KOD TABANLI VE KAFES TABANLI KRİPTOGRAFİK SİSTEMLERİN CEBİRSEL-GEOMETRİK AÇIDAN İNCELENMESİ VE KARŞILAŞTIRMASI

ALGEBRAIC-GEOMETRIC EXAMINATION AND COMPARISON OF CODE-BASED AND LATTICE-BASED CRYPTOGRAPHIC SYSTEMS

BARIŞ SUFRACI

PROF. DR. MESUT ŞAHİN

Tez Danışmanı

Hacettepe Üniversitesi

Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliğinin

Matematik Anabilim Dalı için Öngördüğü

YÜKSEK LİSANS TEZİ olarak hazırlanmıştır.

2025

ÖZET

KOD TABANLI VE KAFES TABANLI KRİPTOGRAFİK SİSTEMLERİN CEBİRSEL-GEOMETRİK AÇIDAN İNCELENMESİ VE KARŞILAŞTIRMASI

Barış SUFRACI

Yüksek Lisans, MATEMATİK Bölümü

Tez Danışmanı: Prof. Dr. Mesut ŞAHİN

Haziran 2025, 282 sayfa

Bu tezde, kuantum-sonrası algoritmalardan en yoğun cebir ve geometri bilgisi gerektiren çeşitler olarak kod-tabanlı ve kafes-tabanlı algoritmalar çalışılmıştır.

Tezin ilk kısmı giriş bölümüdür. Bu bölümde kriptografinin önemi ve tarihsel gelişiminden bahsedilmiştir.

Kriptografiyi ve özellikle kuantum-sonrası kriptografiyi anlamak için fazlaca matematiksel konuya hakim olmak gerektiğinden, tezin ikinci bölümünde, ihtiyaç duyacağımız matematik konularının geniş özetleri anlatılmıştır. Bu konulardan başlıcaları; sayılar teorisi, polinomlar, sonlu cisimler, matrisler, kodlar ve kafeslerdir.

Kuantum-sonrası kriptografiyi daha iyi anlamak için anahtar, açık metin ve şifreli metin gibi kavramları kriptografik algoritmalar üzerinde görüp özümsemek gerekir. Bu özümseme işini, gelişmiş yapıları sebebiyle, doğrudan kuantum-sonrası algoritmalar üzerinden yapmak oldukça zor olacağı için tezin üçüncü bölümünde klasik kriptografi anlatılmıştır.

Kuantum-sonrası algoritmalara neden ihtiyaç duyduğumuz hususundaki gerçekleri anlamak için önce kuantum bilgisayarların neden normal bilgisayarlara kıyasla çok daha hızlı olduğunu bilmek gerekir. Bu sebeple tezin dördüncü bölümünde bu farktan bahsedilerek başlanmıştır. Ardından, bazı kod aileleri ve bu kod ailelerini kullanan kod-tabanlı algoritmaların işlem süreçleri anlatılmıştır. Daha sonra kafes yapılarından bahsedilip bu yapıyı kullanan birkaç algoritmanın işleyişi anlatılmıştır.

Anahtar Kelimeler: Şifreleme, Deşifreleme, Kriptografi, Kuantum-Sonrası Kriptografi, Kod, Kafes, Kod-Tabanlı, Kafes-Tabanlı

ABSTRACT

ALGEBRAIC-GEOMETRIC EXAMINATION AND COMPARISON OF CODE-BASED AND LATTICE-BASED CRYPTOGRAPHIC SYSTEMS

Barış SUFRACI

Master of Science, Department of MATHEMATICS

Supervisor: Prof. Dr. Mesut ŞAHİN

June 2025, 282 pages

In this thesis, code-based and lattice-based algorithms, which are the types of post-quantum algorithms that require the most intensive knowledge of algebra and geometry, have been studied.

The first part of the thesis is the introduction. In this section, importance and historical development of the cryptography are mentioned.

Since it is necessary to have a good command of many mathematical subjects to understand cryptography and especially post-quantum cryptography, the second part of the thesis provides extensive summaries of the mathematical subjects we will need. The main topics are number theory, polynomials, finite fields, matrices, codes and lattices.

In order to better understand post-quantum cryptography, it is necessary to see and internalize concepts such as key, plaintext and ciphertext on cryptographic algorithms. Since it would be very difficult to do this internalization directly on post-quantum algorithms due to their advanced structures, classical cryptography is explained in the third part of the thesis.

To understand the facts about why we need post-quantum algorithms, we must first know why quantum computers are much faster than normal computers. For this reason, the fourth chapter of the thesis begins by mentioning this speed difference. Then, some code families and the process of code-based algorithms using these code families are explained. Then, lattice structures are mentioned and the operation of several algorithms using this structure is explained.

Keywords: Encryption, Decryption, Cryptography, Post-Quantum Cryptography, Codes, Lattices, Code-Based, Lattice-Based

TEŞEKKÜR

Bilmediklerimin aslında eksiklerim değil yol göstericilerim olduğu bilincini bana vererek bu uzun ve zorlu yolda beni cesaretlendiren, tez yazım sürecinde ihtiyaç duyduğum yardımı bana fazlasıyla sunan, mesleki ve insani açıdan tutum ve davranışlarını örnek aldığım saygıdeğer hocam Prof. Dr. Mesut ŞAHİN'e;

Yürümek, konuşmak, sevmek ve saymak gibi en temel özelliklerim de dahil bana bildiklerimin çoğunu öğreten, hayatım boyunca benden maddi ve manevi hiçbir desteklerini esirgemeyen, ilk öğretmenlerim annem Emine SUFRACI, babam Göksel SUFRACI ve ablam Sevgi SUFRACI'ya;

Kriptografi ile tanışmamı sağlayan sayın hocam Doç. Dr. Oğuz YAYLA ve tanışıklığımı ileriye taşımamda büyük katkıları olan sayın hocam Dr. Öğr. Üyesi Talha ARIKAN'a;

Tez yazım üslubumun mimarları olan, çalışma azimleri ve enerjileriyle beni daima dinç tutan manevi öğrencilerim Sebahattin Emre ŞAHİN ve Hasan BİRCAN'a;

Motivasyon kaybı yaşadığım zamanlarda tekrar motive olmamı sağlayan ve yardıma ihtiyaç duyduğum anlarda hep yanımda olan çok sevgili arkadaşlarım Mehmet Fatih TEMİRCİ, Ali GÖK, İshak TEKİN, Mehmet Sait CALAYIR, Mustafa SOLMAZ, Soltan ZEYNALLI, Abdilkadir Efe ŞAHBAZ, Rəvan ZEYNALLI, Murat Erşen ÜNAL, Emre Can ERDEM ve Elif CAN'a;

2210 Yurt İçi Yüksek Lisans Burs Programı kapsamında beni destekleyerek maddi kaygılardan arındıran TÜBİTAK'a;

SONSUZ TEŞEKKÜRLER...

Barış SUFRACI

Haziran 2025

İÇİNDEKİLER

Sayfa

ŞEKİLLER

Sayfa

SİMGELER VE KISALTMALAR

$\mathbb{F}_q$	:	q Elemanlı Cisim
der	:	Derece
kar	:	Karakteristik
$L(V)$	:	V ile Üretilen L Kafesi
$C^{\perp}$	:	C Kodunun Duali
$dist$	:	Uzaklık
$[n, k, d]_q$	:	n, k, d, q Parametreli Kod
$w(c)$	:	c vektörünün ağırlığı
$\oplus$	:	XOR Toplama
$S(y)$	:	y Vektörünün Sendromu
$\Gamma(L, g)$	:	L ve g Parametreli Goppa Kod
$\mathcal{R}(r, m)$	:	r ve m Parametreli Reed-Muller Kod
MDS	:	Maksimum Distance Separable
QC	:	Quasi Cyclic
LRPC	:	Low Rank Parity Check
MDPC	:	Moderate Density Parity Check
HQC	:	Hamming Quasi Cyclic
SVP	:	Shortest Vector Problem
CVP	:	Closest Vector Problem
LWE	:	Learning With Errors
PKE	:	Public Key Encryption
KEM	:	Key Encapsulation Mechanism

1. GİRİŞ

Bu kısımda kriptografinin tarihsel gelişimini anlatan özet bilgilere yer verilerek teze bir giriş yapılmıştır. Ayrıntılı bilgiler için [1], [2], [3], [4], [5], [6] ve [7] numaralı kaynaklar incelenebilir.

İletişim kavramının kökleri insanlık tarihinden de öncesine uzanır. Besin zincirinin alt katmanlarında bulunan canlılar, kendi aralarındaki iletişim yöntemlerini, daha üst katmanlarda bulunan canlıların fark edemeyeceği şekillerde evrimleştirme gayretiyle geliştirmişlerdir. Bu durum tam olarak bir kriptografik örnek olarak verilemese de, iletişimi gizlemek kısmı baz alındığında "kriptografimsi" bir durum sayılabilir. İnsanlığın başlangıcıyla beraber, dünya üzerinde var olan iletişim özellikleri, zeka dokunuşları sayesinde daha da gelişmiştir ve yaşayan insan sayısı arttıkça, iletişimde gizliliğin de önemi artmıştır. Bu sebeple tarih boyunca güvenli, gizli ve özel iletişimi geliştirmeye yönelik büyük çabalar olmuştur ve hala da olmaya devam etmektedir. Yazı ve iletişimi gizleme kısmında önceleri ilkel yöntemler kullanılırken, matematik bilen insanların devreye dahil oluşuyla birlikte güvenilir iletişim konusundaki akım bir hayli artmıştır. Bu girişimlerin çok hızlı artış göstermesi, yıllar içerisinde, kriptografinin kendi başına bir araştırma alanı olmasını sağlamıştır. Daha sonraları, teknolojinin gelişmesiyle birlikte, matematikçilerin yanı sıra makine ve elektrik mühendislerinin de çok uğraştıkları bir alan haline gelmiştir. Hatta, bazı matematikçilerin çözmek için yıllar harcadığı birçok matematiksel problem, kriptologların gözünde "çözülmemeli" veya "umarız çözülemez" kategorilerine dahil olmuşlardır. Çünkü kriptografide, bir iletişimin gizliliğini sağlayan en temel kısımlar, çözülmesi çok zor olan matematiksel problemlerdir. Bu problemlerin neler olduklarından, yeri geldikçe tez içerisinde bahsedeceğiz.

Tam anlamıyla kriptografi diyebileceğimiz durumlara tarihte ilk defa MÖ 1900'lerde Mısırlıların mezar yazıtlarında rastlıyoruz. Bu yazıtların içerisinde bazen bilinçli olarak gizlenmiş sembolik ifadeler yer alırken bazen o dönemin kendi yazım dili olmasına rağmen,

uzun zamandır kullanılmadığı için artık anlaşılamayan metinler yer alır. Bilinçli olarak gizlenmiş kısımlar, sadece dönemin rahiplerinin anlayabileceği şekilde şifrelenerek yazılmıştır. Kriptografi sayesinde bu yazıtların içerikleri çözülmüştür ve herkesin anlayabileceği şekilde tekrardan yazılabilmiştir.

Bir diğer antik dönem kriptografi örneklerine MÖ 500'lü yıllarda Spartalıların silindir şifreleme çubuklarında (Spartan Scytale Cipher) rastlıyoruz. Silindir bir boru üzerine sarılan bir metnin dikey olarak okunmasına dayanan bir mantıkla şifreleme yapılır ve aynı metni çözmek için yine aynı çapa sahip bir boru kullanmak gerekir.

Matematiksel anlamda ilk şifreleme örneğine ise MÖ 100'lü yıllarda Roma'da rastlıyoruz. Bu şifrelemenin adı, o dönemin imparatoru olan Sezar'dan (Julius Caesar) gelir. Bu şifreleme tekniğini tez içerisinde ayrıntılı olarak anlatacağız.

Kriptografi sadece şifreleme ile değil, aynı zamanda şifrelerin ne derecede güvenli olduğuyla da ilgilenir. Şifrelerin güvenlikleri, şifrelenmiş bir metni çözmek için yapılan çalışmalarla belirlenir. Bu çalışmalara kısaca "kripto analiz" denir ve tarihin ilk kripto analisti, 900'lü yıllarda o dönemin Arap topraklarında yaşamış olan El-Kindi'dir. Frekans analizinin mimarı olarak bilinir. Frekans analizi, şifrelemede kullanılan harflerin sıklığına göre tahmin güçlendirmeye dayanan bir yöntemdir.

Daha sonraları ise, frekans analizine karşı şifreleri güçlendirme çabaları başlamış ve tek alfabe yerine çok alfabe kullanan sistemler geliştirilmiştir. Bunlardan en ünlüsü Vigenere şifrelemesidir. Bu şifreleme sistemine de tez içerisinde ayrıntılı yer vereceğiz. Vigenere şifreleme sistemi ilk kullanılmaya başlandığı zamanlarda "kırılamaz" olarak anılıyordu fakat bu ünvanı çok uzun süre taşıdıktan sonra kırılabileceği anlaşılmıştır.

Kriptografinin modernleşmeye başladığı dönemi 1400'lü yıllar olarak söylemek yanlış olmaz. Çünkü bu yıllarda Leon Battista Alberti adındaki bir İtalyan bilim insanı tarafından o dönemin şartları göz önünde bulundurulduğunda oldukça karmaşık bir yapıya sahip olan bir döner diskli şifreleme cihazı icat etmiştir. Bu sebepten ötürü, bazı kaynaklarda "kriptografinin babası" olarak anılır.

Kriptografinin altın çağı olarak adlandırabileceğimiz dönem ise 20. yüzyıldır. Çünkü bu dönemde artık el ile şifreleme devri son bulurken makinelerin hızı devreye girmiştir. Hatta, şifreleme işi o kadar ileri seviyelere ulaşmıştır ki bu bu yüzyıl içerisinde yapılan savaşlarda, sırf düşman taraftan ele geçirilen şifreli metinleri çözmekle görevli birlikler ordulara dahil edilmişlerdir. Dahası, 2. Dünya Savaşı sırasında Nazi Almanyası tarafından çok güçlü şifreler oluşturmak ve tekrar şifreleri çözmek amacıyla geliştirilen Enigma makinesi savaşın seyrini değiştirmiştir. Enigma'nın icadıyla birlikte kriptografinin ve kriptografik algoritmaların ilerleyişi tamamıyla şekil değiştirmiştir ve artık kriptografide makineler dönemi başlamıştır. Enigma'dan bahsetmişken, Enigma tarafından oluşturulan şifreleri kırabilen bilim insanının bir sözünü vermeden geçmeyelim:

"Makinelerle oluşturulan şifreler ancak makinelerle kırılır."

Alan TURING

Bazı tarihçiler, Enigma şifrelerinin kırılmasının savaşın süresini 2 yıl kadar kısalttığını ve yaklaşık 14 milyon insanın ölümünün engellendiğini dile getirmektedir. 2. Dünya Savaşı sırasında yaşanan bu olaylar, kriptografinin insanlık açısından önemini vurgulayan en belirgin olaylardır.

Daha sonraları, 1970'li yıllarda RSA algoritmasıyla birlikte asimetrik şifreleme dönemi başlamıştır. Asimetrik şifreleme ve RSA algoritmasına tez içerisinde ayrıntılı yer vereceğiz.

Klasik kriptografi anlamında zirveler ise DES ve AES algoritmalarının ortaya atılmasıyla olmuştur. Bu dönemler, artık şifrelemeler ve şifre çözmeler konusunda makinelerin hakimiyetine boyun eğdiğimizin kabullenişini yaşadığımız dönemler olarak adlandırılabilir.

Daha sonrasında ise, överek bitiremediğimiz tüm bu algoritmaların sonunu getireceğini düşündüğümüz kuantum bilgisayarlar devri başlamıştır. Aslında "kuantum bilgisayar devrimi" demek daha doğru olabilir. Çünkü bu bilgisayarların potansiyel işlem hızı düşünüldüğünde klasik şifreleme algoritmalarının hiçbirinin tam anlamıyla gizliliği ve

güvenliği sağlayamayacağını görmek zor değildir. Dolayısıyla artık "kuantum-sonrası kriptografi" dönemi başlamış bulunmakta olup bu konudaki çalışmalar hızla devam etmektedir.

Kuantum-sonrası kriptografik algoritmalar dört temel başlığa ayrılıyor. Bu başlıklar, algoritmaların temel aldığı matematiksel yapılara göre belirlenmişlerdir:

- Kod tabalı algoritmalar
- Kafes tabanlı algoritmalar
- Çok değişkenli polinom tabanlı algoritmalar
- Özet tabanlı algoritmalar

Tezimizde bu başlıklardan ilk iki kategoriye ait olan birkaç algoritmayı inceleyeceğiz. Şimdi geniş hatlarıyla birlikte bir kriptografi yolculuğuna çıkabiliriz...

2. MATEMATİKSEL ALTYAPI

Bu bölümde, algoritmaları anlayabilmek için bilinmesi gereken matematik konularının özetleri verilmiştir.

2.1 Modüler Aritmetik ve Sayılar Teorisi

Kriptografide kullanılan temel kümeler, "sonlu cisimler" adındaki matematiksel yapılardır. (Sonlu cisimler konusunu, tezin ilerleyen kısımlarında ele alacağız.) Sonlu cisimleri oluştururken tam sayılar üzerinde denklik sınıfları kullanılır ve buradaki kurallar modüler aritmetik ile belirlenir.

Bu bölümde modüler aritmetik konusu ve ardından sayılar teorisinin kriptografik anlamda önemli olan kısmı [8] ve [9] numaralı kaynaklardan faydalanılarak anlatılmıştır.

Tanım 2.1.1. $d \neq 0$ olmak üzere $a, d \in \mathbb{Z}$ olsun. Eğer $a = c \cdot d$ olacak şekilde bir $c \in \mathbb{Z}$ tam sayısı bulunabiliyorsa d sayısı a sayısını böler denir ve $d \mid a$ şeklinde gösterilir. Aksi takdirde d sayısı a sayısını bölmez denir ve $d \nmid a$ şeklinde gösterilir.

Örnek 2.1.2. $56 = 7 \cdot 8$ olduğundan $7 \mid 56$ ve $8 \mid 56$'dır.

Not : 0 sayısını hiçbir zaman bir sayının böleni olarak alamayız.

Not : Eğer d sayısı a sayısının bir böleni ise aynı şekilde $-d$ sayısı da a sayısının bir bölenidir.

Önerme 2.1.3. $a, b, c \in \mathbb{Z}$ olmak üzere $a \mid b$ ve $b \mid c$ ise $a \mid c$'dir.

Kanıt. $a \mid b \Rightarrow b = e \cdot a$ olacak şekilde bir $e \in \mathbb{Z}$ vardır ve aynı şekilde $b \mid c \Rightarrow c = f \cdot b$ olacak şekilde bir $f \in \mathbb{Z}$ vardır. Burada $c = f \cdot b = f(e \cdot a) = (f \cdot e)a$ yazılabilir. $f \cdot e \in \mathbb{Z}$ ve $c = (f \cdot e)a$ olduğundan $a \mid c$'dir. □

Uyarı : Bir karışıklığa sebep olmadığı müddetçe $\forall\, a, b \in \mathbb{Z}$ için $a \cdot b$ çarpımını kısaca ab şeklinde göstereceğiz.

Önerme 2.1.4. $a, b, d, x, y \in \mathbb{Z}$ olmak üzere $d \mid a$ ve $d \mid b$ ise $d \mid ax + by$'dir.

Kanıt. $d \mid a$ ve $d \mid b$ ise $a = md$ ve $b = nd$ olacak şekilde $m, n \in \mathbb{Z}$ tam sayıları vardır. Buradan, $ax+by = (md)x+(nd)y = d(mx)+d(ny) = d(mx+ny)$ yazılabilir. Dolayısıyla, $d \mid ax + by$'dir. □

Sonuç 2.1.5. $d \mid a$ ve $d \mid b$ ise $d \mid a + b$ ve $d \mid a - b$'dir.

Tanım 2.1.6. (Asal Sayı) 1 ve kendisi hariç pozitif böleni olamayan 1'den büyük sayılara asal sayı denir. Asal sayılar genellikle p ile ifade edilir.

Örnek 2.1.7. $2, 3, 5, 7, 11, 13, 17, 19, 23, 29$ sayıları ilk on asal sayıdır.

Önerme 2.1.8. 1'den büyük her tam sayı ya asaldır ya da bir asala bölünür.

Kanıt. $1 < n \in \mathbb{Z}$ tam sayısını alalım. Eğer n asalsa ispat biter. Asal olmadığını varsayalım. O halde n sayısı $1 < a_1 < n$ olacak şekilde bir $a_1 \in \mathbb{Z}$ sayısına bölünür. Eğer a_1 sayısı asalsa ispat biter. Asal değilse $1 < a_2 < a_1$ olacak şekilde bir $a_2 \in \mathbb{Z}$ sayısına bölünür. Eğer a_2 sayısı asalsa ispat biter. Değilse benzer şekilde ilerleyerek pozitif tam sayılardan oluşan ve giderek azalan bir $a_1 > a_2 > a_3 > \cdots$ dizisini elde ederiz. Pozitif tam sayılardan oluşan ve sonsuza dek azalarak giden bir diziye sahip olamayacağımıza göre bu dizi bir a_m asal sayısında durur ki aradığımız asal bölen de bu sayıdır. □

Sonuç 2.1.9. 1'den büyük olan bir pozitif tam sayı kendinden önce gelen hiçbir asal sayıya bölünmüyorsa asaldır.

Teorem 2.1.10. Sonsuz sayıda asal vardır.

Kanıt. Tersini varsayalım, yani asal sayılar sonlu sayıda olsun ve bunlara $p_1, p_2, \ldots, p_n$ diyelim. Bu sayıların çarpımına 1 ekleyerek $p_1p_2 \cdots p_n + 1$ sayısını elde ederiz. Buradan kolayca görürüz ki elde ettiğimiz yeni sayı $p_1, p_2, \ldots, p_n$ asallarından hiçbirine bölünmez. O halde $p_1, p_2, \ldots, p_n$ asallarından farklı ve hepsinden daha büyük bir asal sayı elde etmiş oluruz. Bu durumda varsayımımız yanlıştır. Yani asal sayılar sonlu değildir. □

Tanım 2.1.11. (Bileşik Sayı) Asal olmayan sayılara bileşik sayı denir.

Önerme 2.1.12. $2 < n \in \mathbb{Z}$ bir bileşik sayı olmak üzere n'nin $p \leq \sqrt{n}$ olacak şekilde bir p asal böleni vardır.

Kanıt. n bir bileşik sayı olduğundan $1 < a \leq b < n$ olmak üzere $n = ab$ yazabiliriz. Bu durumda $a^2 \leq ab = n$ olur ve böylece $a \leq \sqrt{n}$ elde ederiz. p sayısı a'yı bölen bir asal olsun. O halde $p \leq a \leq \sqrt{n}$ olur ve istenilen p asal sayısını elde etmiş oluruz. □

Sonuç 2.1.13. $n, p \leq \sqrt{n}$ olan asallara bölünmüyorsa n asaldır.

2.1.1 Bölme Algoritması

Teorem 2.1.14. $a, b \in \mathbb{Z}^+$ iki tam sayı olsun. $0 \leq r < b$ olmak üzere $a = bq + r$ olacak şekilde tek türlü belirli q ve r tam sayıları vardır.

Kanıt. Önce, istenilen şekilde q ve r sayılarının var olduğunu gösterelim. q sayısı $\frac{a}{b}$'den küçük ya da eşit olan en büyük tam sayı olsun, yani $q \leq \frac{a}{b} < q + 1$ olsun. Bu eşitsizliği b ile çarparsak $bq \leq a < bq + b$'yi elde ederiz. Eşitsizliğin her tarafından bq sayısını çıkardığımızda $0 \leq a - bq < b$'ye ulaşırız.
$r = a - bq$ olsun. $0 \leq a - bq < b$ olduğundan $0 \leq r < b$'dir ve böylece q ve r sayılarının varlığını ispatlamış oluruz.
Şimdi q ve r sayılarının tek türlü olduklarını gösterelim. Tersini varsayalım, yani hem q_1, r_1 hem de q_2, r_2 sayıları istenilen eşitliği sağlasın. Bu durumda $a = bq_1 + r_1 = bq_2 + r_2$ olur ve buradan $b(q_1 - q_2) = r_2 - r_1$ eşitliğini elde ederiz. Eşitliğin sol tarafı b'nin katı olduğundan sağ tarafı da katı olmalıdır. Hem $b \mid r_2 - r_1$ hem de $0 \leq r_1, r_2 < b$ koşulunu sağlayan tek $r_2 - r_1$ sayısı 0'dır. $r_2 - r_1 = 0$ olduğundan $r_1 = r_2$ elde edilir ve $b(q_1 - q_2) = r_2 - r_1$ eşitliğinde yerine yazılarak $b(q_1 - q_2) = 0$ eşitliğine ulaşılır. $b > 0$ olduğundan $q_1 - q_2 = 0$ olmak zorundadır ve dolayısıyla $q_1 = q_2$ elde edilir. □

Tanım 2.1.15. (En Büyük Ortak Bölen) $0 \neq a, b$ tam sayılarının pozitif ortak bölenleri kümesinin en büyük elemanına a ile b'nin en büyük ortak böleni (ebob'u) denir ve $ebob(a, b)$ ile gösterilir.

Not : Bir karışıklığa yol açmadığı sürece $ebob(a,b)$'yi (a,b) şeklinde kullanacağız.

Tanım 2.1.16. Ebob'u 1 olan sayı çiftlerine aralarında asaldır denir.

Tanım 2.1.17. (Euler φ Fonksiyonu) $p_1, p_2, \ldots, p_k$ asal sayılar ve $e_1, e_2, \ldots, e_k \in \mathbb{N}$ olmak üzere $n = p_1^{e_1} p_2^{e_2} \ldots p_k^{e_k}$ olsun. $\alpha \leq n$ ve $(\alpha, n) = 1$ olacak şekildeki α sayılarının adedini veren fonksiyona Euler φ fonksiyonu denir ve aşağıdaki şekilde tanımlanır.

$$\begin{array}{rcccl} \varphi & : & \mathbb{N} & \to & \mathbb{N} \\ & & n & \mapsto \varphi(n) & = p_1^{e_1}(1-\frac{1}{p_1})p_2^{e_2}(1-\frac{1}{p_2})\cdots p_k^{e_k}(1-\frac{1}{p_k}) \\ & & & & = n(1-\frac{1}{p_1})(1-\frac{1}{p_2})\cdots(1-\frac{1}{p_k}) \end{array}$$

Örnek 2.1.18. 300 sayısından küçük olan ve 300 ile aralarında asal olan sayı adedini bulalım, yani $\varphi(n)$ değerini bulalım.

$$n = 300 = 2^2.3.5^2 \;\Rightarrow\; \varphi(300) = 300(1-\frac{1}{2})(1-\frac{1}{3})(1-\frac{1}{5}) = 80.$$

Not : Sıfırdan farklı her tam sayı 0 sayısını böldüğünden $(0,0)$ tanımsızdır.

Not : $0 \neq a \in \mathbb{Z}^+$ sayısı için $(a,0) = a$'dır.

Önerme 2.1.19. $a, b \in \mathbb{Z}^+$ tam sayıları için $(a,b) = d$ olsun. Bu durumda $(\frac{a}{d}, \frac{b}{d}) = 1$'dir.

Kanıt. $(\frac{a}{d}, \frac{b}{d}) = c$ olsun. Bu durumda $c \mid \frac{a}{d}$ ve $c \mid \frac{b}{d}$'dir. Dolayısıyla $\frac{a}{d} = ck_1$ ve $\frac{b}{d} = ck_2$ eşitliklerini veren k_1, k_2 tam sayılarının var olduğu anlaşılır. Buradan, $a = dck_1$ ve $b = dck_2$ yazabiliriz ve a ile b sayılarının bir ortak böleninin dc sayısı olduğunu elde ederiz. d sayısı en büyük ortak bölen olduğundan $dc \leq d$ olur ve buradan $c = 1$ sonucuna ulaşılırız. Sonuç olarak $(\frac{a}{d}, \frac{b}{d}) = 1$ elde edilir. □

Sonuç 2.1.20. a, b, e sıfırdan farklı pozitif tam sayılar olmak üzere $(ea, eb) = e(a,b)$'dir.

2.1.2 Öklid Algoritması

Öklid algoritması, bölme işlemini kullanarak iki sayının ebobunu bulmamızı sağlayan bir algoritmadır ve işleyişi aşağıdaki gibidir.

Teorem 2.1.21. $b \neq 0$ olmak üzere a, b negatif olmayan tam sayılar olsun. Aralarındaki bölme işlemi şu şekildedir:

$$\begin{aligned}
a &= q_1 b + r_1, & 0 \leq r_1 < b \\
b &= q_2 r_1 + r_2, & 0 \leq r_2 < r_1 \\
r_1 &= q_3 r_2 + r_3, & 0 \leq r_3 < r_2 \\
&\vdots & \vdots \\
r_{n-3} &= q_{n-1} r_{n-2} + r_{n-1}, & 0 \leq r_{n-1} < r_{n-2} \\
r_{n-2} &= q_n r_{n-1} + 0
\end{aligned}$$

0 sayısına ulaşınca algoritma durur ve verilen a, b sayılarının ebob'u sıfırdan önceki son kalandır, yani r_{n-1} sayısıdır.

Kanıt. Algoritmadaki son kalan 0 olduğuna göre $r_{n-1} \mid r_{n-2}$ olur. Sondan bir önceki satırda r_{n-1} sayısı eşitliğin sağ tarafını böldüğünden sol tarafını da böler. Yani $r_{n-1} \mid r_{n-3}$ elde edilir. Benzer şekilde adım adım yukarı çıktığımızda ikinci satırdan $r_{n-1} \mid b$'yi ve birinci satırdan da $r_{n-1} \mid a$'yı elde ederiz. r_{n-1} sayısının hem a hem de b'yi böldüğünü gösterdik. Şimdi de ortak bölenlerinin en büyüğü olduğunu gösterelim. d sayısı a ve b'nin keyfi bir ortak böleni olsun. Algoritmanın ilk satırından $d \mid r_1$ elde edilir. $d \mid b$ ve $d \mid r_1$ olduğundan, ikinci satırdan $d \mid r_2$ bulunur. Bu şekilde adım adım aşağı satırlara inildiğinde en son $d \mid r_{n-1}$ bulunur. Dolayısıyla $d \leq r_{n-1}$'dir. d sayısını keyfi seçtiğimiz için, elde ettiğimiz sonuç tüm ortak bölenler için geçerlidir. Yani a ve b sayılarının r_{n-1}'den daha büyük bir ortak böleni yoktur. □

Örnek 2.1.22. 63 ve 45 sayılarının ebob'unu Öklid algoritması ile bulalım.

$$\begin{aligned} 63 &= 1 \cdot 45 + 18 \\ 45 &= 2 \cdot 18 + 9 \\ 18 &= 2 \cdot 9 + 0 \end{aligned}$$

Sıfırdan önceki son kalan 9 olduğundan $(63, 45) = 9$ olarak bulunur.

Teorem 2.1.23. (Genişletilmiş Öklid Algoritması) a, b en az birisi sıfırdan farklı olan tam sayılar olsun. Bu durumda $(a, b) = ax + by$ olacak şekilde x ve y tam sayıları bulunabilir.

Yani a ve b'nin ebob'u a ile b'nin bir lineer kombinasyonu olarak ifade edilebilir. Bunu yaparken genişletilmiş Öklid algoritması kullanılır. Bu teoremin ispatını yapmak yerine bir örnek üzerinde görmek daha anlaşılır olacaktır.

Örnek 2.1.24. $(456, 123) = 3$'tür. Buna göre, $3 = 456x + 123y$ olacak şekilde x ve y tam sayılarını bulalım. Önce Öklid algoritmasıyla sayılarımızı açalım:

$$\begin{aligned} 456 &= 3 \cdot 123 + 87 \\ 123 &= 1 \cdot 87 + 36 \\ 87 &= 2 \cdot 36 + 15 \\ 36 &= 2 \cdot 15 + 6 \\ 15 &= 2 \cdot 6 + 3 \\ 6 &= 2 \cdot 3 + 0 \end{aligned}$$

İlk satırdan $87 = 456 - 3 \cdot 123$ eşitliğini elde ederiz. İkinci satırdan $36 = 123 - 87$ eşitliğini elde ederiz. Burada 87 yerine ilk satırdan elde ettiğimiz eşitliği yazarsak

$$36 = 123 - (456 - 3 \cdot 123) = 4 \cdot 123 - 456$$

eşitliğine ulaşırız. Üçüncü satırdan $15 = 87 - 2 \cdot 36$ eşitliğini elde ederiz. Burada 87 ve 36 sayılarının yerine üst satırlarda elde ettiklerimizi yazdığımızda

$$15 = (456 - 3 \cdot 123) - 2(4 \cdot 123 - 456) = 3 \cdot 456 - 11.123$$

eşitliğine ulaşırız. Dördüncü satırdan $6 = 36 - 2 \cdot 15$ eşitliğini elde ederiz. Daha önceden bulduğumuz 36 ve 15 sayılarını yerine yazdığımızda

$$6 = (4 \cdot 123 - 456) - 2(3 \cdot 456 - 11 \cdot 123) = 26.123 - 7 \cdot 456$$

eşitliğine ulaşırız. Beşinci satırdan $3 = 15 - 2 \cdot 6$ eşitliğini elde ederiz. Üst satırlardan elde ettiğimiz 15 ve 6 sayılarını yerlerine yazdığımızda

$$3 = (3 \cdot 456 - 11 \cdot 123) - 2(26 \cdot 123 - 7 \cdot 456) = 17 \cdot 456 - 521 \cdot 23$$

eşitliğine ulaşırız. Sonuç olarak $(456, 123) = 3 = 17 \cdot 456 - 63 \cdot 123$ elde edilir, dolayısıyla aradığımız sayılar $x = 17$ ve $y = -63$ olarak bulunur.

Genişletilmiş Öklid Algoritması ve tümevarım kullanarak şu sonuca ulaşabiliriz.

Sonuç 2.1.25. $n \leq 2$ olmak üzere, her $a_1, a_2, \ldots, a_n$ tam sayıları için

$$(a_1, a_2, \ldots, a_n) = k_1 a_1 + k_2 a_2 + \cdots + k_n a_n$$

olacak şekilde $k_1, k_2, \ldots, k_n$ tam sayıları vardır.

Sonuç 2.1.26. a, b, e birer tam sayı olmak üzere eğer e sayısı a ve b'nin bir ortak böleni ise $e \mid (a, b)$'dir.

Sonuç 2.1.27. a, b, c birer tam sayı olmak üzere $(a, c) = (b, c) = 1$ ise $(ab, c) = 1$'dir.

Sonuç 2.1.28. $a \neq 0$ olmak üzere a, b, c birer tam sayı ve $(a, b) = 1$ olsun. Eğer $a \mid bc$ ise $a \mid c$'dir.

Sonuç 2.1.29. $a, b \neq 0$ olmak üzere a, b, c birer tam sayı ve $(a, b) = 1$ olsun. Eğer $a \mid c$ ve $b \mid c$ ise $ab \mid c$'dir.

2.1.3 İkilik Taban

İlkokuldan ve hatta daha öncesinden beri kullandığımız sayıları aslında onluk tabanda kullanırız. Fakat bilgisayarların icadıyla birlikte ortaya çıkan kriptografik algoritmaların neredeyse tamamında sayılar ikilik tabanda kullanılır. Onluk tabanda verilen bir sayıyı ikilik tabana nasıl çevireceğimizi bir örnek üzerinden anlatarak görelim.

Örnek 2.1.30. 147 sayısını ikilik tabana çevirelim. Önce verilen sayının onluk tabandaki açık yazımını görelim.

$$\mathbf{1} \times 100 + \mathbf{4} \times 10 + \mathbf{7} \times 1 = \mathbf{1} \times 10^2 + \mathbf{4} \times 10^1 + \mathbf{7} \times 10^0$$

Şimdi de ikilik tabandaki basamakları nasıl oluşturduğumuzu görelim. Bunun için öncelikle verilen sayının içinde 2'nin en büyük kuvvetini bulmalıyız. $2^7 < 147 < 2^8$ olduğundan 147 sayısının içinde 2'nin en büyük kuvveti 7'dir. Daha sonra $147 - 2^7 = 19$ işlemi ile kalan sayıyı buluruz. Ardından kalan sayının içinde 2'nin en büyük kuvvetini ararız ki bu kuvvet 4'tür. İşlemi devam ettirerek elimizdeki sayı bitene kadar devam ederiz.

$$\begin{aligned} 147 &= 2^7 + 19 \\ 19 &= 2^4 + 3 \\ 3 &= 2^1 + 1 \\ 1 &= 2^0 + 0 \end{aligned}$$

Şimdi basamak değerleriyle birlikte 147 sayısının ikilik tabandaki açık halini yazalım. Bu yazımı yaparken en büyük kuvvetten başlayarak elimizde olan kuvvetlerin önüne 1 çarpanını, elimizde olmayan kuvvetlerin önüne ise 0 çarpanını yazarız.

$$147 = \mathbf{1} \times 2^7 + \mathbf{0} \times 2^6 + \mathbf{0} \times 2^5 + \mathbf{1} \times 2^4 + \mathbf{0} \times 2^3 + \mathbf{0} \times 2^2 + \mathbf{1} \times 2^1 + \mathbf{1} \times 2^0$$

Şimdi sadece katsayıları yazarak verilen sayının ikilik tabandaki halini elde ederiz:

$$1\,0\,0\,1\,0\,0\,1\,1$$

Bu örnekte verdiğimiz yöntem taban değiştirmenin formal yöntemidir. Şimdi aynı örnek üzerinden daha pratik bir yöntem verelim.

Örnek 2.1.31. 147 sayısını ikilik tabana çevirelim. Verilen sayıyı 2'ye böldüğümüzde kalan sayı bize son son basamağı verir. Bölme işleminden elde edilen bölümü tekrar 2'ye böldüğümüzde kalan sayı bize sondan bir önceki basamağı verir. Bu şekilde, elimizdeki sayı bitene kadar işleme devam ederiz ve bölme işlemlerinden kalanları sıralayarak sayıyı oluştururuz.

$$\begin{aligned} 147 &= 2 \cdot 73 + 1 \\ 73 &= 2 \cdot 36 + 1 \\ 36 &= 2 \cdot 18 + 0 \\ 18 &= 2 \cdot 9 + 0 \\ 9 &= 2 \cdot 4 + 1 \\ 4 &= 2 \cdot 2 + 0 \\ 2 &= 2 \cdot 1 + 0 \\ 1 &= 2 \cdot 0 + 1 \end{aligned}$$

Kalanları sağdan sola sıralayarak verilen sayının ikilik tabandaki halini elde ederiz:

$$1\,0\,0\,1\,0\,0\,1\,1$$

2.1.4 Eşlenik Sınıfları

Tanım 2.1.32. (Eşlenik) a, b iki tam sayı olmak üzere aralarındaki fark bir m tam sayısının herhangi bir katı ise a ve b'ye m sayısına göre eşleniktir veya kongrüenttir denir.

$$a \equiv b \pmod{m}$$

şeklinde gösterilir. Buradaki m sayısına eşleniklerin modülü denir ve çoğunlukla pozitif kabul edilir.

Örnek 2.1.33.

$$\begin{array}{lll} 14 \equiv 34 \pmod{20} & 7 \equiv 15 \pmod{8} & 4 \equiv 4 \pmod{11} \\ 5 \equiv 17 \pmod{4} & 9 \equiv 3 \pmod{6} & 6 \equiv 1 \pmod{5} \\ 4 \equiv 4 \pmod{11} & 6 \equiv 1 \pmod{5} & 19 \equiv 6 \pmod{13} \end{array}$$

Örnek 2.1.34. 2 modülüne göre bütün çift sayılar 0'a, bütün tek sayılar da 1'e kongrüenttir.

$$\begin{array}{ll} 20 \equiv 0 \pmod{2} & 13 \equiv 1 \pmod{2} \\ 8 \equiv 0 \pmod{2} & 19 \equiv 1 \pmod{2} \end{array}$$

Tanım 2.1.35. (Eşlenik Sınıfları) Tam sayıları m modülüne göre incelediğimizde birbirinden farklı m tane sınıfa ayrıldıklarını görürüz. Bu sınıflar şunlardır:

$$\begin{array}{rcl} \overline{0} & = & m \text{ ile bölümünden } 0 \text{ kalanlar,} \\ \overline{1} & = & m \text{ ile bölümünden } 1 \text{ kalanlar,} \\ \overline{2} & = & m \text{ ile bölümünden } 2 \text{ kalanlar,} \\ & \vdots & \\ \overline{m-1} & = & m \text{ ile bölümünden } m-1 \text{ kalanlar.} \end{array}$$

Bu sınıflara eşlenik veya kongrüens sınıfları denir. Bu eşlenik sınıflarının hepsini bir küme içerisinde yazmak istediğimizde tam sayılar kümesi notasyonunu ve modülü kullanarak $\mathbb{Z}_m$ şeklinde ifade ederiz. $\mathbb{Z}_m = \{\overline{0}, \overline{1}, \ldots, \overline{m-1}\}$

Tanım 2.1.36. $0 < m$ olmak üzere a, m birer tam sayı olsun. Bu durumda $0 \leq r \leq m - 1$ olacak şekilde $a \equiv r \pmod{m}$ denkliğini sağlayan tek bir r tam sayısı vardır. Bu r tam sayısına a'nın $\pmod{m}$'e göre negatif olmayan en küçük kalanı denir. Genellikle bir sayının modülünü aldığımızda bu en küçük pozitif kalan sayısını kullanırız.

Sonuç 2.1.37. $0 < m$ olmak üzere a, b, c, m birer tam sayı olsunlar. Bu durumda aşağıdakiler sağlanır.

1. $a \equiv a \pmod{m}$
2. $a \equiv b \pmod{m} \Rightarrow b \equiv a \pmod{m}$
3. $a \equiv b \pmod{m}$ ve $b \equiv c \pmod{m} \Rightarrow a \equiv c \pmod{m}$

Sonuç 2.1.38. $0 < m$ olmak üzere a, b, c, d, m birer tam sayı olsunlar. Eğer $a \equiv b \pmod{m}$ ve $c \equiv d \pmod{m}$ ise aşağıdakiler sağlanır.

1. $a + c \equiv b + d \pmod{m}$
2. $a - c \equiv b - d \pmod{m}$
3. $ac \equiv bd \pmod{m}$

Sonuç 2.1.39. $0 < m$ olmak üzere a, b, m birer tam sayı olsunlar. Eğer $a \equiv b \pmod{m}$ ise her n pozitif tam sayısı için $a^n \equiv b^n \pmod{m}$'dir.

Önerme 2.1.40. $0 < m$ olmak üzere a, b, c, m birer tam sayı olsunlar. $ac \equiv bc \pmod{m}$ ve $(c, m) = 1$ ise $a \equiv b \pmod{m}$'dir.

Önerme 2.1.41. $0 < m$ olmak üzere a, b, c, d, m birer tam sayı olsunlar. $ac \equiv bc \pmod{m}$ ve $(c, m) = d$ ise $a \equiv b \pmod{\frac{m}{d}}$'dir.

Sonuç 2.1.42. p bir asal ve x bir tam sayı olsun. Bu durumda $x^2 \equiv 1 \pmod{p}$ denkliğinin çözüm kümesi sadece $x \equiv \pm 1 \pmod{p}$'dir.

2.1.5 Modüler Üs Alma

Küçük sayılarla çalışırken üs almak tabi ki kolaydır fakat kriptografide küçük sayılar kullanımının miadı modern bilgisayarların icadıyla birlikte dolmuştur. Burada büyük sayılarla üs alırken kullanacağımız bir yöntem anlatılmıştır. Bu yöntemi bir örnek üzerinde görelim.

Örnek 2.1.43. $3^{385} \pmod{479}$ sayısını sürekli kare alarak hesaplayabiliriz.
$3^2 \equiv 9 \pmod{479}$ ile başlıyoruz.

$$\begin{aligned}
(3^2)^2 &\equiv 3^4 \equiv 9^2 \equiv 81 \equiv 81 \pmod{479} \\
(3^4)^2 &\equiv 3^8 \equiv 81^2 \equiv 6561 \equiv 334 \pmod{479} \\
(3^8)^2 &\equiv 3^{16} \equiv 334^2 \equiv 111556 \equiv 428 \pmod{479} \\
(3^{16})^2 &\equiv 3^{32} \equiv 428^2 \equiv 183184 \equiv 206 \pmod{479} \\
(3^{32})^2 &\equiv 3^{64} \equiv 206^2 \equiv 42436 \equiv 284 \pmod{479} \\
(3^{64})^2 &\equiv 3^{128} \equiv 284^2 \equiv 80656 \equiv 184 \pmod{479} \\
(3^{128})^2 &\equiv 3^{256} \equiv 184^2 \equiv 33856 \equiv 326 \pmod{479}
\end{aligned}$$

Hesaplamaya çalıştığımız sayının kuvveti $385 = 256+128+1$ olduğundan, oluşturduğumuz liste ile birlikte 3^{385} sayısını $3^{256}\ 3^{128}\ 3^{1}$ çarpımıyla kolayca hesaplayabiliriz.

$$3^{385} \equiv 3^{256+128+1} \equiv 3^{256}\ 3^{128}\ 3^{1} \equiv 326 \cdot 184 \cdot 3 \equiv 179952 \equiv 327 \pmod{479}$$

2.1.6 Doğrusal Eşlenik Denklemleri

Teorem 2.1.44. $0 < m$ ve $a \neq 0$ olmak üzere a, b, x, m birer tam sayı olsunlar. Bu durumda $ax \equiv b \pmod{m}$ eşlenik denkleminin bir çözüme sahip olması için gerek ve yeter koşul $d = (a, m)$ olmak üzere $d \mid b$'dir. Çözüm sayısı tam olarak $d \pmod{m}$ tanedir. Denklemin bir çözümü x_0 olmak üzere tüm çözümleri her $0 \leq k < d$ için $x = x_0 + \frac{m}{d}k$'dır.

Sonuç 2.1.45. $0 < m$ ve $a \neq 0$ olmak üzere a, b, x, m birer tam sayı olsunlar. Eğer $(a, m) = 1$ ise $ax \equiv b \pmod{m}$ denkleminin tam olarak $1 \pmod{m}$ tane çözümü vardır.

Şimdi bu teorem ve sonucun nasıl işlediğini bir örnek üzerinde görelim.

Örnek 2.1.46. $9x \equiv 6 \pmod{15}$ olsun. $(9, 15) = 3$ olduğundan denklemin her tarafını 3'e bölerek işimizi kolaylaştırabiliriz. Bu durumda yeni denklem $3x \equiv 2 \pmod{5}$'tir. x yerine sayı deneyerek ilk çözümü elde edip diğer çözümlere ilk çözümden kolayca ulaşabiliriz. Denklemin bir çözümü açıkça görüldüğü üzere $x_0 = 4$'tür. $(9, 15) = 3$ olduğundan toplamda 3 çözüm vardır. O halde diğer çözümler $x_1 = 4 + \frac{15}{3}1 = 9$ ve $x_2 = 4 + \frac{15}{3}2 = 14$ olarak bulunur.

