models, all supersets of authorized sets are also authorized. In the following years, studies began on access models in which the pieces held by the participants had different weightings as they came together to form the secret. These models are generally referred to as hierarchical threshold schemes [57–59]

In the study [60], a novel secret sharing scheme based on binary error-correcting codes is

presented, which can implement arbitrary access structures. In this secret sharing scheme, the secret is a codeword in a binary error-correcting code, and the shares are binary words of the same length. When a group of participants wants to reconstruct the secret, they compute the sum of their shares and apply Hamming decoding to the sum. The feature of the shares is that when the group is authorized, the secret corresponds to the codeword closest to the sum of the shares. Otherwise, the sum results in a different codeword through the Hamming decoding process. The shares can be described as solutions to a system of linear equations that is closely related to first-order Reed-Muller codes. Additionally, the access structure model in this study [60] was constructed through Reed-Muller codes. In another study [25], multiplicative linear secret sharing schemes were examined. Such schemes can be defined through linear codes. In this study, a more general class of Reed-Muller type codes, suitable for multiparty computation, was presented.

There are many studies on error-correcting codes in the context of secret sharing. One of the open problems we leave for future work is the advantages and disadvantages that the use of SRM codes may bring to schemes inspired by these studies. This is because we believe that codes with efficient encoding and decoding capabilities will lead to the development of effective schemes and provide good privacy threshold for secret sharing.

6. CONCLUSION AND FUTURE WORK

Our work aims at determining the set of affine-invariant transformations. The linear automorphism groups of SRM for $n = 2$ and $n = 3$ over the field $\mathbb{F}_p$, where p is any prime number is proven in this study. For $n = 2$ and $n = 3$, we find that the exact set generated by transformations remaining SRM codes invariant is a subgroup of the affine linear group. For different values of n, different techniques were used to determine this set. We give the essential lemmas for our main theorem, which identifies the group created by transformations leaving the SRM code invariant.

This study has offered valuable insights into affine invariant transformations on SRM codes; however, there exists an unresolved question that calls for further investigation in future research. We could not give a general proof for an arbitrary n, and leave it an open problem of the complete determination of the automorphism group $\mathrm{Aut}(\mathrm{SRM})$ for any $n > 3$. We state the problem left for future researches as follows and include our prediction regarding set M associated with the solution of this problem :

Let J_n be the $n \times n$ all one matrix, I_n be the $n \times n$ identity matrix and $\mathcal{P}_n$ be the set of permutations of order n. Let M be a subset of $\mathrm{GL}(n, q)$ defined as $M = \left\{P((b-a)I_n + aJ_n) \mid P \in \mathcal{P}_n, a, b \in \mathbb{F}_q, a \neq b, a \neq (1-n)b\right\} \subset \mathrm{GL}(n, q)$. Then, the automorphism group of the $\mathrm{SRM}_q[n, r]$ for $q > r > \frac{n(n-1)}{2}$ contains a subgroup isomorphic to M, i.e., $\mathrm{SRM}_q[n, r]$ is invariant under the transformations in M.

The proof of the invariance of $\mathrm{SRM}_q[n, r]$ codes under the transformations, which come from the set M, may be similarly done to that of Lemma 4.2.2. Notwithstanding, in order to show that the set M is the complete set in this manner is quite challenging to follow the same techniques.

On the other hand, error correction codes have numerous applications across various fields, some of which have been discussed above. In particular, Reed-Muller codes have significantly accelerated research in terms of efficiency, speed, and accuracy by being integrated into new technologies. Ongoing studies on these codes span a wide range

of applications, from data privacy to satellite technologies. Improvements made through research on Reed-Muller codes are expected to enhance the technologies that utilize these codes.

At this juncture, we anticipate obtaining better results in applications through the use of SRM codes. It is well-known that employing codes with transitive automorphism groups in proof systems will lead to more efficient proofs [49]. Therefore, we believe that SRM codes will prove beneficial in this context. To this end, we have investigated the automorphism groups of SRM codes. This raises an open question regarding the transitivity of this group. Once the transitivity of the automorphism group of SRM codes is established, we expect that their application in proof systems will yield even greater efficiency. Consequently, this will lead to significant advancements across all fields where proof systems are employed, resulting in exponential improvements in areas such as e-voting, homomorphic encryption, secret sharing, and other related domains.

REFERENCES

[1] W. C. Huffman and V. Pless. *Fundamentals of error correcting codes*. Cambridge University Press, **2003**.

[2] F. J. MacWilliams and N. J. A. Sloane. *The theory of error correcting codes*, volume 16. Elsevier, **1977**.