Tanım 2.1.47. $0 < m$ olmak üzere a, b, m birer tam sayı olsunlar. $ab \equiv 1 \pmod{m}$ ise b sayısına a'nın $\pmod{m}$'e göre çarpımsal tersi denir. Herhangi bir karışıklığa yol açmadığı sürece kısaca tersi diyeceğiz.

Örnek 2.1.48. $3.5 \equiv 1 \pmod{7}$ olduğundan 3 ve 5 sayıları $\pmod{7}$'ye göre birbirinin tersidir. Benzer şekilde, $7 \cdot 7 \equiv 1 \pmod{12}$ olduğundan 7 sayısı $\pmod{7}$'ye göre kendisinin tersidir.

Sonuç 2.1.49. $0 < m$ olmak üzere a, m birer tam sayı olsunlar. Bu durumda a sayısının $(\mathrm{mod}\ m)$'e göre bir tersinin olması için gerek ve yeter şart $(a, m) = 1$ olmasıdır.

Örnek 2.1.50. 13 sayısının $(\mathrm{mod}\ 100)$'e göre tersini bulalım.
$(13, 100) = 1$ olduğundan 13 sayısının $(\mathrm{mod}\ 100)$'e göre tersi olan bir sayı vardır. Aradığımız sayı $13x \equiv 1\ (\mathrm{mod}\ 100)$ denkleminin çözümüdür. Bu denklemi düzenleyerek $1 = 13x - 100y$ şekline getirebiliriz. Buradan, genişletilmiş Öklid algoritmasını kullanarak $1 = 100 \cdot 3 - 13 \cdot 23$ eşitliğini elde ederiz ve $13(-23) = 1 - 100 \cdot 13$ eşitliğine ulaşırız ki bu da $13 \cdot (-23) \equiv 1\ (\mathrm{mod}\ 100)$ anlamına gelir. Yani 13 sayısının $(\mathrm{mod}\ 100)$'e göre tersi -23 sayısıdır ve $-23 \equiv 77\ (\mathrm{mod}\ 100)$ demektir. Sonuç olarak 13'ün $(\mathrm{mod}\ 100)$'e göre tersi 77 sayısıdır.

2.1.7 Çin Kalan Teoremi

Şimdiye kadar, elimizde bir tane eşlenik denklem varken çözümün hangi koşullar altında var olduğunu gördük. Çin kalan teoremi ise bize birden fazla denklem varsa ortak çözümlerinin hangi şartlar altında var olduğunu ve varsa nasıl bulunacağını gösterir. Teoremi vermeden önce deneme yoluyla çözmeye çalışalım.

Örnek 2.1.51. $x \equiv 5\ (\mathrm{mod}\ 7)$ ve $x \equiv 7\ (\mathrm{mod}\ 11)$ denklemlerinin bir ortak çözümünü bulalım.
Bunun için verilen iki denklemi de ayrı ayrı sağlayan değerleri yazalım.

$$x \equiv 5 \pmod 7 \quad \Rightarrow \quad x = \ldots, 5, 12, 19, 26, 33, \mathbf{40}, 47, \ldots, 5 + 7m, \ldots$$
$$x \equiv 7 \pmod{11} \quad \Rightarrow \quad x = \ldots, 7, 18, 29, \mathbf{40}, 51, 62, 73, \ldots, 7 + 11n, \ldots$$

Bulduğumuz değerler arasında 40 sayısının iki denklemi de sağladığını görüyoruz. Bu da bize bir ortak çözüm bulduğumuzu söylüyor. İlk denklemde x değerleri yedişerli ilerlerken ikinci denklemde on birerli ilerlemektedir. Bu da bize her iki denklemde de x'in ortak değerlerinin aralarındaki farkın $7.11 = 77$ olacağını söyler. Bu durumda çözüm kümesini $\{x = 40 + 77k \mid k \in \mathbb{Z}\}$ şeklinde yazabiliriz.

Bu örnekten de anlayacağımız üzere sayılar küçük olmasına rağmen çözüm kümesini deneyerek yazmak zahmetli ve zor bir iştir ki kriptografide kullanılan büyük sayılarla bu çözümü bulmak çok daha zordur. Şimdi de Çin kalan teoremini anlatıp onunla bir örnek çözelim.

Teorem 2.1.52. (Çin Kalan Teoremi) : $m_1, m_2, \ldots, m_r$ ikişerli olarak aralarında asal pozitif tam sayılar olsun. Yani $\forall\ i \neq j$ için $(m_i, m_j) = 1$ olsun. Bu durumda

$$\begin{aligned} x &\equiv a_1 \pmod{m_1} \\ x &\equiv a_2 \pmod{m_2} \\ &\vdots \\ x &\equiv a_r \pmod{m_r} \end{aligned}$$

eşlenik denklem sisteminin $\pmod{m_1 m_2 \ldots m_r}$'ye göre tek bir çözümü vardır.

Kanıt. $m = m_1 m_2 \ldots m_r$ ve $\forall\ i$ için $n_i = \frac{m}{m_i}$ olsun. Yani n_i sayısı m_i hariç tüm m_j'lerin çarpımı olsun. $(n_i, m_i) = 1$ olduğunu iddia ediyoruz. p sayısının (n_i, m_i)'yi bölen bir asal sayı olduğunu varsayalım. Bu durumda $p \mid n_i$'dir. O halde $p \mid m_j$ olacak şekilde en az bir $j \neq i$ vardır. Ayrıca $p \mid m_j$ olduğundan $p \mid (m_j, m_i)$'dir ve bu teoremdeki $(m_j, m_i) = 1$ ifadesiyle çelişir. Bu nedenle $(n_i, m_i) = 1$'dir ve buradan $n_i u_i \equiv 1 \pmod{m_i}$ eşitliğini sağlayan bir u_i sayısının var olduğunu söyleyebiliriz.

$$x = a_1 n_1 u_1 + a_2 n_2 u_2 + \cdots + a_r n_r u_r$$

olsun. $\forall\ j \neq i$ için $m_i \mid n_j$ olduğundan $x = a_1 n_1 u_1 + \cdots + a_r n_r u_r$ eşitliğindeki $a_i n_i u_i$ haricindeki tüm terimler $\pmod{m_i}$'ye göre 0'dır ve böylece $x \equiv a_i n_i u_i \equiv a_i \pmod{m_i}$ olur. Bu eşitlik her i için doğru olduğundan x sayısı bir çözümdür. Şimdi x'in tek çözüm

olduğunu tersini varsayarak gösterelim. x_1 ve x_2 iki çözüm olsun. O halde

$$\begin{array}{rcll} x_1 & \equiv & a_1 & (\mathrm{mod}\ m_1) \\ x_1 & \equiv & a_2 & (\mathrm{mod}\ m_2) \\ & \vdots & & \\ x_1 & \equiv & a_r & (\mathrm{mod}\ m_r) \end{array} \qquad \begin{array}{rcll} x_2 & \equiv & a_1 & (\mathrm{mod}\ m_1) \\ x_2 & \equiv & a_2 & (\mathrm{mod}\ m_2) \\ & \vdots & & \\ x_2 & \equiv & a_r & (\mathrm{mod}\ m_r) \end{array}$$

eşlenik denklem sistemlerini yazabiliriz. $\forall\ 1 \leq i \leq r$ için $m_i \mid (x_1 - x_2)$ ve m_i'lerin ikişerli olarak aralarında asal olduğundan $m_1 m_2 \ldots m_r \mid (x_1 - x_2)$ elde edilir. Buradan

$$x_1 \equiv x_2 \pmod{m_1 m_2 \ldots m_r}$$

sonucuna ulaşılır ki bu da iki çözümün birbirine denk olduğu anlamına gelir. □

Örnek 2.1.53.

$$\begin{aligned} x &\equiv 2 \pmod 3 \\ x &\equiv 3 \pmod 5 \\ x &\equiv 2 \pmod 7 \end{aligned}$$

eşlenik denklem sistemini Çin kalan teoremiyle çözelim. $m = 3.5.7 = 105$, $a_1 = 2$, $a_2 = 3$, $a_3 = 2$ ve

$$\begin{array}{rcccl} n_1 & = & \frac{3\cdot 5\cdot 7}{3} & = & 35 \\ n_2 & = & \frac{3\cdot 5\cdot 7}{5} & = & 21 \\ n_3 & = & \frac{3\cdot 5\cdot 7}{7} & = & 15 \end{array}$$

sayılarına sahibiz.

$$\begin{aligned} 35u_1 &\equiv 1 \pmod 3 \\ 21u_2 &\equiv 1 \pmod 5 \\ 15u_3 &\equiv 1 \pmod 7 \end{aligned}$$

denklemlerini çözüldüğünde $u_1 = 2$, $u_2 = 1$, $u_3 = 1$ değerleri bulunur ve buradan x sayısının

$$\begin{aligned} x &= a_1 n_1 u_1 + a_2 n_2 u_2 + a_3 n_3 u_3 \\ &= 2 \cdot 35 \cdot 2 + 3 \cdot 21 \cdot 1 + 2 \cdot 15 \cdot 1 = 233 \\ &\equiv 23 \pmod{105} \end{aligned}$$

olduğu sonucuna ulaşırız.

2.2 Grup, Halka, Cisim Kavramları

Kriptografide kullanılan temel kümeler genellikle sonlu cisimlerdir. Bu kısımda sonlu cisimlere giden yolu oluşturan grup ve halka konuları da özetlenmiştir. Ayrıca halkalar, polinomlar için temel oluşturma görevini de üstlenmektedir. Bu bölümde anlatılanlar [10] kaynağından yararlanılarak hazırlanmıştır.

2.2.1 Gruplar

Tanım 2.2.1. (Grup) G boş olmayan bir küme ve $*$ da tanım kümesi $G \times G$ olan bir fonksiyon olsun. Eğer G kümesi $*$ fonksiyonu ile birlikte aşağıdaki özellikleri sağlıyorsa $(G, *)$ yapısına bir grup denir.

1. $\forall\, a, b \in G$ için $a * b \in G$'dir. (Kapalılık)
2. $\forall\, a, b, c \in G$ için $a * (b * c) = (a * b) * c$'dir. (Birleşme)
3. $\forall\, a \in G$ için $a * e = e * a = a$ olacak şekilde bir tek $e \in G$ vardır. (Birim Eleman)
4. $\forall\, a \in G$ için bir tek $b \in G$ vardır öyle ki $a * b = b * a = e$'dir. (Ters Eleman)

Tanım 2.2.2. Tanım 2.2.1'de bahsedilen e'ye G'nin birim elemanı ve b'ye de a'nın tersi denir. Genel olarak, a'nın tersi, toplamsal gruplarda $-a$, çarpımsal gruplarda a^{-1} ile gösterilir.

Tanım 2.2.3. (Abel Grup) $(G, *)$ bir grup olmak üzere $\forall a, b \in G$ için $a * b = b * a$ eşitliği sağlanıyorsa $(G, *)$ grubuna abel (değişmeli) grup denir.

Tanım 2.2.4. (Sonlu Grup) $(G, *)$ bir grup olmak üzere G kümesi sonlu sayıda elemana sahip bir küme ise $(G, *)$ grubuna sonlu grup denir.

Teorem 2.2.5. Bir grubun birim elemanı tektir.

Kanıt. Teoremdeki ifadenin tersini varsayalım, yani $(G, *)$ bir grup olmak üzere e ve e' elemanlarının her ikisi birden G'nin birim elemanı olsunlar. e birim eleman olduğundan

$e * e' = e'$ olmalıdır. Aynı şekilde e' birim eleman olduğundan $e * e' = e$ olmalıdır. Bu iki eşitliği birlikte yazarsak $e = e * e' = e'$ eşitliklerini elde ederiz. Buradan $e = e'$ sonucuna ulaşılır. □

Teorem 2.2.6. Bir grupta her elemanın tersi tektir.

Kanıt. $(G, *)$ bir grup ve $g \in G$ bir elaman olsun. Teoremdeki ifadenin tersini varsayalım, yani g elemanının iki tane tersi olsun ve bunlara g_1, g_2 diyelim. Buradan, grubun birim elemanı e olmak üzere, $g * g_1 = e$ ve $g * g_2 = e$ eşitliklerini elde ederiz. Bu eşitlikleri kullanarak

$$g_1 = g_1 * e = g_1 * (g * g_2) = (g_1 * g) * g_2 = e * g_2 = g_2$$

denklem sistemine ulaşırız ve bu da bize $g_1 = g_2$ sonucunu verir. □

Örnek 2.2.7. $\mathbb{Z}$ tam sayılar kümesi, $\mathbb{Q}$ rasyonel sayılar kümesi, $\mathbb{R}$ reel sayılar kümesi ve $\mathbb{C}$ kompleks sayılar kümesi bilinen toplama işlemiyle birlikte birer gruptur. $\mathbb{N}$ doğal sayılar kümesi ise ters eleman özelliğini sağlamadığından bilinen toplama işlemiyle birlikte bir grup değildir.

Örnek 2.2.8. $\mathbb{Q}\backslash\{0\}, \mathbb{R}\backslash\{0\}$ ve $\mathbb{C}\backslash\{0\}$ kümeleri bilinen çarpma işlemiyle birlikte birer gruptur. $\mathbb{Z}$ ve $\mathbb{Z}\backslash\{0\}$ kümeleri ters eleman özelliğini sağlamadığından bilinen çarpma işlemiyle birlikte bir grup değillerdir.

Örnek 2.2.9. $G = \{1, -1\}$ kümesi bilinen çarpma işlemiyle birlikte bir sonlu ve değişmeli gruptur.

Tanım 2.2.10. (Altgrup) $(G, *)$ bir grup olmak üzere $H \subseteq G$ boştan farklı bir altküme olsun. Eğer $*$ işlemiyle birlikte H kümesi de bir grup oluyorsa $(H, *)$ grubuna $(G, *)$ grubunun altgrubu denir ve $(H, *) \leq (G, *)$ ile gösterilir. (Yazım kolaylığı açısından kısaca $H \leq G$ şeklinde de gösterilebilir.)

Örnek 2.2.11. (Aşikar Altgruplar) : Her grup kendisinin bir altgrubudur. Aynı zamanda her grubun sadece birim elemanından oluşan alt kümesi de grubun bir altgrubudur. Bu iki altgruba aşikar altgruplar denir.

Örnek 2.2.12. $n \in N$ olmak üzere $\forall\ n\mathbb{Z}$ kümesi (n'nin tam katları kümesi) bilinen toplama işlemi ile birlikte bir gruptur ve $\forall\ (n\mathbb{Z}, +)$ grubu $(\mathbb{Z}, +)$ grubunun bir altgrubudur. $n\mathbb{Z} \leq \mathbb{Z}$.

Teorem 2.2.13. (Altgrup Olma Kriteri) : $(G, *)$ bir grup olmak üzere $H \subseteq G$ boştan farklı bir altküme olsun. $H \leq G$ olması için gerek ve yeter şart aşağıdaki iki özelliğin sağlanmasıdır.

1. $\forall\ a, b \in H$ için $a * b \in H$
2. $\forall\ a \in H$ için $a^{-1} \in H$

Kanıt. ($\Rightarrow$) : H kümesi G'nin bir altgrubu olduğundan kendi başına da bir gruptur ve böylece, tanım gereği, verilen iki şartı da sağlar.
($\Leftarrow$) : Verilen özelliklerin sağlandığını kabul edelim.
Birinci özellik kapalılığı sağlar.
İkinci özellik ters eleman koşulunu sağlar.
Bir $a \in H$ alalım. İkinci özellik gereği $a^{-1} \in H$'dir. Birinci özellikten, $a \in H$ ve $a^{-1} \in H$ olduğundan $a * a^{-1} = e \in H$ olur. Yani birim eleman koşulu da sağlanır.
G kümesinden alınan her eleman üçlüsü için birleşme özelliği sağlandığından ve $H \subseteq G$ olduğundan H kümesinden alınan elemanlar için de birleşme özelliği sağlanır. □

2.2.2 Halkalar

Tanım 2.2.14. (Halka) R boştan farklı bir küme ve $\star$ ve $\bullet$ da tanım kümeleri $R \times R$ olan fonksiyonlar olsunlar. Eğer R kümesi $\star$ ve $\bullet$ fonksiyonları ile birlikte aşağıdaki özellikleri sağlıyorsa $(R, \star, \bullet)$ yapısına bir halka denir.

1. $(R, \star)$ bir değişmeli (abel) gruptur.
2. $\forall\ a, b, c \in R$ için $a \bullet (b \bullet c) = (a \bullet b) \bullet c$'dir.
3. $\forall\ a, b, c \in R$ için $a \bullet (b \star c) = (a \bullet b) \star (a \bullet c)$ ve $(a \star b) \bullet c = (a \bullet b) \star (b \bullet c)$'dir.

Örnek 2.2.15. $\mathbb{Z}, \mathbb{Q}, \mathbb{R}$ ve $\mathbb{C}$ kümeleri bilinen toplama ve çarpma işlemleriyle birer halkadır.

Tanım 2.2.16. (Değişmeli Halka) $(R, \star, \bullet)$ bir halka olmak üzere $\forall\ a, b \in R$ için $a \bullet b = b \bullet a$ eşitliği sağlanıyorsa $(R, \star, \bullet)$ halkasına değişmeli halka denir.

Tanım 2.2.17. Bir $(R, \star, \bullet)$ halkasının $\star$ işlemine göre (halkadaki 1. işleme göre) birim elemanı 0_R ile, $\bullet$ işlemine göre (halkadaki 2. işleme göre) birim elemanı 1_R ile gösterilir.

Tanım 2.2.18. (Birimli Halka) Bir $(R, \star, \bullet)$ halkası her zaman 1_R elemanına sahip olamayabilir. Eğer bir halkanın 1_R elemanı varsa bu halkaya birimli halka denir.

Örnek 2.2.19. $(\mathbb{Z}, +, \cdot)$ tam sayılar halkası bilinen toplama ve çarpma işlemleriyle birlikte bir birimli halkadır ve birim elemanı $1_{\mathbb{Z}}$=1 tam sayısıdır. Fakat $(2\mathbb{Z}, +, \cdot)$ halkası birimli bir halka değildir.

Örnek 2.2.20. $R = \{f \mid f : \mathbb{R} \to \mathbb{R} \text{ fonksiyon}\}$ kümesi üzerinde $+$ ve $\cdot$ işlemleri aşağıdaki gibi tanımlansın.

$\forall\, f, g \in R$ ve $\forall\, x \in \mathbb{R}$ için,

- $(f + g)(x) = f(x) + g(x)$
- $(f \cdot g)(x) = f(x) \cdot g(x)$

Bu tanımlar ile birlikte $(R, +, \cdot)$ halkası birimli ve değişmeli bir halkadır. Halkanın 0_R elemanı sabit 0 (sıfır) fonksiyonu, 1_R elemanı da sabit 1 fonksiyonudur.

Not : Buradan sonra, bir karışıklığa sebep vermediği sürece ve aksi belirtilmedikçe;

- halkaları $(R, +, \cdot)$ şeklinde ifade edeceğiz ve kısaca R şeklinde, kümenin adıyla,
- $\forall\, a \in R$'nin birinci işleme göre tersini $-a$ ile, ikinci işleme göre tersini de a^{-1} ile,
- 0_R elemanını 0 ile, 1_R elemanını da 1 ile,

göstereceğiz.

Tanım 2.2.21. (Althalka) R bir halka olmak üzere $S \subseteq R$ boştan farklı bir altküme olsun. Eğer $+$ ve $\cdot$ işlemleriyle birlikte S kümesi de bir halka oluşturuyorsa S'ye R'nin bir althalkası denir.

Teorem 2.2.22. (Althalka Olma Kriteri) : R bir halka olmak üzere $S \subseteq R$ boştan farklı bir altküme olsun. S'nin R halkasının bir althalkası olabilmesi için gerek ve yeter koşul aşağıdaki şartları sağlamasıdır.

1. $0 \in S$

2. $\forall\, a, b \in S \Rightarrow a - b \in S$ ve $ab \in S$

Kanıt. $(\Rightarrow)$: S kümesi althalka olsun. Althalka olduğundan halka olma özelliklerini de taşımaktadır ve böylece her iki özelliği de sağlar.

$(\Leftarrow)$: S kümesi her iki özelliği de sağlasın. $0_R \in S$ olduğundan, ikinci özellik gereği $\forall\, b \in S$ için $0 - b = -b \in S$ olur (Ters eleman).

$a, b \in S$ elemanlarını alalım. R kümesinde $a + b = b + a$ olduğundan ve $S \subseteq R$ olduğundan S kümesi de $+$ işlemine göre değişme özelliğini sağlar. Buraya kadar söylediklerimiz S'nin $+$ işlemine göre değişmeli bir grup olduğunu gösteriyor.

$+$ ve $\cdot$ işlemleri birer fonksiyon olduklarından ve R kümesinde birleşme ve dağılma özelliklerini sağladıklarından, işlem olma özelliklerini kaybetmedikleri her altkümede de, yani S kümesinde de sağlarlar.

Böylece S kümesi bir halkadır ve $S \subseteq R$ olduğundan R'nin bir althalkasıdır. □

Tanım 2.2.23. (Sıfır Bölen) R bir halka olsun. Bir $0 \neq a \in R$ elemanını alalım. Eğer $a.b = 0$ olacak şekilde bir $0 \neq b \in R$ elemanı varsa a'ya sıfır bölen denir.

Tanım 2.2.24. (Tamlık Bölgesi) Sıfır böleni olmayan birimli ve değişmeli halkaya tamlık bölgesi denir.

Örnek 2.2.25. $\mathbb{Z}$ tam sayılar, $\mathbb{Q}$ rasyonel sayılar, $\mathbb{R}$ reel sayılar ve $\mathbb{C}$ kompleks sayılar bilinen toplama ve çarpma işlemleriyle birlikte birer tamlık bölgeleridir.

Örnek 2.2.26. $\mathbb{Z}^2 = \mathbb{Z}\times\mathbb{Z}$ kümesi üzerinde $+$ ve $\cdot$ işlemleri aşağıdaki gibi tanımlansın: (a, b) ve $(c, d) \in \mathbb{Z}^2$ olmak üzere,

- $(a, b) + (c, d) = (a + c, b + d)$
- $(a, b) \cdot (c, d) = (ac, bd)$

Bu şekilde tanımlanan işlemlerle birlikte $\mathbb{Z}^2$ kümesi birimli ve değişmeli bir halkadır fakat bir tamlık bölgesi değildir çünkü sıfır bölen bulundurmama koşulunu sağlamaz.

Örneğin, $(0, 1), (1, 0) \in \mathbb{Z}^2$ elemanlarının her ikisi birden sıfırdan farklı olduğu halde $(0, 1) \cdot (1, 0) = (0, 0) = 0_{\mathbb{Z}\times\mathbb{Z}}$'dir.

Örnek 2.2.27. $\mathbb{Z}_m$ bir tamlık bölgesidir $\Leftrightarrow$ m asaldır.

Kanıt. $(\Rightarrow)$: Tersini varsayalım, yani m bir asal sayı olmasın. Bu durumda $a, b < m$ ve $m = ab$ koşullarını sağlayan iki pozitif tam sayı bulunabilir. Bu durumda a ve b pozitif olup

çarpımları m olduğundan ve $m \equiv 0 \pmod{m}$ olduğundan $\mathbb{Z}_m$ bir tamlık bölgesi değildir ve bu kabulümüzle çelişir.
$(\Leftarrow)$: m asal olsun. $a, b \in \mathbb{Z}_m$ alalım. $ab \equiv 0 \pmod{m}$ olsun. O halde $ab = km$ olacak şekilde bir $k \in \mathbb{Z}$ vardır. Bu durumda m asal olduğundan $m \mid a$ veya $m \mid b$ olmalıdır. Dolayısıyla $a \equiv 0 \pmod{m}$ veya $b \equiv 0 \pmod{m}$'dir. □

2.2.3 Cisimler

Cisimler, özellikle sonlu cisimler, kriptografinin olmazsa olmaz bir yapı taşıdır. Konuyla ilgisi olsun olmasın herkes, bilgisayarların 0 ve 1'lerle çalıştığını bir yerlerden mutlaka duymuştur. Bu 0 ve 1'ler aslında sonlu bir cismin elemanlarıdır. Bilgisayarlar, özünde sadece elektrik akımıyla hayat bulduklarından, bu sayılar bilgisayar için " 0 : elektrik yok, 1 : elektrik var " anlamlarını taşımaktadır. Bilgisayarlardaki tüm işlemler bu iki anlamın çeşitli kombinasyonlarıyla oluşturulmaktadır.

Bu bölümde cisimler için genel bilgiler verilmiştir, sonlu cisimlerin nasıl oluşturulduğu, daha sonra anlatılacaktır.

Tanım 2.2.28. (Cisim) F boştan farklı bir küme olmak üzere $(F, +, \cdot)$ bir tamlık bölgesi (Tanım 2.2.24) olsun. Eğer sıfırdan farklı her elemanın "$\cdot$" işlemine (2. işleme) göre tersi varsa $(F, +, \cdot)$ yapısına cisim denir.

Not : Yazım kolaylığı açısından, herhangi bir karışıklığa yol açmadığı sürece cisimleri kısaca kümenin adıyla, F şeklinde göstereceğiz ve cismin işlemlerini sırasıyla $+$ ve $\cdot$ olarak kullanacağız.

Örnek 2.2.29. $\mathbb{Q}, \mathbb{R}, \mathbb{C}$ kümeleri bilinen toplama ve çarpma işlemleriyle birlikte bir cisimdir fakat $\mathbb{Z}$ tam sayılar kümesi ters eleman özelliğini sağlamadığı için bilinen toplama ve çarpma işlemleriyle birlikte bir cisim değildir. 1 ve -1 elemanları haricindeki elemanların tam sayılarda tersi yoktur. Örneğin, $2 \in \mathbb{Z}$ elemanının çarpımsal tersi $\frac{1}{2}$'dir fakat $\frac{1}{2} \notin \mathbb{Z}$'dir.

Tanım 2.2.30. (Altcisim) F bir cisim ve $E \subseteq F$ boştan farklı bir altküme olsun. F cismindeki $+$ ve $\cdot$ işlemleriyle birlikte E kümesi de bir cisim oluyorsa E'ye F'nin altcismi denir.

Örnek 2.2.31. Bilinen toplama ve çarpma işlemleriyle birlikte $\mathbb{Q}$ cismi $\mathbb{R}$ cisminin, $\mathbb{R}$ cismi de $\mathbb{C}$ cisminin bir altcismidir. $\mathbb{Q} \subseteq \mathbb{R} \subseteq \mathbb{C}$.

Teorem 2.2.32. (Altcisim Olma Kriteri) : F bir cisim ve $E \subseteq F$ olsun. E kümesinin F'nin bir altcismi olması için gerek ve yeter şart aşağıdaki iki özelliğin sağlanmasıdır.

1. $\forall\, a, b \in E$ için $a - b \in E$

2. $\forall\, a, b \in E$ için $a \cdot b^{-1} \in E$

Kanıt. $(\Rightarrow)$: E bir altcisim olduğundan tanım gereği verilen özellikleri sağladığı açıktır.
$(\Leftarrow)$: E'nin işlemleri birer fonksiyon olduğundan ve $E \subseteq F$ olduğundan işlemler E'de de iyi tanımlıdır ve birleşme, dağılma gibi özellikleri sağlarlar. Buradan sonra E kümesinde toplamsal ve çarpımsal kapalılığın sağlandığını göstermemiz yeterlidir.
$a, b \in E$ olsun. 1. özellik gereği $a - a = 0 \in E$'dir.
$0, b \in E \Rightarrow 0 - b = (-b) \in E$'dir. $\Rightarrow a - (-b) = a + b \in E$ olur ve bu da E'nin toplamsal kapalı olduğunu gösterir.
$1_F, b \in E$ olduğundan $b \neq 0$ iken 2. özellik gereği $1_F \cdot b^{-1} = b^{-1} \in E$ olur.
$a, b^{-1} \in E$ olduğundan yine 2. özellik gereği $a(b^{-1})^{-1} = ab \in E$ olur ve böylece E kümesi çarpımsal kapalılığı da sağlar. □

Tanım 2.2.33. (Sonlu Cisim) $(F, +, \cdot)$ cismini oluştururken kullanılan F kümesi sonlu sayıda elemana sahip bir küme ise (yani sonlu küme ise) F cismine sonlu cisim denir. Aksi takdirde sonlu olmayan (veya kısaca sonsuz) cisim denir.

Örnek 2.2.34. Bilinen toplama ve çarpma işlemleriyle birlikte $\mathbb{Q}, \mathbb{R}, \mathbb{C}$ kümeleri sonlu olmayan cisimler, $\mathbb{Z}_2, \mathbb{Z}_3, \mathbb{Z}_5$ kümeleri ise sonlu cisimlerdir.

Önerme 2.2.35. F bir cisim olsun. $\forall\, a, b \in F$ elemanları için aşağıdakiler sağlanır.

1. $a \cdot 0 = 0$

2. $-(a \cdot b) = (-a) \cdot b = a \cdot (-b)$

3. $a \cdot b = 0 \Rightarrow a = 0$ veya $b = 0$

Kanıt. **1.** $a \cdot 0 = a \cdot (0+0)$'dır. Dağılma özelliğinden $a \cdot (0+0) = a \cdot 0 + a \cdot 0$'dır. Sonuç olarak $a \cdot 0 = a \cdot 0 + a \cdot 0$ elde edilir. Eşitliğin her iki tarafına $-(a \cdot 0)$ eklenerek $0 = a \cdot 0$ eşitliğine ulaşılır.

2. $0 = 0 \cdot b = (a + (-a)) \cdot b = a \cdot b + (-a) \cdot b$'dir. Bu eşitliğin her iki tarafına $-(a \cdot b)$ eklenerek $-(a \cdot b) = (-a) \cdot b$ sonucuna ulaşılır. Aynı işlemler uygulanarak eşitlik $a \cdot (-b)$ için de sağlanır.

3. $a \neq 0$ olsun. O halde $0 = a^{-1} \cdot 0 = a^{-1} \cdot (a \cdot b) = (a^{-1} \cdot a) \cdot b = 1 \cdot b = b$ eşitliği yazılabilir ve buradan $0 = b$ elde edilir. □

Tanım 2.2.36. (Karakteristik) F bir cisim olmak üzere $n \cdot 1_F = 0$ olacak şekilde bir $n \in \mathbb{Z}^+$ pozitif tam sayıları varsa bu n sayılarının en küçüğüne F cisminin karakteristiği denir. Böyle bir n sayısı yoksa F cisminin karakteristiği sıfır olarak kabul edilir. Bir F cisminin karakteristiği $kar(F)$ ile gösterilir.

Örnek 2.2.37. Bilinen toplama ve çarpma işlemleriyle birlikte $\mathbb{Q}, \mathbb{R}, \mathbb{C}$ cisimlerinin karakteristiği sıfırdır. p bir asal sayı olmak üzere, bilinen toplama ve çarpma işlemleriyle birlikte $\mathbb{Z}_p$ karakteristiği p olan bir cisimdir, çünkü her $a \in \mathbb{Z}_p^* = \{1, 2, \ldots, p-1\}$ için $a \cdot \mathbb{Z}_p^* = \{a, 2a, \ldots, (p-1)a\} = \mathbb{Z}_p^*$ olduğundan $ab = 1$ olacak şekilde bir $b \in \mathbb{Z}_p^*$ vardır. Karakteristik $kar(\mathbb{Z}_p) = p$'dir, çünkü $n \cdot 1 \equiv 0 \pmod{p}$ ise $p \mid n$ olur.

Teorem 2.2.38. F bir cisim olmak üzere $kar(F)$ ya sıfırdır ya da bir asal sayıdır.

Kanıt. $kar(F) = 0$ ise ispat biter. $kar(F) = k \neq 0$ olduğunu ve asal olmadığını kabul edelim. k asal olmadığından $k = a \cdot b$ olacak şekilde $1 < a, b < k$ tam sayıları vardır. (Burada $1 \cdot 1 = 1 \neq 0$ olduğundan $k = 1$ olamaz.) $x = a \cdot 1_F$ ve $y = b \cdot 1_F$ olsun.
$\Rightarrow x \cdot y = (a \cdot 1_F) \cdot (b \cdot 1_F) = (a \cdot b) \cdot 1_F = k \cdot 1_F = 0$ elde edilir.
$x \cdot y = 0$ ise $x = 0$ veya $y = 0$'dır. $x = 0$ ise $a \cdot 1_F = 0$ olur, $y = 0$ ise $b \cdot 1_F = 0$ olur ki $x, y < k$ olduğundan bu durum k sayısının $k \cdot 1_F = 0$ eşitliğini sağlayan en küçük pozitif tam sayı olması ile çelişir. O halde k sayısı asal olmak zorundadır. □

2.3 Polinom Halkaları

Polinomlar, başta eliptik eğri şifreleme algoritması olmak üzere kriptografinin birçok yerinde kullanılan matematiksel yapılardır. Bu bölümde polinom halkalarının geniş bir özeti anlatılmıştır. Bu özet [11] kaynağından yararlanılarak oluşturulmuştur.

Tanım 2.3.1. (Polinom) R bir halka, $a_0, a_1, a_2, \ldots, a_n \in R$, x bir değişken ve $n \in \mathbb{N}$ olmak üzere

$$a_o + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

yapısına katsayılarını R'den alan ve x değişkenine bağlı olan bir polinom denir. Buradan da anlaşılacağı üzere $a_0, a_1, \ldots, a_n \in R$ elemanlarına polinomun katsayıları denir.

Tanım 2.3.2. (Polinom Halkası) Katsayılarını R halkasından alan ve x değişkenine bağlı olan tüm polinomların kümesi $R[x]$ ile gösterilir ve $R[x]$'e bir polinom halkası denir. Çoğu zaman, polinomlar $p(x), q(x), f(x), g(x)$ gibi harflerle ifade edilir.

Tanım 2.3.3. $a_0, a_1, \ldots, a_n, b_0, b_1, \ldots, b_k \in R$ olmak üzere $p(x) = a_0 + a_1 x + \cdots + a_n x^n$ ve $q(x) = b_0 + b_1 x + \cdots + b_k x^k$ polinomları $R[x]$ halkasından alınan iki polinom olsun. Bu durumda iki polinomun eşitliği

$$p(x) = q(x) \Leftrightarrow \forall\, i \geq 0,\ a_i = b_i$$

önermesi ile ifade edilir.

Tanım 2.3.4. Bütün katsayıları sıfır olan polinoma sıfır polinomu, x'li terimlerinin tamamının katsayılarının sıfır olduğu polinoma da sabit polinom denir. Yani, R halkasının her bir elemanı tek başına bir sabit polinom belirtirler.

Not : Bir polinomda katsayısı sıfır olan terimler ihtiyaç duyulmadıkça yazılmaz ki çoğunlukla da ihtiyaç duyulmaz.

Örnek 2.3.5. $p(x) = 5 + 6x^2 + 13x^5 \in \mathbb{Z}[x]$ polinomu, katsayılarını $\mathbb{Z}$ halkasından alan ve x değişkenine bağlı olan bir polinomdur.

Tanım 2.3.6. (Derece) Bir polinomda en büyük kuvvete (üsse) sahip terimin katsayısına polinomun başkatsayısı, en büyük kuvvete de polinomun derecesi denir. Bunu matematiksel olarak ifade edecek olursak $p(x) = a_0 + a_1x + \cdots + a_nx^n \in R[x]$ bir polinom olmak üzere $a_n \neq 0$ ise $a_n \in R$'ye $p(x)$ polinomunun başkatsayısı ve $n \in \mathbb{N}$ sayısına da derecesi denir. Bir polinomun derecesi, polinomu ifade eden harf ile birlikte $der(p(x))$ şeklinde veya kısaca $der(p)$ ile gösterilir.

Not : Sıfır polinomu haricinde tüm sabit polinomların derecesi sıfırdır, sıfır polinomunun derecesi ise $-\infty$ olarak kabul edilir.

2.3.1 Polinomlarda İşlemler

Tanım 2.3.7. (Toplama ve Çıkarma) $p(x) = a_0 + a_1x + \cdots + a_nx^n$ ve $q(x) = b_0 + b_1x + \cdots + b_mx^m \in R[x]$ iki polinom olsun. $\forall\, i \geq 0$ için $c_i = a_i \pm b_i$ ve $k = maks\,\{n, m\}$ olmak üzere bu iki polinomun toplamı

$$p(x) \pm q(x) = c_0 + c_1x + c_2x^2 + \cdots + c_kx^k$$

şeklinde hesaplanır.

Tanım 2.3.8. (Çarpma) $p(x) = a_0 + a_1x + \cdots + a_nx^n$ ve $q(x) = b_0 + b_1x + \cdots + b_mx^m$, $R[x]$'deki iki polinom olsun. $\forall\, i \geq 0$ için $c_i = a_ib_0 + a_{i-1}b_1 + a_{i-2}b_2 + \cdots + a_1b_{i-1} + a_0b_i$ ve $k = n + m$ olmak üzere bu iki polinomun çarpımı

$$p(x) \cdot q(x) = c_0 + c_1x + c_2x^2 + \cdots + c_kx^k$$

şeklinde hesaplanır.

Örnek 2.3.9. $p(x) = 3+7x+2x^3$ ve $q(x) = 4x+x^2 \in \mathbb{Z}[x]$ iki polinom olsun. Bu durumda

$$p(x) + q(x) = 3 + 11x + x^2 + 2x^3$$

$$p(x).q(x) = 12x + 31x^2 + 9x^3 + 8x^4 + 2x^5$$

olarak bulunur.

Önerme 2.3.10. R bir halka olmak üzere

1. R birimli ise $R[x]$ halkası da birimlidir.
2. R değişmeli ise $R[x]$ halkası da değişmelidir.
3. R bir tamlık bölgesi ise $R[x]$ halkası da bir tamlık bölgesidir.

Kanıt. **1.** Bir $p(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n \in R[x]$ polinomunu alalım ve R'nin birimi 1_R olsun.

$$1_R \cdot p(x) = 1_R.a_0 + 1_R \cdot a_1x + \cdots + 1_R \cdot a_nx^n = a_0 + a_1x + \cdots + a_nx^n = p(x)$$

olduğundan $R[x]$ halkasının da birimi 1_R sabit polinomudur.

2. $p(x), q(x) \in R[x]$ polinomlarını alalım. $\forall\ a, b \in R$ için $a \cdot b = b \cdot a$ olduğundan ve polinomlardaki çarpma işlemi de katsayıların çarpılması üzerinden hesaplandığından $p(x) \cdot q(x) = q(x) \cdot p(x)$ olur ve buradan $R[x]$ halkasının değişmeli olduğunu söyleriz.

3. R bir tamlık bölgesi ise birimli ve değişmelidir. Böylece $R[x]$ halkası da birimli ve değişmeli olur. Bu durumda tek yapmamız gereken $R[x]$ halkasının sıfır bölensiz olduğunu göstermektir. $a_n, b_n \neq 0$ olmak üzere $p(x) = a_0 + a_1x + \cdots + a_nx^n$ ve $q(x) = b_0 + b_1x + \cdots + b_mx^m \in R[x]$ polinomlarını alalım. Yani $p(x), q(x) \neq 0$ olsunlar. Polinom çarpması tanımından $c_{n+m} = a_{n+m}b_0 + \cdots + a_nb_m + \cdots + a_0b_{n+m}$'dir. $\forall\ i > n$ için $a_i = 0$ ve $\forall\ j > m$ için $b_j = 0$ olduğundan $c_{n+m} = a_nb_m$ olarak bulunur. R bir tamlık bölgesi olduğundan ve $a_n, b_m \neq 0$ olarak kabul ettiğimizden $a_nb_m \neq 0$ olur ve buradan $c_{n+m} \neq 0$ olduğu sonucuna ulaşırız. Bu durumda $R[x]$ halkasından alınan sıfırdan farklı polinomların çarpımının sıfır olamayacağını elde ederiz ki bu da $R[x]$ halkasının bir tamlık bölgesi olduğunu söyler. □

Şimdi doğrudan bu önermeden çıkan iki sonucu verelim:

Sonuç 2.3.11. F bir cisim ise $F[x]$ polinom halkası bir tamlık bölgesidir.

Sonuç 2.3.12. R bir tamlık bölgesi ise bu durumda $\forall\ p(x), q(x) \in R[x]$ polinomu için $der(p(x) \cdot q(x)) = der(p(x)) + der(q(x))$'dir.

Tanım 2.3.13. $p(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n \in R[x]$ bir polinom ve $r \in R$ halkanın bir elemanı olsun. $p(r) = a_0 + a_1r + a_2r^2 + \cdots + a_nr^n \in R$'ye p polinomunun r'deki değeri denir.

2.3.2 Polinomlarda Bölme Algoritması

Teorem 2.3.14. R değişmeli bir halka ve $f(x), g(x) \in R[x]$ iki polinom olsun. $g(x) \neq 0$ ve $g(x)$'in başkatsayısı tersinir olsun. Bu durumda

$$f(x) = q(x).g(x) + r(x)\ ve\ der(r) < der(g)$$

olacak şekilde tek türlü belirli $q(x), r(x) \in R[x]$ polinomları vardır.

Kanıt. $f(x) = a_nx^n + a_{n-1}x^{n-1} + \cdots + a_1x + a_0,\ der(f) = n$ ve $b^{-1} \in R$ olmak üzere $g(x) = b_mx^m + b_{m-1}x^{m-1} + \cdots + b_1x + b_0,\ der(g) = m$ olsun. Önce, bu q ve r polinomlarının varlığını ispatlayalım. Bu ispatı n üzerinden tümevarımla yapacağız.
$n = 0$ olsun. $0 = der(f) < der(g)$ ise $q = 0$ ve $r = f$ alınabilir.
$0 = der(f) = der(g)$ ise $r = 0$ ve $q = a_0b_0^{-1}$ alınabilir.
Tümevarım hipotezi olarak derecesi n'den küçük olan polinomlar için teoremdeki ifadenin doğru olduğunu kabul edelim. Genelliği bozmadan, $der(g) \leq der(f)$ alabiliriz çünkü aksi durumda $q = 0$ ve $r = f$ alınarak istenilen ispat edilmiş olur.
$f_1(x) = f(x) - a_nb_m^{-1}x^{n-m}g(x)$ diyelim. $der(f_1) < n$ olduğundan, tümevarım hipotezine göre $der(r) < der(g)$ ve $f_1(x) = q_1(x)g(x) + r(x)$ olacak şekilde $q_1, r \in R[x]$ vardır.

$$\begin{aligned} f(x) &= a_nb_m^{-1}x^{n-m}g(x) + q_1(x)g(x) + r(x) \\ &= [a_nb_m^{-1}x^{n-m} + q_1(x)]g(x) + r(x),\ der(r) < der(g) \end{aligned}$$

olacak şekilde $q(x) = a_n b_m^{-1} x^{n-m} + q_1(x)$ ve $r(x) \in R[x]$ bulunmuş olur.
Şimdi de bulduğumuz bu $q, r \in R[x]$ polinomlarının tek türlü olduklarını ispatlayalım.
$f(x) = q_1(x)g(x)+r_1(x) = q_2(x)g(x)+r_2(x),\ der(r_1) < der(g),\ der(r_2) < der(g)$ olsun.
$(q_1(x) - q_2(x) = r_2(x) - r_1(x)$ eşitliğinden ve $g(x)$'in başkatsayısının tersinir oluşundan derece karşılaştırması yapılırsa

$$der(r_2 - r_1) = der(q_1 - q_2) = der(q_1 - q_2) + der(g)$$

elde edilir. $der(r_2 - r_1) < der(g)$ olduğundan, son eşitlik ancak $q_1(x) = q_2(x)$ olması ile sağlanır. Dolayısıyla da $r_1(x) = r_2(x)$ elde edilir. □

Sonuç 2.3.15. F bir cisim olmak üzere $\forall\ f, g \in F[x]$ polinomlarına $g \neq 0$ iken bölme algoritması uygulanabilir.

Sonuç 2.3.16. $f \in R[x]$ ve $\forall\ a \in R$ olmak üzere $f(x) = (x - a)q(x) + f(a)$ olacak şekilde tek türlü belirli bir $q(x) \in R[x]$ polinomu vardır.

Kanıt. $f(x)$ ve $(x - a)$ polinomlarına bölme algoritması uygulandığında

$$f(x) = (x - a)q(x) + r(x)\ ve\ der(r) < der(x - a)$$

olacak şekilde tek türlü belirli $q, r \in R[x]$ polinomlarının bulunabildiğini görürüz. Ayrıca $der(r) < der(x - a)$ ve $der(x - a) = 1$ olduğundan $der(r) = 0$ olarak bulunur. Yani $r(x) = r \in R$ bir sabittir. Yukarıdaki eşitlikte $x = a$ alınırsa $f(a) = r(a) = r$ bulunur ve buradan $q(x)$'in tek türlü belirli olduğu elde edilir. □

Tanım 2.3.17. (İndirgenemez Polinom, Asal Polinom) $f \in R[x]$ bir polinom olmak üzere, Eğer f polinomunu derecesi en az bir olan birden fazla polinomun çarpımı şeklinde yazamıyorsak, f polinomuna indirgenemez polinom denir. Eğer f polinomu, başkatsayısı 1 olan bir indirgenemez polinom ise f'ye asal polinom denir.

Tanım 2.3.18. (Primitif Polinom) Eğer bir $p(x) \in R[x]$ bir indirgenemez polinom ve katsayıları R cismi üzerinde ikişerli olarak aralarında asal (Tanım 2.1.16) ise $p(x)$ polinomuna primitif (veya ilkel) polinom denir.

Önerme 2.3.19. Bir $\mathbb{F}_q$ cismi üzerinde n dereceli $\frac{\varphi(q^n-1)}{n}$ (Tanım 2.1.17) tane primitif polinom vardır.

Örnek 2.3.20. $p(x) = x^{13} + x^4 + x^3 + x + 1$ polinomu $\mathbb{F}_2$ cismi üzerinde, derecesi 13 olan bir primitif polinomdur.

2.4 Sonlu Cisimler

Daha önce de belirttiğimiz gibi, sonlu cisimler, kriptografinin olmazsa olmaz yapı taşlarıdır. Ayrıca sonlu cisimler konusu matematiksel anlamda çok derin bilgiler içeren bir konudur fakat kriptografi açısından, başlangıç seviyelerinde nasıl inşa edildiklerini bilmek yeterlidir. Bu kısımda sonlu cisimlerin nasıl inşa edildiği olabildiğince kısa ve pratik bir yolla anlatılmıştır ve bu anlatım yapılırken [12] ve [13] numaralı kaynaklardan faydalanılmıştır.

Sonlu cisimler, esasında bölüm halkalarının birer özel halleridirler. Bu sebeple, inşaya geçmeden önce bölüm halkalarını tanımak gerekir.

2.4.1 Bölüm Halkaları

Tanım 2.4.1. (**İdeal**) R bir halka ve $I \subseteq R$ boştan farklı bir altküme olsun. Aşağıdaki şartlar sağlanıyorsa I kümesine R halkasının bir ideali denir.

1. $\forall\, x, y \in I$ için $x - y \in I$
2. $\forall\, r \in R$ ve $\forall\, x \in I$ için $rx, xr \in I$

Örnek 2.4.2. $\{0\}$ ve R kümeleri R halkasının birer idealidirler.

Teorem 2.4.3. R bir birimli halka ve $I \subseteq R$ bir ideal olsun. Eğer $1 \in I$ ise $I = R$'dir.

Kanıt. $\forall\, r \in R$ elemanını $r = 1 \cdot r$ şeklinde yazabiliriz. Bu sebeple $1 \in I$ olduğundan ve $I \subseteq R$ bir ideal olduğundan $\forall\, r \in R$ için $r = 1 \cdot r \in I$ olur ve böylece $I = R$ eşitliğine ulaşırız. □

Tanım 2.4.4. R bir halka olsun. $c_1, \ldots, c_n \in R$ alalım.

$$\langle c_1, \ldots, c_n \rangle = \{r_1 c_1 + \cdots + r_n c_n \mid r_1, \ldots, r_n \in R\}$$

kümesi R halkasının bir idealidir ve bu ideale $c_1, \ldots, c_n$ tarafından üretilen ideal denir.

Tanım 2.4.5. R bir halka ve $I \subseteq R$ bir ideal olsun. Eğer I ideali R halkasının tek bir elemanı tarafından üretiliyorsa I idealine tek üreteçli ideal denir.

Tanım 2.4.6. (**Temel İdeal Bölgesi**) Bir R halkasının her ideali tek bir elemanla üretilebiliyorsa R halkasına temel ideal bölgesi denir.

Tanım 2.4.7. (**Bölüm Halkası**) R bir halka ve $I \subseteq R$ bir ideal olsun. $\forall\, x, y \in R$ için

$$x \equiv y \Leftrightarrow x - y \in I$$

şeklinde tanımlanan bağıntı R halkası üzerinde bir denklik bağıntısıdır. Bu denklik bağıntısı kullanılarak oluşturulan eşlenik sınıflarını (Tanım 2.1.35) $\forall\, r \in R$ için

$$\overline{r} = \{r + i \mid i \in I\}$$

şeklinde gösterelim. Bu kümeyi de kısaca $r + I$ şeklinde yazalım. R üzerinden oluşturulmuş tüm eşlenik sınıflarının kümesi R/I şeklinde gösterilir.

$$R/I = \{r + I \mid r \in R\}$$

Bu küme aşağıda tanımlanan işlemler ile birlikte bir halka olur.

$$\begin{array}{rccl} + & : & R/I \times R/I & \longrightarrow & R/I \\ & & (x+I),(y+I) & \longmapsto & (x+I)+(y+I) = (x+y)+I \end{array}$$

$$\begin{array}{rccl} \cdot & : & R/I \times R/I & \longrightarrow & R/I \\ & & (x+I),(y+I) & \longmapsto & (x+I)\cdot(y+I) = (x\cdot y)+I \end{array}$$

Bu işlemlerle birlikte oluşan $({}^{R}\!/_{I}, +, \cdot)$ halkasına bölüm halkası denir. Bu halkanın etkisiz elemanı $0 + I = I$, birim elemanı ise $1 + I$'dır. $\forall\ r + I \in {}^{R}\!/_{I}$ için aşağıdaki eşitlikler sağlanır:

$$(r + I) + (0 + I) = (r + 0) + I = r + I$$

$$(r + I) \cdot (1 + I) = (r \cdot 1) + I = r + I$$

Örnek 2.4.8. $2\mathbb{Z} \subseteq \mathbb{Z}$ ve $5\mathbb{Z} \subseteq \mathbb{Z}$ altkümeleri, tam sayılar halkasının birer idealleridir ve bu idealler kullanılarak oluşturulan bölüm halkaları aşağıdaki şekildedir:

$${}^{\mathbb{Z}}\!/_{2\mathbb{Z}} = \{0 + 2\mathbb{Z},\ \ 1 + 2\mathbb{Z}\} = \{2\mathbb{Z},\ \ 1 + 2\mathbb{Z}\}$$

$${}^{\mathbb{Z}}\!/_{5\mathbb{Z}} = \{5\mathbb{Z},\ \ 1 + 5\mathbb{Z},\ \ 2 + 5\mathbb{Z},\ \ 3 + 5\mathbb{Z},\ \ 4 + 5\mathbb{Z}\}$$

Katsayılarını bir cisimden alan polinomlar kümesi, polinomlardaki bilinen toplama (2.3.7) ve çarpma (2.3.8) işlemleriyle birlikte bir halka yapısına sahip olurlar. Doğal olarak, bu halka yapıları üzerinde de bölüm grupları oluşturabiliriz. Bunun nasıl yapıldığını bir örnek üzerinden görelim:

Örnek 2.4.9. $R = \mathbb{Z}_3[x]$ halkasını ve bu halka içerisinden x^2+2x+1 polinomunu alalım. $I = \langle x^2+2x+1 \rangle$ olsun. Burada I kümesinin, R halkasının x^2+2x+1 elemanı tarafından üretilen ideali olduğunu görmek zor değildir. Polinomlarda bölme algoritmasını (2.3.2) kullanarak R halkasındaki tüm polinomları x^2+2x+1 polinomuna böldüğümüzde, bu bölümler sonucunda elde edilen kalanlar, derecesi 2'den küçük olan ve katsayılarını $\mathbb{Z}_3$ cisminden alan polinomlar olarak bulunur. Yani ${}^{R}\!/_{I}$ bölüm halkasını aşağıdaki şekilde yazabiliriz:

$${}^{R}\!/_{I} = {}^{\mathbb{Z}_3[x]}\!/_{\langle x^2 + 2x + 1 \rangle} = \{c_1 x + c_0 \mid c_1, c_0 \in \mathbb{Z}_3\}$$

Bu örnekte elde edilen bölüm halkası bir cisim değildir, çünkü $(x+1) + I \in {}^{R}\!/_{I}$ elemanı sıfırdan farklı olduğu halde bu elemanı kendisiyle çarptığımızda

$$((x + 1) + I) \cdot ((x + 1) + I) = ((x + 1) \cdot (x + 1)) + I = (x + 1)^2 + I = (x^2 + 2x + 1) + I$$

sonucunu elde ederiz ki bu sonuç $0 + I$'ya eşittir. Sıfırdan farklı iki elemanı çarparak sıfırı elde ettiğimiz için bu bölüm halkası bir cisim yapısı oluşturamaz.

Bir bölüm halkasının hangi koşullar altında bir cisim olabileceğini aşağıdaki teoremle söyleyebiliriz:

Teorem 2.4.10. F bir cisim ve $F[x]$ de katsayılarını F cisminden alan tüm polinomların kümesi olsun. Bir $p(x) \in F[x]$ elemanını alalım. ($I = \langle p(x) \rangle$ kümesinin $F[x]$ halkasının bir ideali olduğu aşikardır.) Bu durumda aşağıdaki önerme doğrudur.

$F[x]/\langle p(x) \rangle$ bölüm halkası bir cisimdir $\Leftrightarrow$ $p(x)$ polinomu F üzerinde indirgenemezdir

Kanıt. $(\Rightarrow)$: Tersini varsayalım, yani $p(x)$ polinomu indirgenemez olmasın. Bu durumda $q(x) \cdot r(x) = p(x)$ olacak şekilde $p(x), q(x) \in F[x]$ polinomları vardır ve aynı zamanda $q(x) + I, r(x) + I \in F[x]/I$'dır. O halde, bu iki polinomu çarptığımızda

$$(q(x) + I) \cdot ((r(x) + I)) = ((q(x) \cdot r(x)) + I = p(x) + I = I$$

sonucu elde edilir ki bu da $F[x]/I$ bölüm halkasının bir cisim oluşuyla çelişir. Dolayısıyla varsayımımız yanlıştır, yani $p(x)$ polinomu indirgenemezdir.
$(\Leftarrow)$: F bir cisim olduğundan $F[x]/I$ bölüm halkasının birimli ve değişmeli bir halka olduğu aşikardır. Dolayısıyla bu bölüm halkasının bir cisim olduğunu göstermek için her elemanın tersinin var olduğunu göstermek yeterlidir. Sıfırdan farklı bir $q(x) + I \in F[x]/I$ alalım. $der(q) < der(p)$ olduğundan ve $p(x)$ indirgenemez olduğundan $(q(x), p(x)) = 1$'dir. Bu durumda $a(x)q(x) + b(x)p(x) = 1$ olacak şekilde $a(x), b(x) \in F[x]$ polinomları vardır. Buradan $a(x)q(x) = 1 - b(x)p(x)$ eşitliği elde edilir.

$$a(x)q(x) = 1 - b(x)p(x) \equiv (1 - b(x)p(x)) + I \equiv 1 + I$$

olduğundan $a(x)q(x) \equiv 1 + I$ olur. Dolayısıyla $q(x) + I$ polinomunun $F[x]/I$ bölüm halkasındaki tersi $a(x) + I$ olarak bulunur. $\square$

Bir bölüm halkasının hangi koşul altında bir cisim olduğunu gördüğümüze göre artık bir sonlu cismin nasıl inşa edildiğine geçebiliriz.

2.4.2 Sonlu Cisimlerin İnşası

Eğer bir cismin eleman sayısı sonlu ise bu cisme sonlu cisim (Tanım 2.2.33) dendiğini ve sonlu bir cismin eleman sayısının ya bir asal sayı ya da bir asal sayının kuvveti olduğunu daha önce belirtmiştik. Eleman sayısı asal olan cisimler, $\mathbb{Z}_q$ sonlu halkalarıdır (Burada q bir asaldır). Biz bu kısımda eleman sayısı bir asalın 1'den farklı bir kuvveti olan cisimlerin nasıl inşa edildiğini göreceğiz.

q bir asal sayı olmak üzere $\mathbb{Z}_q$ kümesi q elemanlı bir cisimdir. Aynı eleman sayısına sahip cisimler birbirlerine izomorf olduğundan, genel bir yazım olması amacıyla biz bu cismi $\mathbb{F}_q$ ile gösterelim. m, 1'den farklı bir pozitif sayı olmak üzere $\mathbb{F}_{q^m}$ cismini inşa edelim.

İlk adım olarak $\mathbb{F}_q[x]$ polinom halkasından derecesi m olan indirgenemez bir polinom seçilir. Seçilen polinom $p(x)$ olsun. Ardından bu polinom kullanılarak $\mathbb{F}_q[x]/\langle p(x)\rangle$ bölüm halkası oluşturulur:

$$\mathbb{F}_q[x]/\langle p(x)\rangle = \{c_0 + c_1x + \cdots + c_{m-1}x^{m-1} \mid c_0, c_1, \ldots, c_{m-1} \in \mathbb{F}_q\}$$

Bu kümedeki her bir polinom cismin bir elemanıdır. Bu elemanları polinomlar şeklinde kullanmak büyük bir işlem yükü gerektirir. Bu sebeple, polinomlar yerine her bir polinomun bir sembolle gösterildiği bir yol izleyeceğiz. Şimdi bunu nasıl yaptığımızı görelim:

Bildiğimiz üzere, $p(x)$ polinomunun $\mathbb{F}_q$ cisminde bir kökü yoktur. Eğer $\mathbb{F}_q$ cisminde bulunmayan bir sembolü, $p(x)$'in bir kökü olarak kabul edersek, ve inşa ettiğimiz cisimde x yerine bu sembolü kullanırsak cismimizi daha sade bir şekilde gösterebiliriz. Kullanacağımız sembolü, genel kullanımda da kabul gören α olarak seçelim. Bu durumda $p(\alpha) = 0$ olur. α'yı kullanarak cismin tüm elemanlarını teorik dille yazmak hem yazım hem de anlaşılırlık bakımından negatif bir etki yaratacağı için bir örnek üzerinden anlatmak daha işlevsel olur.

Örnek 2.4.11. $q = 2$ ve $m = 3$ olmak üzere $\mathbb{F}_{q^m} = \mathbb{F}_{2^3}$ cismini inşa edelim:

$\mathbb{F}_2 = \{0, 1\}$ olduğunu biliyoruz. $\mathbb{F}_2[x]$ polinom halkasından $p(x) = x^3+x+1$ polinomunu alalım.

$$p(0) = 0^3 + 0 + 1 = 1 \equiv 1 \pmod 2 \quad \text{ve} \quad p(1) = 1^3 + 1 + 1 = 3 \equiv 1 \pmod 2$$

olduğundan $p(x)$ polinomu $\mathbb{F}_2$ cismi üzerinde indirgenemezdir. Buradan söyleyebiliriz ki $\mathbb{F}_2[x]/\langle x^3 + x + 1\rangle$ bölüm halkası bir cisimdir.

$$\mathbb{F}_2[x]/\langle x^3 + x + 1\rangle = \{c_0 + c_1x + c_2x^2 \mid c_0, c_1, c_2 \in \mathbb{F}_2\}$$

Burada $x \leftrightarrow \alpha$ eşlemesi yapılarak cismin tüm elemanları α'nın kuvvetleri şeklinde yazılır. Bu yazım yapılırken öncelikle 0 polinomu 0 ile, 1 polinomu da $1 = \alpha^0$ ile eşleştirilir. Eşleme işlemini yaparken $x \leftrightarrow \alpha$ eşlemesi ile başladığımız için doğal olarak x polinomu α'nın kendisiyle eşleşir. Tahmin edilebileceği üzere x^2 polinomu α^2 ile eşleşir. Buradan hareketle α^3 sembolünü x^3 polinomu ile eşlememiz gerektiğini düşünebiliriz fakat cismimizde x^3 polinomu bulunmamaktadır. O yüzden x^3 polinomunun cisimde hangi eşlenik sınıfında olduğunu bulmamız gerekir. Cismimiz kalan sınıflarından oluştuğu için x^3 polinomunun eşlenik sınıfı $x^3 + x + 1$ polinomuna bölümünden kalandır.

$$x^3 = 1 \cdot (x^3 + x + 1) + (-x - 1)$$

olduğundan x^3 polinomunun kalan sınıfı $-x - 1$ polinomunun sınıfıdır. $\mathbb{F}_2$ cisminde $-1 = 1$ olduğundan bu polinom $x + 1$ polinomudur. Dolayısıyla $\alpha^3 \leftrightarrow x + 1$ eşlemesi yapılır. α^4 ise x^4 polinomunun $x^3 + x + 1$ polinomuna bölümünden kalandır.

$$x^4 = x(x^3 + x + 1) + (x^2 + x)$$

olduğundan α^4 sembolü $x^2 + x$ ile eşleşir.

$$x^5 = (x^2 + 1)(x^3 + x + 1) + (x^2 + x + 1)$$

olduğundan $\alpha^5 \leftrightarrow x^2 + x + 1$ eşlemesi yapılır. Ve son olarak

$$x^6 = (x^3 + x + 1)(x^3 + x + 1) + (x^2 + 1)$$

olduğundan $\alpha^6 \leftrightarrow x^2 + 1$ eşlemesi yapılarak bitirilir çünkü $\mathbb{F}_2[x]$ halkasında derecesi en fazla 2 olan başka bütün polinomların eşlemesi tamamlanmıştır. O halde cismimizi aşağıdaki şekilde, sadeleşmiş görünümüyle yazabiliriz:

$$\mathbb{F}_{2^3} = \{0, 1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6\}$$

Sonlu cisimlerin bu sade yazımı, işlemlerde rahatlık sağlasa da bazen α üzerinden işlem yapmak zorlayıcı olabiliyor. Örneğin, bilgisayarlar 0 ve 1 mantığıyla çalıştığı için α sembolünü ve sonlu cisimleri onların da anlayacağı bir hale getirmeliyiz. Bunun için sonlu cisimleri vektör uzayları ile ifade edebileceğimiz bir yöntem geliştirilmiştir. Bu yöntem, sonlu cismi oluşturan polinomların katsayılarını kullanarak vektör uzayının elemanlarını yazmaya dayanır. Bu yazım ve eşleştirmeyi yukarıda verdiğimiz örnek üzerinden aşağıdaki şekilde yapabiliriz:

$\langle\alpha\rangle \longleftrightarrow \mathbb{F}_{2^3} \longleftrightarrow \mathbb{F}_2^3$

0	0	$(0,0,0)$
1	1	$(0,0,1)$
α	x	$(0,1,0)$
α^2	x^2	$(1,0,0)$
α^3	$x+1$	$(0,1,1)$
α^4	x^2+x	$(1,1,0)$
α^5	x^2+x+1	$(1,1,1)$
α^6	x^2+1	$(1,0,1)$

Şekil 2.1. $\mathbb{F}_{2^3}$ ve $\mathbb{F}_2^3$ eşleştirmesi

2.5 Matrisler

Başta Hill şifreleme algoritması olmak üzere kriptografinin birçok yerinde matrislerden oldukça fazla faydalanırız. Bu bölümde matrislerle ilgili kriptografik anlamda önemli olan kısımlar anlatılmıştır ve bu anlatım yapılırken [14] numaralı kaynaktan faydalanılmıştır.

Tanım 2.5.1. (Matris) Satırlar ve sütunlardan oluşan ve bu satır ve sütunlarında girdileri olan dikdörtgensel yapıya matris denir. Bu girdilere matrisin elemanları da denilir. Matrisler genellikle $A, B, C, \ldots$ gibi büyük harflerle gösterilir.

Örnek 2.5.2. $A = \begin{bmatrix} 2 & 3 \\ -3 & \sqrt{7} \\ \pi & 0 \end{bmatrix}$, $B = \begin{bmatrix} 9 & 3,2 & 10^4 \\ -5 & \sqrt{2} & \frac{4}{9} \end{bmatrix}$, $C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ birer matristir.

Tanım 2.5.3. Bir matrisin en temel karakteristik özelliği, sahip olduğu satır ve sütun sayılarıyla ifade edilen büyüklüğü (tipi, formu)'dür. Örneğin a tane satıra ve b tane sütuna sahip bir A matrisi "$a \times b$ tipinde bir A matrisi" şeklinde ifade edilir.

Örnek 2.5.4. $Y = \begin{bmatrix} 1 & 4 & 7 & 19 \\ 5 & 0 & \sqrt{11} & 3 \\ 0 & 0 & 2 & 1 \end{bmatrix}$ matrisi 3×4 tipinde bir Y matrisidir.

Tanım 2.5.5. (Satır / Sütun Matrisi) Bir matris tek bir sütundan oluşuyorsa bu matrise sütun matrisi (veya sütun vektörü), tek bir satırdan oluşuyorsa satır matrisi (veya satır vektörü) denir.

Girdileri belirtmek istemediğimiz zaman, yani $m \times n$ tipinde genel bir A matrisi yazmak istediğimizde bunu

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}_{m \times n}$$

şeklinde yaparız. Veya daha kısa yazmak istersek doğrudan $A = [a_{ij}]_{m\times n}$ şeklinde de yazabiliriz.

Tanım 2.5.6. (Kare Matris, Köşegen, İz) Satır ve sütun sayısı eşit olan bir matrise kare matris denir. Kare matrislerin $a_{11}, a_{22}, \ldots, a_{nn}$ konumlarındaki elemanlarına köşegen elemanları denir. Bir kare matrisin köşegen elemanlarının toplamına matrisin izi denir ve $tr(A)$ veya $iz(A)$ ile gösterilir.

Tanım 2.5.7. (Birim Matris) Köşegen elemanları 1 ve diğer tüm girdileri 0 olan matrise birim matris denir. Birim matris, matrisin büyüklüğü de kullanılarak, $I_{n\times n}$ veya kısaca I_n ile gösterilir. (Herhangi bir karışıklığa yol açmadığı sürece doğrudan I şeklinde de yazılabilir.)

Tanım 2.5.8. (Sıfır Matrisi) Tüm girdileri 0 olan matrise sıfır matrisi denir.

Tanım 2.5.9. $A = [a_{ij}]_{m\times n}$ ve $B = [b_{ij}]_{m\times n}$ aynı satır ve sütun sayılarına sahip iki matris olsun. $\forall\ 1 \leq i \leq m$ ve $1 \leq j \leq n$ için $a_{ij} = b_{ij}$ ise A ve B matrislerine eşit matrisler denir ve $A = B$ ile gösterilir.

2.5.1 Matrislerde İşlemler

Tanım 2.5.10. (Toplama ve Çıkarma) $A = [a_{ij}]_{m\times n}$ ve $B = [b_{ij}]_{m\times n}$ aynı satır ve sütun sayılarına sahip iki matris olsun.
$\forall\ 1 \leq i \leq m$ ve $1 \leq j \leq n$ için $c_{ij} = a_{ij} \pm b_{ij}$ olmak üzere A ile B matrislerinin toplamı $A \pm B = [c_{ij}]_{m\times n}$ şeklinde hesaplanır. (Satır sayıları ve sütun sayıları aynı olmayan matrisler toplanamaz.)

Tanım 2.5.11. (Skalerle Çarpma) $A = [a_{ij}]_{m\times n}$ bir matris ve c de bir skaler olsun.
$\forall\ 1 \leq i \leq m$ ve $1 \leq j \leq n$ için $c_{ij} = c.a_{ij}$ olmak üzere c skaleri ile A matrisinin çarpımı $c.A = [c_{ij}]_{m\times n}$ şeklinde hesaplanır.

Tanım 2.5.12. (Çarpma) $A = [a_{ij}]_{m\times k}$ ve $B = [b_{ij}]_{k\times n}$ iki matris olsun.
$\forall\ 1 \leq i \leq m$, $1 \leq j \leq n$ için $c_{ij} = a_{i1}b_{1j} + a_{i2} + b_{2j} + \cdots + a_{ik}b_{kj}$ olmak üzere A matrisi ile B matrisinin çarpımı $AB = [c_{ij}]_{m\times n}$ şeklindedir. (Matrislerde çarpma işleminde

değişme özelliği yoktur. Ayrıca iki matrisi çarpabilmemiz için ilk matrisin sütun sayısı ile ikinci matrisin satır sayısı mutlaka eşit olmalıdır.)

Örnek 2.5.13. $A = \begin{bmatrix} 2 & 1 & 3 \\ 3 & 6 & 0 \end{bmatrix}_{2\times 3}$ ve $B = \begin{bmatrix} 1 & 4 & 1 & 3 \\ -1 & 0 & 3 & 1 \\ 7 & 2 & 5 & -2 \end{bmatrix}_{3\times 4}$ matrislerini çarpalım.

$$\begin{array}{rcccccl} c_{11} & = & a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} & = & 2.1 + 1.(-1) + 3.7 & = & 22 \\ c_{12} & = & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} & = & 2.4 + 1.0 + 3.2 & = & 14 \\ c_{13} & = & a_{11}b_{13} + a_{12}b_{23} + a_{13}b_{33} & = & 2.1 + 1.3 + 3.5 & = & 20 \\ \vdots & & & & & & \\ c_{24} & = & a_{21}b_{14} + a_{22}b_{24} + a_{23}b_{34} & = & 3.3 + 6.1 + 0.(-2) & = & 15 \end{array}$$

$\Rightarrow AB = \begin{bmatrix} 22 & 14 & 20 & 1 \\ -3 & 12 & 21 & 15 \end{bmatrix}_{2\times 4}$ olarak bulunur.

Tanım 2.5.14. (Transpoz) $A = [a_{ij}]_{m\times n}$ bir matris olsun. $\forall\ 1 \leq i \leq m$, $1 \leq j \leq n$ için $c_{ji} = a_{ij}$ eşitliği ile oluşturulan $C = [c_{ji}]_{n\times m}$ matrisine A matrisinin transpozu denir. $A^\top$ ile gösterilir.

Önerme 2.5.15. A, B iki matris olmak üzere, büyüklükleri işleme uygun olduğu sürece aşağıda verilenler sağlanır.

1. $(A^\top)^\top = A$

2. $(A \pm B)^\top = A^\top \pm B^\top$

3. $(AB)^\top = B^\top A^\top$

Kanıt. **1.** A $m \times n$ tipinde bir matris olsun.

$(a_{ij})^\top = (a_{ji}) \Rightarrow ((a_{ij})^\top)^\top = (a_{ji})^\top = a_{ij}$'dir.

$\Rightarrow (A^\top)^\top = A$'dır.

2. $A = [a_{ij}]_{m\times n}$ ve $B = [b_{ij}]_{m\times n}$ olmak üzere $a_{ij} \pm b_{ij} = c_{ij}$ olsun.

$\Rightarrow (a_{ij} \pm b_{ij})^\top = (c_{ij})^\top = c_{ji} = a_{ji} \pm b_{ji} = (a_{ij})^\top \pm (b_{ij})^\top$

$\Rightarrow (A \pm B)^\top = A^\top \pm B^\top$

3. $A = [a_{ij}]_{m\times k},\ B = [b_{ij}]_{k\times n}$ olmak üzere $AB = C = [c_{ij}]_{m\times n}$ olsun.
$(AB)^\top = (c_{ij})^\top = c_{ji} = \sum_{r=1}^{n} a_{jr}b_{ri} = \sum_{r=1}^{n}(a_{rj})^\top(b_{ir})^\top = B^\top A^\top$ □

Tanım 2.5.16. A bir kare matris olmak üzere $AB = BA = I_n$ olacak şekilde bir B matrisi bulunabiliyorsa A matrisine tersinir, B matrisine de A'nın tersi denir. Bir A matrisinin tersi A^{-1} ile gösterilir.

Teorem 2.5.17. A ve B $n \times n$ tipinde iki tersinir matris olsun. Bu durumda AB matrisi de tersinirdir.

Kanıt. $(AB)B^{-1}A^{-1} = A(BB^{-1})A^{-1} = AI_nA^{-1} = AA^{-1} = I_n$. Çarpma işlemini diğe taraftan da yapalım. $B^{-1}A^{-1}(AB) = B^{-1}(A^{-1}A)B = B^{-1}I_nB = B^{-1}B = I_n$. Bu iki eşitlikten AB çarpım matrisinin tersinin $B^{-1}A^{-1} = (AB)^{-1}$ matrisi olduğunu görürüz. □

Sonuç 2.5.18. n tane A matrisi kullanılarak yapılan $AAA\ldots A$ çarpımı kısaca A^n şeklinde gösterilir ve aşağıdaki özellikler tersinir matrisler tarafından sağlanır.

1. $A^0 = I$
2. $\forall\, n \in \mathbb{Z},\ (A^{-1})^n = (A^n)^{-1}$
3. $\forall\, r, s \in \mathbb{Z},\ A^rA^s = A^{r+s}$
4. $(A^r)^s = A^{rs}$
5. $(A^{-1})^{-1} = A$
6. $(A^n)^{-1} = (A^{-1})^n$
7. $(kA)^{-1} = \frac{1}{k}A^{-1}$
8. $(A^\top)^{-1} = (A^{-1})^\top$

Tanım 2.5.19. (Alt / Üst Üçgensel Matris) Bir kare matrisin köşegen elemanlarının üstündeki girdilerin tamamı sıfır ise bu matrise alt-üçgensel matris, köşegen elemanlarının altındaki girdilerin tamamı sıfır ise bu matrise üst-üçgensel matris denir.

Örnek 2.5.20. $A = \begin{bmatrix} 2 & 0 & 0 \\ 5 & 9 & 0 \\ 1 & 4 & 2 \end{bmatrix}$ alt-üçgensel, $B = \begin{bmatrix} 2 & 5 & 8 \\ 0 & 9 & 7 \\ 0 & 0 & 2 \end{bmatrix}$ üst-üçgensel bir matristir.

Tanım 2.5.21. (Simetrik ve Ters-Simetrik Matris) A bir kare matris olmak üzere $A = A^\top$ ise A matrisine simetrik matris, $A = -A^\top$ ise ters-simetrik matris denir. Ters-simetrik matrislerin köşegen elemanları daima sıfır olmak zorundadır.

Örnek 2.5.22. $C = \begin{bmatrix} 2 & 5 & 1 \\ 5 & 9 & 4 \\ 1 & 4 & 2 \end{bmatrix}$ simetrik, $D = \begin{bmatrix} 0 & 5 & -8 \\ -5 & 0 & 7 \\ 8 & -7 & 0 \end{bmatrix}$ ters-simetriktir.

2.5.2 Determinantlar

Tanım 2.5.23. (Determinant Fonksiyonu) Kare matrisleri, matrisin elemanlarını yazarken kullandığımız cismin bir elemanına eşleyen ve her kare matris için belirli bir sonuç veren bir fonksiyon vardır. Bu fonksiyona determinant fonksiyonu denir. Bir A kare matrisinin determinantı $det(A)$ veya kısaca $|A|$ ile gösterilir

Şimdi determinant fonksiyonunun nasıl hesaplandığını görelim:

- 1×1 formundaki bir $A = [a_{11}]_{1\times 1}$ matrisinin determinantı a_{11}'dir.
  Örneğin, $A = [7]_{1\times 1}$ matrisi için $det(A) = 7$'dir.
- 2×2 formundaki bir $A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$ matrisinin determinantı $a_{11}a_{22} - a_{12}a_{21}$'dir.

  Örneğin, $A = \begin{bmatrix} 3 & 7 \\ 2 & 9 \end{bmatrix}$ matrisinin için $det(A) = 3 \cdot 9 - 7 \cdot 2 = 13$'dir.
- 3×3 formundaki bir $A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$ matrisinin determinantı

  $$|A| = (a_{11})(-1)^{1+1}\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} + (a_{12})(-1)^{1+2}\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + (a_{13})(-1)^{1+3}\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}$$

  şeklinde hesaplanır.
- İlk üç durumdan da anlayacağımız üzere determinant hesaplanırken öncelikle bir satır veya sütun seçilir. Ardından seçilen satırın veya sütunun elemanları sırasıyla işleme alınır. İşleme alınan elemanın bulunduğu satır ve sütundaki elamanlar görmezden gelinir ve kalan kısmın determinantı hesaplanır. Ve sırasıyla, seçilen eleman, -1 sayısı üzerinde seçilen elamanın bulunduğu satır ve sütun numaralarının toplamı ve ardından

kalan kısmın determinantı çarpılır. Bu işlem o sıradaki tüm elemanlar için ayrı ayrı yapılarak toplanır.

Tanım 2.5.24. (Minör ve Kofaktör) $A = [a_{ij}]_{n\times n}$ bir kare matris olmak üzere A'nın $i.$ satır ve $j.$ sütununun silinmesi ile oluşan matrisin determinantına a_{ij} elamanının minörü denir. M_{ij} ile gösterilir. Ek olarak, $c_{ij} = (-1)^{i+j}|M_{ij}|$ değerine de a_{ij} elemanının eş çarpanı *(kofaktörü)* denir.

Örnek 2.5.25. $A = \begin{bmatrix} 7 & 9 & 1 \\ 5 & 2 & 4 \\ 2 & 7 & 6 \end{bmatrix}$ matrisi için c_{31} kofaktörünü bulalım.

$$c_{31} = (-1)^{3+1}\begin{vmatrix} 9 & 1 \\ 2 & 4 \end{vmatrix} = (9 \cdot 4 - 1 \cdot 2) = 34$$

Sonuç 2.5.26. Bir A kare matrisinin determinantı herhangi bir satır veya sütunundaki tüm elemanlar ile bu elamanların kofaktörlerinin çarpımlarının toplamına eşittir. Yani;
$det(A) = a_{i1}c_{i1} + a_{i2}c_{i2} + \cdots + a_{in}c_{in}$ veya $det(A) = a_{1j}c_{1j} + a_{2j}c_{2j} + \cdots + a_{mj}c_{mj}$'dir.

Önerme 2.5.27. Eğer bir matrisin elemanları bir cisimden alınıyorsa tersinir olması için gerek ve yeter şart determinantının sıfırdan farklı olmasıdır.

Önerme 2.5.28. A bir tersinir matris olmak üzere $det(A^{-1}) = \frac{1}{det(A)}$'dır.

Tanım 2.5.29. (Katsayılar Matrisi) Bir denklem sistemindeki her bir denklemin katsayılarını satır elemanı olarak kabul eden matrise katsayılar matrisi denir. Eğer denklem sisteminde eşitliğin sağ tarafındaki sabitleri de matrise eklersek oluşturduğumuz matrise ilaveli (artırılmış) matris denir.

Örnek 2.5.30.
$\begin{aligned} 2x + 3y + 4z &= 5 \\ 4x + 2y + 7z &= 2 \end{aligned}$ denklem sisteminin katsayılar matrisi ve ilaveli matrisi sırasıyla

$\begin{bmatrix} 2 & 3 & 4 \\ 4 & 2 & 7 \end{bmatrix}$ ve $\left[\begin{array}{ccc|c} 2 & 3 & 4 & 5 \\ 4 & 2 & 7 & 2 \end{array}\right]$ matrisleridir.

Denklem sistemlerini matrisler yardımıyla çözmek büyük kolaylık sağlar. Önce matrislerde elementer satır işlemlerini, ardından da bir denklem sistemini matrisler yardımıyla nasıl çözeceğimizi görelim.

2.5.3 Elementer İşlemler

Tanım 2.5.31. (Elementer Satır İşlemleri) Bir matris üzerinde yapılan aşağıdaki üç işlem tipine elementer satır işlemleri denir.

Tip 1 : Matrisin bir satırını bir skalerle çarpmak

Tip 2 : Matrisin bir satırının skaler bir katını başka bir satırına eklemek

Tip 3 : Matrisin iki satırının yerlerini değiştirmek

Tanım 2.5.32. (Elementer Matris) Bir önceki tanımda bahsedilen işlemleri birim matrise uygulayarak elde edilen matrise elementer matris denir.

Buradan da anlaşılacağı üzere, bir A matrisine bir elementer işlem uygulamak istediğimizde, aynı işlemi birim matrise uygulayarak elde edilen elementer matrisi A matrisi ile soldan çarpmak yeterlidir.

Örnek 2.5.33. $A = \begin{bmatrix} 5 & 1 & -6 & -7 \\ 4 & -3 & 9 & 1 \\ 6 & 2 & -5 & 8 \end{bmatrix}$ matrisini alalım ve bu matrise üç tip elementer işlemi de ayrı ayrı uygulayalım.

- **Tip 1 :** 2. satırını 4 ile çarpalım:

$$A_1 = \begin{bmatrix} 5 & 1 & -6 & -7 \\ 4.4 & 4.(-3) & 4.9 & 4.1 \\ 6 & 2 & -5 & 8 \end{bmatrix} = \begin{bmatrix} 5 & 1 & -6 & -7 \\ 16 & -12 & 36 & 4 \\ 6 & 2 & -5 & 8 \end{bmatrix}$$

Şimdi de I_3 birim matrisinin 2. satırının 4 ile çarpılmış halini A matrisi ile çarpalım:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 5 & 1 & -6 & -7 \\ 4 & -3 & 9 & 1 \\ 6 & 2 & -5 & 8 \end{bmatrix} = \begin{bmatrix} 5 & 1 & -6 & -7 \\ 16 & -12 & 36 & 4 \\ 6 & 2 & -5 & 8 \end{bmatrix}$$

Görüldüğü gibi her iki durumda da aynı sonuca ulaşılmaktadır.

- **Tip 2 :** 3. satırın 2 katını 1. satıra ekleyelim:

$$A_2 = \begin{bmatrix} 5+2\cdot 6 & 1+2\cdot 2 & -6+2\cdot(-5) & -7+2\cdot 8 \\ 4 & -3 & 9 & 1 \\ 6 & 2 & -5 & 8 \end{bmatrix} = \begin{bmatrix} 17 & 5 & -16 & 9 \\ 4 & -3 & 9 & 1 \\ 6 & 2 & -5 & 8 \end{bmatrix}$$

Şimdi de I_3 birim matrisine aynı işlem uygulanmış hali ile A matrisini soldan çarpalım:

$$\begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 5 & 1 & -6 & -7 \\ 4 & -3 & 9 & 1 \\ 6 & 2 & -5 & 8 \end{bmatrix} = \begin{bmatrix} 17 & 5 & -16 & 9 \\ 4 & -3 & 9 & 1 \\ 6 & 2 & -5 & 8 \end{bmatrix}$$

Görüldüğü gibi her iki durumda da aynı sonuca ulaşılmaktadır.

- **Tip 3 :** 2. satır ile 3. satırın yerlerini değiştirelim:

$$A_3 = \begin{bmatrix} 5 & 1 & -6 & -7 \\ 6 & 2 & -5 & 8 \\ 4 & -3 & 9 & 1 \end{bmatrix}$$

Şimdi de I_3 birim matrisine aynı işlemin uygulanmış ahalini A ile soldan çarpalım:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 5 & 1 & -6 & -7 \\ 4 & -3 & 9 & 1 \\ 6 & 2 & -5 & 8 \end{bmatrix} = \begin{bmatrix} 5 & 1 & -6 & -7 \\ 6 & 2 & -5 & 8 \\ 4 & -3 & 9 & 1 \end{bmatrix}$$

Görüldüğü üzere her iki durumda da aynı sonuca ulaşılmaktadır.

Tanım 2.5.34. Matrislerin satırlarında soldan başlandığında sıfırdan farklı olan ilk elemana satırın baş girdisi (baş elamanı) denir. Eğer bu eleman 1 ise pivot eleman veya kısaca pivot denir.

Tanım 2.5.35. (Satır İndirgenmiş Eşelon Form) Aşağıdaki koşulları sağlayan matrise satır indirgenmiş eşelon form denir.

1. Her satırın baş elemanı pivottur.
2. Her satırın pivot elemanı bir üst satırın pivotundan daha sağdadır.
3. Pivot bulunduran sütunda pivot haricindeki tüm girdiler sıfırdır.
4. Eğer tamamen sıfırdan oluşan satırlar varsa bu satırlar matrisin en altındadır.

Bir denklem sisteminin ilaveli katsayılar matrisini elementer satır işlemlerini kullanarak satır indirgenmiş eşelon forma getirip yeni katsayılarla birlikte yeni denklem sistemini yazabiliriz. Yeni denklem sistemi ile eski denklem sisteminin çözüm kümeleri aynıdır fakat yeni denklem sistemini çözmek çok daha kolaydır. Bunu bir örnek üzerinde görelim:

Örnek 2.5.36.

$$\begin{array}{rcrcrcl} x & + & 2y & + & z & = & 1 \\ 3x & + & 5y & + & 7z & = & 3 \\ 2x & + & 6y & + & 7z & = & 1 \end{array}$$ denklem sistemini çözelim.

Denklem sisteminin ilaveli katsayılar matrisini yazalım:

$$A = \left[\begin{array}{ccc|c} 1 & 2 & 1 & 1 \\ 3 & 5 & 7 & 3 \\ 2 & 6 & 7 & 1 \end{array}\right]$$

1. satırın -3 katını 2. satıra ekleyelim:

$$A \to A_1 = \left[\begin{array}{ccc|c} 1 & 2 & 1 & 1 \\ 3-3\cdot 1 & 5-3\cdot 2 & 7-3\cdot 1 & 3-3\cdot 1 \\ 2 & 6 & 7 & 1 \end{array}\right] = \left[\begin{array}{ccc|c} 1 & 2 & 1 & 1 \\ 0 & -1 & 4 & 0 \\ 2 & 6 & 7 & 1 \end{array}\right]$$

2. satırı -1 ile çarpalım:

$$A_1 \to A_2 = \left[\begin{array}{ccc|c} 1 & 2 & 1 & 1 \\ 0 & 1 & -4 & 0 \\ 2 & 6 & 7 & 1 \end{array}\right]$$

1. satırın -2 katını 3. satıra ekleyelim:

$$A_2 \to A_3 = \left[\begin{array}{ccc|c} 1 & 2 & 1 & 1 \\ 0 & 1 & -4 & 0 \\ 2-2\cdot 1 & 6-2\cdot 2 & 7-2\cdot 1 & 1-2\cdot 1 \end{array}\right] = \left[\begin{array}{ccc|c} 1 & 2 & 1 & 1 \\ 0 & 1 & -4 & 0 \\ 0 & 2 & 5 & -1 \end{array}\right]$$

3. satırı $\frac{1}{2}$ ile çarpalım:

$$A_3 \to A_4 = \left[\begin{array}{ccc|c} 1 & 2 & 1 & 1 \\ 0 & 1 & -4 & 0 \\ 0 & 1 & 5/2 & -1/2 \end{array}\right]$$

2. satırın -1 katını 3. satıra ekleyelim:

$$A_4 \to A_5 = \left[\begin{array}{ccc|c} 1 & 2 & 1 & 1 \\ 0 & 1 & -4 & 0 \\ 0 & 1-1 & 5/2+4 & -1/2 \end{array}\right] = \left[\begin{array}{ccc|c} 1 & 2 & 1 & 1 \\ 0 & 1 & -4 & 0 \\ 0 & 0 & 13/2 & -1/2 \end{array}\right]$$

3. satırı $\frac{2}{13}$ ile çarpalım:

$$A_5 \to A_6 = \left[\begin{array}{ccc|c} 1 & 2 & 1 & 1 \\ 0 & 1 & -4 & 0 \\ 0 & 0 & 1 & -1/13 \end{array}\right]$$

3. satırın 4 katını 2. satıra ekleyelim:

$$A_6 \to A_7 = \left[\begin{array}{ccc|c} 1 & 2 & 1 & 1 \\ 0 & 1 & -4+4 & 0-4/13 \\ 0 & 0 & 1 & -1/13 \end{array}\right] = \left[\begin{array}{ccc|c} 1 & 2 & 1 & 1 \\ 0 & 1 & 0 & -4/13 \\ 0 & 0 & 1 & -1/13 \end{array}\right]$$

3. satırın -1 katını 1. satıra ekleyelim:

$$A_7 \to A_8 = \left[\begin{array}{ccc|c} 1 & 2 & 1-1 & 1+1/13 \\ 0 & 1 & 0 & -4/13 \\ 0 & 0 & 1 & -1/13 \end{array}\right] = \left[\begin{array}{ccc|c} 1 & 2 & 0 & 14/13 \\ 0 & 1 & 0 & -4/13 \\ 0 & 0 & 1 & -1/13 \end{array}\right]$$

2. satırın -2 katını 1. satıra ekleyelim:

$$A_8 \to A_9 = \left[\begin{array}{ccc|c} 1 & 2-2 & 0 & 14/13+8/13 \\ 0 & 1 & 0 & -4/13 \\ 0 & 0 & 1 & -1/13 \end{array}\right] = \left[\begin{array}{ccc|c} 1 & 0 & 0 & 22/13 \\ 0 & 1 & 0 & -4/13 \\ 0 & 0 & 1 & -1/13 \end{array}\right]$$

Elementer işlemler sonucunda elde elde ettiğimiz A_9 matrisini kullanarak denklem sisteminin yeni halini ve çözümünü yazalım:

$$\begin{array}{rcrcrcl} x + 0\cdot y + 0\cdot z & = & 22/13 & & & x & = & 22/13 \\ 0\cdot x + y + 0\cdot z & = & -4/13 & \Rightarrow & & y & = & -4/13 \\ 0\cdot x + 0\cdot y + z & = & -1/13 & & & z & = & -1/13 \end{array}$$

Sonuç 2.5.37. Bir denklem sisteminin katsayılar matrisi tersinir kare matris ise denklemin çözüm kümesi tek türlüdür. Çünkü katsayılar matrisinin satır indirgenmiş eşelon formu birim matris olur. Aksi taktirde matrisin satır indirgenmiş eşelon formundaki sıfırdan farklı satırların sayısına bakılır: Eğer değişken sayısından daha az ise denklemin çözüm kümesi parametrelere bağlı çıkar.

Tanım 2.5.38. (Rank) Bir matrisin satır indirgenmiş eşelon formundaki sıfırdan farklı satır sayısına matrisin rankı denir. Bir A matrisinin rankı $Rank(A)$ veya kısaca $Rk(A)$ ile gösterilir.

2.6 Vektör Uzayları

Kafes yapılarını ve kafes-tabanlı kriptosislemlerin işleyişlerini daha iyi anlayabilmek için vektör uzayları konusuna hakim olmak gerekir. Bu bölümde vektör uzaylarıyla ilgili bilmemiz gerekenler anlatılmıştır. Bu anlatım yapılırken [15] kaynağından yararlanılmıştır.

Tanım 2.6.1. (Vektör Uzayı) $V \neq \emptyset$ bir küme ve $(F, +, \cdot)$ bir cisim olmak üzere aşağıda tanımlanan işlemlerle birlikte verilen aksiyomlar sağlanıyorsa V'ye bir vektör uzayı denir. V_F ile gösterilir.

$$\begin{array}{rcccl} \oplus & : & V \times V & \to & V \\ & & (u, v) & \mapsto & u \oplus v \end{array} \qquad \begin{array}{rcccl} \odot & : & F \times V & \to & V \\ & & (\lambda, u) & \mapsto & \lambda \odot u \end{array}$$

1. $\forall\, u, v \in V$ için $u \oplus v \in V$
2. $\forall\, u, v, w \in V$ için $(u \oplus v) \oplus w = u \oplus (v \oplus w)$
3. $\forall\, u \in V$ için $\exists\, e \in V$ öyle ki $e \oplus u = u \oplus e = u$
4. $\forall\, u \in V$ için $\exists\, u' \in V$ öyle ki $u \oplus u' = u' \oplus u = e$
5. $\forall\, u, v \in V$ için $u \oplus v = v \oplus u$
6. $\forall\, \lambda \in F$ ve $\forall\, u, v \in V$ için $\lambda \odot (u \oplus v) = (\lambda \odot u) \oplus (\lambda \odot v)$
7. $\forall\, \lambda, \mu \in F$ ve $\forall\, u \in V$ için $(\lambda + \mu) \odot u = (\lambda \odot u) \oplus (\mu \odot u)$
8. $\forall\, \lambda, \mu \in F$ ve $\forall\, u \in V$ için $\lambda \odot (\mu \odot u) = (\lambda \cdot \mu) \odot u$
9. $\forall\, u \in V$ için $\exists\, 1_F \in F$ öyle ki $1_F \odot u = u \odot 1_F = u$

Örnek 2.6.2. Her cisim kendi üzerinde bir vektör uzayıdır. Dolayısıyla $\mathbb{Q}_{\mathbb{Q}}, \mathbb{R}_{\mathbb{R}}, \mathbb{C}_{\mathbb{C}}$ birer vektör uzayıdır.

Örnek 2.6.3. $x = (x_1, x_2, \ldots, x_n)$ ve $y = (y_1, y_2, \ldots, y_n)$ olmak üzere aşağıda tanımlanan işlemlerle birlikte $\mathbb{R}^n = \left\{(x_1, x_2, \ldots, x_n) \mid \forall\, i,\ x_i \in \mathbb{R}\right\}$ kümesi $\mathbb{R}$ üzerinde bir vektör uzayıdır.

$$\begin{array}{rcccl} \oplus & : & \mathbb{R}^n \times \mathbb{R}^n & \to & \mathbb{R}^n \\ & & (x, y) & \mapsto & x \oplus y = (x_1 + y_1, x_2 + y_2, \ldots, x_n + y_n) \\ \odot & : & \mathbb{R} \times \mathbb{R}^n & & \\ & & (r, x) & \mapsto & r \odot x = (r \cdot x_1, r \cdot x_2, \ldots, r \cdot x_n) \end{array}$$

Örnek 2.6.4. Girdilerini $\mathbb{R}$'den alan $m \times n$ tipindeki matrislerin kümesi matrislerde toplama ve skalerle çarpma işlemleri ile birlikte $\mathbb{R}$ üzerinde bir vektör uzayıdır. Vektör uzayı aksiyomlarını sağladığını sırasıyla gösterelim.