[3] P. Delsarte, J. M. Goethals, and F. J. MacWilliams. On generalized reed-muller codes and their relatives. *Information and control*, 16(5):403–442, **1970**.

[4] V. Pless, R. A. Brualdi, and W. C. Huffman. *Handbook of coding theory*. Elsevier Science Inc., **1998**.

[5] W. Yan and S. J. Lin. Symmetric reed–muller codes. *IEEE Transactions on Communications*, 68(7):3937–3947, **2020**.

[6] T. Kasami, S. Lin, and W. Peterson. Some results on cyclic codes which are invariant under the affine group and their applications. *Information and Control*, 11(5-6):475–496, **1967**.

[7] P. Delsarte. On cyclic codes that are invariant under the general linear group. *IEEE Transactions on Information Theory*, 16(6):760–769, **1970**.

[8] A. Dür. The automorphism groups of reed-solomon codes. *Journal of Combinatorial Theory, Series A*, 44(1):69–82, **1987**.

[9] T. P. Berger and P. Charpin. The automorphism group of generalized reed-muller codes. *Discrete mathematics*, 117(1-3):1–17, **1993**.

[10] T. P. Berger and P. Charpin. The permutation group of affine-invariant extended cyclic codes. *IEEE transactions on Information theory*, 42(6):2194–2209, **1996**.

[11] T. P. Berger. Automorphism groups of homogeneous and projective reed-muller codes. *IEEE Transactions on Information Theory*, 48(5):1035–1045, **2006**.

[12] W. Yan and S. J. Lin. Local correctabilities and dual codes of symmetric reed–muller codes. In *2021 IEEE Information Theory Workshop (ITW)*, pages 1–5. IEEE, **2021**.

[13] D. E. Muller. Application of boolean algebra to switching circuit design and to error detection. *Transactions of the IRE professional group on electronic computers*, (3):6–12, **1954**.

[14] I. S. Reed. A class of multiple-error-correcting codes and the decoding scheme. *IEEE Transactions on Information Theory*, 4(4):38–49, **1954**.

[15] E. Abbe, O. Sberlo, A. Shpilka, M. Ye, et al. Reed-muller codes. *Foundations and Trends® in Communications and Information Theory*, 20(1–2):1–156, **2023**.

[16] R. Knörr and W. Willems. The automorphism groups of generalized reed-muller codes. *Astérisque*, 181:182, **1990**.

[17] T. P. Berger. On the automorphism groups of affine-invariant codes. *Designs, codes and cryptography*, 7:215–221, **1996**.

[18] V. Lint and J. Hendricus. *Introduction to coding theory*, volume 86. Springer Science & Business Media, **1998**.

[19] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, 8(2):300–304, **1960**.

[20] Raj Chandra Bose and Dwijendra K Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and control*, 3(1):68–79, **1960**.

[21] Alexis Hocquenghem. Codes correcteurs d'erreurs. *Chiffers*, 2:147–156, **1959**.

[22] E. Abbe, A. Shpilka, and M. Ye. Reed–muller codes: Theory and algorithms. *IEEE Transactions on Information Theory*, 67(6):3251–3277, **2020**.

[23] T. Kasami, S. Lin, and W. Peterson. New generalizations of the reed-muller codes–i: Primitive codes. *IEEE Transactions on information theory*, 14(2):189–199, **1968**.

[24] J. Cho, Y.-S. Kim, and J. S. No. Homomorphic computation in reed-muller codes. *IEEE Access*, 8:108622–108628, **2020**.

[25] I. Duursma and J. Shen. Multiplicative secret sharing schemes from reed-muller type codes. In *2012 IEEE International Symposium on Information Theory Proceedings*, pages 264–268. IEEE, **2012**.

[26] R. Pellikaan and X. W. Wu. List decoding of q-ary reed-muller codes. *IEEE Transactions on Information Theory*, 50(4):679–682, **2004**.

[27] N. Alon, T. Kaufman, M. Krivelevich, S. Litsyn, and D. Ron. Testing reed-muller codes. *IEEE Transactions on Information Theory*, 51(11):4032–4039, **2005**.

[28] S. Yekhanin et al. Locally decodable codes. *Foundations and Trends® in Theoretical Computer Science*, 6(3):139–255, **2012**.

[29] E. F. Assmus and J. D. Key. *Designs and their Codes*. 103. Cambridge University Press, **1992**.

[30] A. B. Sorensen. Projective reed-muller codes. *IEEE Transactions on Information Theory*, 37(6):1567–1576, **1991**.

[31] P. Beelen, S. R. Ghorpade, and T. Hoholdt. Affine grassmann codes. *IEEE Transactions on Information theory*, 56(7):3166–3176, **2010**.