(Yazımda kolaylık olması amacıyla $m \times n$ tipindeki matrisleri $[a_{ij}]$ şeklinde yazalım ve girdilerini reel sayılardan alan $m \times n$ tipindeki matrislerin kümesini de $\mathbb{R}_{m\times n}$ ile gösterelim.)

1. $[a_{ij}], [b_{ij}] \in \mathbb{R}_{m\times n}$ olsun.
   $[a_{ij}] + [b_{ij}] = [a_{ij} + b_{ij}] \in \mathbb{R}_{m\times n}$'dir.

2. $[a_{ij}], [b_{ij}], [c_{ij}] \in \mathbb{R}_{m\times n}$ olsun.
   $([a_{ij}] + [b_{ij}]) + [c_{ij}] = [a_{ij}] + ([b_{ij}] + [c_{ij}])$'dir.

3. $\forall\, [a_{ij}] \in \mathbb{R}_{m\times n}$ için $[a_{ij}] + [0]_{m\times n} = [0]_{m\times n} + [a_{ij}] = [a_{ij}]$'dir. Yani $e = [0]_{m\times n}$'dir.

4. $\forall\, [a_{ij}] \in \mathbb{R}_{m\times n}$ için $[a_{ij}] + [-a_{ij}] = [-a_{ij}] + [a_{ij}] = [0]_{m\times n} = e$'dir.

5. $\forall\, [a_{ij}], [b_{ij}] \in \mathbb{R}_{m\times n}$ için $[a_{ij}] + [b_{ij}] = [a_{ij} + b_{ij}] = [b_{ij} + a_{ij}] = [b_{ij}] + [a_{ij}]$'dir.

6. $\forall\, \lambda \in \mathbb{R}$ ve $\forall\, [a_{ij}], [b_{ij}] \in \mathbb{R}_{m\times n}$ için,
   $\lambda([a_{ij}]+[b_{ij}]) = \lambda[a_{ij}+b_{ij}] = [\lambda(a_{ij}+b_{ij})] = [(\lambda a)_{ij}+(\lambda b)_{ij}] = [(\lambda a)_{ij}]+[(\lambda b)_{ij}] = \lambda[a_{ij}] + \lambda[b_{ij}]$'dir.

7. $\forall\, \lambda, \mu \in \mathbb{R}$ ve $\forall\, [a_{ij}] \in \mathbb{R}_{m\times n}$ için,
   $(\lambda + \mu)[a_{ij}] = [((\lambda + \mu)a)_{ij}] = [(\lambda a)_{ij}] + [(\mu a)_{ij}] = \lambda[a_{ij}] + \mu[a_{ij}]$'dir.

8. $\forall\, \lambda, \mu \in \mathbb{R}$ ve $\forall\, [a_{ij}] \in \mathbb{R}_{m\times n}$ için,
   $\lambda(\mu[a_{ij}]) = \lambda[(\mu a)_{ij}] = [(\lambda\mu a)_{ij}] = (\lambda\mu)[a_{ij}]$'dir.

9. $\forall\, [a_{ij}] \in \mathbb{R}_{m\times n}$ için $[a_{ij}].1_{\mathbb{R}} = 1_{\mathbb{R}}.[a_{ij}] = [a_{ij}]$'dir.

Örnek 2.6.5. $\varphi(I) = \left\{ f \mid f : I \subseteq \mathbb{R} \to \mathbb{R} \right\}$ kümesi $a, b \in \mathbb{R}$ olmak üzere $I = [a, b]$ aralığında tanımlı reel değerli fonksiyonların kümesi olsun. Aşağıda tanımlanan işlemlerle birlikte $\varphi(I)$ kümesi $\mathbb{R}$ üzerinde bir vektör uzayıdır.

$$
\begin{array}{rccl}
+ : & \varphi(I) \times \varphi(I) & \to & \varphi(I) \\
 & (f, g) & \mapsto & f + g \;\; (\forall\, t \in \mathbb{R},\; (f + g)(t) = f(t) + g(t)) \\
\cdot : & \mathbb{R} \times \varphi(I) & \to & \varphi(I) \\
 & (r, f) & \mapsto & r \cdot f \;\; (\forall\, t \in \mathbb{R},\; (r \cdot f)(t) = r \cdot f(t))
\end{array}
$$

Tanım 2.6.6. (Altuzay) $V_{\mathbb{F}}$ bir vektör uzayı ve $W \subseteq V$ bir alt kümesi olsun. Aşağıdaki koşullar sağlanıyorsa W'ye $V_{\mathbb{F}}$'nin bir altuzayı denir.

1. $0 \in W$
2. $\forall\, v, w \in W$ için $v + w \in W$
3. $\forall\, \lambda \in \mathbb{F}$ ve $\forall\, v \in W$ için $\lambda v \in W$

Örnek 2.6.7. Her vektör uzayı kendisinin bir altuzayıdır.

Örnek 2.6.8. $m \times n$ tipindeki reel girdili matrislerin kümesinin matrislerdeki toplama ve skalerle çarpma işlemleri ile birlikte $\mathbb{R}$ üzerinde bir vektör uzayı olduğunu söyledik. Buradan söyleyebiliriz ki, genelliği bozmayacağı için $n \times n$ tipindeki reel girdili kare matrisler kümesi de $\mathbb{R}$ üzerinde bir vektör uzayıdır ve aşağıda verilen matris kümeleri bu uzayın birer altuzayıdır.

$$\begin{aligned} \boldsymbol{U}_{n\times n} &= \{n \times n \text{ tipindeki üst üçgensel matrisler }\} \\ \boldsymbol{L}_{n\times n} &= \{n \times n \text{ tipindeki alt üçgensel matrisler }\} \\ \boldsymbol{D}_{n\times n} &= \{n \times n \text{ tipindeki köşegen matrisler }\} \end{aligned}$$

Tanım 2.6.9. $V_{\mathbb{F}}$ bir vektör uzayı ve $v_1, v_2, \ldots, v_k \in V$ olsun.

$$Span(v_1, v_2, \ldots, v_k) = \left\{ c_1v_1 + c_2v_2 + \cdots + c_kv_k \mid \forall\, 1 \leq i \leq k;\ c_1, c_2, \ldots, c_k \in \mathbb{F} \right\}$$

kümesine $v_1, v_2, \ldots, v_k \in V$ elemanları tarafından üretilen (gerilen) küme denir. Kısaca $\langle v_1, v_2, \ldots, v_k \rangle$ ile gösterilir.

2.6.1 Lineer Bağımsızlık, Taban ve Boyut

Tanım 2.6.10. (Lineer Bağımsızlık) $c_1, c_2, \ldots, c_k \in \mathbb{F}$ ve $v_1, v_2, \ldots, v_k \in V$ olmak üzere $c_1v_1 + c_2v_2 + \cdots + c_kv_k = 0$ eşitliği sadece $c_1 = c_2 = \cdots = c_k = 0$ iken sağlanıyorsa $\{c_1, c_2, \ldots, c_k\}$ kümesine lineer bağımsız denir. Aksi takdirde lineer bağımlı denir.

Tanım 2.6.11. (Taban) $v_1, v_2, \ldots, v_k \in V$ olmak üzere $S = \{v_1, v_2, \ldots, v_k\}$ kümesi için aşağıdaki koşullar sağlanıyorsa S kümesine V için bir tabandır denir.

1. $\{v_1, v_2, \ldots, v_k\}$ lineer bağımsız
2. $\langle v_1, v_2, \ldots, v_k \rangle = V$

Örnek 2.6.12. $S = \{(1,0),(0,1)\} \subseteq \mathbb{R}^2$ kümesi $\mathbb{R}^2_{\mathbb{R}}$ vektör uzayı için bir tabandır.

1. $c_1(1,0) + c_2(0,1) = (0,0)$
   $\Rightarrow (c_1,0) + (0,c_2) = (0,0)$
   $\Rightarrow (c_1,c_2) = (0,0)$
   $\Rightarrow c_1 = c_2 = 0$
   $\Rightarrow S$ kümesi lineer bağımsızdır.

2. $\langle (1,0),(0,1)\rangle = \mathbb{R}^2$ olduğunu göstereceğiz.
   $(\subseteq) : \langle (1,0),(0,1)\rangle$ kümesinden aldığımız her elemanın $\mathbb{R}^2$'de de olduğu aşikardır.
   $(\supseteq)$: $\mathbb{R}^2$den aldığımız her (a,b) elemanın $\langle (1,0),(0,1)\rangle$ kümesinde olması için $(a,b) = c_1(1,0) + c_2(0,1)$ denkleminin bir (c_1,c_2) çözümünün olması gerekir.
   $c_1(1,0) + c_2(0,1) = (a,b)$
   $\Rightarrow (c_1,0) + (0,c_2) = (a,b)$
   $\Rightarrow (c_1,c_2) = (a,b)$
   $\Rightarrow c_1 = a$ ve $c_2 = b$ olarak bulunur.

Tanım 2.6.13. (Boyut) Bir vektör uzayının herhangi bir tabanındaki eleman sayısına vektör uzayının boyutu denir. V bir vektör uzayı olmak üzere V'nin boyutu $dim(V)$ ile gösterilir.

Tanım 2.6.14. Taban kümesi sonlu sayıda elemana sahip olan vektör uzayına sonlu boyutlu vektör uzayı denir.

Önerme 2.6.15. V bir sonlu boyutlu vektör uzayı ve $\{v_1, v_2, \ldots, v_k\}$ kümesi de V için bir taban olsun. $w_1, w_2, \ldots, w_l \in V$ ve $k < l$ ise $\{w_1, w_2, \ldots, w_l\}$ kümesi lineer bağımlıdır.

Kanıt. $\{v_1, v_2, \ldots, v_k\}$ kümesi V'nin bir tabanı ve $\forall\, i$ için $w_i \in V$ olduğundan w_i'leri

$$\begin{array}{rcccccccc}
w_1 & = & a_{11}v_1 & + & a_{21}v_2 & + & \cdots & + & a_{k1}v_k \\
w_2 & = & a_{12}v_1 & + & a_{22}v_2 & + & \cdots & + & a_{k2}v_k \\
\vdots & & & & & & & & \\
w_l & = & a_{1l}v_1 & + & a_{2l}v_2 & + & \cdots & + & a_{kl}v_k
\end{array}$$

şeklinde yazarak bir denklem sistemi elde ederiz ve bu denklem sisteminin $A = [a_{ij}]$ katsayılar matrisini oluşturabiliriz. $k < l$ olduğundan $A.x = 0$ denklemini sağlayan sıfırdan farklı bir c vektörü bulunabilir.

$$\Rightarrow \sum_{j=1}^{l} c_j w_j = \sum_{j=1}^{l} c_j (\sum_{i=1}^{k} a_{ij} v_i) = \sum_{i=1}^{k} (\sum_{j=1}^{l} a_{ij} c_j) = 0$$

O halde $w_1, w_2, \ldots, w_l$'ler için aşikar olmayan bir çözüm vardır ve böylelikle $\{w_1, w_2, \ldots, w_l\}$ kümesi lineer bağımlıdır. □

Sonuç 2.6.16. Bir vektör uzayının tüm tabanlarındaki eleman sayıları eşittir. Bu sebeple bir vektör uzayının boyutu sabittir.

2.7 Kafesler

Kafes-tabanlı kriptografik algoritmalar, kafes yapıları üzerinde çözümünün henüz bulunamadığı matematiksel problemler temel alınarak oluşturulmuştur. Kafes-tabanlı algoritmaların özel olarak hangi kafes problemlerini kullandığı kafes-tabanlı algoritmalar bölümünde anlatılmıştır. Bu kısımda sadece temel bir kafes tanımı ve birkaç örneği verilmiştir. Bu anlatım yapılırken [16] numaralı kaynaktan faydalanılmıştır.

Kısaca söylemek gerekirse, kafesler, elemanları periyodik olarak sıralanmış n-boyutlu uzaylardır. Daha resmi bir tanım yapalım.

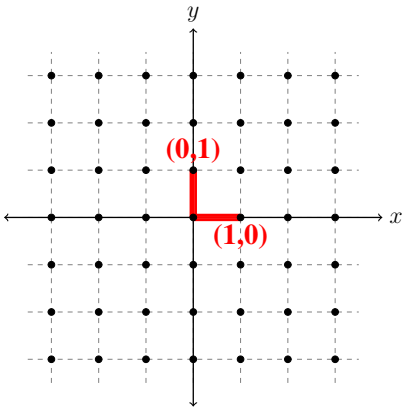
Tanım 2.7.1. (**Kafes**) V bir vektör uzayı, $v_1, v_2, \ldots, v_k \in V$ lineer bağımsız vektörler olsun. Bu vektörlerin, katsayılarını tam sayılardan alarak oluşturulmuş lineer kombinasyonlarıyla elde edilen kümeye kafes denir. Kafesler genellikle L ile gösterilir. Bir kafesi matematiksel olarak aşağıdaki şekilde gösterebiliriz.

$$L(v_1, v_2, \ldots, v_k) = \left\{ \sum_{i=1}^{k} \alpha_i v_i \mid \alpha_i \in \mathbb{Z} \right\}.$$

Tanım 2.7.2. $L(v_1, v_2, \ldots, v_k)$ bir kafes olsun. L kafesini oluşturan $v_1, v_2, \ldots, v_k$ vektörlerine kafesin taban elemanları denir. Dolayısıyla $\{v_1, v_2, \ldots, v_k\} \subseteq V$ altkümesine de kafesin tabanı denir.

Örnek 2.7.3. $\{\{1,0\},(0,1)\} \subset \mathbb{R}^2$ altkümesini taban kabul eden kafes aşağıda verilen kümedir.

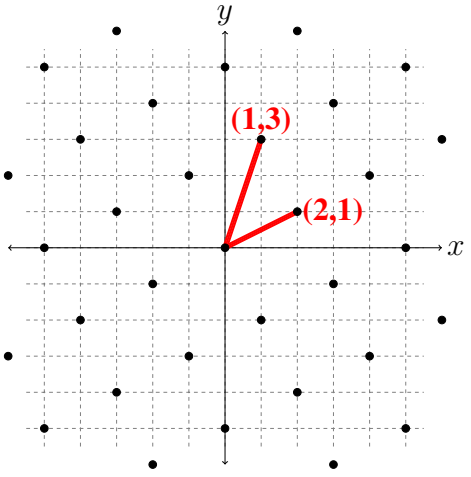
$$L\big((0,1),(0,1)\big) = \big\{\alpha_1(1,0) + \alpha_2(0,1) \mid \alpha_1, \alpha_2 \in \mathbb{Z}\big\} = \mathbb{Z}^2$$



Şekil 2.2. $(1,0)$ ve $(0,1)$ vektörleri tarafından üretilen kafes

Örnek 2.7.4. $\big\{(1,3),(2,1)\big\} \subseteq \mathbb{R}^2$ altkümesini taban kabul eden kafes aşağıda verilen kümedir.

$$L\big((1,3),(2,1)\big) = \big\{\alpha_1(1,3) + \alpha_2(2,1) \mid \alpha_1, \alpha_2 \in \mathbb{Z}\big\}$$

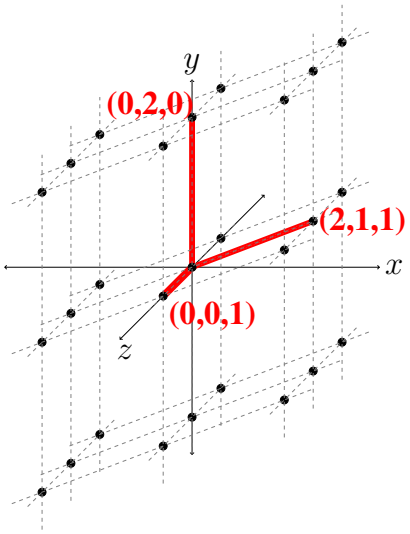


Şekil 2.3. $(1,3)$ ve $(2,1)$ vektörleri ile üretilen kafes

Kriptografide kullanılan kafesler çok yüksek boyutludur. Fakat bu kafesleri çizerek göremiyoruz. Çizerek görebileceğimiz en yüksek boyut 3 olduğundan 3 boyutlu bir kafes örneği verelim.

Örnek 2.7.5. $\{(0,0,1),(0,2,0),(2,1,1)\} \subseteq \mathbb{R}^3$ kümesini taban kabul eden kafes aşağıda verilen kümedir.

$$L\left((0,0,1),(0,2,0),(2,1,1)\right) = \{\alpha_1(0,0,1) + \alpha_2(0,2,0), \alpha_3(2,1,1) \mid \alpha_1,\alpha_2,\alpha_3 \in \mathbb{Z}\}$$



Şekil 2.4. $(0,0,1),(0,2,0)$ ve $(2,1,1)$ vektörleri ile üretilen kafes

2.8 Kodlar

Kod-tabanlı kriptografik algoritmalar, kırılması en zor şifreleme algoritmalarındandır. Bunun en büyük sebeplerinden birisi algoritmalarda kullanılan kodların çok fazla karmaşık matematiksel işlemlerle oluşturulmuş olmalarıdır. Kod-tabanlı kriptografik algoritmaları anlayabilmek için öncelikle kodlama teorisini öğrenip kodların yapı ve özelliklerini anlamak gerekir. Bu kısımda kod kavramı anlatılmıştır ve bu anlatım yapılırken [17] numaralı kaynaktan faydalanılmıştır. Algoritmaların özel olarak kullandığı kod çeşitleri, kod-tabanlı algoritmaların bölümünde anlatılmıştır.

Veriler kanal boyunca iletilirken bazen olumsuz durumlarla karşılaşılır ve veride bozulmalar oluşabilir. Kodlama teorisinin amacı, iletilmiş olan veride hatanın nerede olduğunu saptamak ve eğer mümkünse düzeltmektir. Bunu daha anlaşılır ve kolay yapmak için kodlar matematiksel yapılar halinde inşa edilmiştir.

Tanım 2.8.1. (Kod, Lineer kod) A bir sonlu küme olmak üzere $A^n = A \times A \times \cdots \times A$ (n tane) kümesinin alt kümelerinin her birine kod denir ve genellikle C ile gösterilir. Eğer A kümesi bir sonlu cisim ve C kümesi bir vektör uzayı yapısındaysa C koduna lineer kod denir.

Tanım 2.8.2. (Harf, Kelime) Bir C kodu üretilirken kullanılan A kümesinin elemanlarının her birine harf denir. Dolayısıyla A kümesine de alfabe denir. Oluşturulan C kodunun elemanlarına ise kod kelimesi veya kısaca kelime denir.

Modern bilgisayarlara geçiş sebebiyle kodlama teorisinde kullanılan kodların büyük bir bölümü harf kümesi olarak $\mathbb{F}_2$ cismini kullanır. Bu cisim haricinde kullanılan harf kümeleri genellikle q bir asalın kuvveti olmak üzere $\mathbb{F}_q = \{a_0, a_1, \ldots, a_{q-1}\}$ sonlu cisimleridir.

Örnek 2.8.3. $A = \mathbb{F}_2 = \{0, 1\}$ cismini alalım.

$$C = \{(1,0,1,0), (0,0,1,0), (1,0,0,0), (0,1,1,0)\} \subseteq A^4$$

altkümesi bir koddur fakat lineer kod değildir. Çünkü bir vektör uzayı yapısı belirtmez. Örneğin, $(1,0,0,0) + (0,1,1,0) = (1,1,1,0)$'dır fakat $(1,1,1,0) \notin C$'dir.

Not : Buradan sonra bir karışıklığa yol açmadığı sürece kod kelimelerini yazarken $(a_1, a_2, \ldots, a_n)$ yerine kısaca $a_1 a_2 \ldots a_n$ yazacağız.

Örnek 2.8.4. $A = \mathbb{F}_2 = \{0, 1\}$ cismini alalım.

$$C = \{00000, 10000, 01000, 00100, 11000, 10100, 01100, 111000\} \subseteq A^5$$

altkümesi bir lineer koddur çünkü vektör uzayı yapısındadır. Vektör uzayı yapısında olduğundan bir tabana sahiptir. Bu örnekte C kodunun tabanı $\{10000, 01000, 00100\}$ kümesidir. Yani C kodunu yazarken tüm elemanları uzun uzun yazmak yerine kısaca $C = \langle\{10000, 01000, 00100\}\rangle$ şeklinde yazabiliriz.

2.8.1 Kodların Parametreleri

Tanım 2.8.5. (Uzunluk, Boyut) $C \subseteq A^n$ altkümesi bir lineer kod olmak üzere buradaki n sayısına kodun uzunluğu denir. C lineer kodunun bir vektör uzayı olarak tabanındaki eleman sayısına kodun boyutu denir ve $dim(C) = k$ ile gösterilir. Eğer kodun tabanını bilmiyorsak q alfabenin eleman sayısı ve $|C|$ kodun eleman sayısı olmak üzere k sayısını $k = \log_q |C|$ ile hesaplayabiliriz.

Tanım 2.8.6. (Uzaklık / Hamming Uzaklığı, Minimum Uzaklık) $C \subseteq A^n$ altkümesi bir lineer kod olmak üzere $c, e \in C$ iki kod kelimesini alalım. Bu durumda kod kelimelerini $c = c_1c_2 \dots c_n$ ve $e = e_1e_2 \dots e_n$ şeklinde yazabiliriz. Bu iki kod kelimesinin birbirine olan uzaklığı (Hamming uzaklığı) $dist(c, e)$ ile gösterilir (distance kelimesinden gelir) ve tanımı şu şekildedir:

$$dist(c, e) =\mid \{i \mid c_i \neq e_i\} \mid$$

Yani, iki kod kelimesinin birbirine göre uzaklığı farklı harflerinin sayılarıyla tanımlanır. Minimum uzaklık ise bir kodun içindeki tüm kod ikililerinin uzaklıklarının en küçüğü ile tanımlanır ve d ile gösterilir. Matematiksel olarak şu şekilde yazabiliriz:

$$d = \min \{dist(c_1, c_2) \mid c_1, c_2 \in C \text{ ve } c_1 \neq c_2\}$$

Tanım 2.8.7. (Ağırlık) $C \subseteq A^n$ altkümesi bir lineer kod ve $c \in C$ bir kod kelimesi olmak üzere $dist(c, 0)$ uzaklığına c kod kelimesinin ağırlığı denir ve kısaca $w(c)$ ile gösterilir. (Burada bahsedilen 0 kod kelimesi $0 = 000 \dots 0$'dır)
Yani bir kod kelimesinin ağırlığı, barındırdığı sıfırdan farklı harf sayısı ile tanımlanır.

Tanım 2.8.8. Uzunluğu n, boyutu k, minimum uzaklığı d ve alfabesindeki eleman sayısı q olan bir koda kısaca n, k, d kod denir ve $[n, k, d]_q$ ile gösterilir.

Örnek 2.8.9. $C \subseteq \mathbb{F}_2^4$ altkümesi $\{1001, 0100, 0011\}$ kümesi ile üretilen lineer kod olsun. Bu kodun parametrelerini yazalım.

Uzunluk	:	$n = 4$'tür çünkü kod kelimeleri4 harflidir.
Boyut	:	$k = 3$'tür çünkü $\log_2 8 = 3$.
Minimum Uzaklık	:	$d = 1$'dir çünkü $dist(0100, 0000) = 1$.

$q = 2$ sayısı ile birlikte C kodu bir $[4, 3, 1]_2$ koddur.

2.8.2 Dual Kod

Bir C lineer kodu aynı zamanda bir vektör uzayı yapısına sahip olduğundan üzerinde bir iç çarpım tanımlanabilir. Kodlarda kullanılan iç çarpım çoğunlukla bilinen nokta çarpımıdır.

Tanım 2.8.10. (İç Çarpım / Nokta Çarpımı) $C \subseteq \mathbb{F}_q^n$ altkümesi bir lineer kod olsun ve $c, d \in C$ kod kelimelerini alalım. $c = c_1c_2 \dots c_n$ ile $d = d_1d_2 \dots d_n$ kod kelimelerinin nokta çarpımı $\langle c, d \rangle$ şeklinde gösterilir ve aşağıdaki gibi tanımlanır.

$$\langle c, d \rangle = c_1d_1 + c_2d_2 + \cdots + c_nd_n$$

Tanım 2.8.11. (Dual Kod) $C \subseteq \mathbb{F}_q^n$ bir lineer kod olmak üzere C kodunun duali $C^\perp$ ile gösterilir ve aşağıdaki şekilde tanımlanır.

$$C^\perp = \left\{ c' \in \mathbb{F}_q^n \mid \forall\, c \in C \text{ için } \langle c, c' \rangle = 0 \right\}$$

Pratik Yöntem : Dual kodun tanımından da anlaşılacağı üzere bir C kodunun dualini bulmak için aşağıdaki işlemler sırasıyla yapılabilir:

1. C kodunun taban elemanlarını satır kabul eden G matrisi oluşturulur.

2. $GX = 0$ denklem sistemini sağlayan X sütun matrisleri (vektörleri) bulunur.

3. X vektörlerini kod kelimesi kabul eden kod $C^{\perp}$ dual kodudur.

Örnek 2.8.12. $C = \langle\{101, 011\}\rangle \subseteq \mathbb{F}_2^3$ lineer kodunun dual kodunu bulalım.

1. $G = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$

2. $\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ denklem sisteminin çözümü $\{000, 111\}$ kümesidir.

3. $C^{\perp} = \{000, 111\} = \langle\{111\}\rangle$'dir.

Önerme 2.8.13. $C \subseteq \mathbb{F}_q^n$ bir lineer kod olmak üzere aşağıdakiler sağlanır.

1. $(C^{\perp})^{\perp} = C$

2. $dim(C^{\perp}) = n - dim(C)$

Kanıt. **1.** G matrisi C kodunun taban elemanlarını satır kabul eden matris olsun. Bu durumda $C^{\perp}$ dual kodunun kelimeleri X sütun matrisi (vektörü) olmak üzere $GX = 0$ denklem sisteminin çözümleridir. Yani $C^{\perp}$ kodunun taban elemanları $GX = 0$ denklem sisteminin çözümlerinin oluşturduğu uzayın tabanıdır. Bu taban elemanlarını sütun kabul eden matrise H dersek $GH = 0$ sağlanır. Bu durumda $C^{\perp}$ kodunun taban elemanlarını satır kabul eden matris $(H)^{\top}$ matrisidir. Y sütun matrisi olmak üzere $C^{\perp}$ kodunun dual kodu $(H)^{\top}Y = 0$ denklem sisteminin çözümleridir. O halde $(C^{\perp})^{\perp}$ kodunun taban elemanları Y çözümlerinin oluşturduğu uzayın tabanıdır. Y çözümlerinin taban elemanlarını sütun kabul eden matrise Y' dersek $(H)^{\top}Y' = 0$ denklem sistemi sağlanır. Yani $(C^{\perp})^{\perp}$ kodunun taban elemanlarını satır kabul eden matris $(Y')^{\top}$ matrisidir. Elde ettiklerimizi toparlayacak olursak aşağıdaki sistemi elde ederiz:

$$GX = 0 \Rightarrow (GH)^{\top} = 0 \Rightarrow (H)^{\top}G^{\top} = 0 = (H)^{\top}Y'$$

$G^\top$ matrisinin sütunlarının ürettiği uzayın elemanları X vektörleri, Y' matrisinin sütunlarının ürettiği uzayın elemanları Y vektörleri olduğundan ve $H^\top G^\top = H^\top Y' = 0$ olduğundan X ve Y vektörlerinden oluşan uzaylar aynıdır. Bu da $C = (C^\top)^\top$ olduğu anlamına gelir.

2. $k < n$ olmak üzere $dim(C) = k$ ve C kodunun taban elemanlarını satır kabul eden matris G olsun. O halde X sütun matrisi olmak üzere $GX = 0$ denklem sisteminin çözüm uzayı $n - k$ tane parametreye sahiptir. Bu da çözüm uzayının tabanının $n - k$ elemanlı olduğunu söyler. Yani $C^\perp$ kodunun tabanı $n - k$ elemanlıdır. Dolayısıyla $dim(C^\perp) = n - k$'dır. $dim(C) = k$ olduğundan $dim(C^\perp) = n - dim(C)$ elde edilir. □

Tanım 2.8.14. (Üreteç Matrisi, Eşlik Denetim Matrisi) $C \subseteq \mathbb{F}_q^n$ altkümesi bir lineer kod olmak üzere C'nin taban elemanlarını satır kabul eden matrise C kodunun üreteç matrisi denir ve genellikle G ile gösterilir. $C^\perp$ kodunun taban elemanlarını satır kabul eden matrise ise eşlik denetim matrisi denir. Genellikle H ile gösterilir.

Yukarıdaki örneklerden ve önermeden açıkça gördüğümüz üzere GH çarpımı sıfıra eşittir.

Tanım 2.8.15. (Sendrom) $C \subseteq \mathbb{F}_q$ lineer kodunun eşlik denetim matrisi H olsun. Bir $w \in \mathbb{F}_q$ vektörünün sendromu $wH^\top$ ile tanımlanır. Ayrıca $w \in C$ olması için gerek ve yeter koşul $wH^\top = 0$ olmasıdır.

Tanım 2.8.16. (Singleton Sınırı) $C \subseteq \mathbb{F}_q^n$ altkümesi bir $[n, k, d]_q$ kod olmak üzere $d \leq n - k + 1$ eşitsizliği sağlanır. Minimum uzaklık için geçerli bu üst sınıra Singleton sınırı denir.

Tanım 2.8.17. (MDS Kod) $C \subseteq \mathbb{F}_q^n$ altkümesi bir $[n, k, d]_q$ kod olmak üzere eğer C kodunun parametreleri arasında $d = n - k + 1$ eşitliği sağlanıyorsa bu C koduna maksimum uzaklığa ayrılabilir (maximum distance separable) kod veya kısaca MDS kod denir. MDS kodlar iyi kodlar olarak bilinir.

3. KLASİK KRİPTOGRAFİ

Bu bölümde kriptografinin işleyişini ve amaçlarını daha iyi kavramak amacıyla klasik kriptosistemler anlatılmıştır. Bu anlatım yapılırken [18] numaralı kaynaktan faydalanılmıştır.

Kriptografi kelimesi adını Yunanca'da "gizli" anlamına gelen "kryptos" ve "çizmek, yazmak" anlamına gelen "graphein" kelimelerinin birleşiminden alır. Adından da anlaşılacağı üzere kriptografi bir şifreli yazma işidir. Çalışma alanı oldukça geniş olan kriptografi, bilgisayarların da icadıyla birlikte hemen her yerde kendine bir mesken tutmuş durumdadır. Bizler farkında olmasak da hayatımızın neredeyse her anında bizimle olan birlikteliğini sürdürmektedir. Kriptografinin hayatımızdaki yerine basit örnekler vermek istersek, cep telefonlarımızdaki mesajlaşma ve sosyal medya uygulamalarından başlayarak bankacılık ve bitcoin sektörlerine kadar birçok alan sayabiliriz.

3.1 Temel Kriptografik Kavramlar

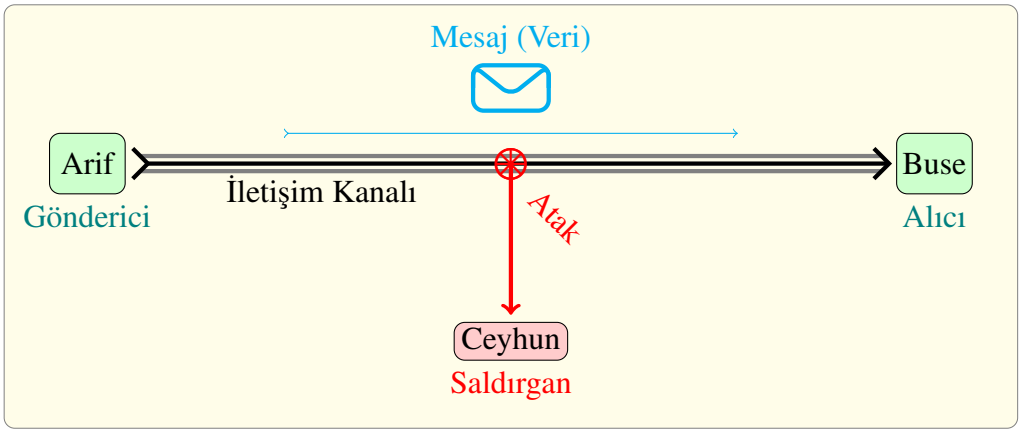
Kriptografik şifreleme algoritmalarının amacını basitçe söylemek istersek, bir veriyi şifrelemek ve gerektiğinde istenilen miktarının istenilen kişi tarafından deşifrelenebilmesini sağlamak, diyebiliriz. Kriptografik algoritmalar bu amacı anahtarlar kullanarak yerine getirirler. Eğer veriyi şifrelerken ve açarken aynı anahtar kullanılıyorsa, kullanılan algoritmaya simetrik şifreleme algoritması, farklı anahtarlar kullanılıyorsa asimetrik şifreleme algoritması diyoruz. Klasik kriptografik algoritmaları bu iki ana başlık altında aktaracağız. Öncesinde temel bir iletişim senaryosu üzerinden kriptografiyi biraz daha açıklayalım. Açıklamaların daha iyi anlaşılması için üretilen senaryoyu tanımlar ve örnekler halinde yazalım.

Tanım 3.1.1. (İletişim Kanalı) Arif ile Buse adında iki kişinin haberleştiği ve Ceyhun ismindeki birinin de bu haberleşmeyi ele geçirmek istediği bir senaryoyu düşünelim. Haberleştikleri platforma iletişim kanalı denir. İletişim kanalına mektup, telefon, bilgisayar gibi örnekler verilebilir.

Tanım 3.1.2. (Atak / Saldırı) Ceyhun'un, Arif ile Buse'nin haberleşirken alışverişte bulundukları verileri ele geçirmek için kullandığı yöntemlerin tümüne atak (saldırı), dolayısıyla

Ceyhun'a da saldırgan denir. Ceyhun'un uygulayabileceği atak türleri aşağıda verilenlerdir:

1. İletilen mesajları kaydetmek
2. Mesajları değiştirmek
3. Mesajlara ekleme yapmak
4. Mesajların tümünü veya bir kısmını silmek



Şekil 3.1. Temel iletişim senaryosu

Tanım 3.1.3. Kriptografi sayesinde Ceyhun'un yapabileceği atak türlerine karşı önlem alabiliyoruz. Buradan yola çıkarak kriptografinin amaçlarını şu şekilde tanımlayabiliriz:

1. **Gizliliği Sağlama :** Arif (gönderici) tarafından gönderilen bir mesajın Buse (alıcı) haricindeki kişiler tarafından anlaşılabilirliği engellemek.
2. **Bütünlüğü Sağlama :** Arif'in gönderdiği bir mesajın Buse'ye değişmeden (bozulmadan) gittiğini garantilemek.
3. **Kimliği Doğrulama :** Buse'ye gelen bir mesajın Arif tarafından gönderildiğini bilmesini sağlamak.
4. **İnkarı Engelleme :** Arif'in Buse'ye gönderdiği bir mesajı kendinin gönderdiğini inkar etmesini önlemek ve Buse'ye gönderilen bir mesajın Buse tarafından okunduğunun inkar edilmesini önlemek.

Bu görevlerin aksi durumlarını açıklayan bir örnek verelim:

Örnek 3.1.4. Arif'in Buse'nin ev sahibi olduğunu ve Buse'ye *"Yeni kira bedelin 5000 TL'dir. 4 gün içerisinde TR1000***01 IBAN numaralı hesabıma gönder."* şeklinde bir mesaj gönderdiği bir senaryoyu düşünelim.

1. Eğer Ceyhun bu mesajı ele geçirip Buse'nin 4 gün içerisinde Arif'e para göndereceği bilgisine ulaşırsa bu durum gizlilik ihlaline girer.

2. Eğer Buse mesajı aldığında Arif'in gönderdiği mesajın orijinal halindekinden farklı bir IBAN numarası görüyorsa veya 5000 yerine farklı bir miktar görüyorsa veya mesaj tamamen farklı bir şekilde görünüyorsa bu durum bütünlük ihlaline girer. Saldırgan mesajı değiştirip kendi banka hesabının IBAN numarasını eklemiş olabilir veya yeni kira bedelini 2000 TL olarak yazıp Buse ile Arif arasında bir tartışma çıkarabilir.

3. Eğer Buse, Arif'ten gelen mesajı aldığında başka birisinin Arif adına kendisine mesaj gönderdiğinden şüphelenir ve mesajı ciddiye almazsa bu durum kimlik doğrulama ihlaline girer.

4. Eğer kirasının artmasını istemeyen Buse, Arif'ten gelen mesajı görmediğini iddia eder ve eski kira bedelini gönderirse bu durum inkar edilememe ihlaline girer.

Tüm bunların yanı sıra iyi bir kriptografik algoritma süreyi de verimli kullanmalıdır. Zamanında yapılamayan görevler bazen işe yaramayabilir. Bunu bir örnekle açıklayalım:

Örnek 3.1.5. Yaşadığınız şehirden uzakta bir üniversitede okuduğunuzu düşünün ve iki gün sonraki bir bayram tatili için memleketinize gideceğinizi, ailenize bir mesaj yoluyla bildirdiğinizi varsayın. Eğer kullandığınız iletişim kanalı kriptografinin dört temel amacını da sağlamak uğruna sizin mesajınızı güvenilir bir şekilde dört gün sonra iletebiliyorsa bu durum sizin pek işinize yaramaz, çünkü siz mesajdan önce eve ulaşmış olursunuz.

Kriptografinin tüm bunları nasıl sağladığına ve algoritmaların işlem süreçlerine geçmeden son temel tanımları verelim.

Tanım 3.1.6. (Açık Metin) Arif'in gönderdiği mesajın şifrelenmeden önceki orijinal haline açık metin denir. Genelde m (message) veya P (Plaintext) ile gösterilir.

Tanım 3.1.7. (Kapalı / Şifreli Metin) Arif'in göndereceği mesajın kriptografik algoritmalar kullanılarak şifrelenmiş haline şifreli metin denir. Genellikle c (ciphertext) veya C ile gösterilir.

Tanım 3.1.8. (Şifreleme) Açık metni kapalı metne çevirme işine şifreleme denir. Bir açıdan, şifreleme işlemi bir fonksiyon gibi düşünülebilir.

Tanım 3.1.9. (Deşifreleme) Kapalı metni açık metne çevirme işine deşifreleme denir. Bir açıdan, deşifreleme işlemi şifreleme fonksiyonunun tersi olarak görülebilir.

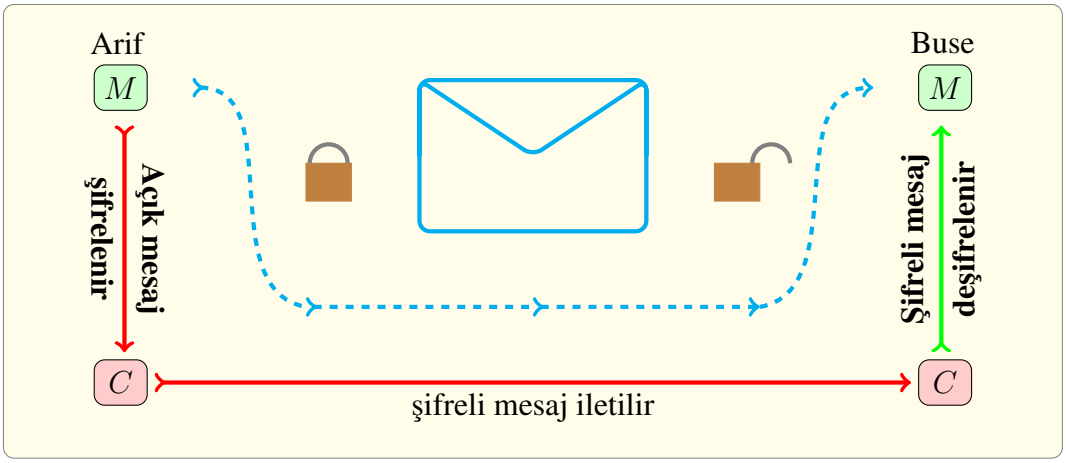
Tanım 3.1.10. (Kriptoanaliz) Bazen şifreli metinlerden, bazen açık metin ve şifreli metin ikililerinden, bazen de algoritmaların işleyişleri gibi araçlar kullanarak şifre kırma yöntemi arayışına kriptoanaliz denir. Bu yöntem arayışı aynı zamanda algoritmaların zayıf yönlerini ortaya çıkarmak için de kullanılabilir ve böylece algoritmalar geliştirilerek yüksek güvenlik seviyelerine sahip yeni şifreleme sistemleri oluşturulabilir.

Tanım 3.1.11. (Anahtar) Bir açık metni şifrelerken ve deşifrelerken kullanılan yapıya anahtar denir. Genellikle k ile gösterilir. Bir kriptografik sistemin güvenliği çoğunlukla anahtar ile sağlanır.

Bir mesajın iletilme aşamaları kısaca mesajın şifrelenmesi, şifreli mesajın gönderilmesi, gönderilen mesajın deşifrelenmesi şeklinde özetlenebilir. Eğer şifreleme ve deşifreleme işlemlerini sırasıyla $\mathcal{E}$ ve $\mathcal{D}$ harfleriyle temsil edip birer fonksiyon şeklinde gösterecek olursak şu şekilde yazabiliriz:

1. $\mathcal{E}(M, k) = C$
2. $\mathcal{D}(C, k) = \mathcal{E}^{-1}(C, k) = M$

$\mathcal{E}$ ve $\mathcal{D}$ fonksiyonlarındaki anahtarlar kullanılan algoritmanın türüne göre değişiklik gösterebilir. İkisi her zaman aynı olmak zorunda değildir.



Şekil 3.2. Mesaj iletim aşamaları

Şimdi aynı anahtar ve farklı anahtar kullanan algoritmalara geçebiliriz. Önce aynı anahtarı kullanan sistemlerden başlayacağız.

3.2 Simetrik Şifreleme Algoritmaları

Biraz önce belirttiğimiz gibi simetrik şifreleme algoritmalarında şifrelemede kullanılan anahtar ile deşifrelemede kullanılan anahtar aynıdır. Bu yüzden bir mesaj şifrelenirken kullanılan anahtar hem mesajı gönderen kişide hem de mesajı alan kişide olmalıdır. Bu kişiler haricinde hiç kimsede anahtar bulunmamalıdır. Bu sebeple, simetrik şifreleme algoritmalarına gizli anahtarlı şifreleme de denilmektedir. Matematiksel anlamda şifreleme algoritması kabul edilebilecek ilk şifreleme Sezar şifrelemedir.

3.2.1 Sezar

Tanım 3.2.1. (Kaydırmalı Şifreleme) Bir alfabedeki harfler sırasıyla sıfırdan başlayarak sayılarla eşleştirilir. Ardından bir sayı belirlenir ve mesajın açık halinde kullanılan her bir harf bu sayı kadar ötelenir ve öteleme sonucunda elde edilen harf kullanılır. Bu şekilde yapılan şifrelemeye kaydırmalı (ötelemeli) şifreleme denir. Bu şifreleme yönteminin özel olarak 3 ötelemeli halini Julius Ceasar (Jül Sezar) kullanmıştır. Böylece bu şifreleme algoritması kriptografi literatürüne Sezar şifreleme olarak geçmiştir.

Şifreleme sistemin daha iyi anlaşılabilmesi için bir örnek verelim. Türkçe alfabe kullanalım.

$0 \to$ A	$5 \to$ E	$10 \to$ I	$15 \to$ M	$20 \to$ R	$25 \to$ Ü
$1 \to$ B	$6 \to$ F	$11 \to$ İ	$16 \to$ N	$21 \to$ S	$26 \to$ V
$2 \to$ C	$7 \to$ G	$12 \to$ J	$17 \to$ O	$22 \to$ Ş	$27 \to$ Y
$3 \to$ Ç	$8 \to$ Ğ	$13 \to$ K	$18 \to$ Ö	$23 \to$ T	$28 \to$ Z
$4 \to$ D	$9 \to$ H	$14 \to$ L	$19 \to$ P	$24 \to$ U	$29{=}0 \to$ A

Şekil 3.3. Türkçe alfabe ve sayı eşleştirmesi

Örnek 3.2.2. GEOMETRİ kelimesini 3 harf öteleyerek şifreleyelim. Bunun anlamı, her bir harfin yerine kendisinden sonra gelen üçüncü harfi kullanacağız.

$$
\begin{array}{lllllllll}
\text{G} & = & 7 & \Rightarrow & \text{3 harf sonrası} & 7+3 & = & 10 & = & \text{I} \\
\text{E} & = & 5 & \Rightarrow & \text{3 harf sonrası} & 5+3 & = & 8 & = & \text{Ğ} \\
\text{O} & = & 17 & \Rightarrow & \text{3 harf sonrası} & 17+3 & = & 20 & = & \text{R} \\
\text{M} & = & 15 & \Rightarrow & \text{3 harf sonrası} & 15+3 & = & 18 & = & \text{Ö} \\
\text{E} & = & 5 & \Rightarrow & \text{3 harf sonrası} & 5+3 & = & 8 & = & \text{Ğ} \\
\text{T} & = & 23 & \Rightarrow & \text{3 harf sonrası} & 23+3 & = & 26 & = & \text{V} \\
\text{R} & = & 20 & \Rightarrow & \text{3 harf sonrası} & 20+3 & = & 23 & = & \text{T} \\
\text{İ} & = & 11 & \Rightarrow & \text{3 harf sonrası} & 11+3 & = & 14 & = & \text{L}
\end{array}
$$

Sonuç olarak, GEOMETRİ kelimesinin 3 harf kaydırarak şifrelenmiş hali IĞRÖĞVTL kelimesidir. Eğer bulduğumuz şifreli metinden tekrar mesajın orijinal halini elde etmek istiyorsak her harfin sayı değerinin 3 eksiğindeki sayıya karşılık gelen harfi yazmamız yeterlidir. Örneğin, şifreli metindeki I harfi 10 sayısına denk gelmektedir. $10-3=7$ olduğundan ve $7=$ G olduğundan I harfinin deşifrelenmiş hali G harfidir. Bunu şifreli metnin her harfi için

yaptığımızda deşifreleme işlemini yapmış oluruz. Gösterelim:

$$
\begin{array}{lllllllll}
\text{I} & = & 10 & \Rightarrow & \text{3 harf öncesi} & 10-3 & = & 7 & = & \text{G} \\
\text{Ğ} & = & 8 & \Rightarrow & \text{3 harf öncesi} & 8-3 & = & 5 & = & \text{E} \\
\text{R} & = & 20 & \Rightarrow & \text{3 harf öncesi} & 20-3 & = & 17 & = & \text{O} \\
\text{Ö} & = & 18 & \Rightarrow & \text{3 harf öncesi} & 18-3 & = & 15 & = & \text{M} \\
\text{Ğ} & = & 8 & \Rightarrow & \text{3 harf öncesi} & 8-3 & = & 5 & = & \text{E} \\
\text{V} & = & 26 & \Rightarrow & \text{3 harf öncesi} & 26-3 & = & 23 & = & \text{T} \\
\text{T} & = & 23 & \Rightarrow & \text{3 harf öncesi} & 23-3 & = & 20 & = & \text{R} \\
\text{L} & = & 14 & \Rightarrow & \text{3 harf öncesi} & 14-3 & = & 11 & = & \text{İ}
\end{array}
$$

Kriptografik tanımları bu örnekte göstermek istersek aşağıdaki şekilde gösterebiliriz:

- Açık metin: GEOMETRİ
- Şifreli metin: IĞRÖĞVTL
- Anahtar: 3
- Şifreleme: 3 harf ileri kaydırma, yani $\mathcal{E}(M,3) = M+3$
- Deşifreleme: 3 harf geri kaydırma, yani $\mathcal{D}(C,3) = C-3$

Tanım 3.2.3. (Anahtar Uzayı) Bir şifreleme algoritmasında kullanılabilecek tüm olası anahtarların bulunduğu kümeye anahtar uzayı denir.

Not : Eğer toplama ve çıkarma işlemleri sırasında işlem sonucu 28'i geçer veya 0'ın altına düşer ise sayıların değeri $(\mathrm{mod}\ 29)$'a göre hesaplanır. Yani kullanılan alfabedeki harf sayısına göre modüler işlem yapılır. Ayrıca biraz önce bahsettiğimiz üzere Jül Sezar 3 harf kaydırarak şifreleme yaptı diyerekten biz de 3 harf kaydırmak zorunda değiliz. Eğer m harfli bir alfabe kullanıyorsak $m-1$ farklı anahtar kullanabiliriz. (0 ve m sayılarını anahtar olarak kullanamayız, çünkü bu sayılarda açık metin ve şifreli metin aynı olur.) Bu durumda Sezar şifreleme sistemi için anahtar uzayının eleman sayısı $m-1$'dir.

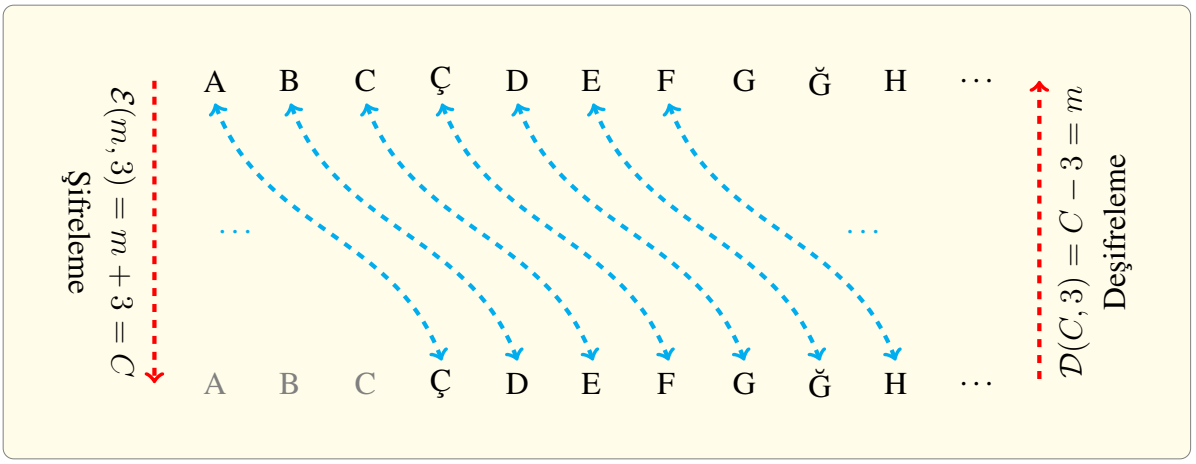
Kaydırmalı şifrelemeyi matematik diliyle yazalım:

Açık mesaj m, şifreli mesaj C, anahtar k ve alfabedeki harf sayısı p olmak üzere kaydırmalı şifrelemeyi aşağıdaki gibi ifade edebiliriz;

$$\mathcal{E}(m,k) = m + k = C$$

$$\mathcal{D}(C,k) = \mathcal{E}^{-1}(C,k) = C - k = m$$

Özetlemek gerekirse, sezar şifrelemeyi aşağıdaki şekilde resmedebiliriz.



Şekil 3.4. Türkçe alfabe ile Sezar şifreleme

Tanım 3.2.4. (Frekans Analizi) Sezar gibi, karmaşık olmayan şifreleme sistemlerinde genelde bir harf daima aynı harfle şifrelenir. Şifrelemede kullanılan dilde genelde çok veya az kullanılan harfler ile bir şifreli metinde çok veya az sayıda geçen harflerin eşleştirmesine dayanan kriptoanaliz yöntemine frekans analizi denir. Frekans analizinde, çok veya az geçen harflerin yanı sıra harf ikililerini, üçlülerini ve hatta o dilde çok kullanılan kelimeleri de kullanabiliriz. Elimizdeki veri miktarı (açık ve şifreli metin miktarı) arttıkça frekans analizi yapmak daha da kolaylaşır.

Örnek 3.2.5. GEOMETRİ kelimesini 3 harf kaydırarak IĞRÖĞVTL şifreli metnini elde etmiştik. Şifreli metne baktığımızda Ğ harfinin diğer harflere göre daha çok bulunduğunu görüyoruz. Türkçe metinlerde en çok geçen harflerin A ve E harfleri olduğu bilgisini göz

önünde bulundurduğumuzda Ğ ile şifrelenen harfin A veya E olduğu tahmininde bulunuyoruz. Sırayla deneyelim. Önce açık metindeki A harfinin Ğ ile şifrelendiği varsayımında bulunalım. Harf tablosunda $\mathrm{A} = 0$ ve $\check{\mathrm{G}} = 8$ olduğundan anahtarın $k = 8 - 0 = 8$ olduğu sonucuna ulaşırız. Yani elimizdeki şifreli metindeki her bir harfi 8 harf geriye götürdüğümüzde açık metni elde etmiş oluruz. Fakat her harfi 8 harf geriye götürdüğümüzde

$$\text{IĞRÖĞVTL} \rightarrow \text{CAJIAÖMF}$$

şeklinde anlamsız bir metne ulaşırız. Bu da bize Ğ harfinin A'nın şifrelenmiş hali olmadığını söyler. Diğer aday olan harfe, yani E'ye geçelim. $\mathrm{E} = 5$ ve $\check{\mathrm{G}} = 8$ olduğundan anahtarın bu defa $k = 8 - 5 = 3$ olduğunu anlıyoruz ve şifreli metindeki her bir harfi 3 harf geriye götürdüğümüzde

$$\text{IĞRÖĞVTL} \rightarrow \text{GEOMETRİ}$$

şeklinde anlamlı bir kelimeye ulaşırız. Bu da bize yaptığımız frekans analizi sonucunda anahtarın $k = 3$ olduğunu söyler.

3.2.2 Yerine Koyma

Sezar şifreleme algoritmasında her harf için kaydırma miktarı aynıdır. Bu sebeple Sezar ile şifrelenmiş bir metnin kırılması kolaydır. Her harfin farklı bir harfle gelişigüzel bir şekilde eşleştirildiği bir şifreleme sistemi Sezar'dan daha güvenilir olur.

Tanım 3.2.6. (Yerine Koyma Şifreleme Sistemi) Belirlenen alfabedeki her harfin yerine başka bir harfin kullanıldığı şifreleme sistemine yerine koyma şifreleme denir.

Türkçe alfabe kullanarak yapılan bir yerine koyma şifrelemede A harfinin yerine kullanmak için 29 farklı harf seçeneğimiz var. Geriye kalan 28 harften birisini de B harfi ile eşleştirebiliriz. Kalanlardan birini de C ile eşleştiririz. Bu şekilde devam ederek Türkçe alfabe ile yerine koyma şifreleme yapmak için $29!$ farklı seçeneğimiz olduğunu kolayca görebiliriz. Yine de bu seçeneklerden biri seçildiği taktirde her harf şifreleme boyunca aynı harf ile şifreleneceğinden bu şifreleme sistemi de frekans analizine yenik düşer.

3.2.3 Afin

Yerine koyma şifreleme sistemiyle bir metni şifrelediğinizde mesajı deşifreleyecek kişi her harfin hangi harfle eşleştirildiğini bilmek zorundadır. Örneğin, Türkçe alfabe ile oluşturulan bir yerine koyma şifrelemede deşifreleme yapmak için 29 farklı bilgiyi bilmemiz gerekiyor. Bunun yerine tüm bu alfabe listesini sadece iki sayı bilerek oluşturabiliriz.

Tanım 3.2.7. (Afin Fonksiyon) Kullanılan alfabedeki eleman sayısı n olsun. $(\alpha, n) = 1$ olacak şekilde $\alpha, \beta \in \mathbb{Z}_n$'iki tam sayı olsun.

$$\begin{array}{rccl} \mathcal{E}: & \mathbb{Z}_n & \to & \mathbb{Z}_n \\ & x & \mapsto & \alpha x + \beta \pmod{n} \end{array}$$

şeklinde oluşturulan fonksiyona bir afin fonksiyon denir.

Adından da anlaşılacağı üzere Afin şifreleme sistemi de seçilen iki tam sayıya karşılık gelen bir afin fonksiyon ile birlikte oluşturulan şifreleme sistemidir. n sayısı asal olduğu sürece α için $n-1$, β için n farklı seçeneğimiz vardır. Dolayısıyla anahtar uzayının eleman sayısı $n(n-1) = n^2 - n$'dir.

Örnek 3.2.8. Türkçe alfabe kullanalım, yani $n = 29$ olsun. 29 sayısı asal olduğundan $\forall\, \alpha \in \mathbb{Z}_{29}\backslash\{0\}$ için $(\alpha, n) = 1$ olur. $\alpha = 3$ ve $\beta = 17$ alalım. Seçtiğimiz bu sayılarla birlikte GEOMETRİ kelimesini $\mathcal{E}(x) = \alpha.x + \beta$ fonksiyonu ile şifreleyelim. Harflerin sayı değerleri için daha önce oluşturduğumuz eşleştirme tablosunu kullanalım.

$$\begin{array}{rcrcrcrcrcrcr}
\text{G} & = & 7 & \Rightarrow & \mathcal{E}(7) & = & 3.7+17 & \equiv & 9 \pmod{29} & \Rightarrow & 9 & = & \text{H} \\
\text{E} & = & 5 & \Rightarrow & \mathcal{E}(5) & = & 3.5+17 & \equiv & 3 \pmod{29} & \Rightarrow & 3 & = & \text{Ç} \\
\text{O} & = & 17 & \Rightarrow & \mathcal{E}(17) & = & 3.17+17 & \equiv & 10 \pmod{29} & \Rightarrow & 10 & = & \text{I} \\
\text{M} & = & 15 & \Rightarrow & \mathcal{E}(15) & = & 3.15+17 & \equiv & 4 \pmod{29} & \Rightarrow & 4 & = & \text{D} \\
\text{E} & = & 5 & \Rightarrow & \mathcal{E}(5) & = & 3.5+17 & \equiv & 3 \pmod{29} & \Rightarrow & 3 & = & \text{Ç} \\
\text{T} & = & 23 & \Rightarrow & \mathcal{E}(23) & = & 3.23+17 & \equiv & 28 \pmod{29} & \Rightarrow & 28 & = & \text{Z} \\
\text{R} & = & 20 & \Rightarrow & \mathcal{E}(20) & = & 3.20+17 & \equiv & 19 \pmod{29} & \Rightarrow & 19 & = & \text{P} \\
\text{İ} & = & 11 & \Rightarrow & \mathcal{E}(11) & = & 3.11+17 & \equiv & 21 \pmod{29} & \Rightarrow & 21 & = & \text{S}
\end{array}$$

Böylelikle GEOMETRİ kelimesinin şifrelenmiş hali HÇIDÇZPS olarak bulunur. Şimdi de şifreli metin ve anahtar elimizde iken nasıl tekrar açık metine döneceğimizi görelim. Örneğin M harfinin şifrelenmiş halini D olarak bulmuşuz. D harfini geri döndürelim.

$$\begin{aligned} \mathcal{E}(x) &= \alpha.x + \beta = 3.x + 17 \equiv 4 \pmod{29} \\ &\Rightarrow 3.x = 4 - 17 = -13 \equiv 16 \pmod{29} \end{aligned}$$

olur ki $3.x \equiv 16 \pmod{29}$ denkliğinde her iki tarafı da 3 sayısının $\pmod{29}$'daki tersi olan 10 sayısı ile çarparsak

$$\begin{aligned} 10.3.x &\equiv 10.16 \pmod{29} \\ \Rightarrow \quad 30.x &\equiv 160 \pmod{29} \\ \Rightarrow \quad 1.x &\equiv 15 \pmod{29} \\ \Rightarrow \quad x &\equiv 15 \pmod{29} \end{aligned}$$

elde edilir ki bu da x harfinin 15 sayısına denk gelen harf olduğunu söyler, yani M harfi. Şifreli metindeki diğer harfler de aynı yolla geri döndürülerek tekrar GEOMETRİ kelimesine ulaşılarak deşifreleme işlemi tamamlanır.

Türkçe alfabedeki harf sayısı 29 olduğundan α sayısını belirlemek zor değildir fakat asal olmayan bir n sayısına göre $(\alpha, n) = 1$ olacak şekilde bir α sayısını bulmak çok da kolay değildir, özellikle n sayısı çok büyükse. Bir n sayısı asal çarpanlarına ayrılmış bir şekilde verildiğinde her ne kadar $(\alpha, n) = 1$ olacak şekildeki α sayılarını tam olarak bulamasak da bu koşulu sağlayan kaç tane α sayısının var olduğunu Euler φ fonksiyonu (Tanım 2.1.17) ile bulabiliyoruz.

3.2.4 Permütasyon

Açık metni şifreli metne çevirirken her harf için farklı bir işlem yapmak zorunda değiliz. Bir metni belirli sayıda harf içeren parçalara ayırıp her bir parçayı kendi içinde bir bütün olarak şifreleyebiliriz.

Tanım 3.2.9. (Blok) Metni, belirli sayıda harf içeren parçalara ayırdığımızda bu parçalardan her birine blok denir. Bloklar içerdiği harf sayısına göre isim alır. Örneğin 4 harf içeren bloklara dörtlü bloklar denir.

Tanım 3.2.10. (Permütasyon / Yer Değiştirme Şifreleme Algoritması) Bir metni n harf içeren bloklara ayırdığımızı düşünelim. Her bir bloktaki harfleri başka bir bloğa bulaşmadan kendi içerisinde karıştırarak yapılan şifrelemeye permütasyon (veya yer değiştirme) şifreleme algoritması denir.

Bunu bir örnekle açıklayalım.

Örnek 3.2.11. Beşli bloklar halinde permütasyon şifreleme ile şifrelenmiş bir açık metnin şifreli hali "RTKÜİNYİENŞBKAEİNYTOAZTGMIIRD?" olsun. Açık metni bulalım.
Daha rahat çıkarım yapabilmek için öncelikle şifreli metni beşli bloklara ayırarak yazalım.

RTKÜİ NYİEN ŞBKAE İNYTO AZTGM IIRD?

Son beşli bloğa baktığımızda sonunda soru işareti olduğunu görüyoruz. Biliyoruz ki soru işareti cümlenin sonunda olur. O halde bloklardaki son harfin yerinin değiştirilmediğini anlıyoruz. ilk bloğa baktığımızda ise RTKÜİ harflerini görüyoruz. Beşinci harfin sabit olduğunu bildiğimizden İ harfine dokunmadan diğer dört harfi kullanarak bir çıkarım yapabiliriz. Örneğin bu dört harften TÜRK kelimesi oluşabiliyor. Yani açık metin şifrelenirken birinci harfin ikinci sıraya, ikinci harfin dördüncü sıraya, üçüncü harfin ilk sıraya, dördüncü harfin üçüncü sıraya ve beşinci harfin kendi yerine yazıldığı çıkarımını yapabiliyoruz. Bunu matematiksel olarak ifade etmek istersek

$$(1,2,3,4,5) \mapsto (3,1,4,2,5) \quad \text{veya} \quad \alpha_1\alpha_2\alpha_3\alpha_4\alpha_5 \mapsto \alpha_3\alpha_1\alpha_4\alpha_2\alpha_5$$

şeklinde yazabiliriz. Bulduğumuz bu permütasyonun tersini şifreli metne uygulayarak açık metne ulaşabiliriz. Bunun için önce permütasyonun tersini bulalım. Şifrelerken ilk harf ikinci sıraya yazıldığına göre deşifrelerken ikinci harfi ilk sıraya getireceğiz, benzer şekilde dördüncü harfi ikinci sıraya, ilk harfi üçüncü sıraya, üçüncü harfi dördüncü sıraya getireceğiz

ve son olarak beşinci harfi sabit bırakacağız. Yani deşifreleme permütasyonumuz

$$(1, 2, 3, 4, 5) \mapsto (2, 4, 1, 3, 5)$$

şeklinde olur. Bu permütasyonu diğer bloklara da uyguladığımızda açık metni

TÜRKİ YENİN BAŞKE NTİYO ZGATM IDIR?

olarak buluruz. Türkçe dil yapısına uygun hale getirmek için açık metnimizi, harf sırasını bozmadan, "TÜRKİYE'NİN BAŞKENTİ YOZGAT MIDIR?" şeklinde düzenleyebiliriz.

Örnekten de anlaşılacağı üzere permütasyon şifreleme sistemini frekans analizi ile kıramayız fakat küçük blok uzunlukları kullanıldığında dile hakim olan birisi için sistemi kırmak hiç zor değildir. Ayrıca, aşikar olarak söylenebilir ki n'li bloklar kullanan bir permütasyon şifreleme algoritmasının anahtar uzayının eleman sayısı $n!$'dir.

3.2.5 Vigenere

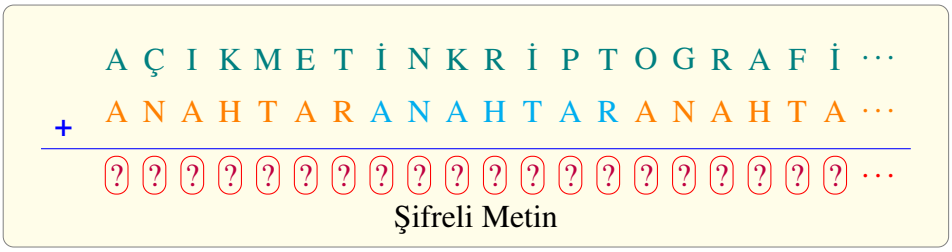
Kaydırmalı (ötelemeli) şifreleme sistemlerinin en zayıf noktası bir harfi her defasında aynı harfle eşleştirmemizden faydalanan frekans analizidir. Frekans analizine karşı koyabilecek bir şifreleme sistemi olan Vigenere, 1553 yılında oluşturulmuştur ve yüzyıllarca kullanılmıştır. Günümüzde her ne kadar modern bilgisayarlar sayesinde kolayca kırılabilse de kendi zamanında "kırılamayan şifreleme" olarak tarihe geçmiştir.

Tanım 3.2.12. (Vigenere Şifreleme Algoritması) Vigenere şifreleme algoritmasını Türkçe alfabe üzerinden adım adım anlatalım:

1. Bir anahtar belirlenir. Bu defa anahtarımız bir sayı değil bir kelimedir.
2. Şifrelenecek olan açık metin düz bir şekilde yazılır.
3. Açık metnin harf sayısına ulaşana kadar anahtar kelime tekrar tekrar yazılarak anahtar uzunluğu artırılır.

4. Uzatılmış olan anahtar kelime açık metnin altına yazılarak iki metin harf harf toplanır. Türkçe alfabe için bu toplama işlemi harfleri sayılarla eşleştirdiğimiz tabloya göre ve $(\mathrm{mod}\ 29)$'a göre yapılır.

5. Bulunan toplamlar tekrar harflere çevirilerek şifreli metin elde edilir.



Şekil 3.5. Vigenere şifreleme diyagramı

Algoritmayı daha iyi anlamak için bir örnek verelim:

Örnek 3.2.13. Açık metnimiz "ADEM MADENE İNMİŞ. ADEM MADENDE BADEM YEMİŞ. MADEM ADEM MADENDE BADEM YEMİŞ, NİYE BİZE GETİRMEMİŞ." olsun ve bu metni k = PASTEL anahtarı ile şifreleyelim. (İstersek noktalama işaretleri için de sayı değeri belirleyip onları da şifrelemeye dahil edebiliriz ama şimdilik biz noktalama işaretlerini göz ardı edeceğiz.)
Anahtar uzunluğumuz 6 olduğundan önce açık metni altılı bloklara ayıralım.

A	D	E	M	M	A	D	E	N	E	İ	N
M	İ	Ş	A	D	E	M	M	A	D	E	N
D	E	B	A	D	E	M	Y	E	M	İ	Ş
M	A	D	E	M	A	D	E	M	M	A	D
E	N	D	E	B	A	D	E	M	Y	E	M
İ	Ş	N	İ	Y	E	B	İ	Z	E	G	E
T	İ	R	M	E	M	İ	Ş				

Her bir bloğun altına anahtarı yazdıktan sonra harflerin sayı değerlerini $(\mathrm{mod}\ 29)$'a göre toplayalım. Çıkan sonuçların tekrar harf değerlerini yazdığımızda şifreli metne ulaşırız. (Bu

işlemleri sadece iki blok için ayrıntılı gösterelim. Diğer blokları kısaca yazalım.)

	A	D	E	M	M	A
	0	4	5	15	15	0
	P	A	S	T	E	L
	19	0	21	23	5	14
+						
	19	4	26	9	20	14
	P	D	V	H	R	L

Blok 1'in Şifrelenmesi

	D	E	N	E	İ	N
	4	5	16	5	11	16
	P	A	S	T	E	L
	19	0	21	23	5	14
+						
	23	5	8	28	16	1
	T	E	Ğ	Z	N	B

Blok 2'nin Şifrelenmesi

Şimdi diğer blokların şifrelenmiş hallerini doğrudan verelim:

$$\begin{aligned}
\text{ADEMMA} &+ \text{PASTEL} = \text{PDVHRL}\\
\text{DENEİN} &+ \text{PASTEL} = \text{TEĞZNB}\\
\text{MİŞADE} &+ \text{PASTEL} = \text{EİLTHP}\\
\text{MMADEN} &+ \text{PASTEL} = \text{EMSYIB}\\
\text{DEBADE} &+ \text{PASTEL} = \text{TEŞTHP}\\
\text{MYEMİŞ} &+ \text{PASTEL} = \text{EYVHNG}\\
\text{MADEMA} &+ \text{PASTEL} = \text{EAÜZRL}\\
\text{DEMMAD} &+ \text{PASTEL} = \text{TEGHEÖ}\\
\text{ENDEBA} &+ \text{PASTEL} = \text{UNÜZFL}\\
\text{DEMYEM} &+ \text{PASTEL} = \text{TEGSIA}\\
\text{İŞNİYE} &+ \text{PASTEL} = \text{BŞĞEÇP}\\
\text{BİZEGE} &+ \text{PASTEL} = \text{RİRZJP}\\
\text{TİRMEM} &+ \text{PASTEL} = \text{KİJHIE}\\
\text{İŞ} &+ \text{PA} = \text{BŞ}
\end{aligned}$$

Sonuç olarak PDVHRL TEĞZNB EİLTHP EMSYIB TEŞTHP EYVHNG EAÜZRL TEGHEÖ UNÜZFL TEGSIA BŞĞEÇP RİRZJP KİJHIE BŞ şifreli metnini elde ederiz.

3.2.5.1 Kasiski Yöntemi

Bu örnekten de anlayacağımız üzere hangi harfin hangi harfle şifreleneceğini doğrudan bilemeyiz. Hatta anahtar uzunluğunu belirlemek için bile çok fazla sayıda işlem yapılması gerekiyor. Bu sebeplerden ötürü, Vigenere şifreleme algoritmasına "kırılamayan şifre" denilmiştir ve yüzyıllar boyunca kullanılmıştır. Fakat 1863 yılında Friedrich Kasiski tarafından Vigenere'yi kırmaya yönelik bir yöntem yayınlanmıştır. Yöntemi anlatmadan önce şunu da belirtmeliyiz ki, bu yöntemin 1846 gibi daha erken bir tarihte Charles Babbage tarafından bağımsız olarak bulunduğu da biliniyor.

Kasiski yöntemi iki aşamadan oluşuyor. İlk aşama anahtar uzunluğunu belirlemek ve ikinci aşama da anahtar uzunluğuna göre metni periyodik olarak ayırıp her bir periyot için ayrı olarak frekans analizi uygulamaktır. Biraz önce Vigenere ile şifreleme yaptığımız örnek üzerinden bu yöntemi uygulayalım.

Örnek 3.2.14. Elimizde sadece

PDVHRLTEĞZNBEİLTHPEMSYIBTEŞTHPEYVHNGEAÜ
ZRLTEGHEÖUNÜZFLTEGSIABŞĞEÇPRİRZJPKİJHIEBŞ

şifreli metni var olsun. Önce anahtar uzunluğunu tahmin edeceğiz. Bu tahmini yaparken şifreli metin içerisinde kendini tekrar eden ikili harfleri bularak başlayacağız.

PDVHRLTEĞZNBEİLTHPEMSYIBTEŞTHPEYVHNGEAÜ
ZRLTEGHEÖUNÜZFLTEGSIABŞĞEÇPRİRZJPKİJHIEBŞ

Gördüğümüz üzere şifreli metin içerisinde kendini tekrar eden TE harf ikilisini bulduk. Şimdi bu harf ikilisinin tekrarlama aralıklarını sayalım. İlk TE ikilisi kendisini ilk kez 18 harf sonra tekrar etmiş ve ikinci tekrarını kendisinden 36 harf sonra yapmış ve son olarak üçüncü tekrarını kendisinden 48 harf sonra yapmıştır. Anahtar uzunluğumuz büyük olasılıkla bu sayıların ortak çarpanlarından biridir.

$$
\begin{aligned}
18 &= 2.3.3 \\
36 &= 2.2.3.3 \\
48 &= 2.2.2.3
\end{aligned}
$$

$18, 36, 48$ sayılarının ortak çarpanlar kümesi $\{1, 2, 3, 6\}$'dır. Bu durumda anahtar uzunluğumuzun 6 olduğu varsayımını yapabiliriz. Çünkü anahtar uzunluğu $1, 2, 3$ gibi sayılar olsaydı şifreleme zayıf olurdu. Tabi ki 6 sayısından bir sonuca varamazsak $1, 2, 3$ sayılarını deneyeceğiz ama şimdilik en iyi adayımız 6'dır. Anahtar uzunluğu tahminimizi yaptığımıza göre şimdilik temsili olarak anahtarımıza $k = a_1a_2a_3a_4a_5a_6$ diyebiliriz.

Şimdi şifreli metni altılı bloklara ayırıp alt alta yazdıktan sonra her bir sütun için ayrı ayrı frekans analizi yapalım.

1. Sütun	2. Sütun	3. Sütun	4. Sütun	5. Sütun	6. Sütun
P	D	V	H	R	L
T	E	Ğ	Z	N	B
E	İ	L	T	H	P
E	M	S	Y	I	B
T	E	Ş	T	H	P
E	Y	V	H	N	G
E	A	Ü	Z	R	L
T	E	G	H	E	Ö
U	N	Ü	Z	F	L
T	E	G	S	I	A
B	Ş	Ğ	E	Ç	P
R	İ	R	Z	J	P
K	İ	J	H	I	E
B	Ş				

Altıncı sütun için frekans analizi yaptığımızda sütun içerisinde en çok L ve P harflerinin bulunduğunu görüyoruz. Türkçede en çok kullanılan harflerin A ve E olduğunu biliyoruz. Bu durumda L ve P'nin A ve E ile eşleştiği varsayımında bulunabiliriz. Diyelim ki L harfi A ile eşleşsin, A'nın sayı değeri 0 ve L'nin sayı değeri 14 olduğundan ve altıncı sütundaki harfler anahtarımızdaki a_6 karakteri ile şifrelendiğinden

$$\mathrm{A} + a_6 = \mathrm{L} \quad \Rightarrow \quad 0 + a_6 = 14$$

eşitliğini yazabiliriz ve buradan $a_6 = 14$ sonucuna ulaşırız. Bulduğumuz bu sonucu frekans analizimizdeki diğer aday harf için denediğimizde

$$\mathrm{E} + a_6 = \mathrm{P} \quad \Rightarrow \quad 5 + 14 = 19$$

eşitliğinin sağlandığını görüyoruz. Bu eşitliğin sağlanması frekans analizimizde büyük ihtimalle doğru yolda olduğumuzu söyler. Buradan yola çıkarak a_6 karakterinin sayı değerinin 14 olduğunu kabul edebiliriz.

Şimdi de beşinci sütuna göre frekans analizi yapalım. Beşinci sütuna baktığımızda en çok bulunan harfin I olduğunu görüyoruz. Yine Türkçede en çok kullanılan harflerin A ve E olduğu bilgisinden yola çıkarak I harfini E ile eşleştiği varsayımını yapalım. Beşinci sütundaki harfler anahtarımızın a_5 karakteri ile şifrelendiğinden ve I harfinin sayı değeri 10 olduğundan

$$\mathrm{E} + a_5 = \mathrm{I} \quad \Rightarrow \quad 5 + a_5 = 10$$

eşitliğini yazabiliriz ki bu da bize $a_5 = 5$ olduğunu söyler.

Son olarak ikinci sütuna göre frekans analizimizi yapıp topladığımız verilerle birlikte metnimizi çözmeye çalışalım. İkinci sütunda en çok bulunan harfin E olduğunu görüyoruz. Bu E harfinin Türkçede en çok kullanılan harflerden biri olan E ile eşleştiği varsayımını yapıp anahtarımızdaki a_2 karakterinin aslında 0 olduğunu düşünebiliriz.

Yaptığımız varsayımlara ve çıkarımlara göre anahtarımızın son hali

$$k = a_1 \ \ 0 \ \ a_3 \ \ a_4 \ \ 10 \ \ 14$$

şeklindedir. Sadece anahtarda bulduğumuz basamakları kullanarak şifreli metnin ilgili yerlerini deşifreleyip yazdığımızda

PDVHMA TEĞZİN EİLTDE EMSYEN TEŞTDE EYVHİŞ EAÜZMA
TEGHAD UNÜZBA TEGSEM BŞĞEYE RİRZGE KİJHEM BŞ

metnini elde ederiz. Elde ettiğimiz metin üzerinden tahminlerde bulunarak ilerleyeceğiz. Cümlenin sonuna baktığımızda M ve Ş harflerini bulduğumuzu fakat arasındaki harfin hala bulunamadığını görüyoruz. Türkçede cümle sonlarında M ve Ş varsa o cümlenin sonunun ya MIŞ ya da MİŞ olduğunu biliyoruz. O halde elde ettiğimiz metindeki M ve Ş harflerinin ortasındaki B harfi ya I ya da İ'dir. İ olduğunu varsayalım. B harfinin sayı değeri 1 ve İ harfinin sayı değeri 11 olduğundan ve blokların ilk harfi anahtardaki a_1 karakteri ile şifrelendiğinden

$$\text{İ} + a_1 = \text{B} \quad \Rightarrow \quad 11 + a_1 = 1$$

eşitliğine ulaşırız ve bu da $a_1 = -10$ anlamına gelir ki $\pmod{29}$'a göre -10 sayısı 19'a eşittir. Yani $a_1 = 19$'dur.

Son duruma göre anahtarımız

$$k = 19 \ \ 0 \ \ a_3 \ \ a_4 \ \ 10 \ \ 14$$

olur. Şifreli metnimizin bu anahtara göre deşifrelenmiş halini yazalım.

ADVHMA DEĞZİN MİLTDE MMSYEN DEŞTDE MYVHİŞ MAÜZMA
DEGHAD ENÜZBA DEGSEM İŞĞEYE BİRZGE TİJHEM İŞ

Bu metinde altı çizili kısma baktığımızda burada gizlenen kelimenin "GETİRMEMİŞ" kelimesi olduğunu anlarız. O halde J harfinin R ile, H harfinin de M ile eşleştiğini elde ederiz.

J harfi TİJHEM bloğunun üçüncü harfi olduğundan anahtarın a_3 karakteri ile, R harfi de aynı bloğun dördüncü harfi olduğundan anahtarın a_4 karakteri ile şifrelenmişlerdir. Harflerin sayı değerleri $\mathrm{R} = 20, \mathrm{J} = 12, \mathrm{M} = 15, \mathrm{H} = 9$ olduğundan

$$\mathrm{R} + a_3 = \mathrm{J} \Rightarrow 20 + a_3 = 12 \Rightarrow a_3 = -8 \equiv 21 \pmod{29}$$

$$\mathrm{M} + a_4 = \mathrm{H} \Rightarrow 15 + a_4 = 9 \Rightarrow a_4 = -6 \equiv 23 \pmod{29}$$

eşitlikleri elde edilir. Sonuç olarak anahtarımız

$$k = 19\ \ 0\ \ 21\ \ 23\ \ 5\ \ 14$$

şeklinde bulunmuştur ve bu anahtara göre deşifreleme yapıldığında

ADEMMA DENEİN MİŞADE MMADEN DEBADE MYEMİŞ MADEMA DEMMAD ENDEBA DEMYEM İŞNİYE BİZEGE TİRMEM İŞ

metni elde edilir. Harf sıralamasını bozmadan Türkçeye uygun bir cümle haline getirdiğimizde

ADEM MADENE İNMİŞ. ADEM MADENDE BADEM YEMİŞ. MADEM ADEM MADENDE BADEM YEMİŞ, NİYE BİZE GETİRMEMİŞ.

tekerlemesini elde ederiz.

3.2.6 Kullan-At

Vigenere şifreleme sisteminin yüzyıllarca kullanılmasının sebebi her harfi başka bir harfle eşlerken anahtar uzunluğuna bağlı olarak her defasında başka bir harfle eşlemesinden kaynaklanan üst seviye sağlamlığıdır. Fakat sonunda Kasiski atağı ile kırılabilmesinin asıl sebebi de yine anahtar uzunluğuna bağlı olarak periyodik bir şifreleme yapmasıdır. Yaptığımız

örnekten de anlaşıldığı üzere, anahtar uzunluğu ne kadar fazla olursa Vigenere ile şifrelenmiş bir metnin kırılma olasılığı da o kadar düşük olur. Buradan çıkarılabilecek sonuç, sonlu uzunlukta olmayan bir anahtar kullanırsak mükemmel sağlamlıkta bir şifreleme yapabileceğimizdir.

Tanım 3.2.15. (Kullan-At Şifreleme) Her şifreleme işleminde farklı bir anahtarın kullanıldığı şifreleme algoritmalarına kullan-at şifreleme algoritması (OTP: One Time Pad) denir. 1917'de Gilbert Vernarm ve Major Joseph tarafından tasarlanmıştır.

Her defasında farklı bir anahtar kullanılması, anahtar gizliliğinin önemini azaltır çünkü şifrelemede kullanılan anahtar bir daha kullanılmadığı için şifreli metinde herhangi bir periyodik duruma rastlanmaz. Bu sebeple anahtarı elde etmek isteyen bir saldırgan tüm olasılıkları denemek zorundadır ki bu hem çok zaman alır hem de bir sonuç bulsa bile doğruluğundan emin olamaz. Sonuç olarak anahtarı elde etse bile, bulduğu anahtar bir sonraki şifreleme için işe yaramayacaktır.

Kullan-at şifreleme mükemmel gizliliği, her defasında farklı bir anahtar kullanarak sağlıyor fakat her defasında şifreleme yaparken kullanılan anahtarı şifreli metinle birlikte karşı tarafa göndermek pek kolay olmuyor. Bunu bir örnekle açıklayalım.

Örnek 3.2.16. "SOĞUK BİR ORTAMDA UYUMAK KABUS GÖRME İHTİMALİNİZİ ARTIRIR" cümlesini Vigenere ve kullan-at şifreleme ile ayrı ayrı şifrelediğimizi düşünelim. Vigenerede kullanılan anahtarın uzunluğu 8 harf olsun. Açık metin 43 harften oluşmaktadır. Dolayısıyla şifreli metin de 43 harf olur. Vigenere ile şifrelediğinizde şifreli metin ve anahtar için toplamda $43 + 8 = 51$ harf iletmeniz gerekirken aynı cümleyi kullan-at şifreleme ile şifrelediğinizde toplam $43 + 43 = 86$ harf göndermeniz gerekecektir.

3.2.7 Playfair

Playfair şifreleme algoritması tarih sahnesinde kendisini I. Dünya Savaşı'nda İngilizler tarafından kullanılarak göstermiştir. Fakat algoritmanın tasarlanışı 1854 yılında Sir Charles Wheatstone tarafından gerçekleştirilmiş ve adını Wheatstone'un bir arkadaşı olan Baron

Playfair'den almıştır. Algoritma, yapısı gereği, kırılması kolay gibi görünmektedir fakat onu kırmaya çalışan her birey çok fazla çaba gerekeceği konusunda hemfikir olurlar.

Playfair algoritması, anahtar olarak bir kelime kullanır. Genelde bu anahtar kelime playfair seçilerek anlatılır. Bu yüzden geleneği bozmamak adına biz de anahtarı "playfair" kelimesi seçelim ve bu seferlik İngiliz alfabesini kullanalım.

A a	B b	C c	D d	E e	F f	G g	H h	I i	J j	K k	L l	M m
0	1	2	3	4	5	6	7	8	9	10	11	12
N n	O o	P p	Q q	R r	S s	T t	U u	V v	W w	X x	Y y	Z z
13	14	15	16	17	18	19	20	21	22	23	24	25

Şekil 3.6. Sıralı İngiliz alfabesi ve sayı eşleştirmesi

İlk adım olarak "playfair" kelimesinin harflerinden başlayıp alfabedeki tüm harfler bir defa kullanılarak tüm harfler 5×5 formunda bir matrise yerleştirilir. (Bilindiği üzere alfabede 26 harf olmasına karşın matris içerisinde 25 boşluk vardır. Bu yüzden i ve j harfi aynı kabul edilip matrise sadece i harfi yazılır.)

p	l	a	y	f
i	r	b	c	d
e	g	h	k	m
n	o	q	s	t
u	v	w	x	z

Şekil 3.7. Playfair algoritması şifreleme matrisi

İkinci adım olarak, açık metindeki boşluklar ve noktalama işaretleri kaldırılıp metin ikili bloklara ayrılır. Her bir blok için şifreleme ayrı yapılır ve her ikili blok için aşağıdaki kriterlere göre şifreleme yapılır. (Metni ikili bloklara ayırınca son blokta bir harf kalıyorsa x harfi eklenerek iki harfe tamamlanır.)

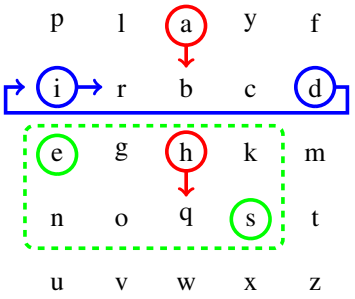
- Eğer iki harf de aynı satırda ise sırasıyla harflerin bir sağındaki harfler seçilerek şifreleme yapılır. Eğer iki harften biri satırın sonuna denk gelmişse satırın birinci harfi ile şifrelenir.
- Eğer iki harf de aynı sütunda ise sırasıyla harflerin bir altındaki harfler seçilerek şifreleme yapılır.
- Eğer iki harfin hem satırı hem sütünu farklı ise ilk harfin bulunduğu satır ve ikinci harfin bulunduğu sütun numarası şifrelemedeki ilk harfin konumunu, ilk harfin bulunduğu sütun ve ikinci harfin bulunduğu satır numarası şifrelemedeki ikinci harfin konumunu belirler.

Anlattıklarımızın daha iyi anlaşılması için bir örnek verelim.

Örnek 3.2.17. "hadise" açık metnini Playfair algoritmasıyla $k =$ playfair anahtarını kullanarak şifreleyelim.
Önce açık metni ikili bloklara ayıralım.

h a d i s e

Algoritmanın matrisini yazalım ve şifrelemeyi matris üzerinde gösterelim.



- h ile a harfleri aynı sütunda olduğu için sırasıyla bir altlarındaki q ve b harflerini seçeriz ve böylelikle "ha" bloğu "qb" şeklinde şifrelenmiş olur.
- d ile i harfleri aynı satırda olduğu için sırasıyla bir adım sağ taraflarındaki i ve r harflerini seçeriz ve böylelikle "di" bloğu "ir şeklinde şifrelenmiş olur. (d harfi satırın sonuna geldiği için, d harfinin bir adım sağındaki harf satırın başındaki i harfi oldu)
- s ve e harfi hem farklı sütunda hem de farklı satırdalar. e harfinin satır-sütun konumu $(3, 1)$ ve s harfinin satır-sütun konumu $(4, 4)$ olduğundan şifreli bloğun ilk harfi 3. satır 4. sütunda yer alan k harfidir ve şifreli bloğun ikinci harfi 4. satır 1. sütunda yer alan n harfidir. Böylelikle "se" bloğu "kn" şeklinde şifrelenmiş olur.

Sonuç olarak Playfair şifreleme algoritmasında k = playfair anahtarı kullanıldığında "hadise" kelimesi "qbirkn" şeklinde şifrelenir.

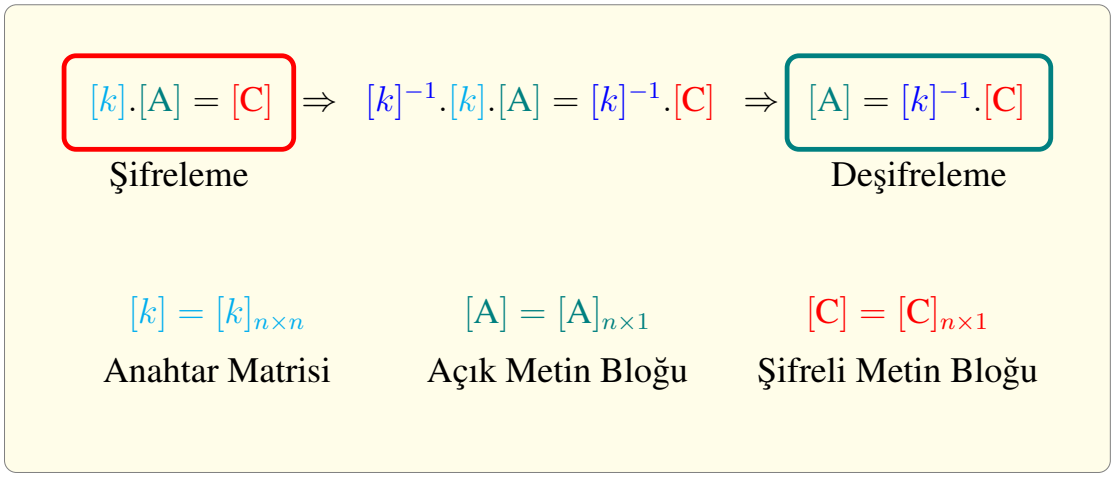
3.2.8 Hill

Tanım 3.2.18. (Blok Şifreleme) Şimdiye kadar gördüğümüz algoritmalardan Sezar, afin, yerine koyma, permütasyon, Vigenere ve kullan-at şifrelemeler harfi harfle eşleyen şifrelemelerdi. Bu şifrelemelerde anahtarın veya açık metnin bir karakteri değiştiğinde şifreli metnin de bir karakteri değişir. Playfair ve birazdan göreceğimiz Hill algoritmasında ise anahtarın bir karakteri değiştiğinde, bu değişim şifreli metnin birden fazla karakterini etkilemektedir. Çalışma prensibi bu şekilde olan algoritmalara blok şifreleme algoritmaları denir.

Hill şifreleme algoritması anahtar olarak matrisleri kullanır. Şimdi, algoritmanın işleyişini, harflerin Türkçe alfabedeki sayı değerlerine göre anlatalım ve kullanılan matrislerin türünü ve özelliklerini, belirtelim.

Önce, bir n sayısı, blok uzunluğu olarak belirlenir ve açık metin belirlenen blok uzunluklarına göre parçalara ayrılır. Her bir blok sütun vektörü şeklinde yazılır. Ardından $(\mathrm{mod}\ 29)$'a göre tersinir olan $n \times n$ tipinde tersinir bir matris seçilir. Son olarak, seçilen

matris ile her bir açık metin bloğu, bilinen matris çarpmasıyla çarpılır ve çıkan sonuç şifreli metin bloğu olarak seçilir. Her bir açık metin bloğu için bu çarpma işlemi ayrı ayrı yapıldıktan sonra, elde edilen şifreli bloklar yan yana dizilerek şifreli metin elde edilir.



Şekil 3.8. Hill şifreleme diyagramı

Algoritmanın işleyişini daha iyi anlamak için bir örnek verelim.

Örnek 3.2.19. "HACETTEPE" kelimesini $k = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 1 & 7 \\ 4 & 2 & 5 \end{bmatrix}$ anahtarını ile şifreleyelim.

Anahtar matrisi 3×3 formunda olduğu için açık metnimizi üçlü bloklara ayırmalıyız.

H A C E T T E P E

Açık metindeki her bir bloğu harflerin sayı değerlerini kullanarak sütun matrisi şeklinde yazalım.

$$\mathrm{H\,A\,C} = \begin{bmatrix} 9 \\ 0 \\ 2 \end{bmatrix} \qquad \mathrm{E\,T\,T} = \begin{bmatrix} 5 \\ 23 \\ 23 \end{bmatrix} \qquad \mathrm{E\,P\,E} = \begin{bmatrix} 5 \\ 19 \\ 5 \end{bmatrix}$$

Her bir sütunu $(\mathrm{mod}\ 29)$'a göre k ile çarpalım ve çarpım sonuçlarını tekrar harflere çevirelim.

$$k.(\text{H A C}) = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 1 & 7 \\ 4 & 2 & 5 \end{bmatrix} \begin{bmatrix} 9 \\ 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 15 \\ 41 \\ 46 \end{bmatrix} \equiv \begin{bmatrix} 15 \\ 12 \\ 17 \end{bmatrix} = \text{M J O}$$

$$k.(\text{E T T}) = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 1 & 7 \\ 4 & 2 & 5 \end{bmatrix} \begin{bmatrix} 5 \\ 23 \\ 23 \end{bmatrix} = \begin{bmatrix} 120 \\ 199 \\ 181 \end{bmatrix} \equiv \begin{bmatrix} 4 \\ 25 \\ 7 \end{bmatrix} = \text{D Ü G}$$

$$k.(\text{E T T}) = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 1 & 7 \\ 4 & 2 & 5 \end{bmatrix} \begin{bmatrix} 5 \\ 19 \\ 5 \end{bmatrix} = \begin{bmatrix} 58 \\ 69 \\ 83 \end{bmatrix} \equiv \begin{bmatrix} 0 \\ 11 \\ 25 \end{bmatrix} = \text{A İ Ü}$$

Çarpımlardan bulunan blokları yan yana getirerek "MJODÜGAİÜ" şifreli metnini elde ederiz.