[32] A. M. Steane. Quantum reed-muller codes. *IEEE Transactions on Information Theory*, 45(5):1701–1703, **1999**.

[33] M. Bhaintwal and S. K. Wasan. Generalized reed–muller codes over. *Designs, Codes and Cryptography*, 54(2):149–166, **2010**.

[34] E. Weldon. New generalizations of the reed-muller codes–ii: Nonprimitive codes. *IEEE Transactions on Information Theory*, 14(2):199–205, **1968**.

[35] SageMath. *The Sage Mathematics Software System (Version 9.3)*, **2021**. `https://www.sagemath.org`.

[36] V. Mulder, A. Mermoud, V. Lenders, and B. Tellenbach. *Trends in Data Protection and Encryption Technologies*. Springer Nature, **2023**.

[37] O. Goldreich. *Foundations of Cryptography, Volume 2*. Cambridge university press Cambridge, **2004**.

[38] M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications. In *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 329–349. **2019**.

[39] Eduardo Morais, Tommy Koens, Cees Van Wijk, and Aleksei Koren. A survey on zero knowledge range proofs and applications. *SN Applied Sciences*, 1:1–17, **2019**.

[40] Jens Ernstberger, Stefanos Chaliasos, Liyi Zhou, Philipp Jovanovic, and Arthur Gervais. Do you need a zero knowledge proof? *Cryptology ePrint Archive*, **2024**.

[41] Ian Miers, Christina Garman, Matthew Green, and Aviel D Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *2013 IEEE symposium on security and privacy*, pages 397–411. IEEE, **2013**.

[42] Ahmed Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *2016 IEEE symposium on security and privacy (SP)*, pages 839–858. IEEE, **2016**.

[43] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous

payments from bitcoin. In *2014 IEEE symposium on security and privacy*, pages 459–474. IEEE, **2014**.

[44] Nicolas Van Saberhagen. Cryptonote v 2.0. **2013**.

[45] Malte Möser, Kyle Soska, Ethan Heilman, Kevin Lee, Henry Heffan, Shashvat Srivastava, Kyle Hogan, Jason Hennessey, Andrew Miller, Arvind Narayanan, et al. An empirical analysis of traceability in the monero blockchain. *arXiv preprint arXiv:1704.04299*, **2017**.

[46] Benedikt Bünz, Shashank Agrawal, Mahdi Zamani, and Dan Boneh. Zether: Towards privacy in a smart contract world. In *International Conference on Financial Cryptography and Data Security*, pages 423–443. Springer, **2020**.

[47] Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Lattice-based zero-knowledge arguments for integer relations. In *Annual International Cryptology Conference*, pages 700–732. Springer, **2018**.

[48] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. Scalable, transparent, and post-quantum secure computational integrity. *Cryptology ePrint Archive*, **2018**.

[49] E. Ben-Sasson, Y. Kaplan, S. Kopparty, O. Meir, and H. Stichtenoth. Constant rate pcps for circuit-sat with sublinear query complexity. *Journal of the ACM (JACM)*, 63(4):1–57, **2016**.

[50] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM (JACM)*, 45(3):501–555, **1998**.

[51] S. Arora. How np got a new definition: a survey of probabilistically checkable proofs. *arXiv preprint cs/0304038*, **2003**.

[52] Y. Kaplan and E. Ben-Sasson. *Multi-variate Abstractions of Algebraic Geometry Codes, With Applications*. Ph.D. thesis, Computer Science Department, Technion, **2016**.

[53] Craig Gentry. *A fully homomorphic encryption scheme*. Stanford university, **2009**.

[54] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, **1979**.

[55] George Robert Blakley. Safeguarding cryptographic keys. In *Managing requirements knowledge, international workshop on*, pages 313–313. IEEE Computer Society, **1979**.

[56] Gustavus J Simmons. Geometric shared secret and/or shared control schemes. In *Conference on the Theory and Application of Cryptography*, pages 216–241. Springer, **1990**.

[57] Tamir Tassa. Hierarchical threshold secret sharing. *Journal of cryptology*, 20:237–264, **2007**.

[58] Mihir Bellare, Alexandra Boldyreva, and Jessica Staddon. Randomness re-use in multi-recipient encryption schemeas. In *Public Key Cryptography—PKC 2003: 6th International Workshop on Practice and Theory in Public Key Cryptography Miami, FL, USA, January 6–8, 2003 Proceedings 6*, pages 85–99. Springer, **2002**.

[59] Muhammed Ali Bingol, Sermin Kocaman, Ali Dogan, and Sibel Kurt Toplu. Flexhi: A flexible hierarchical threshold signature scheme. In *Science and Information Conference*, pages 509–529. Springer, **2024**.