Anahtar olarak seçilen matrisin tersinir olma koşulunun sebebi, deşifreleme yapabilmektir. Bir açık metin bloğunun sütun matris hali A_1 olsun ve bu bloğu bir k anahtarıyla şifreleyerek $k.A_1 = A_2$ şifreli metin bloğuna ulaşmış olduğumuzu varsayalım. A_2 bloğundan açık metne ulaşabilmek için $k.A_1 = A_2$ eşitliğinde her iki tarafı da k^{-1} matrisi ile çarpmalıyız. Bunu daha iyi anlamak için biraz önceki örnekten elde ettiğimiz şifreli metin üzerinden örnekleyelim.

Örnek 3.2.20. $a_1a_2a_3a_4a_5a_6a_7a_8a_9$ açık metni $k = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 1 & 7 \\ 4 & 2 & 5 \end{bmatrix}$ anahtarı ile şifrelenerek "MJODÜGAİÜ" şifreli metni elde edilmiş olsun. Açık metni bulalım. Deşifreleme için öncelikle şifreli metni üçlü bloklara ayıralım:

M J O D Ü G A İ Ü

Şifreli metnin her bir bloğu için harflerin sayı karşılıklarını kullanarak sütun matrisleri şeklinde yazalım.

$$\text{M J O} = \begin{bmatrix} 15 \\ 12 \\ 17 \end{bmatrix} \qquad \text{D Ü G} = \begin{bmatrix} 4 \\ 25 \\ 7 \end{bmatrix} \qquad \text{A İ Ü} = \begin{bmatrix} 0 \\ 11 \\ 25 \end{bmatrix}$$

Şimdi anahtarımızın $(\mathrm{mod}\ 29)$'a göre tersini yazalım.

$$k = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 1 & 7 \\ 4 & 2 & 5 \end{bmatrix} \Rightarrow k^{-1} = \begin{bmatrix} 16 & 20 & 3 \\ 22 & 6 & 19 \\ 19 & 28 & 25 \end{bmatrix}$$

Şifreli metin bloklarını k^{-1} ile çarparak açık metin bloklarını elde edelim.

$$k^{-1}.(\text{M J O}) = \begin{bmatrix} 16 & 20 & 3 \\ 22 & 6 & 19 \\ 19 & 28 & 25 \end{bmatrix} \begin{bmatrix} 15 \\ 12 \\ 17 \end{bmatrix} = \begin{bmatrix} 531 \\ 725 \\ 1046 \end{bmatrix} \equiv \begin{bmatrix} 9 \\ 0 \\ 2 \end{bmatrix} = \text{H A C}$$

$$k^{-1}.(\text{D Ü G}) = \begin{bmatrix} 16 & 20 & 3 \\ 22 & 6 & 19 \\ 19 & 28 & 25 \end{bmatrix} \begin{bmatrix} 4 \\ 25 \\ 7 \end{bmatrix} = \begin{bmatrix} 585 \\ 371 \\ 951 \end{bmatrix} \equiv \begin{bmatrix} 5 \\ 23 \\ 23 \end{bmatrix} = \text{E T T}$$

$$k^{-1}.(\text{A İ Ü}) = \begin{bmatrix} 16 & 20 & 3 \\ 22 & 6 & 19 \\ 19 & 28 & 25 \end{bmatrix} \begin{bmatrix} 0 \\ 11 \\ 25 \end{bmatrix} = \begin{bmatrix} 295 \\ 541 \\ 933 \end{bmatrix} \equiv \begin{bmatrix} 5 \\ 19 \\ 5 \end{bmatrix} = \text{E P E}$$

Son olarak, çarpımlar sonucunda ulaşılan açık metin blokları yan yana getirilerek "HACETTEPE" açık metni elde edilir.

Önerme 3.2.21. p bir asal olmak üzere p adet harfe sahip bir alfabe kullanarak n'li bloklar ile şifreleme yapan bir hill sistemi için tüm olası anahtarların sayısı aşağıdaki şekilde hesaplanır.

$$p^{n^2}\left(1-\frac{1}{p}\right)\left(1-\frac{1}{p^2}\right)\cdots\left(1-\frac{1}{p^n}\right)$$

Kanıt. Tersinir matris elde etmek için, satırların hiçbirisi diğerlerinin lineer kombinasyonu şeklinde yazılamaz kriteri üzerinden gideceğiz. Kullanılan bloklar n'li olduğundan amaçlanan tersinir matris $n \times n$ formunda olmalıdır. Matrisin ilk satırında n tane konum olduğundan ve her bir konum için alfabedeki harf sayısı kadar, yani p tane seçeneğimiz olduğundan matrisin ilk satırı $p.p.p \ldots p = p^n$ farklı şekilde oluşturulabilir. Fakat, biliyoruz ki, satırın tamamı sıfırlardan oluşamaz, aksi halde matris tersinir olmaz. O halde, matrisi tersinir yapabilecek $p^n - 1$ tane ilk satır seçeneğimiz var.

İkinci satır için koşulumuz, ilk satırın bir tam katı olmaması yönündedir. Yani ilk satırı a_1, ikinci satırı a_2 olarak isimlendirirsek $\forall\ \lambda \in \mathbb{Z}_p$ için $a_2 \neq \lambda.a_1$ koşulunun sağlanmasını istiyoruz. $|\mathbb{Z}_p| = p$ ve $\lambda \in \mathbb{Z}_p$ olduğundan, eşitlik sağlanması için p tane farklı λ seçeneğimiz var. Eşitliğin sağlandığı durumları tüm durumlardan çıkarırsak ikinci satır için toplamda $p^n - p$ tane farklı satır seçeneğimiz olduğunu elde ederiz.

Üçüncü satır için de aynı koşulu kullanacağız. Normalde üçüncü satırda da n tane konum olduğundan p^n farklı satır seçeneğimiz vardır fakat bu seçenekler arasından ilk iki satırın tüm lineer kombinasyonlarını çıkarmalıyız. ilk iki satırın tüm lineer kombinasyonları $\forall\ \lambda, \mu \in \mathbb{Z}_p$ için $\lambda.a_1 + \mu.a_2$ toplamlarıdır ve bun toplamların sayısı (λ, μ) ikililerinin sayısı ile belirlenir. Toplamda p tane λ ve p tane μ olduğundan p^2 tane farklı (λ, μ) ikilisi vardır. Dolayısıyla üçüncü satır için toplam $p^n - p^2$ farklı seçeneğimiz vardır.

Genelliği bozmadan işlemlere devam ederek, $n.$ satır için $p^n - p^{n-1}$ tane seçeneğimiz olduğunu elde ederiz. Sonuç olarak, $\mathbb{Z}_p$ cisminde $n \times n$ formundaki tüm tersinir matrislerin sayısı

$$\left(p^n-1\right)\left(p^n-p\right)\cdots\left(p^n-p^{n-1}\right) = p^{n^2}\left(1-\frac{1}{p}\right)\left(1-\frac{1}{p^2}\right)\cdots\left(1-\frac{1}{p^n}\right)$$

olarak bulunur. □

Sonuç 3.2.22. $p_1, p_2, \ldots, p_k$ sayıları asal olmak üzere $m = p_1 p_2 \ldots p_k$ şeklinde çarpanlara ayrılan bir sayı olsun. m elemanlı bir alfabe kullanılan bir Hill şifreleme sistemi için anahtar uzayının eleman sayısı aşağıdaki şekilde hesaplanır.

$$\left[p_1^{n^2}\left(1-\frac{1}{p_1}\right)\cdots\left(1-\frac{1}{p_1^n}\right)\right]\cdots\left[p_k^{n^2}\left(1-\frac{1}{p_k}\right)\cdots\left(1-\frac{1}{p_k^n}\right)\right]$$

$$= p_1^{n^2}\ldots p_k^{n^2}\left[\left(1-\frac{1}{p_1}\right)\cdots\left(1-\frac{1}{p_1^n}\right)\right]\cdots\left[\left(1-\frac{1}{p_k}\right)\cdots\left(1-\frac{1}{p_k^n}\right)\right]$$

$$= m^{n^2}\left[\left(1-\frac{1}{p_1}\right)\cdots\left(1-\frac{1}{p_1^n}\right)\right]\cdots\left[\left(1-\frac{1}{p_k}\right)\cdots\left(1-\frac{1}{p_k^n}\right)\right]$$

Yani, m sayısının tüm asal çarpanları için ayrı ayrı tersinir matris sayısı hesaplanır ve sonuçlar çarpılır.

3.2.9 DES (Data Encryption Standard)

Şimdiye kadar bahsettiğimiz algoritmalar, bilgisayarların icadına kadar iyi anlamda iş görüyorlardı çünkü insan eliyle çözülmeleri çok kolay değildi. Fakat bilgisayarların icadıyla birlikte çok hızlı işlem yapabilen makineler devreye girince bu algoritmalar raflarda tozlanmaya bırakılmak zorunda kaldılar. Bu evre, insanlığın gizlilik ihtiyacını ortadan kaldırmadığı için modern bilgisayarların bile karşı koyamayacağı şifreleme algoritmaları arayışına girdik.

1973 yılında, ilk adı Ulusal Standartlar Bürosu (NBS: National Bureau of Standards) olan Ulusal Standartlar ve Teknoloji Enstitüsü (NIST: National Institute of Standards and Technology) kamuya açık bir ilan yayınlayarak ulusal standart hale gelecek bir şifreleme algoritması arayışında olduklarını belirttiler. 1974 yılında Uluslararası İş Makineleri (IBM: International Business Machines) şirketi LUCIFER adında bir algoritma önerdi. Ulusal Standartlar Bürosu da bu algoritmaları Ulusal Güvenlik Ajansı'na (NSA: National Security Agency) iletti. LUCIFER algoritması incelendi ve üzerinde bazı değişiklikler yapıldı. Son haline

Veri Şifreleme Standartı (DES: Data Encryption Standard) algoritması ismi verildi ve 1977 yılında NBS, DES'i resmi standart veri şifreleme algoritması yaptı.

DES elektronik ticarette çok yaygınlaşmıştır. Örneğin, iki farklı banka bir veri transferinde bulunacakları sırada öncelikle, ilerleyen bölümlerde anlatacağımız RSA algoritmasıyla bir ortak anahtar belirleyip ardından bu anahtarı ve DES algoritmasını kullanarak veri aktarımını sağlıyorlardı. DES çok güvenliydi ve bu yüksek güvenlik seviyesine rağmen oldukça da hızlıydı.

Bunların yanı sıra DES, çok fazla tartışma konusu haline geldi. Bazıları algoritmada kullanılan anahtarın boyunun kısa olup olmadığını tartışırken bazıları da algoritmanın güvenliğinden şüphe ettiklerini dile getiriyorlardı. Algoritma, yapısı gereği fazla karışık olduğundan içerisine bilerek gizlenmiş bir tuzak hemen fark edilemeyebilir. Bu yüzden algoritmayı kullanıma sunan taraflarca herhangi bir tuzak yerleştirilmiş olabileceğinden şüphe duyuluyordu.

DES için birçok kripto analiz tekniği uygulandı fakat hiçbirinden kayda değer bir sonuç çıkmadı. Yıllarca verimli bir şekilde kullanıldı ama her ne kadar pratikte hala iş görür olsa da teoride, gelişen teknolojiye yenik düştü.

DES algoritması bir veriyi 16 ardışık aşamayla şifreliyor. Yapılan kriptoanalizler gösteriyor ki 15 aşamayla şifrelemeye göre 16 aşamayla şifrelemek çok daha verimlidir. Eğer 17 aşama ile şifrelenirse aynı verim elde edilemiyor. Bu yüzden depolama alanını gereksiz yere işgal etmemek için ve algoritmanın işleyiş süresini çok fazla uzatmamak için 16 aşama ile çalışmasına karar verilmiştir.

Şimdi algoritmanın işleyişini örnek üzerinden bir aşama için gösterelim. (Aşamalardan her birine round denir)

Örnek 3.2.23. DES algoritmasında sonlu cisim olarak $\mathbb{F}_2 = \mathbb{Z}_2 = \{0, 1\}$ kullanılır çünkü bilgisayarlar için tasarlanmıştır. Şifrelenecek olan açık metindeki her bir harf önce ikilik tabana çevrilir. Bu çevirme yapılırken harflerin ve sembollerin ASCII adında halihazırda oluşturulmuş olan bir tablodaki değerleri kullanılır. ASCII tablosunda her karakter 8 bit ile ifade

edilir. Örneğin A harfinin ASCII karşılığı 41'dir ve 41 sayısı ikilik tabandaki karşılığı olan 01000001 ile ifade edilir. DES algoritması 64 bitlik bloklar ile şifreleme yapar ve şifreleme yaparken kullanılan anahtar da 64 bitten oluşur. Bu sebeple, fazlalık olmasın diye açık metnimizi ve anahtarımızı 8 harf, yani 64 bit uzunluğunda seçelim.

Açık metnimiz $m =$ Cebirsel ve anahtarımız $k =$ Geometri olsun. ASCII tablosunda bu harflerin karşılıkları aşağıdaki şekildedir:

C	=	01000011	G	=	01000111
e	=	01100101	e	=	01100101
b	=	01100010	o	=	01101111
i	=	01101001	m	=	01101101
r	=	01110010	e	=	01100101
s	=	01110011	t	=	01110100
e	=	01100101	r	=	01110010
l	=	01101100	i	=	01101001

Açık metnimizi, açık metindeki her harfin bit karşılığını satır kabul eden 8×8 formunda bir matris olarak yazarız.

$$m = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

Açık metin matrisini karıştırmak için bir permütasyon matrisi kullanılır. Bu permütasyon matrisi de halihazırda oluşturulmuş bir matristir ve aşağıdaki şekildedir.

$$P = \begin{bmatrix} 58 & 50 & 42 & 34 & 26 & 18 & 10 & 2 \\ 60 & 52 & 44 & 36 & 28 & 20 & 12 & 4 \\ 62 & 54 & 46 & 38 & 30 & 22 & 14 & 6 \\ 64 & 56 & 48 & 40 & 32 & 24 & 16 & 8 \\ 57 & 49 & 41 & 33 & 25 & 17 & 9 & 1 \\ 59 & 51 & 43 & 35 & 27 & 19 & 11 & 3 \\ 61 & 53 & 45 & 37 & 29 & 21 & 13 & 5 \\ 63 & 55 & 47 & 39 & 31 & 23 & 15 & 7 \end{bmatrix}$$

Karıştırma işlemini yaparken doğrudan matris çarpması veya benzeri bir işlem yapılmaz. Permütasyon matrisinin i. satır ve j. sütununda yazan sayı açık metinden seçilecek bitin sıra numarasıdır. Örneğin, permütasyon matrisinin 1. sütun ve 1. satırında yazan sayı 58 olduğundan m matrisinde en baştan başlayarak sağa doğru sayma yapıldığında 58. sayıyı seçeriz ve karıştırma sonucunda oluşacak matrisin 1. satır ve 1. sütununu bu bit oluşturur. m matrisimizdeki 58. bit kırmızı renkte ve altı çizili şekilde verilen bittir, yani 1 bitidir. Bu şekilde tüm matris karıştırıldığında aşağıdaki matris elde edilir.

$$P(m) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Bu karıştırma işlemini sözel olarak ifade edelim:
m matrisinin 2, 4, 6 ve 8. sütunlarının tersten yazımı $P(m)$ matrisinin 1, 2, 3 ve 4. satırlarını oluşturur. m matrisinin 1, 3, 5 ve 7. sütunlarının tersten yazımı $P(m)$ matrisinin 5, 6, 7 ve 8. satırlarını oluşturur. Karıştırma işlemi sonucunda elde edilen matrisin ilk 4 satırı ile son 4 satırı birbirinden ayrılır ve ilk 4 satır sol (L), son 4 satır sağ (R) olarak isimlendirilir.

$$L = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \qquad R = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Daha sonra anahtarımızdaki harflerin bit karşılıklarını satır kabul eden 8×8 formundaki k matrisi oluşturulur.

$$k = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Oluşturulan k matrisi aşağıda verilen 8×7 formunda olan PC_1 adındaki matris ile karıştırılır ve C ve D olarak isimlendirilen iki matrisin birleşimi şeklinde yazılarak 56 bite düşürülür.

$$PC_1 = \left[\begin{array}{ccccccc} 57 & 49 & 41 & 33 & 25 & 17 & 9 \\ 1 & 58 & 50 & 42 & 34 & 26 & 18 \\ 10 & 2 & 59 & 51 & 43 & 35 & 27 \\ 19 & 11 & 3 & 60 & 52 & 44 & 36 \\ \hline 63 & 55 & 47 & 39 & 31 & 23 & 15 \\ 7 & 62 & 54 & 46 & 38 & 30 & 22 \\ 14 & 6 & 61 & 53 & 45 & 37 & 29 \\ 21 & 13 & 5 & 28 & 20 & 12 & 4 \end{array}\right] \Rightarrow PC_1(k) = \begin{array}{rl} C & = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \\ D & = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Ardından C ve D ayrı ayrı sola kaydırma işlemi uygulanır. Bu kaydırma işleminde tüm bitler bir sola yazılır. Satırın başındaki bitler bir üst satırın en sağına yazılır. Kaydırma işlemi sonucunda oluşan matrisleri C_s ve D_s şeklinde isimlendirelim.

$$C = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \Rightarrow C_s = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \Rightarrow D_s = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Kaydırma işlemi sonucunda oluşan C_s ve D_s matrisleri tekrar birleştirilerek tek bir matris şeklinde yazılır ve aşağıda verilen 8×6 formunda olan PC_2 adındaki matris ile karıştırılarak 48 bite düşürülür.

$$PC_2 = \begin{bmatrix} 14 & 17 & 11 & 24 & 1 & 5 \\ 3 & 28 & 15 & 6 & 21 & 10 \\ 23 & 19 & 12 & 4 & 26 & 8 \\ 16 & 7 & 27 & 20 & 13 & 2 \\ 41 & 52 & 31 & 37 & 47 & 55 \\ 30 & 40 & 51 & 45 & 33 & 48 \\ 44 & 49 & 39 & 56 & 34 & 53 \\ 46 & 42 & 50 & 36 & 29 & 32 \end{bmatrix} \Rightarrow PC_2\left(\left[\frac{C_s}{D_s}\right]\right) = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Şimdi m matrisinin ikinci yarısını temsil eden 4×8 formundaki R matrisi aşağıda verilen E matrisi yardımıyla karıştırarak 8×6 formundaki R_1 matrisi elde edilir. Buradan açıkca

anlaşılacağı üzere 32 bitten oluşan R matrisi bu karıştırma işlemi sonucunda genişleyerek 48 bite tamamlanır.

$$E = \begin{bmatrix} 32 & 1 & 2 & 3 & 4 & 5 \\ 4 & 5 & 6 & 7 & 8 & 9 \\ 8 & 9 & 10 & 11 & 12 & 13 \\ 12 & 13 & 14 & 15 & 16 & 17 \\ 16 & 17 & 18 & 19 & 20 & 21 \\ 20 & 21 & 22 & 23 & 24 & 25 \\ 24 & 25 & 26 & 27 & 28 & 29 \\ 28 & 29 & 30 & 31 & 32 & 1 \end{bmatrix} \Rightarrow E(R) = R_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Bu işlem ile birlikte $PC_2\left(\left[\frac{C_s}{D_s}\right]\right)$ ve $E(R_1)$ matrislerinin her ikisi de 8×6 formuna getirilmiş olur. Şimdi bu iki matrisi toplayalım. Yapacağımız toplama işlemi normalden biraz farklıdır.

Tanım 3.2.24. (XOR Toplama) XOR işlemi ikilik tabanda yapılan bir işlemdir ve aşağıdaki şekilde tanımlanır:

$$0 \oplus 0 = 0 \qquad 0 \oplus 1 = 1 \qquad 1 \oplus 1 = 0 \qquad 1 \oplus 0 = 1$$

Bu işleme göre toplama gerçekleştiğinde aşağıdaki matris elde edilir. Bu matrise R_2 adını verelim.

$$PC_2\left(\left[\frac{C_s}{D_s}\right]\right) + R_1 = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} = R_2$$

Şimdi R_2 matrisinin satırlarını S-Box adı verilen, 4 satır ve 16 sütundan oluşan karıştırma kutularıyla karıştıralım. (S-Box'ların satırları sıfırdan üçe kadar, sütunları ise sıfırdan on beşe kadar olan sayılarla isimlendirilir) Her bir satır farklı bir S-Box ile karıştırılır. Bu sebeple 8 tane S-Box vardır. S-Box'lar her şifreleme için farklı değildir, halihazırda oluşturulmuş kutulardır. Önce S-Box'ları, ardından S-Box'larla karıştırma işleminin nasıl yapıldığını gösterelim.

S-Box 1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S-Box 2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S-Box 3	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S-Box 4	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S-Box 5	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
3	11	8	12	7	1	10	2	13	6	15	0	9	10	4	5	3

S-Box 6	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
2	9	14	155	5	2	8	12	3	7	0	4	10	1	13	11	6
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S-Box 7	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S-Box 8	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Şekil 3.9. DES S-Box

Şimdi karıştırma işlemini nasıl yapacağımızı görelim. R_2 matrisinin ilk satırı 011000 sayısıdır ve bu satırı S-Box 1 ile karıştıracağız. Bu satırın birinci ve altıncı hanelerinden oluşan 00 sayısının onluk tabandaki değeri bize satır numarasını, 2, 3, 4 ve 5. hanelerinden oluşan 1100 sayısının onluk tabandaki değeri sütun numarasını verir. Buradan hareketle S-Box 1'deki o satır ve sütunun gösterdiği sayı R_2 matrisinin birinci satırının onluk tabandaki değeridir. Bu değeri tekrar ikilik tabana çevirdiğimizde R_2'nin ilk satırının karıştırma sonucundaki halini elde ederiz.

İkilik tabanda 00 sayısı onluk tabanda 0'a denktir. İkilik tabandaki 1100 sayısı da onluk tabanda 12 sayısına denktir. Buradan S-Box 1'in 0. satır ve 12. sütunundaki sayı olan 5'e ulaşırız. 5 sayısının ikilik tabandaki değeri 110 sayısıdır. Dört haneye tamamlamak için başına sıfırlar ekleyerek 0110 sayısını elde ederiz. Sonuç olarak R_2'nin ilk satırı olan 011000 sayısının S-Box 1 ile karıştırılmış hali 0110 olarak bulunur. Diğer satırları da benzer şekilde yaparak aşağıdaki değerleri elde ederiz.

1. Satır	:	**0** 1100 **0**	$\rightarrow$	**00** - 1100	$\rightarrow$	0. satır, 12. sütun	$\rightarrow$	5	$\rightarrow$	0110
2. Satır	:	**0** 0101 **0**	$\rightarrow$	**00** - 0101	$\rightarrow$	0. satır, 5. sütun	$\rightarrow$	11	$\rightarrow$	1011
3. Satır	:	**0** 0010 **0**	$\rightarrow$	**00** - 0010	$\rightarrow$	0. satır, 2. sütun	$\rightarrow$	9	$\rightarrow$	1001
4. Satır	:	**0** 1101 **1**	$\rightarrow$	**01** - 1101	$\rightarrow$	1. satır, 13. sütun	$\rightarrow$	10	$\rightarrow$	1010
5. Satır	:	**1** 1000 **1**	$\rightarrow$	**11** - 1000	$\rightarrow$	3. satır, 8. sütun	$\rightarrow$	6	$\rightarrow$	0110
6. Satır	:	**0** 0001 **1**	$\rightarrow$	**01** - 0001	$\rightarrow$	1. satır, 1. sütun	$\rightarrow$	15	$\rightarrow$	1111
7. Satır	:	**1** 1111 **0**	$\rightarrow$	**10** - 1111	$\rightarrow$	2. satır, 15. sütun	$\rightarrow$	2	$\rightarrow$	0010
8. Satır	:	**1** 1100 **1**	$\rightarrow$	**11** - 1100	$\rightarrow$	3. satır, 12. sütun	$\rightarrow$	3	$\rightarrow$	0011

Elde edilen sonuçları satır kabul eden matris yazılır (bu matrise R_3 diyelim) ve bu matris yine halihazırda oluşturulmuş 8×4 formunda bir P_2 permütasyon matrisi ile karıştırılır. Karıştırma sonucunda elde edilen matrise R_4 diyelim.

$$P_2 = \begin{bmatrix} 16 & 7 & 20 & 21 \\ 29 & 12 & 28 & 17 \\ 1 & 15 & 23 & 26 \\ 5 & 18 & 31 & 10 \\ 2 & 8 & 24 & 14 \\ 32 & 27 & 3 & 9 \\ 19 & 13 & 30 & 6 \\ 22 & 11 & 4 & 25 \end{bmatrix} \Rightarrow P_2(R_3) = R_4 = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Daha sonra, R_4 matrisinin 1. satırını 8. sütun olarak, 2. satırını 7. sütun olarak, 3. satırını 6. sütun olarak, ... , 8. satırını 1. sütun olarak kabul eden 4×8 formunda bir matris yazılır. Bu matrise R_5 diyelim.

$$R_4 = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \Rightarrow R_5 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

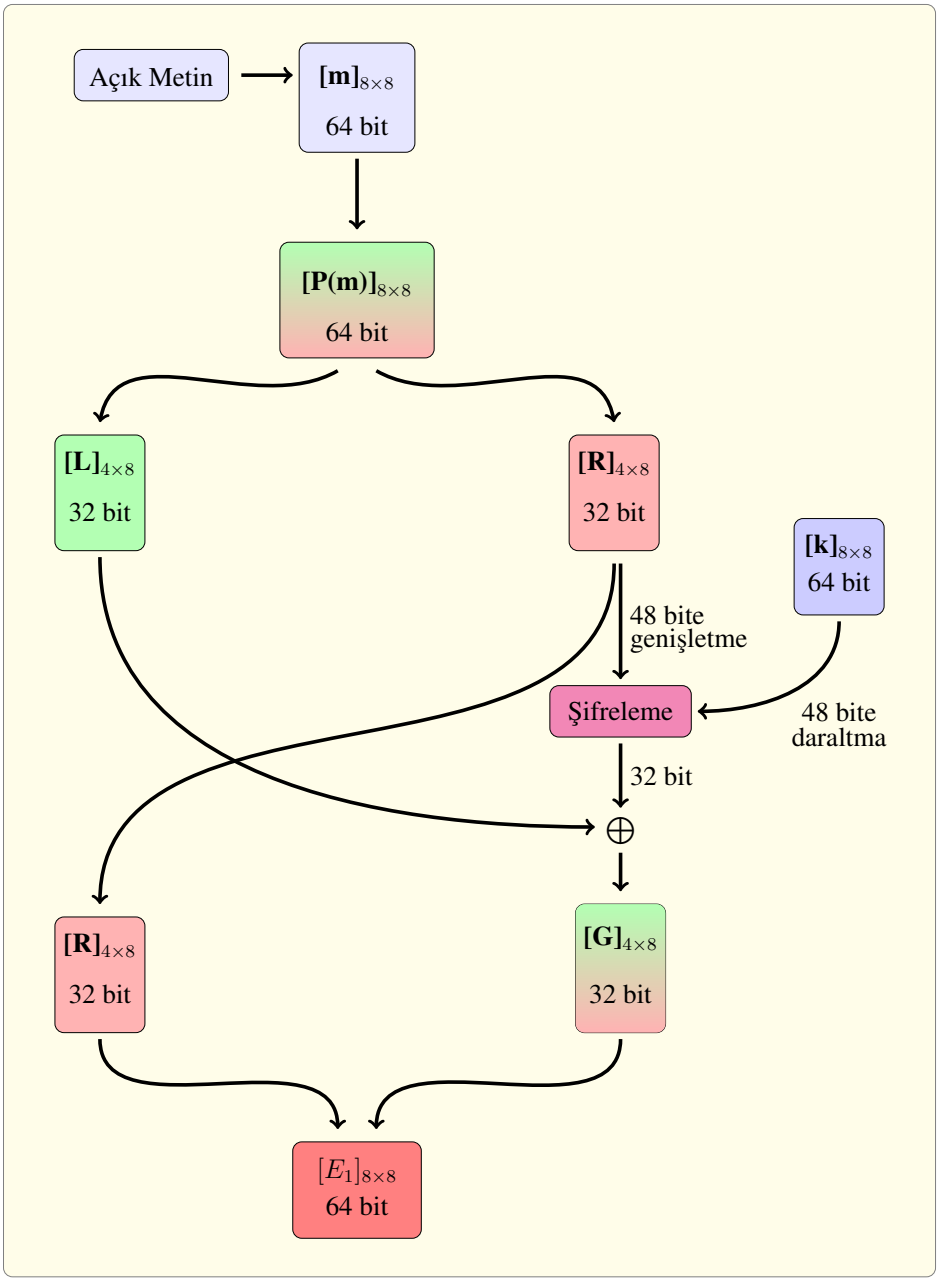
Elde edilen R_5 matrisi ile L matrisi XOR işlemi ile toplanır.

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \oplus \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Toplamadan elde edilen matrise G diyelim. Son olarak R matrisi ile G matrisi birleştirilerek ilk round tamamlanır. Birleştirme sonucunda elde edilen matrise E_1 diyelim.

$$\left[\frac{R}{G}\right] = E_1 = \left[\begin{array}{cccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{array}\right]$$

İşlemler tekrardan en başa alınarak E_1 matrisi üzerinden 15 round daha devam ettirildiğinde DES algoritması ile bir bloğun şifrelenmesi tamamlanmış olur.



Şekil 3.10. DES ile 1 round şifreleme

Tanım 3.2.25. (Feistel Yapı) DES şifreleme algoritmasının en ilgi çekici özelliği deşifreleme işleminin de aynı olmasıdır. Yani DES ile şifrelenmiş bir metni, aynı anahtarı

kullanarak tekrar şifrelersek açık metne ulaşırız. Şifreleme ve deşifreleme işlemlerinin çok benzer olduğu, hatta bazen aynı olduğu yapılara feistel yapılar denir.

DES algoritması 1998 yılında 250.000$ maliyetli bir sistem kullanılarak kırılmıştır. Algoritmayı yeniden düzenleyerek 2DES ve 3DES gibi girişimler olmuştur fakat bütün çabalara rağmen istenilen verim elde edilememiştir. Bu sebeple Ulusal Standartlar ve Teknoloji Enstitüsü (NIST) yeni bir standart algoritma için 2 Ocak 1997 tarihinde bir yarışma çağrısı açmıştır. Yarışmaya katılan birçok algoritma yıllarca kriptoataklarla denendikten ve elemeler yapıldıktan sonra kazanan Rijndael adında bir algoritma olmuştur.

3.2.10 AES (Advanced Encryption Standard)

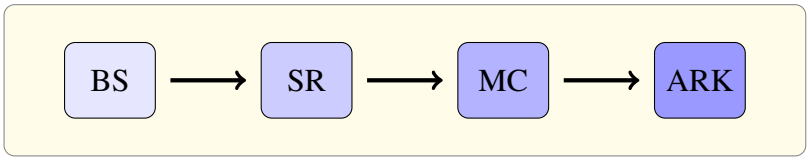
Rijndael algoritmasının standartlaştırılmış versiyonuna Gelişmiş Şifreleme Standartı (AES: Advanced Encryption Standard) algoritması denilmektedir. 26 Kasım 2001 tarihinden itibaren standart algoritma olarak kabul edilmiştir. DES algoritmasının aksine AES, sadece 64 bit uzunluğundaki anahtarla a değil, 128, 192 ve 256 bitlik versiyonlarla çalışabilmektedir. Algoritma 128 bit uzunluğunda anahtarlarla çalışırken 10 round, 192 ile çalışırken 12 round ve 256 bitle çalışırken 14 round şeklinde işlemektedir. Her bir roundda kullanılan anahtarlar orijinal anahtardan üretilmektedir. Her bir round 128 bitle başlayıp 128 bitle biter. Algoritma dört temel adımdan oluşur.

1. **Byte karıştırma dönüşümü (BS: The Byte Sub Transformation):** Diferansiyel ve lineer kriptoanaliz ataklarına karşı dayanıklılık sağlamak için byte'ların lineer olmayan bir şekilde karıştırıldığı adımdır.

2. **Satır kaydırma dönüşümü (SR: The Shift Row Transformation):** Bir bitin birden fazla biti etkilemesini sağlayan kaydırma işlemidir.

3. **Sütun karıştırma dönüşümü (MC: The Mix Column Transformation):** Satır kaydırma adımıyla aynı amacı yerine getiren işlemdir.

4. **Anahtar ekleme (ARK: Add Round Key):** Diğer adımlardan elde edilen sonuç ile anahtarın XOR işlemiyle toplandığı adımdır.



Şekil 3.11. AES 1 round şifreleme adımları

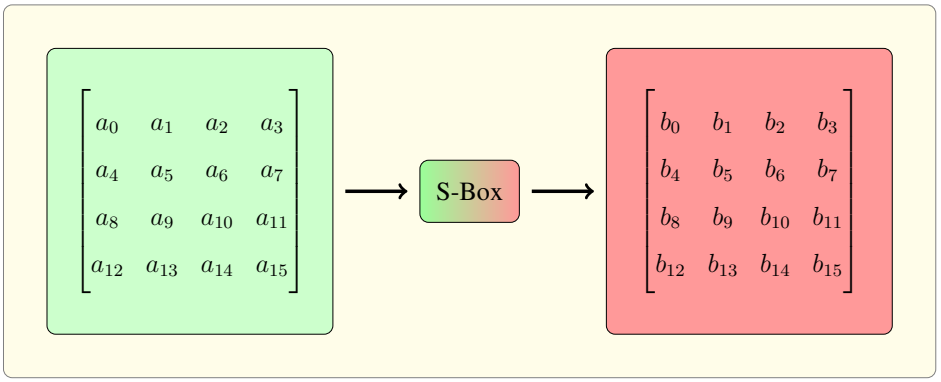
Şimdi algoritmanın işleyişini 128 bitlik versiyon üzerinden görelim.

1 byte 8 bit olduğundan 128 bitlik açık metin 16 byte eder. Açık metni 8 bitlik 16 karakter olacak şekilde ayırırız. İlk byte'tan başlayarak her birini 4×4 formundaki bir matrise soldan sağa doğru yerleştiririz. Byte'lara verdiğimiz isimler $a_0, a_1, a_2, \ldots, a_{15}$ olsun. Bu durumda açık metinden elde edilen matris aşağıdaki gibi olur. (Burada her $i = 0, \ldots, 15$ için a_i'lerin 8 bitten oluştuğunu unutmayalım. Yani matrisin her girdisi aslında 8 karakteri temsil ediyor)

$$m = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 & a_7 \\ a_8 & a_9 & a_{10} & a_{11} \\ a_{12} & a_{13} & a_{14} & a_{15} \end{bmatrix}$$

Şimdi byte karıştırma dönüşümünün nasıl işlediğini açıklayalım. AES algoritmasında da tıpkı DES'te olduğu gibi byte'ları karıştırmak için kullanılan hali hazırda oluşturulmuş S-Box vardır. DES'ten farklı olarak AES'teki S-Box 16 satır ve 16 sütundan oluşur ve 8 tane yerine sadece 1 tane S-Box vardır. Satır numaraları ve sütun numaraları sıfırdan başlayıp 15'te biter. Her bir byte için satır ve sütun numaralarını belirlerken byte'ların ilk dört ve son dört bitine bakarız. İlk dört bitin onluk tabandaki değeri satır numarasını, son dört bitin onluk tabandaki değeri ise sütun numarasını verir. Örneğin, $a_0 = 00110100$ ise ilk dört biti olan 0011 sayısının onluk tabandaki değeri 3'tür ve son dört biti olan 0100 sayısının onluk tabandaki değeri 4'tür. Yani a_0 byte'ının S-Box'taki karşılığı 3 numaralı satır ve dört numaralı sütundaki sayıdır. Bu sayı bulunduktan sonra tekrar ikilik tabandaki değerine çevrilir. Çevirme

sonucunda tekrar 8 bitlik bir sayı elde edilir çünkü S-Box'taki en büyük sayı 255'tir. Bulunan 8 bitlik sayıyı olduğu gibi matrise yazmak yerine tek karakterden oluşan 1 byte halinde yazarız. Bu byte'ı b_0 olarak isimlendirirsek b_0 karakterini matriste yazacağımız konum a_0 karakterinin konumudur.



Şekil 3.12. AES byte karıştırma adımı

S-Box	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	99	124	119	123	242	107	111	197	48	1	103	43	254	215	171	118
1	202	130	201	125	250	89	71	240	173	212	162	175	156	164	114	192
2	183	253	147	38	54	63	247	204	52	165	229	241	113	216	49	21
3	4	199	35	195	24	150	5	154	7	18	128	226	235	39	178	117
4	9	131	44	26	27	110	90	160	82	59	214	179	41	227	47	132
5	83	209	0	237	32	252	177	91	106	203	190	57	74	76	88	207
6	208	239	170	251	67	77	51	133	69	249	2	127	80	60	159	168
7	81	163	64	143	146	157	56	245	188	182	218	33	16	255	243	210
8	205	12	19	236	95	151	68	23	196	167	126	61	100	93	25	115
9	96	129	79	220	34	42	144	136	70	238	184	20	222	94	11	219
10	224	50	58	10	73	6	36	92	194	211	172	98	145	149	228	121
11	231	200	55	109	141	213	78	169	108	86	244	234	101	122	174	8
12	186	120	37	46	28	166	180	198	232	221	116	31	75	189	139	138
13	112	62	181	102	72	3	246	14	97	53	87	185	134	193	29	158
14	225	248	152	17	105	217	142	148	155	30	135	233	206	85	40	223
15	140	161	137	13	191	230	66	104	65	153	45	15	176	84	187	22

Şekil 3.13. AES S-Box

Byte karıştırma işlemi sonucunda elde edilem matrise $S(m)$ diyelim ve satır kaydırma işleminin nasıl yapıldığını görelim. $S(m)$ matrisinin ilk satırındaki byte'lar aynen kalır, ikinci satırdaki her bir byte bir sola kaydırılır, üçüncü satırdakiler iki birim sola, dördüncü satırdakiler de üç birim sola kaydırılır. Satır kaydırma sonucunda elde edilen matris aşağıdaki gibidir. (Bu matrise $R(m)$ diyelim.)

$$R(m) = \begin{bmatrix} c_0 & c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 & c_7 \\ c_8 & c_9 & c_{10} & c_{11} \\ c_{12} & c_{13} & c_{14} & c_{15} \end{bmatrix} = \begin{bmatrix} b_0 & b_1 & b_2 & b_3 \\ b_5 & b_6 & b_7 & b_4 \\ b_{10} & b_{11} & b_8 & b_9 \\ b_{15} & b_{12} & b_{13} & b_{14} \end{bmatrix}$$

Bu kaydırma işlemi sonucunda elde edilen $R(m)$ matrisinin sütunlarını kaydırma dönüşümü aşağıda verilen ve girdileri 8 bitlik olan matris ile çarparak yapılır. (Bu çarpma işlemi sonucunda oluşan matrise $C(m)$ diyelim.)

$$\begin{bmatrix} 00000010 & 00000011 & 00000001 & 00000001 \\ 00000001 & 00000010 & 00000011 & 00000001 \\ 00000001 & 00000001 & 00000010 & 00000011 \\ 00000011 & 00000001 & 00000001 & 00000010 \end{bmatrix} \cdot \begin{bmatrix} c_0 & c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 & c_7 \\ c_8 & c_9 & c_{10} & c_{11} \\ c_{12} & c_{13} & c_{14} & c_{15} \end{bmatrix}$$

$$= \begin{bmatrix} d_0 & d_1 & d_2 & d_3 \\ d_4 & d_5 & d_6 & d_7 \\ d_8 & d_9 & d_{10} & d_{11} \\ d_{12} & d_{13} & d_{14} & d_{15} \end{bmatrix} = C(m)$$

Buradaki çarpma işlemi bilinen çarpma değildir. Katsayılarını $\mathbb{F}_2$'den alan tüm polinomların $x^8 + x^4 + x^3 + x + 1$ indirgenemez polinomuyla bölümünden kalanların kümesi olan $\mathbb{F}_{2^8}$ cismindeki çarpmadır. (Bu cismi başka bir indirgenemez polinomla da üretebiliriz fakat

Rijndael algoritması için bu polinom seçilmiştir.)

$$\mathbb{F}_{2^8} = \left\{ a_7x^7 + a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0 \mid \forall\, i = 0, \ldots, 7; a_i \in \mathbb{F}_2 \right\}$$

Her bir 8 bitlik sayının bu cisimdeki karşılığı olan polinom her bir biti katsayı kabul eden polinomdur. Örneğin,10010101 için aşağıdaki eşitlik yazılır.

$$\begin{aligned} 10010101 &= 1.x^7 + 0.x^6 + 0.x^5 + 1.x^4 + 0.x^3 + 1.x^2 + 0.x + 1 \\ &= x^7 + x^4 + x^2 + 1 \end{aligned}$$

Her çarpmanın sonucu yine 8 bittir ve bu sonuçlar kullanılarak $C(m)$ matrisi elde edilir.

Anahtar ekleme adımında ise $C(m)$ matrisi ile 128 bitlik (16 byte) anahtarın matrisi XOR işlemi ile toplanır. (Anahtarın matrisine K diyelim ve girdileri k_i şeklinde olsun.)

$$\begin{bmatrix} d_0 & d_1 & d_2 & d_3 \\ d_4 & d_5 & d_6 & d_7 \\ d_8 & d_9 & d_{10} & d_{11} \\ d_{12} & d_{13} & d_{14} & d_{15} \end{bmatrix} \bigoplus \begin{bmatrix} k_0 & k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 & k_7 \\ k_8 & k_9 & k_{10} & k_{11} \\ k_{12} & k_{13} & k_{14} & k_{15} \end{bmatrix} = \begin{bmatrix} e_0 & e_1 & e_2 & e_3 \\ e_4 & e_5 & e_6 & e_7 \\ e_8 & e_9 & e_{10} & e_{11} \\ e_{12} & e_{13} & e_{14} & e_{15} \end{bmatrix} = E_1$$

Böylelikle E_1 şifreli metni elde edilerek 1 roundluk şifreleme işlemi biter ve benzer şekilde 9 round daha devam edilerek 128 bitlik bir açık metin şifrelenmiş olur.

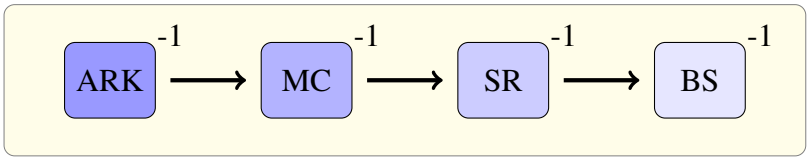
Şimdi de her bir roundda kullanılan anahtarın orijinal anahtardan nasıl elde edildiğini görelim. Anahtar matrisine fazladan 40 sütun eklenerek 44 sütunluk bir matris elde edilir. (Bu matrisin sütun numaraları 0'dan başlatılır, dolayısıyla 43'te biter.) En baştaki ilk 4 sütun başlangıç anahtarı ve ardından gelen her 4 sütunluk parçanın her biri birer roundun anahtarı olarak kullanılır. Bu 40 sütunu ekleme işlemi ise aşağıdaki şekilde yapılır.

i sayısı sütun numarası olmak üzere $W(i)$ gösterimi matrisin i. sütununu temsil etsin. $\forall\, i \geq 4$ için eğer i sayısı 4'ün tam katı değilse i. sütun $W(i) = W(i-4) \oplus W(i-1)$ ile elde

edilir. Eğer i sayısı 4'ün tam katı değilse $W(i) = W(i-4) \oplus T(W(i-1))$ ile elde edilir. Burada verilen T dönüşümü $W(i-1)$'den aşağıdaki şekilde elde edilir.

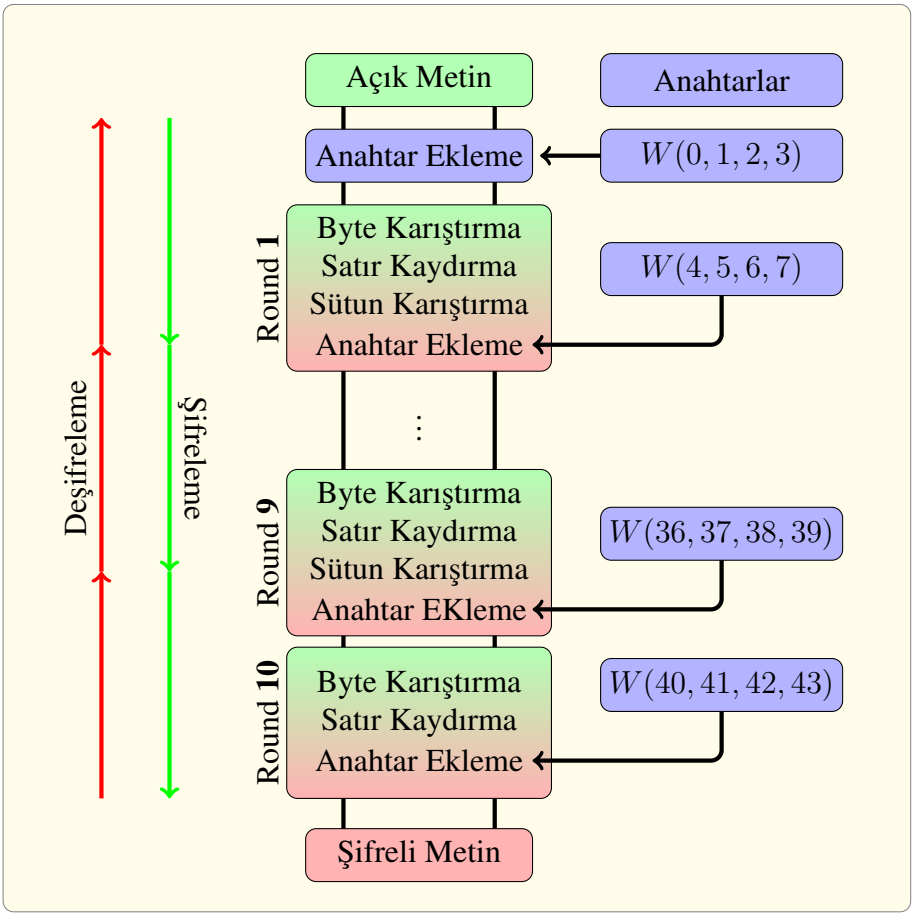
$W(i-1) = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$ olsun. Bu sütun matrisinin girdilerini bir satır yukarı kaydırarak $\begin{bmatrix} b \\ c \\ d \\ a \end{bmatrix}$ matrisini elde edelim. Ardından her bir a, b, c, d byte'ını S-Box'taki karşılıklarıyla değiştirelim. Bu değiştirme sonucunda elde edilen yeni girdilere e, f, g, h diyelim. Daha sonra $r(i) = 00000010^{\frac{i-4}{4}}$ şeklinde round sabitini hesaplayalım. Tüm bunlarla birlikte $T(W(i-1)) = \begin{bmatrix} e \oplus r(i) \\ f \\ g \\ h \end{bmatrix}$ olarak yazılır. Böylelikle 11 adet anahtar elde edilir.

AES algoritmasıyla şifrelenmiş bir metni deşifrelemek için DES'te olduğu gibi tekrar aynı şekilde şifrelemek yanlış sonuç verir. AES ile deşifrele yaparken şifreleme adımlarını tersten uygulamak gerekir. Ayrıca uygulanan adımların sırasını tersten işletmek yetmez, adımları uygularken yapılan işlemlerin de tersini yapmak gerekir. Örneğin şifrelerken sola kaydırma işlemi yapıldıysa, deşifrelerken sağa kaydırma işlemi yapılmalıdır. (Son roundda sütun karıştırma işlemi yapılmaz.)



Şekil 3.14. AES ile 1 round deşifreleme adımları

AES algoritmasını aşağıdaki şekil ile özetleyebiliriz.



Şekil 3.15. AES ile şifreleme ve deşifreleme

3.3 Kerckhoffs Prensibi

Algoritmaların yapılarından, bir algoritma ne kadar karmaşık yapıda olursa ve açık metni ne kadar iyi şifrelerse o kadar güvenilir olacağını anlayabiliriz fakat burada kilit rol anahtardadır. Bir algoritmanın yapısı ne olursa olsun, anahtarı biliyorsak deşifrelemeyi kolayca yapabiliriz. Bir şifreleme algoritmasından beklenilen de tam olarak budur. Yani güvenliği sağlayan kavram algoritmanın yapısı değil, anahtardır. Ancak, anahtarı bildiğimiz halde algoritmayı bilmiyorsak, anahtar işimize yaramayacaktır. Bu durum kriptografide büyük bir tartışmayı ortaya çıkarmıştır: "Algoritmanın yapısı gizli mi

olmalı yoksa açık mı?" Bu tartışmaya son noktayı, Hollandalı kriptograf Auguste Kerckhoffs koymuştur. Kerckhoffs'a göre bir şifreleme algoritması aşağıdaki şartları sağlamalıdır.

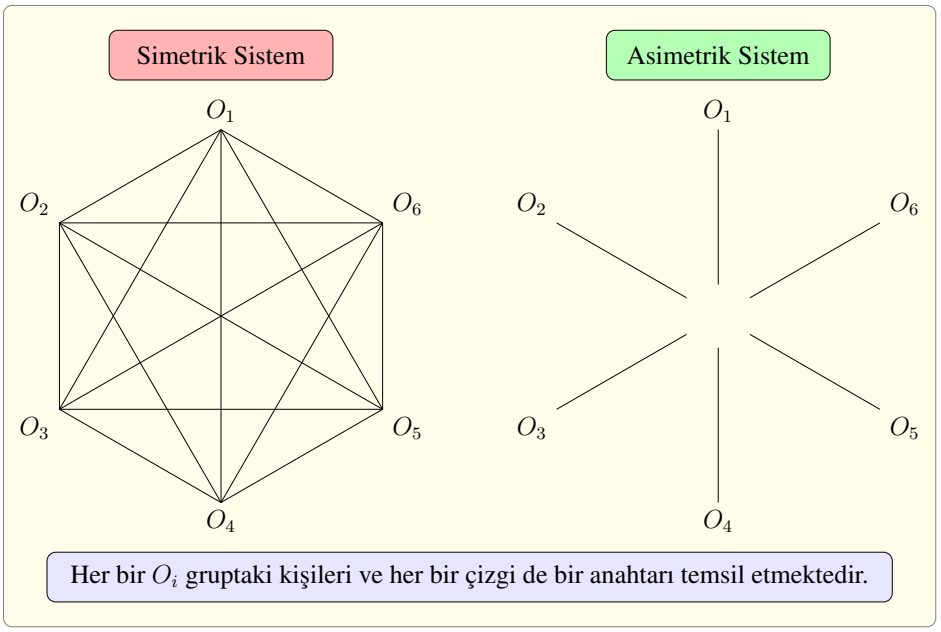
1. Algoritmanın en az bir kısmı matematiksel olarak geri çevrilemez olmalı.
2. Algoritmanın gizli tutulması gerekmemeli, anahtarın gizli olması güvenliği sağlamalı.
3. Taraflar yan yana gelmeden anahtar alışverişi yapabilmeli.
4. Algoritmalar anlaşılır olmalı.
5. Telgraf iletişimine uyarlanabilir olmalı.

Bu ilkelerden de anlayabileceğimiz üzere, algoritma değil, anahtar gizli tutulmalıdır.

3.4 Asimetrik Şifreleme Algoritmaları

Eğer mesajlaşma veya herhangi bir veri gönderimi iki kişi (veya grup) arasındaysa şimdiye kadar gördüğümüz şifreleme algoritmaları gayet iyi iş görürler. Fakat ikiden fazla kişiden oluşan bir ortam düşünelim. Bu ortamdan herhangi iki kişinin mesajlaşmasını diğerlerinin görmemesi için mesajlaşan iki kişiye özel olarak bir ortak anahtar belirlenmelidir. Ortamdaki her ikili grup için bunun yapılması gerekir. Yani iki kişiden fazla birey bulunduran bir grupta mesajlaşmanın güvenli olabilmesi için birçok anahtar bulunmalıdır. Örneğin n kişiden oluşan bir grup için belirlenmesi gereken anahtar sayısı $C(n,2) = \frac{n.(n-1)}{2}$'dir. Bu da kişi sayısı çoğaldıkça anahtar sayısının da katlanarak çoğalacağını söyler. Çok fazla anahtar ise mesajlaşma sisteminde çok fazla depolama alanı kaplar. Eğer ortamdaki herkesin kendine ait anahtarı olursa ve mesajları şifrelerken ve açarken herkes kendi anahtarlarıyla işlem yaparsa n kişilik bir ortamda sadece n adet anahtar verisinin olması yeterli olur ki bu da depolama alanından kazanç ve gereksiz işlem yükünün kaybettireceği zamandan tasarruf sağlar.

Tanım 3.4.1. (Asimetrik Şifreleme Sistemi) Gönderici ve alıcının anahtarlarının farklı olup, mesajları şifrelerken ve deşifrelerken farklı anahtarlar kullanıldığı, dolayısıyla kimsenin bir diğerinin anahtarını bilmediği şifreleme sistemlerine asimetrik şifreleme (açık anahtarlı) sistemleri denir.



Şekil 3.16. 6 kişi için simetrik ve asimetrik sistemlerdeki anahtar sayısı farkı

Tanım 3.4.2. (Açık Anahtar, Gizli Anahtar) Asimetrik şifreleme sistemlerinde her bireyin, sistemdeki herkes tarafından görünebilen anahtarlarına açık anahtarlar (public keys), sadece bireyin kendisinin görebildiği anahtarlarına gizli anahtarlar (private keys) denir.

Veri alışverişi sırasında açık anahtarı rahatlıkla paylaşabiliriz fakat gizli anahtarımız bir başkası tarafından öğrenilirse iletişim asla güvenli olmayacaktır, bizim adımıza veri gönderilebilir veya gönderdiğimiz bir veri değiştirilebilir. Yani kriptografinin temel amaçları sekteye uğrar.

3.4.1 Diffie-Hellman Anahtar Değişimi

Bir gönderici, veriyi bir anahtar kullanarak şifreledikten sonra alıcıya gönderdiğinde, alıcı kullanılan anahtarı bilmeden veriyi deşifreleyemez. Bu sebeple ikiden fazla kişi veya grubun bulunduğu bir ortamda iki kişinin ortak bir anahtar belirlemesi gerekir ve belirlenen ortak anahtar sistemdeki diğer kişiler tarafından bilinmemelidir. Bu sorunu matematikte çözülmesi çok zor olan ve zaman alan bir yöntemle çözüyoruz. Diffie-Hellman anahtar değişimi adı

verilen bu yöntem 1976 yılında Whitfield Diffie ve Martin Hellman adında Amerikalı şifrebilimciler tarafından "Kriptografide Yeni Yönelimler" (New Directions in Cryptography) makalesinde yayımlanmıştır. Her ne kadar algoritmanın adı anahtar değişimi olarak geçse de aslında sistemde bir değişimden ziyade ortak bir anahtar belirleme amaçlanmıştır. Bu sebeple kriptografi alanında çalışan birçok bilim insanı bu sistemi "Diffie-Hellman Anahtar Anlaşması" olarak isimlendirmektedir. Önce sistemin işleyişini anlatıp ardından bir örnek verelim.

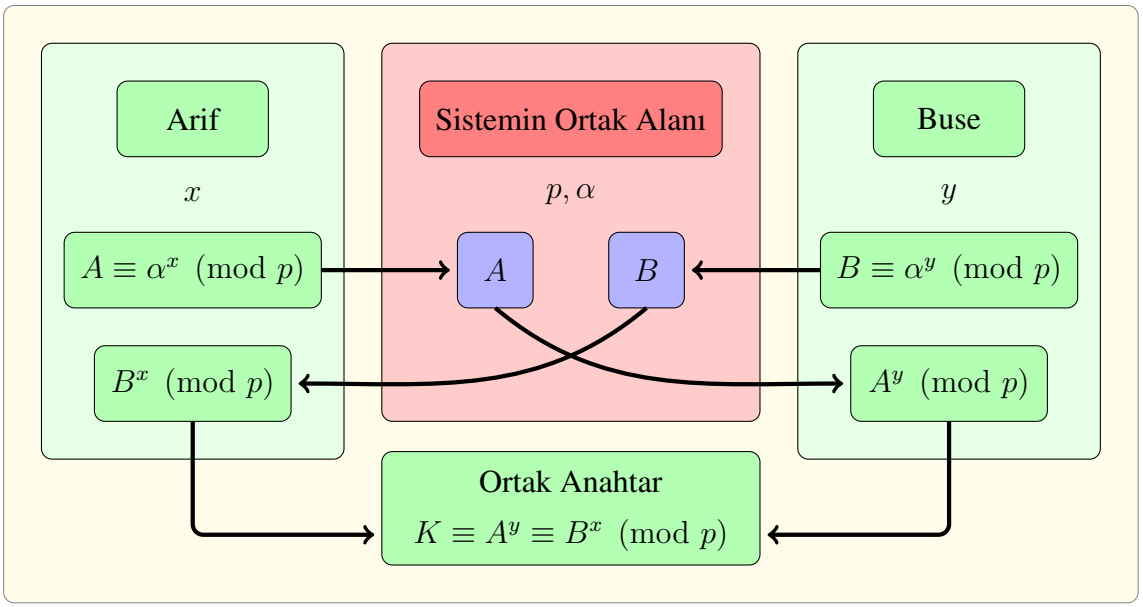
Sistemin işleyişini temel iletişim senaryomuzdaki Arif ve Buse üzerinden anlatalım. İlk olarak Arif veya Buse'den birisi çok büyük bir p asal sayısı ve yine çok büyük bir α sayısı belirler. Bu sayılar herkes tarafından görülebilir. Daha sonra hem Arif hem de Buse sadece kendilerinin görebileceği birer sayı seçerler. Bu sayılar Arif ve Buse'nin gizli anahtarlarıdır. Arif'in seçtiği sayıya x ve Buse'nin seçtiği sayıya y diyelim. Bu sistemdeki tüm süreç $(\mathrm{mod}\ p)$'ye göre işleyecektir. Bu sebeple hem α sayısı hem de x ve y sayıları p'den küçük olmalıdır.

Arif $A \equiv \alpha^x \pmod{p}$ sayısını hesaplayıp Buse'ye gönderir. Sistemde bulunan diğer kişiler A sayısını görebilirler fakat A, α ve p sayılarının bilinmesi, kullanılan sayıların çok büyük olması sebebiyle bu eşlenik denklemin çözülebilmesi için yeterli değildir. Dolayısıyla Buse de dahil olmak üzere A sayısını bilen hiçkimse Arif'in gizli anahtarı olan x sayısını elde edemez. Daha sonra, Arif'in gönderdiği A sayısını alan Buse, kendi gizli anahtarını kullanarak $A^y \pmod{p}$ sayısını hesaplar.

Benzer şekilde Buse de $B \equiv \alpha^y \pmod{p}$ sayısını hesaplayıp Arif'e gönderir. Yine aynı sebeplerden, Arif de dahil olmak üzere B, α ve p sayılarını bilen hiçkimse y sayısını elde edemez. Buse'den B sayısını alan Arif kendi gizli anahtarını kullanarak $B^x \pmod{p}$ sayısını hesaplar.

$$A^y \equiv (\alpha^x)^y \equiv \alpha^{xy} \equiv (\alpha^y)^x \equiv B^x \pmod{p}$$

olduğundan $A^y \equiv B^x$ elde edilir ve bu sayı, sistemde Arif ve Buse haricindeki kişilerde bulunmamaktadır. Böylece Arif ve Buse arasında $K \equiv A^y \equiv B^x \pmod{p}$ ortak anahtarı belirlenmiş olur.



Şekil 3.17. Diffie-Hellman anahtar anlaşması

Örnek 3.4.3. Algoritmada kullanılan sayılar çok büyük sayılardır fakat örnek yapmaktaki amacımız güvenli bir anahtar değişimi yapmak değil, algoritmanın işleyişini daha anlaşılır hale getirmektir. Bu sebeple örneğimizi daha küçük sayılarla oluşturalım.

$p = 211$ ve $\alpha = 139$ olsun. Arif, gizli anahtarını $x = 109$ ve Buse, gizli anahtarını $y = 167$ olarak belirlemiş olsunlar. Bu durumda Arif'in Buse'ye göndereceği sayı

$$A = \alpha^x = 139^{109} \equiv 52 \pmod{211}$$

şeklinde hesaplanır. $A = 52$ sayısını sistemdeki herkes görebilir fakat eşlenik denklemi çözemeyeceklerinden 109 sayısına ulaşamazlar. Buse, Arif'ten gelen 52 sayısını alır ve kendi gizli anahtarını kullanarak

$$A^y = 52^{167} \equiv 46 \pmod{211}$$

sayısını hesaplar ve bu sayı ortak anahtardır. $y = 167$ sayısını sadece Buse bildiğinden sistemdeki hiçkimse 46 sayısına ulaşamaz. Bu sayının Arif'te de olması için Buse,

$$B = \alpha^y = 139^{167} \equiv 189 \pmod{211}$$

hesabını yaparak 189 sayısını Arif'e gönderir. 189 sayısını alan Arif kendi gizli anahtarını kullanarak

$$B^x = 189^{109} \equiv 46 \pmod{211}$$

hesabını yapar. İşlem sonucunda elde ettiği 46 sayısı Buse'nin elde ettiği anahtar ile aynıdır. $x = 109$ sayısını sadece Arif bildiğinden sistemdeki hiçkimse 46 sayısını elde edemeyecektir. Dolayısıyla 46 sayısı sistemde sadece Arif ve Buse'de vardır ve bu sayı, ikisi arasında veri aktarımında kullanılacak ortak anahtardır. $K = 46$.

Tanım 3.4.4. (Ayrık Logaritma Problemi) Diffie-Hellman anahtar anlaşmasında kullanılan eşlenik denklemi reel sayılar cisminde olsaydı logaritma fonksiyonu kullanılarak kolaylıkla çözülebilirdi fakat sonlu cisimlerde çalışırken problemi çözmek imkansız denecek kadar zordur. Diffie-Hellman algoritmasının dayandığı bu probleme ayrık logaritma problemi denir.

3.4.2 RSA (Rivest-Shamir-Adleman)

"91 sayısı asal mıdır?" sorusu sorulduğunda birçoğumuz ilk etapta asal olduğunu düşünürüz fakat bu sayı 7 ile 13 sayılarının çarpımıdır, yani asal değildir. Bize bir sayı verildiğinde, sayının çarpanlarını bulmak bir kenara, o sayının asal olup olmadığını bile söylemek çok zordur. Hatta bu soruya cevap vermenin tek kesin yolu karekökünden küçük olan tüm asallara bölmektir. Bunun haricindeki tüm yollar küçük de olsa hata paylarına sahiptirler. Verilen sayı çok büyük olduğunda elimizdeki tek kesin yolu kullanmak ise yıllarımızı alır. Buna matematikte "çarpanlara ayırma problemi" denir ve 1978 yılında MIT'den üç kişi bu problemi kullanan bir algoritma açıklamışlardır. Bu kişiler, Ron Rivest, Adi Shamir ve Leonard Adleman'dır. Algoritmanın adı da bu kişilerin soyadlarının baş harflerinden oluşmaktadır. RSA algoritması, ortamdaki herkesin anahtarlarının farklı olduğu ve iletileri herkesin gördüğü halde gönderici ve alıcı haricindeki kimsenin orijinal metne ulaşamadığı bir şifreleme sistemidir, yani asimetriktir. Önce algoritmanın işleyişini temel iletişim senaryomuz üzerinden görelim, ardından daha iyi anlaşılması için bir örnek verelim.

Arif çok büyük iki p ve q asal sayıları seçer ve $n = p.q$ sayısını hesaplar. Ardından n sayısının Euler φ fonksiyonu altındaki görüntüsü olan $(p-1)(q-1)$ değerini hesaplar ve bir de $(e, (p-1)(q-1)) = 1$ olacak şekilde bir e sayısı seçer.

n ve e sayılarını herkese açık bir şekilde Buse'ye gönderir. Buse de dahil olmak üzere ortamdaki hiçkimse n ve e sayılarından yola çıkarak p ve q sayılarını elde edemezler, çünkü n sayısı çok büyük olduğundan çarpanlarına ayıramazlar. Buse mesajını n sayısından küçük değere sahip olacak şekilde bloklara ayırır. Bir bloğun değeri m olsun.

$$c \equiv m^e \pmod{n}$$

değerini hesaplar ve yine herkese açık bir şekilde Arif'e gönderir.

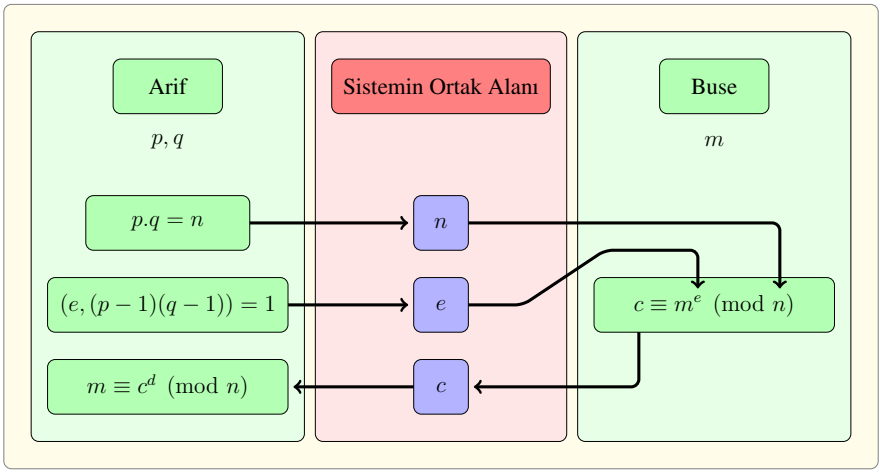
Arif p ve q sayılarını bildiğinden genişletilmiş Öklid algoritmasını kullanarak

$$d.e \equiv 1 \pmod{(p-1)(q-1)}$$

denkliğini veren d sayısını hesaplar. Ardından

$$c \equiv m^e \quad \Rightarrow \quad c^d \equiv (m^e)^d \quad \Rightarrow \quad m \equiv c^d \pmod{n}$$

işlemini yaparak m açık metnine ulaşır.



Şekil 3.18. RSA algoritması

Örnek 3.4.5. Örnekteki amacımız güvenli bir şifreleme oluşturmaktan ziyade algoritmanın işleyişini daha iyi anlatmak olduğu için sayı seçimimizi küçük sayılardan yapalım. Arifin seçtiği asal sayılar $p = 227$ ve $q = 379$ olsun. Bu durumda $n = 227.279 = 86033$ olarak hesaplanır. Buradan

$$\varphi(n) = (p-1)(q-1) = 226.378 = 85428$$

sayısını hesaplarız. Arif'in $(e, \varphi(n)) = 1$ olacak şekildeki sayısı da $e = 97$ olsun. Arif $n = 86033$ ve $e = 97$ sayılarını herkese açık bir şekilde Buse'ye gönderir. Ortamdaki hiçkimse bu sayılara bakarak $p = 227$ ve $q = 379$ sayılarına ulaşamaz. (Sayıları küçük seçtiğimiz için 86033 sayısı bilgisayarlar yardımıyla kolayca çarpanlarına ayrılabilir fakat sayılar büyük seçildiğinde bu işlem günler hatta aylar sürer.)

Buse'nin göndereceği mesaj "SELAM" olsun. Türkçe harfleri sayılarla eşleştirdiğimiz tablodan faydalanarak bu mesajın sayısal değerini 21 05 14 00 15 olarak yazarız. Yani Buse'nin göndereceği mesaj $m = 2105140015$'tir. Mesajın değeri n sayısından küçük olması gerektiği için mesajımızı $m_1 = 21051$ ve $m_2 = 40015$ şeklinde iki bloğa ayırırız. Ardından Buse,

$$\begin{aligned} c_1 &\equiv m_1^e \equiv 21051^{97} \equiv 84314 \pmod{86033} \\ c_2 &\equiv m_2^e \equiv 40015^{97} \equiv 81348 \pmod{86033} \end{aligned}$$

hesaplamalarını yaparak Arif'e $c_1 = 84314$ ve $c_2 = 81348$ sayılarını herkese açık bir şekilde gönderir. Ortamdaki hiçkimse, ayrık logaritma problemi gereği, bu sayıları kullanarak m_1 ve m_2 değerlerine ulaşamaz.

Arif önce, genişletilmiş Öklid algoritmasını kullanarak

$$d.e \equiv 1 \pmod{\varphi(n)}$$

olacak şekildeki d sayısını hesaplayarak $d = 76621$ sayısına ulaşır ve daha sonra

$$\begin{aligned} c_1^d &\equiv 84314^{76621} \equiv m_1 = 21051 \pmod{86033} \\ c_2^d &\equiv 84348^{76621} \equiv m_2 = 40015 \pmod{86033} \end{aligned}$$

hesaplarını yaparak $m_1 = 21051$ ve $m_2 = 40015$ değerlerine ulaşır. Elde ettiği değerleri birleştirerek $m = 2105140015$ açık metnin sayısal değerini bulur ve bunların harf karşılıklarını yazarak "SELAM" mesajına ulaşır.

3.4.3 Eliptik Eğri

Asimetrik şifreleme sistemlerinden en karmaşık ve derin matematiksel konuları barındıran algoritma, eliptik eğri şifreleme algoritmasıdır. Sonlu cisimler üzerinde oluşturulmuş eliptik eğriler temel alınarak inşa edilmiştir. Eliptik eğri algoritmasının güvenliğinin dayandırıldığı matematiksel probleme "eliptik eğrilerde ayrık logaritma problemi" denir. Bu problem, eliptik eğri üzerinde oluşturulmuş olan ve bilinen toplama ve çarpma işlemlerinden biraz farklı olarak tanımlanan işlemler sayesinde, süreç sonucunda elde edilen noktalardan yola çıkarak asıl noktaları bulmanın imkansız oluşuna dayanır.

Ayrıca eliptik eğri algoritmasının sağladığı en büyük fayda, kullanılan anahtarların boyunun rahat bir şekilde küçültülebilmesidir. Örneğin eliptik eğri kriptografisinde 256 bit uzunluğunda bir anahtarın sağladığı güvenlik seviyesini RSA algoritması ancak 15360 bit uzunluğunda bir anahtar kullanarak sağlayabilmektedir. Bu sayede, bir şifreleme yaparken eliptik eğriler kullanıldığında depolama alanından büyük kazanç elde edilir.

Bu kadar çok matematiksel bilgiye dayanan bir algoritma doğal olarak matematikçilerin elinden çıkmıştır diye düşünebiliriz ki böyle düşünmekte haklıyız. Eliptik eğrilerin kriptografik anlamda kullanımı ilk kez 1985 yılında birbirinden bağımsız olarak Neal Koblitz ve Victor Saul Miller adlarında iki matematikçi tarafından gündeme getirilmiştir ve 2000'li yılların başlarında çok geniş kullanım alanlarına yayılmıştır.

Algoritmanın işleyişine geçmeden eliptik eğrinin ne olduğunu ve üzerinde tanımlanan toplama ve çarpma işlemlerini görelim. Burada belirtilen "çarpma" işlemi, iki elemanın birbiriyle çarpılması değil, bir elemanın iki katının alınması işlemidir. Bir elemanın iki katının alınması, bildiğimiz üzere, bir elemanı kendisiyle toplamaktır. Dolayısıyla "eliptik eğri kriptografisinde tek işlem vardır, o da toplamadır" dersek pek de haksız sayılmayız.

Tanım 3.4.6. (Eliptik Eğri) Bir eliptik eğri $E : y^2 = x^3 + ax^2 + bx + c$ denklemiyle verilen ve (x, y) değerlerini, $\mathbb{F}$ bir cisim olmak üzere $\mathbb{F}^2 = \mathbb{F} \times \mathbb{F}$ kümesinden alan bir düzlem eğrisidir. Yani kısaca bir eliptik eğriyi aşağıdaki şekilde yazabiliriz:

$$a, b, c \in \mathbb{F} \text{ sabitler olmak üzere } E = \left\{(x, y) \in \mathbb{F}^2 \mid y^2 = x^3 + ax^2 + bx + c\right\}$$

Önerme 3.4.7. (Weierstrass Form) $y^2 = x^3 + ax^2 + bx + c$ formundaki her eliptik eğri $y^2 = x^3 + Ax + B$ formunda yazılabilir.

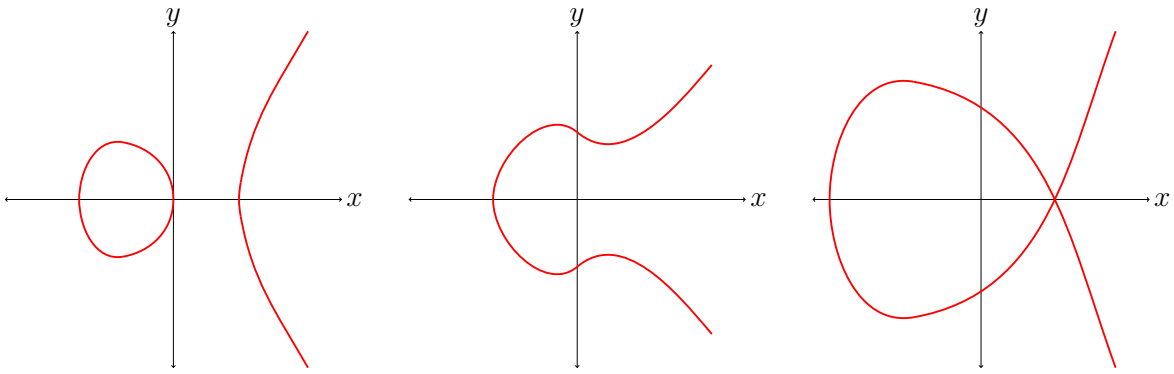
Kanıt. x yerine $x - \frac{1}{3}a$ yazılarak kolayca görülür. □

Bu önerme sayesinde, kullanacağımız eliptik eğrileri kısaca $E(a, b) : y^2 = x^3 + ax + b$ şeklinde yazabiliriz.

Tanım 3.4.8. $E : y^2 = x^3 + ax + b$ eliptik eğrisinin diskriminantı $\Delta = -16(4a^3 + 27b^2)$'dir. Diskriminantı sıfıra eşit olan eğrilere tekil (singular veya non-smooth) eğriler denir. Tekil eğriler şifreleme için güvenli değillerdir.

Not : Bir $E : y^2 = x^3 + ax + b$ eliptik eğrisi aynı zamanda $y = \pm\sqrt{x^3 + ax + b}$ şeklinde de yazılabileceğinden yatay eksene (x-eksenine) göre simetriktir.

Örnek 3.4.9. Aşağıda üç farklı eliptik eğri örneği verilmiştir.



$y^2 = x^3 - x$ $\Delta = 64$ | $y^2 = x^3 - x + 1$ $\Delta = -368$ | $y^2 = x^3 - 3x + 2$ $\Delta = 0$

Eliptik eğrilerde işlemlere geçmeden önce şunu da belirtelim: Eliptik eğriler üzerinde tanımlanacak olan toplama işleminde birim elemanın olmayışından doğan bir tanımsızlık söz konusudur. İşlemlerde tanımsızlığı gidermek adına E eliptik eğri kümesine bir de "sonsuzdaki nokta" adında bir eleman eklenir. Bu eleman toplama işleminin birim elemanı olarak kabul edilir. Eklenen noktanın nerede olduğunu ve nasıl sonuç verdiğini, toplama işlemini tanımlarken daha iyi anlayacağız. Ayrıca bu nokta, bilinen sonsuz sembolüyle (∞) gösterilir. Böylelikle eliptik eğri kümemiz

$$E = \left\{(x, y) \mid x, y \in \mathbb{F}, y^2 = x^3 + ax^2 + b\right\} \cup \{\infty\}$$

şeklinde yazılır.

Örnek 3.4.10. $\mathbb{F}_5 = \mathbb{Z}_5$ cismi üzerinde tanımlanan $E : y^2 = x^3 + 2x - 1$ eliptik eğrisi üzerindeki noktalar $(0, 2), (0, 3), (2, 1), (2, 4), (4, 1), (4, 4), \infty$ şeklindedir.

Eliptik eğrileri $\mathbb{R}^2$ düzleminde çizerek görebiliyoruz fakat eliptik eğri kriptografisindeki eğrileri tanımlarken kullanılan cisim bir sonlu cisim olduğundan, eğrinin geometrik olarak görüntüsü ayrık noktalar şeklindedir. Ayrık noktalı bir yapıda, işlemleri yürütmek $\mathbb{R}^2$ düzleminde yürütmek kadar kolay değildir, derin bir cebirsel geometri bilgisi gerektirir. Eğer işlemlerin geometrik olarak ne ifade ettiklerini merak ederseniz, yapılan işlemleri $\mathbb{R}^2$ düzleminde yapılıyormuş gibi çizerek görebilirsiniz.

3.4.3.1 İki Farklı Noktanın Toplamı

Bir $E : y^2 = x^3 + ax + b$ eliptik eğrisi üzerinde $A \neq B$ olacak şekilde $A = (x_1, y_1)$ ve $B = (x_2, y_2)$ noktalarını alalım ve $A + B$ toplamının sonucunun nasıl bulunduğunu görelim. İlk olarak A ve B'den geçen doğru çizilir. Şimdilik çizilen doğrunun dikey olmadığını varsayalım, dikey olduğu duruma daha sonra bakalım, böylelikle "sonsuzdaki nokta" kavramı da daha iyi anlaşılacaktır.

Eğer doğru dikey değilse E eğrisini mutlaka üçüncü bir noktada tekrar keser. (Bunun sebebi, eliptik eğrilerin denklemlerinin 3. dereceden bir polinom şeklinde olmasıdır. A ve B noktaları, kesişim noktalarından ikisi olduğu için ve birbirinden farklı noktalar olduğu için, eğri ile doğrunun, aşağıda göreceğimiz gibi, mutlaka üçüncü bir kesişim noktası daha olmalıdır. Bu durumda toplamın sonucu, doğrunun eğriyi kestiği üçüncü noktanın yatay eksene göre simetriği olan nokta olarak tanımlanır.

A ve B'den geçen doğrunun eğimi $m = \frac{y_2 - y_1}{x_2 - x_1}$'dir. Eğimi ve A noktasını kullanarak doğrunun denklemini

$$L : y = m(x - x_1) + y_1$$

şeklinde yazabiliriz. L doğrusunun denklemini E eğrisinde yerine yazarak kesişim noktasını bulabiliriz.

Yerine yazma işlemini yaptığımızda

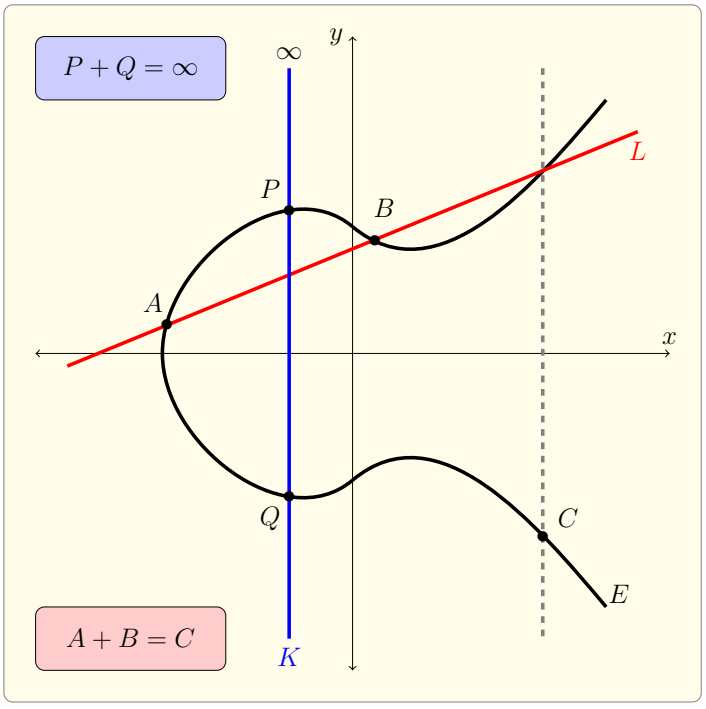
$$(m(x - x_1) + y_1)^2 = x^3 + ax + b \Rightarrow 0 = x^3 - m^2x^2 + \cdots$$

eşitliğine ulaşırız. (Denklemin kalan kısmı lazım olmadığı için yazılmamıştır.) Elde edilen eşitlik üçüncü dereceden bir polinom olduğundan bu denklemi kökleri cinsinden yazabiliriz. Biliyoruz ki iki kök A ve B noktalarının x-koordinatlarıdır. Üçüncü kök de bize, doğru ile eğrinin kesiştiği üçüncü noktanın x-koordinatını verir. Ayrıca üçüncü dereceden bir denklemin köklerinin toplamı x^2'li terimin negatif değerini verdiğinden $m^2 = x_1 + x_2 + x_3$ eşitliği elde edilir ve böylece üçüncü kesişim noktasının x-koordinatı $x_3 = m^2 - x_1 - x_2$ olarak bulunur. Bulunan x_3 değeri L doğrusunun denkleminde yerine yazılarak üçüncü noktanın

y-koordinatı bulunur. $y_3 = m(x_3 - x_1) + y_1$. Böylelikle toplamın sonucu aşağıdaki şekilde bulunur:

$$A + B = C = (x_3, -y_3) = (m^2 - x_1 - x_2, -m(x_3 - x_1) - y_1)$$

Şimdi, doğrunun dikey olduğu duruma geçelim. Bu durumda $y^2 = x^3 + ax + b$ denkleminde $x = x_1 = x_2$ yazarak tam iki y değeri buluruz ki bunlar A ve B noktalarının y-bileşenleridir. Dolayısıyla, A ve B noktalarının x-koordinatları eşit ise A ve B'den geçen doğru E eğrisini sadece iki noktada keser ve doğrunun iki ucu da eğriyi kesmeden sonsuza doğru giderler. Böyle bir durumda "$A + B$ toplamının sonucu sonsuz noktası adındaki elemana eşittir" tanımı yapılarak tanımsızlık ortadan kaldırılmış olur.



Şekil 3.19. Eliptik eğrilerde iki farklı noktanın toplamı

3.4.3.2 Bir Noktanın İki Katı

Bir $E : y^2 = x^3 + ax + b$ eliptik eğrisi üzerinde bir $A = (x_1, y_1)$ noktasını alalım ve işlem sonucunda elde edeceğimiz noktaya $A + A = 2A = C = (x_3, y_3)$ diyelim. Burada teğet

doğrusu kullanılacağından birkaç hatırlatma yapalım. $F(x,y) = \sum c_{i,j}x^i y^j$ polinomu için

$$\frac{\partial F}{\partial x} = \sum c_{i,j} i x^{i-1} y^j \quad \text{ve} \quad \frac{\partial F}{\partial y} = \sum c_{i,j} j x^i y^{j-1}$$

olarak tanımlanır. Böylece, A'daki teğet doğrusunun denklemi aşağıdaki eşitlik ile verilir:

$$\frac{\partial F}{\partial x}(x_1, y_1)(x - x_1) + \frac{\partial F}{\partial y}(x_1, y_1)(y - y_1) = 0$$

Eğer A noktası, eliptik eğrinin x-eksenini kestiği nokta ise toplamın sonucu ∞ noktasıdır. Çünkü bu noktadaki teğet doğrusu dikey eksene paraleldir. Eğer A noktası eliptik eğrinin x-eksenini kestiği noktadan farklı bir nokta ise önce eliptik eğriye A noktasında teğet olan L doğrusu çizilir. (Eğri tekil olarak seçilmediği için her noktasından tek bir teğet doğrusu çizilebilir.) Çizilen teğet doğrusu eliptik eğriyi mutlaka başka bir noktada da keser. Kestiği noktanın yatay eksene göre simetriği olan nokta $2A = C$ işleminin sonucudur.

A noktasındaki teğet doğrusunun denklemini, yukarıda verdiğimiz eşitlik yardımıyla aşağıdaki şekilde hesaplarız:

$$y^2 = x^3 + ax + b \quad \Rightarrow \quad x^3 + ax + b - y^2 = 0$$

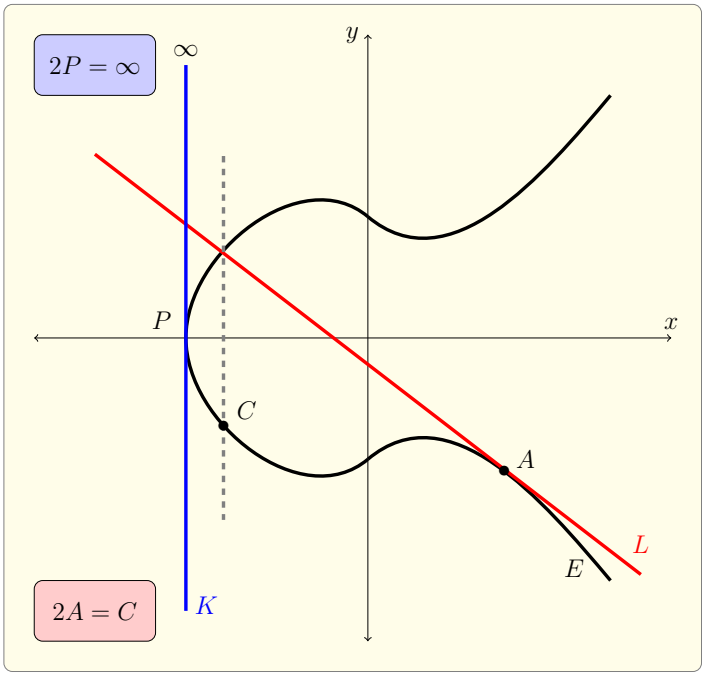
$$\frac{\partial F}{\partial x} = 3x^2 + a \quad \text{ve} \quad \frac{\partial F}{\partial y} = -2y$$

$$L : (3x_1^2 + a)(x - x_1) - 2y_1(y - y_1) = 0 \quad \Rightarrow \quad L : \ y = \frac{3x_1^2 + a}{2y_1}(x - x_1) + y_1$$

Bu denklemden anlıyoruz ki teğet doğrusunun eğimi $m = \frac{3x_1^2+a}{2y_1}$'dir.

Geri kalan işlemler "iki farklı noktanın toplamı"nda yaptıklarımıza benzerdir. İşlemler sonucunda $x_3 = m^2 - 2x_1$ ve $y_3 = m(x_3 - x_1) + y_1$ elde edilir.

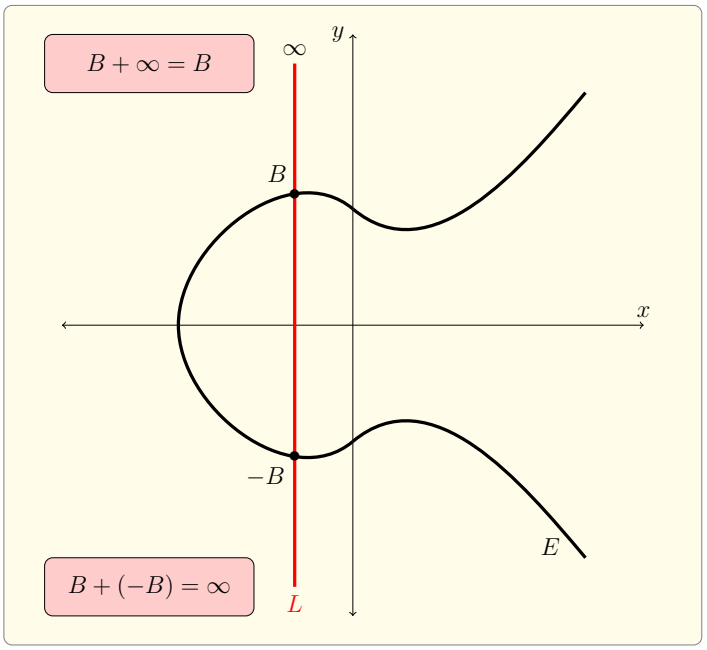
Buradan $2A = C = (x_3, -y_3) = (m^2 - 2x_1, -m(x_3 - x_1) - y_1)$ noktasına ulaşılır.



Şekil 3.20. Eliptik eğrilerde bir noktanın iki katı

Toplama İşlemine Göre Birim Eleman : Eliptik eğrilerde toplama işleminin birim elemanı ∞ noktasıdır. E eğrisi üzerinde bir A noktası alalım ve bu noktayı ∞ ile toplayalım. ∞ ve A noktasından geçen doğru, dikey olduğundan, E eğrisini A'nın simetriği olan üçüncü bir noktada keser. Toplama işleminin sonucu, doğrunun eğriyi kestiği bu üçüncü noktanın simetriği olduğundan ve A noktasının simetriğinin simetriği kendisi olduğundan $\infty + A$ işleminin sonucu A noktasıdır. Bu durum E eğrisi üzerindeki her nokta için geçerlidir. Yani, $\forall\ A \in E$ için $A + \infty = A$'dır.

Toplama İşlemine Göre Ters Eleman : Eliptik eğri üzerindeki bir B noktası için B'nin tersi ($-B$ ile gösterilir) B noktasının yatay eksene göre simetriği olan noktadır. Çünkü B'nin simetriğinden ve B'den geçen doğru dikeydir ve ∞ noktasına gider. Yani $\forall\ B \in E$ için $B + (-B) = \infty$'dir.



Şekil 3.21. Eliptik eğrilerde toplamaya göre birim ve ters eleman

Eliptik eğrilerde toplama ve iki kat alma işlemlerinin hesaplama zorluğu açıkça görülüyor. Üstelik bir E eğrisi üzerinden seçilen bir A noktasının $17A, 487A, 1549A, \ldots$ gibi katlarını hesaplamak çok daha zor olacaktır. Örneğin, bir A noktasının 3 katı tek bir işlem süreciyle hesaplanamıyor, önce $2A$, daha sonra $2A + A = 3A$ hesabının yapılması gerekiyor, yani çok küçük sayılarda bile oldukça zahmetli bir iş. Eliptik eğrilerdeki ayrık logaritma problemi de aslında tam olarak bu zorluğa dayanır. Bu problem kısaca; eğri üzerindeki bir A noktası için, k bir tamsayı olmak üzere, $k.A = B$ sonucu verildiğinde A'nın hangi katının B'ye eşit olacağının bulunması zorluğudur.

Örnek 3.4.11. $\mathbb{F}_{19} = \mathbb{Z}_{19}$ cismi üzerinde $E : y^2 = x^3 - x + 1$ eğrisini alalım ve eğri üzerinden $A = (8, 7)$ ve $B = (3, 6)$ noktalarını seçelim.

A+B : A ve B'den geçen doğrunun eğimi

$$m = \frac{6-7}{3-6} = \frac{14}{16} = 14.16^{-1} \equiv 14.6 \equiv 8 \pmod{19}$$

olarak bulunur. Buradan koordinatları aşağıdaki şekilde hesaplarız.

$$\begin{array}{rcrcrcrcl} x_3 & = & m^2 - x_1 - x_2 & = & 8^2 - 8 - 3 & = & 53 & \equiv & 15 \pmod{19} \\ y_3 & = & -m(x_3 - x_1) - y_1 & = & -8(15 - 8) - 7 & = & -63 & \equiv & 13 \pmod{19} \end{array}$$

Böylece $A + B = (15, 13) \pmod{19}$ olarak bulunur.

2A : E eğrisinin A noktasındaki teğetinin eğimi

$$m = \frac{3x_1^2 + a}{2y_1} = \frac{192 + (-1)}{14} = \frac{191}{14} = 191.14^{-1} \equiv 1.15 \equiv 15 \pmod{19}$$

olarak bulunur. Buradan koordinatları aşağıdaki şekilde hesaplarız.

$$\begin{array}{rcrcrcrcl} x_3 & = & m^2 - x_1 - x_2 & = & 15^2 - 8 - 3 & = & 214 & \equiv & 5 \pmod{19} \\ y_3 & = & -m(x_3 - x_1) - y_1 & = & -15(5 - 8) - 7 & = & 38 & \equiv & 0 \pmod{19} \end{array}$$

Böylece $2A = (5, 0) \pmod{19}$ olarak bulunur.

Artık eliptik eğrilerin kullanıldığı algoritmalara geçebiliriz. Algoritmaları temel iletişim senaryomuz üzerinden anlatalım.

3.4.3.3 Eliptik Eğrilerde Diffie-Hellman Anahtar Değişimi

Arif ve Buse ilk olarak herkese açık bir şekilde bir p asal sayısı (veya 2'nin bir tam kuvveti olan $q = 2^m$ sayısı) ve bir $E(a, b) : y^2 \equiv x^3 + ax + b$ eğrisi belirlerler. Belirledikleri E eğrisi üzerinde yine herkese açık bir şekilde bir G noktası seçerler. Yani kısacası, a, b, p, E, G ortamdaki herkes tarafından biliniyor.

Daha sonra Arif ve Buse kendilerine sırasıyla A ve B sayılarını seçerler. Bu sayılar Arif ve Buse'nin gizli anahtarlarıdır. Arif gizli anahtarını ve G noktasını kullanarak $A' = A.G \pmod{p}$ noktasını hesaplar, aynı şekilde Buse de $B' = B.G \pmod{p}$ noktasını hesaplar ve hesapladıkları noktaları herkese açık bir şekilde birbirlerine gönderirler. Ortamdaki hekes A', B' ve G noktalarını bilmelerine rağmen, eliptik eğrilerde ayrık logaritma problemi gereği, A ve B sayılarını elde edemezler.