[60] Claudia Kässer. *Secret sharing schemes based on error-correcting codes*. Ph.D. thesis, Universität Tübingen, **2016**.

7. Appendix

```
reset()
forget(assumptions())

# Define the variables
x,y,z = var('x y z')

# Define the expression T
a,b,c,d, e, f, g, h, i = var('a b c d e f g h i')

X = a*x+b*y+c*z
Y = d*x+e*y+f*z
Z = g*x+h*y+i*z

D1 = matrix([[1,1,1],[X,Y,Z],[X^2,Y^2,Z^2]]).determinant()
D1s = (D1.full_simplify())

D2 = matrix([[1,1,1],[X,Y,Z],[X^3,Y^3,Z^3]]).determinant()
D2s = (D2.full_simplify())
#print(D1s)

eqns1 =
    ↪  [D1s.coefficient(x*y*z),D1s.coefficient(x^3),D1s.coefficient(y^3),
    ↪  \\D1s.coefficient(z^3),D1s.coefficient(x*y^2)+ \\
    ↪  D1s.coefficient(x^2*y),D1s.coefficient(z*y^2)+ \\
    ↪  D1s.coefficient(z^2*y),D1s.coefficient(x*z^2)+ \\
    ↪  D1s.coefficient(x^2*z)]
```

```
#show(eqns1)

eqns2 = [D2s.coefficient(x*y*z^2),D2s.coefficient(x*y^2*z),\\
    ↪  D2s.coefficient(x^2*y*z), D2s.coefficient(x^2*y^2),\\
    ↪  D2s.coefficient(x^2*z^2), D2s.coefficient(z^2*y^2),\\
    ↪  D2s.coefficient(x^4),D2s.coefficient(y^4),D2s.coefficient(z^4),\\
    ↪  D2s.coefficient(x*y^3)+D2s.coefficient(x^3*y),\\
    ↪  D2s.coefficient(z*y^3)+D2s.coefficient(z^3*y),\\
    ↪  D2s.coefficient(x*z^3)+D2s.coefficient(x^3*z)]
#show(eqns2)
eqns = eqns1+eqns2
print("------------------------------")
# for eq in eqns:
#     show(eq)
#     #print(eq(a=2,b=1,c=1,d=1,e=2,f=1,g=1,h=1,i=2))
#     print("------------------------------")
det=matrix([[a,b,c],[d,e,f],[g,h,i]]).determinant()

sol =
    ↪  solve([eqns[1]==0,eqns[2]==0,eqns[13]==0,eqns[14]==0,eqns[4]==0,
eqns[16]==0],a,b,d,e,g,h)
#print(sol)
sol_dict =
    ↪  solve([eqns[1]==0,eqns[2]==0,eqns[13]==0,eqns[14]==0,eqns[4]==0,
eqns[16]==0],a,b,d,e,g,h,solution_dict=True)

# #sol=solve([eqns[1],eqns[13]],a,d,g)
# sol =
    ↪  solve([eqns[1]==0,eqns[2]==0,eqns[3]==0,eqns[13]==0,eqns[14]==0,
```

```
42 eqns[15]==0,eqns[4]==0,eqns[16]==0],a,b,c,d,e,f,g,h,i)
43 # sol_dict =
   ↪ solve([eqns[1]==0,eqns[2]==0,eqns[3]==0,eqns[13]==0,eqns[14]==0,
44 eqns[15]==0,eqns[4]==0,eqns[16]==0],a,b,c,d,e,f,g,h,i,
45 solution_dict=True)
46 # #sol= solve(eqns,a,b,c,d,e,f,g,h,i)
47 # #print(len(sol),sol)
48
49 eqns_ex = [eqns[0],eqns[3],eqns[15]]+eqns[5:13]+eqns[17:19]
50
51 for j in range(len(sol_dict)):
52     if ((det(sol_dict[j])).full_simplify())!=0:
53         print(j,sol_dict[j])
54         sol_g =
           ↪ solve(eqns_ex+sol[j],a,b,c,d,e,f,g,h,i,solution_dict=True)
55         #print(sol_g)
56         for k in range(len(sol_g)):
57             if ((det(sol_g[k])).full_simplify())!=0:
58                 print(j,k,sol_g[k])
59                 print("----------------------")
60
61 #         #eqns_ex_subt=[eqns_ex[k](sol[j]) for k in
   ↪ range(len(eqns_ex))]
62 #         #print(eqns_ex_subt[0])
63 #         #print(solve(eqns_ex_subt,a,b,c,d,e,f,g,h,i))
64
65 # #sol_p = [e==a,i==a,c==b,d==b,f==b,g==b,h==b]
66 # #solve(eqns+sol_p,a,b,c,d,e,f,g,h,i)
```