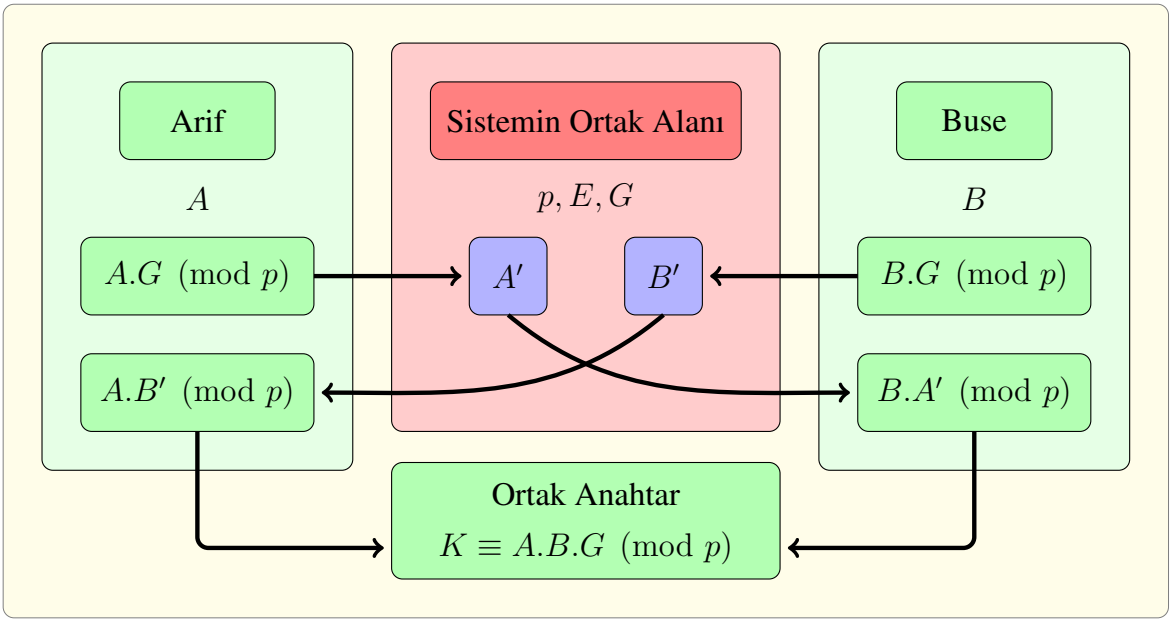
Arif B' noktasını alır ve $A.B' \pmod p$ noktasını hesaplar. Buse de aynı şekilde A' noktasını alır ve $B.A' \pmod p$ noktasını hesaplar. Buradan

$$A.B' \equiv A.B.G \equiv B.A.G \equiv B.A' \pmod p$$

eşitliği elde edilir ki bu nokta ortamda sadece Arif ve Buse'de bulunmaktadır. Yani işlemler sonucunda Arif ve Buse'nin elde ettiği ortak anahtar $K \equiv A.B.G \pmod p$ noktasıdır.

Tanım 3.4.12. (Noktanın Derecesi) Bir E eğrisi üzerinden alınan bir A noktası için $t.A \equiv \infty \pmod p$ denkliğini sağlayan en küçük t pozitif tamsayısına A noktasının derecesi denir. $der(A)$ ile gösterilir.

Eliptik eğrilerde şifreleme yaparken güvenliği artırmak istiyorsak p sayısını ve G noktasının derecesini daha büyük seçmeliyiz.



Şekil 3.22. Eliptik eğrilerde Diffie-Hellman anahtar değişimi

3.4.3.4 Eliptik Eğrilerde El-Gamal

Diffie-Hellman'da olduğu gibi Arif ve Buse herkese açık olacak şekilde p, a, b, E, G parametrelerini belirlerler. Arif kendisine $A < der(G)$ olacak şekilde bir gizli anahtar seçer. Ardından $A' \equiv A.G \pmod p$ noktasını hesaplar.

Buse de kendisine gizli bir B sayısı seçip $B' \equiv B.G \pmod{p}$ değerini hesaplar. Daha sonra Buse, Arif'e göndermek istediği mesajı E eğrisi üzerinde bir M noktası ile eşleştirip $N \equiv M + B.A'$ noktasını hesaplar. Herkese açık bir şekilde N ve B' noktalarını Arif'e gönderir.

Arif, mesajı aşağıdaki şekilde hesaplar.

$$\begin{aligned} N - A.B' &\equiv (M + B.A') - A.(B.G) \\ &\equiv M + (B.A.G) - A.(B.G) \\ &\equiv M + (B.A.G) - (A.B.G) \equiv M \pmod{p} \end{aligned}$$

3.4.4 Özet (Hash) Fonksiyonları

Birçoğumuz, sosyal medya veya mesajlaşma uygulamalarına kaydolurken uygulama sahiplerinin şifrelerimizi görüp göremediğini merak etmişizdir. Eğer kullandığımız uygulama güvenlik açısından kendini kanıtlamış bir uygulama ise bu sorunun cevabı kısaca "hayır"dır, siz hariç hiç kimse şifrenizi göremez. Bu durumda aklımıza yeni bir soru gelir: "Şifremizi bilmiyorlarsa uygulamaya giriş yaparken şifremizi doğru yazıp yazmadığımızı nasıl belirliyorlar?" Bu sorunun kriptografik açıdan karşılığı "kimlik doğrulama nasıl sağlanıyor?" sorusudur. Bu amaç için kullanılan birçok yöntem vardır, bunlardan birisi de özet fonksiyonlarıdır. Özet fonksiyonları, adından da anlaşılacağı üzere bir fonksiyondur ve bu fonksiyonların bazı değişmez özellikleri vardır:

1. **Sabit Çıktı Uzunluğu (Fixed Length Output) :** Fonksiyonun girdisinin uzunluğu ne olursa olsun görüntüsünün uzunluğu sabittir.

2. **Ön-görüntü Direnci (Pre-image Resistance) :** Bir girdinin özet fonksiyonu altındaki görüntüsüne bakarak girdinin ne olduğunu bulmak zordur.

3. **İkincil Ön-görüntü Direnci (Second Pre-image Resistence) :** Bir girdi ve bu girdinin özet fonksiyonu altındaki görüntüsü verildiğinde aynı görüntüye sahip farklı bir girdinin bulunması zordur.

4. **Çakışma Direnci (Collision Resistance) :** Bir özet fonksiyonu altındaki görüntüleri aynı olan iki girdi bulmak zordur.

Özet fonksiyonları altındaki görüntülere bakarak öngörüntüyü bulmanın zorluğu bu fonksiyonlarının şifreleme için kullanılamamasına sebep olur. Zaten özet fonksiyonlarının kullanım amacı da bu değildir. Peki bu fonksiyonlar bizim şifrelerimizin güvende kalmasını nasıl sağlıyorlar? Bir mesajlaşma uygulamasına üye olduğumuzu düşünelim. Şifremizi oluşturduğumuzda, uygulamanın veri tabanında şifremiz değil, şifremizin bir özet fonksiyonu altındaki görüntüsü kaydedilir. Özet fonksiyonlarının ön-görüntü direnci özelliği sayesinde, uygulama sahipleri şifremizin görüntüsüne bakarak öngörüntüyü (yani şifremizi) elde edemezler. Fakat biz uygulamaya her giriş yapışımızda girdiğimiz şifrenin görüntüsü, uygulamanın veri tabanındaki görüntü ile rahatlıkla karşılaştırılabilir ve şifremizi doğru yazıp yazmadığımız kolay bir şekilde belirlenebilir. Benzer şekilde, ikincil-ön görüntü özelliği gereği, şifremizin özet altındaki görüntüsüne bakılarak, şifremizin yerine geçebilecek farklı bir şifre de belirlenemez.

Şimdi bir fonksiyon örneği verelim ve bu fonksiyonun bir özet fonksiyonu olup olamayacağını belirleyelim.

Örnek 3.4.13. Girdilerini $\mathbb{Z}_{10}$'dan alan 3×8 formundaki tüm matrislerin kümesini $\mathbb{Z}_{10}^{3\times 8}$ ile gösterelim ve f fonksiyonunu aşağıdaki şekilde tanımlayalım:

$$\begin{array}{rccl} f & : & \mathbb{Z}_{10}^{3\times 8} & \to & \mathbb{Z}_{10}^{8} \\ & & [a_{ij}]_{3\times 8} & \mapsto & (b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8) \ni b_k = a_{1k} + a_{2k} + a_{3k} \pmod{10} \end{array}$$

$\mathbb{Z}_{10}^{3\times 8}$ kümesinden alınan her elemanın f altındaki görüntüsünün uzunluğu 8 olduğundan f fonksiyonu "sabit çıktı uzunluğu" özelliğini sağlar.
Bir $A \in \mathbb{Z}_{10}^{3\times 8}$ elemanının f altındaki görüntüsüne bakarak tekrar A'ya dönmek zordur. Çünkü $f^{-1}(A)$'nın çok fazla değeri vardır. Bu yüzden f fonksiyonu "ön-görüntü direnci" özelliğini sağlar. Fakat $f(A)$ değerini veren birçok $B \in \mathbb{Z}_{10}^{3\times 8}$ bulunabileceğinden f fonksiyonu "ikincil ön-görüntü" ve "çakışma direnci" özelliklerini sağlamaz.
Sonuç olarak şunu söyleyebiliriz, eğer bu f fonksiyonunu özet fonksiyonu olarak kullanan

bir uygulamaya kayıt olursak, uygulama sahipleri bizim şifremizi bilmeseler bile şifremiz yerine geçecek başka bir girdiyi kullanarak hesabımıza rahatlıkla erişim sağlayabilirler. Örneğin, şifremizin matris görünümü

$$A = \begin{bmatrix} 3 & 5 & 7 & 3 & 6 & 1 & 9 & 3 \\ 5 & 8 & 1 & 4 & 5 & 6 & 7 & 3 \\ 9 & 4 & 5 & 7 & 1 & 2 & 3 & 5 \end{bmatrix}$$

olsun. Bu durumda, şifremizin f fonksiyonu altındaki görüntüsü aşağıdaki şekilde hesaplanır.

$$\begin{array}{llll} 3+5+9 = 17 \equiv 7 \pmod{10} & & 6+5+1 = 12 \equiv 2 \pmod{10} \\ 5+8+4 = 17 \equiv 7 \pmod{10} & & 1+6+2 = 9 \equiv 9 \pmod{10} \\ 7+1+5 = 13 \equiv 3 \pmod{10} & & 9+7+3 = 19 \equiv 9 \pmod{10} \\ 3+4+7 = 14 \equiv 4 \pmod{10} & & 3+3+5 = 11 \equiv 1 \pmod{10} \end{array}$$

$$\Rightarrow f(A) = (7, 7, 3, 4, 2, 9, 9, 1)$$

Bu görüntüden yola çıkarak, aynı görüntüye sahip birçok matris yazılabilir. Bu matrislerin birkaç örneği aşağıdaki gibidir.

$$B = \begin{bmatrix} 7 & 7 & 3 & 4 & 2 & 9 & 9 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad C = \begin{bmatrix} 7 & 0 & 0 & 0 & 2 & 0 & 9 & 0 \\ 0 & 0 & 3 & 4 & 0 & 0 & 0 & 1 \\ 0 & 7 & 0 & 0 & 0 & 9 & 0 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 7 & 2 & 9 & 0 & 7 & 3 & 8 & 6 \\ 5 & 2 & 9 & 7 & 6 & 3 & 8 & 6 \\ 5 & 3 & 5 & 7 & 7 & 3 & 3 & 9 \end{bmatrix} \qquad E = \begin{bmatrix} 5 & 5 & 1 & 2 & 2 & 7 & 7 & 9 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Yani, bizim şifremiz A olduğu halde şifremizi bilmeyen birileri uygulamaya giriş yaparken B, C, D, E şifrelerini denerse bizim hesabımıza girebilirler. Dolayısıyla f fonksiyonu bir özet fonksiyonu olarak kullanılamaz.

Örnek 3.4.14. Kriptografik anlamda kendi zamanlarında güvenilirliklerini kanıtlamış özet fonksiyonları MD (Message Digest) ve SHA (Secure Hash Algorithm) algoritma aileleridir. SHA ailesinin en popüler üyeleri SHA-1, SHA-2 ve SHA-3 algoritmalarıdır. SHA-1 algoritması en fazla 2^{64} bit uzunluğundaki girdileri alıp sabit 160 bit uzunluğunda görüntüler verirken SHA-2 algoritmasının girdi uzunluğu 2^{128} bite kadar çıkabilir ve görüntü uzunluğu da 256 bite kadar ulaşabilir. SHA-3 algoritmasında girdi uzunluğu için herhangi bir sınır bulunmamaktadır. Çıktı uzunluğu ise 512 bite kadar uzayabilmektedir. MD ailesinden MD5 özet algoritması da girdi uzunluğu açısından bir sınır koymadığı halde çıktı uzunluğu olarak 128 bitlik görüntüler vermektedir.

3.4.5 Elektronik İmza

Kriptografiyi tanımlarken temel amaçlarından bazılarının "kimlik doğrulama" ve "inkarı engelleme" olduğunu belirtmiştik. Kimlik doğrulama amacını özet fonksiyonları gayet başarılı bir şekilde yapmaktadır. Dijital imzalar ise her iki amacı da yerine getiren ve kimlik doğrulama için kullanıldığı yere göre özet fonksiyonlarına kıyasla daha üstün başarılar elde edebilen algoritmalardır.

Üzerinde yazılanları onaylamanız için size yazılı bir belge verildiğini ve yazılanlardan bazıları hoşunuza gitmediği için imzalamayı reddettiğinizi düşünelim. Bir başkası gelip imzanızı taklit ederek belgeyi sizin yerinize imzaladığında, grafoloji uzmanları gelişen teknoloji ile birlikte bu imzanın size ait olmadığı anlayabiliyor. Bu ayrımın yapılabilmesi için gerçekten sizin elinizden çıkan bir imzaya ihtiyaç duyuluyor çünkü size ait olan imzadaki yazının ayrıntılı incelemesi yapılırken karakteristik özelliklerin kıyaslanması gerekiyor. Fakat dijital ortamda sizin elinizden çıkan bir yazı ile bir başkası tarafından yazılmış yazılar fiziksel yolla kıyaslanamaz. Bu durumu özetleyecek bir cümle kullanmak istersek "klavye iz tutmaz" diyebiliriz. Elektronik imza algoritmaları da tam olarak böyle durumlarda devreye giriyorlar.

Öncelikle elektronik imzaların iki temel aşama ile çalıştığını belirtelim: "imzalama" ve "doğrulama" aşamaları. Bu aşamalar, adlarından da anlaşılacağı üzere, imzalamanın yapıldığı ve imzanın size ait olduğunun doğrulandığı aşamalardır.

Kriptografik açıdan elektronik imza işlemi yapılırken en çok kullanılan iki algoritma vardır. Bunlar RSA ve El-Gamal elektronik imza algoritmalarıdır. Bu algoritmalarla imzalamanın nasıl yapıldığını ayrı ayrı inceleyelim.

3.4.5.1 RSA İmzalama Algoritması

Algoritmayı temel iletişim senaryomuz üzerinden görelim.

Arif'in elindeki bir belgeyi Buse imzalamayı kabul etmiş olsun. Belgenin sayısal değerine m diyelim. Arif m değerine sahip belgeyi herkese açık bir şekilde Buse'ye gönderir.

Buse, sadece kendisinin bileceği, çok büyük p ve q asal sayılarını seçer ve $n = p.q$ ile $\varphi(n) = (p-1)(q-1)$ sayılarını hesaplar. Ardından $1 < B < \varphi(n)$ ve $(B, \varphi(n)) = 1$ olacak şekilde bir B sayısı seçer. Daha sonra $B.B' \equiv 1 \pmod{\varphi(n)}$ denkliğini veren B' değerini hesaplar. Son olarak B ve n sayılarını herkese açık bir şekilde Arif'e gönderir. Arif de dahil olmak üzere ortamda B ve n sayılarını bilen hiç kimse, ayrık logaritma ve çarpanlara ayırma problemleri sebebiyle, $p, q, \varphi(n), B'$ sayılarını hesaplayamaz. Tüm bu işlemler sonucunda Buse'nin elektronik imzası $y \equiv m^{B'} \pmod{n}$ değeri olarak belirlenir. Sonuç olarak Arif'in imzalatmak istediği m değerindeki belge ve Buse'nin imzası olan y değeri herkese açık bir şekilde görünmektedir. Böylece imzalama olayı gerçekleşmiştir. Buse'nin yerine kimse y değerini hesaplayamaz, çünkü B' değerini sadece Buse biliyor. Dolayısıyla ortamdaki birisi m belgesini farklı bir y' sayısı kullanarak imzalamaya kalkarsa Buse rahatlıkla imzanın kendisine ait olmadığını kanıtlayabilir.

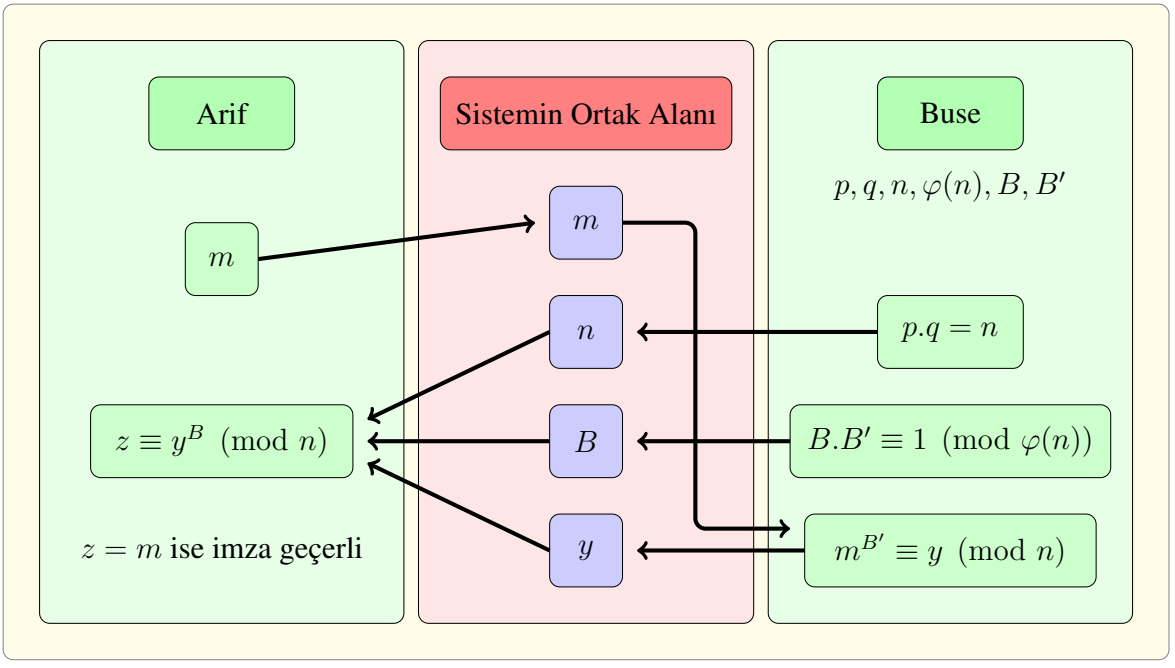
Arif, Buse'den aldığı y ve B değerlerini kullanarak $z \equiv y^B \pmod{n}$ değerini hesaplar.

$$z \equiv y^B \equiv (m^{B'})^B \equiv m^{B'.B} \pmod{n}$$

ve $B.B' \equiv 1 \pmod{\varphi(n)}$ olduğundan, eğer Arif işlem sonucunda $z = m$ eşitliğine ulaşıyorsa imzanın geçerli olduğunu kabul eder, aksi halde imza geçersizdir.

Eğer Buse, Arif'in gönderdiği belgenin sayısal değeri olan m'den farklı bir m' değerindeki belgeyi imzalamak isterse, Arif'in yaptığı işlem sonucunda $z \neq m'$ sonucuna ulaşılır. Çünkü $(m')^{B'}$ ile $m^{B'}$ değerleri $\pmod{n}$'e göre denk değildirler. Dolayısıyla $y^B \not\equiv ((m')^{B'})^B \pmod{n}$'dir.

Not : Eğer Arif, Buse'ye imzalatmak istediği belgenin sadece Buse tarafından bilinmesini isterse, yani imzalama işlemini gizli yaptırmak isterse belgeyi önce RSA şifreleme algoritmasını kullanarak şifreli bir şekilde gönderebilir ve bu işlemin ardından RSA imzalama algoritmasını kullanarak imzalatabilir.



Şekil 3.23. RSA elektronik imza algoritması

Örnek 3.4.15. Arif'in Buseye imzalatmak istediği belgenin sayısal değeri $m = 114$ olsun. Buse de imzalama yaparken kullanacağı asal sayıları $p = 881$ ve $q = 1151$ olarak belirlesin. (Örnekteki amacımız kriptografik anlamda güvenli bir imzalama yapmak değil, algoritmanın daha iyi anlaşılmasını sağlamaktır. Bu sebeple asal sayıları küçük seçtik.) Ardından Buse

$p.q = 881.1151 = 1014031 = n$ ve $\varphi(n) = (p-1).(q-1) = 880.1150 = 1012000$ hesaplamalarını yapar. Daha sonra Buse'nin $1 < B < 1012000$ ve $(B, 1012000) = 1$ olacak şekildeki B sayısını 5421 olarak seçelim. $5421.602981 \equiv \pmod{1012000}$ olduğundan $B' = 602981$'dir. Son olarak Buse

$$y \equiv m^{B'} = 114^{602981} \equiv 981502 \pmod{1014031}$$

hesabını yaparak $n = 1014031, B = 5421$ ve $y = 981502$ sayılarını herkese açık bir şekilde Arif'e gönderir.

Buse'den n, B, y sayılarını alan Arif

$$z \equiv y^B = 981502^{5421} \equiv 114 \pmod{1014031}$$

hesabını yapar ve $z = 114 = m$ eşitliğinin sağlandığını görür. Böylece imzanın geçerli olduğu kararını verir.

3.4.5.2 El-Gamal İmzalama Algoritması

Algoritmayı temel iletişim senaryomuz üzerinden görelim.

Arif, elindeki bir belgeyi Buse'ye imzalatmak istesin ve Buse de imzalamayı kabul etsin. Belgenin sayısal değerine m diyelim. Arif m değerini herkese açık bir şekilde Buse'ye gönderir.

Buse çok büyük bir p asal sayısı ve $\mathbb{Z}_p^*$ cisminin üreteci olan bir α elemanını seçer. Ardından $1 < B < p-2$ olacak şekilde bir B tamsayısı seçer ve $\beta \equiv \alpha^B \pmod{p}$ değerini hesaplar. p, α ve β sayıları ortamdaki herkes tarafından görülebilir. Ayrık logaritma problemi sebebiyle, Arif de dahil olmak üzere ortamda p, α, β sayılarını bilen hiç kimse B sayısını hesaplayamaz. Sayılar belirlendikten sonra Buse $(k, p-1) = 1$ olacak şekilde bir k sayısı seçer ve $r \equiv \alpha^k \pmod{p}$ değerini hesaplar. Son olarak $s \equiv k^{-1}(m - B.r) \pmod{p-1}$ sayısını hesaplar ve m, r, s sayılarını herkese açık bir şekilde Arif'e gönderir.

Arif, Buse'den aldığı p, α, β, r, s sayılarını ve belgenin sayısal değeri olan m sayısını kullanarak

$$v_1 \equiv \beta^r . r^s \pmod{p} \quad \text{ve} \quad v_2 \equiv \alpha^m \pmod{p}$$

hesaplamalarını yapar. Eğer $v_1 = v_2$ eşitliği sağlanıyorsa imza doğrudur, aksi halde imza geçersizdir.

$s \equiv k^{-1}(m - B.r) \pmod{p-1}$ olduğundan $s.k \equiv m - B.r \pmod{p-1}$'dir. Dolayısıyla $m \equiv sk + B.r \pmod{p-1}$'dir. p asal sayısı çok büyük olduğundan $\pmod{p-1}$'e göre eşlenik olan üstel ifadeler $\pmod{p}$'ye göre de eşleniktir. Tüm bu bilgiler ışığında $v_1 = v_2$ iken imzanın neden doğru olduğunu aşağıdaki işlemle görebiliriz.

$$v_2 \equiv \alpha^m \equiv \alpha^{sk+Br} \equiv (\alpha^k)^s . (\alpha^B)^r \equiv \beta^r . r^s \equiv v_1 \pmod{p}$$

B sayısını sadece Buse bildiğinden, ortamdaki hiç kimse $v_1 = v_2$ eşitliği sağlanacak şekilde sayılar belirleyemez, bu sebeple $v_1 = v_2$ eşitliğine ulaşan Arif m değerindeki belgeye imza atan kişinin Buse olduğundan emin olur.

4. KUANTUM-SONRASI KRİPTOGRAFİ

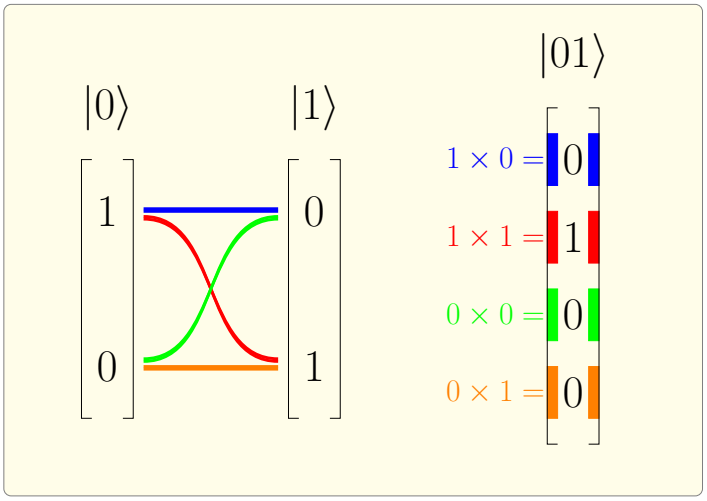
Gelişen teknoloji ile birlikte kuantum bilgisayarlar gündeme gelmiştir. Kuantum bilgisayarları günümüz bilgisayarlarından ayıran en belirgin özellik daha hızlı işlem yapabilmeleridir. Bunun sebebi, günümüz bilgisayarlarının işlem yaparken transistörleri kullanmasıdır. Transistörler sadece iki gerilim seviyesine sahiptir. Bunlar, hepimizin aşina olduğu 0'lar ve 1'lerdir. Bu bilgisayarların aksine, kuantum bilgisayarlarda bitler, "süperpozisyon" adı verilen "hem 0 hem 1" şeklinde tanımlanan bir durumda olabilirler. Kuantum hesaplamayla uğraşan bilim insanları, kuantum bitleri normal bitlerden ayırmak için "kübit" (q-bit) terimini kullanırlar. Kübitler, $|0\rangle$ ve $|1\rangle$ şeklinde gösterilir ve aşağıdaki gibi ifade edilir:

$$0 \text{ Kübiti} : |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad 1 \text{ Kübiti} : |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Buradan da anlaşılacağı üzere bir kübit iki bite karşılık gelmektedir. Benzer şekilde düşünerek iki kübit olduğu durumları aşağıdaki şekilde ifade ederiz:

$$|00\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad |01\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad |10\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad |11\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Bu gösterimler oluşturulurken yapılan işlem dağılma işlemine çok benzemektedir. Örneğin, 0 kübiti ile 1 kübitini birleştirirken 0 kübitinin ilk bileşeniyle 1 kübitinin ilk bileşeni ikilik tabanda çarpılıp sütun matrisinin ilk girdisi olarak yazılır. Ardından 0 kübitinin ilk bileşeniyle 1 kübitinin ikinci bileşeni çarpılıp sütun matrisinin ikinci girdisi olarak yazılır. Benzer şekilde, 0 kübitinin ikinci bileşeni de sırasıyla 1 kübitinin ilk ve ikinci bileşenleriyle çarpılarak sütun matrisinin üçüncü ve dördüncü girdileri olarak yazılır.



Şekil 4.1. İkili kübit oluşturma

Bu işlemden de anlaşılacağı üzere 1 kübit 2 bite, 2 kübit 4 bite, 3 kübit 8 bite karşılık gelmektedir. Dolayısıyla n tane kübit 2^n tane bite karşılık gelmektedir. Buradan şu çıkarımı yapabiliriz: Bir klasik bilgisayarın belirli bir zamanda işleyebildiği veri sayısına A dersek, bir kuantum bilgisayarın aynı zaman aralığında işleyebileceği veri sayısı 2^A kadardır. Böylece, kuantum bilgisayarların neden çok daha hızlı olduğu sorusuna da cevap vermiş oluruz. Burada, şunu unutmamak gerekir: Günümüzde kullanılan klasik bilgisayarların bit okuma hızı ile kuantum bilgisayarların kübit okuma hızı aynı değildir. Çünkü kuantum bilgisayarlar henüz bebeklik dönemlerini yaşamaktadırlar. Dolayısıyla, yukarıda yaptığımız hesabı, birim zamanda bit ve kübit okuma hızlarının aynı olduğunu varsayarak yaptık.

Kuantum bilgisayarların daha hızlı olmasının diğer bir sebebini de şu şekilde açıklayabiliriz: Bir verinin bizim gözümüzdeki görüntüsü ne olursa olsun (yani metin, fotoğraf, video, oyun, sosyal uygulamalar vs.) klasik bilgisayarların gözündeki görüntü sadece 0 ve 1'lerden oluşur. Verinin büyüklüğüne göre 0 ve 1'lerin sayısı artmaktadır. Yani tüm veriler, klasik bilgisayarların gözünde değişen uzunluklara sahip 0-1 dizisi olarak görülür. Klasik bilgisayarlar da bu diziyi en baştan en sona kadar tek tek okuyarak işlem gerçekleştirirler. Bilgisayarın işlemci gücüne bağlı olarak bu okuma hızı değişse de tek tek okuma olayı daima aynıdır. Bu sebeple, büyük verilerde veya karmaşık problemlerin çözümünde klasik bilgisayarlar çok uzun zamanda yanıt vermektedirler. Klasik bilgisayarların aksine kuantum bilgisayarlar ise

tek tek okuma yerine aynı anda birden fazla veriyi işleme olanağı sunmaktadır. Yani aynı işlemi çok daha kısa zamanda halledebilirler.

Meselenin özünün daha iyi anlaşılabilmesi için günlük yaşamdan bir örnek verelim: Sadece bizim bildiğimiz bir bilginin var olduğunu düşünelim. Bu bilgiyi bizden kilometrelerce uzakta olan bir yere gidip kullanmamız ve ardından sonuçları alarak tekrar eski yerimize gelip elde edilenleri işlememiz gerektiğini düşünelim. Yani seyahat etmemiz gerekiyor ki bu da zaman harcamamız anlamına gelir. Fakat biz, aynı anda farklı konumlarda bulunabilme yeteneğine sahip olsaydık, bilgiyi kullanmamız gereken konumda ve sonuçları işleyeceğimiz konumda aynı anda bulunabilirdik. Dolayısıyla yerimizden hiç hareket etmeden bilgiyi kullanabilir ve sonuçları anında işleyebilirdik. Klasik bilgisayarlarda bitler, aynı anda hem 0 hem 1 olamazlar, fakat kuantum bilgisayarlarda, biraz önce açıkladığımız gibi, bitler aynı anda hem 0 hem de 1 olabilirler. Özetlemek amacıyla bir benzetme yapmak istersek şunu söyleyebiliriz: Klasik bilgisayarları uçakla seyahat etmek gibi düşünürsek kuantum bilgisayarları ışınlanma olarak görebiliriz.

Kuantum bilgisayarların bu kadar hızlı olması çoğu bilim insanını heyecanlandırsa da çok sayıda bilim insanını da endişelendirmektedir. Örneğin klasik kriptografide bahsettiğimiz güvenli algoritmaların neredeyse tamamı matematikte çözülmesi imkansız veya çok zaman alan problemlere dayanmaktadır. Şu anda hepimizin bilgisayarlarında güvenliğimiz ve gizliliğimiz, temelinde bu problemlerle sağlanmaktadır. Kuantum bilgisayarlar bu problemlerden birkaçının çözümüne kısa zamanda ulaşabilmektedir. Kalan problemleri çözmeleri de an meselesidir. Bu sebeple klasik kriptosistemler kuantum bilgisayarların hayatımıza girmesiyle birlikte güvenliklerini kaybedeceklerdir. Dolayısıyla kuantum bilgisayarlara karşı da dayanıklı olabilecek yeni kriptosistemler üretmek zorundayız. Bu konuda halihazırda birçok algoritma yapılmış durumdadır. Bu algoritmaların bir kısmı sıfırdan, bir kısmı da klasik kriptosistemlerin geliştirilmesiyle oluşturulmuştur. Kuantum algoritmalar kullandıkları temele göre sınıflandırılmaktadır. Tezimizin başlığında da belirttiğimiz gibi, algoritma sınıflarından kod-tabanlı ve kafes-tabanlı olanları inceleyeceğiz.

4.1 Kod-Tabanlı Kriptografi

Kod-tabanlı kriptografi, adından da anlaşılacağı üzere, temelinde kodları ve kod yapılarını kullanan kriptografik sistemlerin bütünüdür. Bu kriptografi sınıfı hem içerisinde barındırdıkları işlem süreçleri hem kullandıkları kodların sağlamlıkları hem de birçok denemeye rağmen ataklara karşı gösterdikleri direnç bakımından güvenilirliklerini belirli bir seviyenin üzerinde tutabilen algoritmaları barındırır. NIST (National Institute of Standards and Technology) tarafından açılan ve son tarihi kasım 2017 olan yarışma çağrısında ön elemeyi geçerek adaylığa yükselebilen 69 algoritmadan 20 tanesi kod-tabanlı algoritmalardır. Bu bilgi, (yarışmanın dünya çapında olduğu da göz önünde bulundurularak) algoritmaların güvenilirliklerinin uluslararası alanda kanıtlandığını gösterir. Bu bölümde belirli bir güvenlik seviyesinin üzerinde performans gösterebilen kod-tabanlı algoritmaları anlatacağız.

Algoritmaların kullandıkları kod ailelerini tekrar tekrar yazmamak için algoritmalara geçmeden önce kullanılan kod ailelerini anlatalım.

4.1.1 Kod Aileleri

Bu bölümde kod-tabanlı kuantum-sonrası algoritmaların temelinde kullanılan kod aileleri anlatılmıştır.

4.1.1.1 QC-MDPC Kodlar

Bu kısımda QC-MDPC kod ailesi anlatılmıştır ve anlatım yapılırken [19] numaralı kaynaktan faydalanılmıştır. QC-MDPC kod ailesinin tam adı "yarı devirli orta yoğunluklu denklik kontrol (quasi cyclic moderate density parity check) kodları"dır. QC-MDPC kodlarından önce devirli (cyclic) ve yarı devirli (quasi cyclic) kodların ne olduğunu görelim.

Tanım 4.1.1. (Devirli Kodlar) C bir lineer kod ve G matrisi de C kodunun üreteç matrisi olsun. Eğer G matrisi bir kare matris ve her bir satırı, bir üst satırın girdilerinin bir birim sağa kaydırılmasıyla oluşuyorsa C koduna devirli kod, G matrisine de devirli (döngüsel) matris denir.

Örnek 4.1.2. Aşağıda verilen matrisler devirli matrisler ve bu matrislerle üretilen kodlar da devirli kodlardır.

$$G_1 = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 2 & 3 \\ 3 & 4 & 1 & 2 \\ 2 & 3 & 4 & 1 \end{bmatrix} \qquad G_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad G_3 = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

Tanımdan ve örnekten de anlaşılacağı üzere devirli bir matrisin ilk satırını biliyorsak matrisin geri kalanını biz tamamlayabiliriz. Böylelikle kullanılan anahtarlar devirli kodlar olduğunda iletilmesi gereken veri boyutunun azalması yönünde bir fayda sağlanır. Çünkü bir matris devirli değilse karşı tarafa matrisin tamamını göndermemiz gerekir, fakat devirli bir matrisi göndermek istediğimizde sadece ilk satırı göndermemiz yeterlidir.

Tanım 4.1.3. (Yarı Devirli Kodlar) C bir lineer kod ve G matrisi de C kodunun üreteç matrisi olsun. Eğer G matrisi birden fazla devirli matrisin yan yana ve alt alta eklenmesiyle oluşuyorsa G matrisine yarı devirli matris ve G matrisinin ürettiği C koduna da yarı devirli kod denir. Özel olarak, eğer bir yarı devirli G matrisi $u.v$ tane aynı formdaki devirli matrisin u kadarının yan yana, v kadarının alt alta birleşmesiyle oluşuyorsa bu matrise (u, v)-yarı devirli matrisi denir. Eğer bir G yarı devirli matrisi sadece u tane aynı formdaki devirli matrislerin yan yana eklenmesiyle oluştuysa bu matrise kısaca u-yarı devirli matrisi denir.

Tanım 4.1.4. (Blok Uzunluğu) $k \times n$ formunda bir G yarı devirli matrisi yan yana u tane ve alt alta v tane devirli matrisin birleşmesiyle oluşuyorsa bu matrisin blok uzunluğu $\frac{n}{u}$ veya $\frac{k}{v}$ ile hesaplanır. Açıkça görüleceği üzere $\frac{n}{u} = \frac{k}{v}$'dir.

Örnek 4.1.5. Aşağıda verilen matrisler yarı devirlidir.

$$G_1 = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \end{bmatrix} = \left[\begin{array}{cc|cc} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \end{array}\right]$$

$$G_2 = \left[\begin{array}{cccccc} 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{ccc|ccc} 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

$$G_3 = \left[\begin{array}{cccccccc} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{cccc|cccc} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

Bu matrislerin her biri, 2 tane devirli matrisin yan yana birleşmesiyle oluştuğu için $(2, 1)$-yarı devirli (veya kısaca 2-yarı devirli) matrisler olarak adlandırılırlar. Aşağıda verilen matrisler ise 3 tane devirli matrisin birleşmesiyle oluştuğu için $(3, 1)$ yarı devirli (veya kısaca 3-yarı devirli) matrisler şeklinde adlandırılırlar. Benzer şekilde n tane devirli matrisin birleşmesiyle oluşan matrisler de $(n, 1)$ yarı devirli matrisler şeklinde isimlendirilirler.

$$G_4 = \left[\begin{array}{cccccc} 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \end{array}\right] = \left[\begin{array}{cc|cc|cc} 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \end{array}\right]$$

$$G_5 = \left[\begin{array}{ccccccccc} 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \end{array}\right] = \left[\begin{array}{ccc|ccc|ccc} 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \end{array}\right]$$

Devirli matrislerin ilk satırını bildiğimiz zaman matrisin geri kalanını tamamlayabileceğimizi söylemiştik. Buradan hareketle, hem işlemlerde hem de yazımda kolaylık sağlayan, r terimli polinomlar ile $r \times r$ formundaki devirli matrislerin ilk satırlarını eşleştiren bir dönüşüme sahibiz. Bu dönüşüm doğal halka izomorfizmasıdır ve aşağıdaki şekilde tanımlanır.

$$\begin{array}{rccc} \varphi & : & \mathbb{F}_2^r & \to & \mathbb{F}_2[x]/(x^r - 1) \\ & & a_0 \dots a_{r-1} & \mapsto & a_0 + \cdots + a_{r-1}x^{r-1} \end{array}$$

Açıkça görüleceği üzere, bu dönüşüm bize aşağıdaki eşleşmeyi vermektedir.

$$a_0 + a_1x + \cdots + a_{r-1}x^{r-1} \quad \equiv \quad \begin{bmatrix} a_0 & a_1 & \cdots & a_{r-2} & a_{r-1} \\ a_{r-1} & a_0 & \cdots & a_{r-3} & a_{r-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_2 & a_3 & \cdots & a_0 & a_1 \\ a_1 & a_2 & \cdots & a_{r-1} & a_0 \end{bmatrix}$$

Tanımladığımız φ dönüşümüne göre, bir $a = a_0 + a_1x + \cdots + a_{r-1}x^{r-1}$ polinomu ile eşleştirilen matrisin transpozu $a^\top = a_0 + a_{r-1}x + a_{r-2}x^2 + \cdots + a_2x^{r-2} + a_1x^{r-1}$ polinomunun ifade ettiği matristir. Yukarıda gösterdiğimiz polinom ve matris eşleştirmesinden açıkça görülmektedir. Buradan hareketle, bir G devirli matrisinin ilk satırını g ile gösterirsek, $\varphi(g^\top) = (\varphi(g))^\top$ eşitliğini elde etmiş oluruz. Eğer $\varphi(g)$ gösterimi yerine $\varphi(G)$ gösterimini kullanırsak φ dönüşümünü $\forall v \in \mathbb{F}_2^r$ vektörü üzerine $\varphi(v) = v_0 + v_1x + \cdots + v_{r-1}x^{r-1}$ şeklinde genişleterek kolayca $\varphi(G.v^\top) = \varphi(G).\varphi(v)^\top$ eşitliğinin sağlandığını görürüz. Bu da bize bir vektör ile devirli matrisin çarpımını kolay yoldan nasıl yapacağımızı gösterir.

Yarı devirli matrisler de tıpkı devirli matrisler gibi polinomlarla ifade edilebilirler. Örneğin, elimizde 5×15 formunda bir G matrisinin polinom temsilcisi olan $a_0 + a_1x + \cdots + a_{14}x^{14}$ polinomu var olsun. Matrisin boyutlarına baktığımızda ($15/5 = 3$ olduğundan) bir $(3, 1)$ yarı devirli matris olduğunu anlıyoruz. Polinomun katsayılarını matrisin ilk satırına yazıp, 5×5 formunda üç tane devirli matris olacak şekilde girdileri kaydırarak matrisi tamamlayabiliriz. Böylece aşağıdaki matrisi elde ederiz.

$$G = \left[\begin{array}{ccccc|ccccc|ccccc} a_0 & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_8 & a_9 & a_{10} & a_{11} & a_{12} & a_{13} & a_{14} \\ a_4 & a_0 & a_1 & a_2 & a_3 & a_9 & a_5 & a_6 & a_7 & a_8 & a_{14} & a_{10} & a_{11} & a_{12} & a_{13} \\ a_3 & a_4 & a_0 & a_1 & a_2 & a_8 & a_9 & a_5 & a_6 & a_7 & a_{13} & a_{14} & a_{10} & a_{11} & a_{12} \\ a_2 & a_3 & a_4 & a_0 & a_1 & a_7 & a_8 & a_9 & a_5 & a_6 & a_{12} & a_{13} & a_{14} & a_{10} & a_{11} \\ a_1 & a_2 & a_3 & a_4 & a_0 & a_6 & a_7 & a_8 & a_9 & a_5 & a_{11} & a_{12} & a_{13} & a_{14} & a_{10} \end{array}\right]_{5\times15}$$

Şimdiye kadar öğrendiklerimizden şu çıkarımı rahatça yapabiliriz: Bir $(u, 1)$ yarı devirli matrisinin sütun sayısı, u sayısının bir tam katı olmalıdır. Dolayısıyla G'nin ürettiği yarı devirli kodun kelimelerinin uzunluğu da aynı şekilde u sayısının bir tam katı olmalıdır.

Son olarak şunu da belirtelim: Bir C yarı devirli kodunun denklik kontrol matrisi olan H matrisi de yarı devirli bir matristir. Tek bir fark vardır; C kodunun matrisi $(u, 1)$ yarı devirli matrisi iken, H matrisi $(u-1, u)$ yarı devirli matristir. Yani H matrisi, yan yana $u-1$, alt alta u tane devirli matrisin birleşmesiyle oluşur. Örneğin, G_1, G_2, G_3 birer devirli matris olmak üzere

$$G = \left[\begin{array}{c|c|c} G_1 & G_2 & G_3 \end{array}\right]$$

şeklinde bir matris ise $H_1, H_2, \ldots, H_6$ devirli matrisler olmak üzere H matrisi de

$$H = \left[\begin{array}{c|c} H_1 & H_4 \\ \hline H_2 & H_5 \\ \hline H_3 & H_6 \end{array}\right]$$

şeklinde bir matristir.

Devirli ve yarı devirli kodları öğrendiğimize göre artık QC-MDPC kodlara geçebiliriz.

Tanım 4.1.6. (QC-MDPC Kodları) QC-MDPC kodlar 4 parametre ile, "(n', k', r, w) QC-MDPC kod" şeklinde ifade edilirler. Burada n' ve k' parametreleri, kullanılan C kodunun bir (n', k') yarı devirli kod olduğunu belirtir. Bu da C kodunun üreteç matrisi olan G yarı devirli matrisinin yan yana n' tane ve alt alta k' tane devirli matrisin birleşmesiyle oluştuğunu söyler. Yani $\forall\ 0 \leq i \leq n'$ ve $\forall\ 0 \leq j \leq k'$ için G_{ij} matrisleri devirli matrisler olmak üzere C kodunun üreteç matrisi aşağıdaki şekildedir:

$$G = \begin{bmatrix} G_{11} & G_{12} & \cdots & G_{1n'} \\ G_{21} & G_{22} & \cdots & G_{2n'} \\ \vdots & \vdots & \ddots & \vdots \\ G_{k'1} & G_{k'2} & \cdots & G_{k'n'} \end{bmatrix}$$

Diğer yandan, r sayısı ise G_{ij} devirli matrislerinin satır ve sütun sayılarını ifade eder. (Devirli matrisler aynı zamanda birer kare matris olduğundan satır ve sütun sayıları aynıdır. Yani $\forall\ G_{ij}$ matrisi $r \times r$ formundadır) Buradan hareketle C kodunun uzunluğunu $n = r.n'$ ve boyutunu da $k = r.k'$ şeklinde hesaplarız. Tanımdaki w sayısı ise C kodunun denklik kontrol matrisi olan H matrisinin girdilerinden oluşan vektör uzayının $\mathbb{F}$ cismi içerisindeki tabanının eleman sayısını ifade eder ve H matrisinin satır ağırlığı olarak adlandırılır. Bir C yarı devirli kodunun QC-MDPC kod olabilmesi için $w \approx \sqrt{n}$ olması gerekir. Buradan da anlayacağımız üzere her yarı devirli kod, bir QC-MDPC kod olamayabilir.

QC-MDPC kodlarda kod uzunluğu genellikle çok yüksektir. Yaklaşık olarak $10000 - 100000$ aralığındadır.

4.1.1.2 QC-LRPC Kodlar

Bu bölümde QC-LRPC kod ailesi anlatılmıştır ve anlatım yapılırken [20] numaralı kaynaktan faydalanılmıştır. QC-LRPC kodlarının tam adı "yarı devirli düşük ranklı denklik kontrol" (quasi cyclic low rank parity check) kodlarıdır.

QC-LRPC kodları tanımlamadan önce rank metriğini ve rank metrik kodları (veya kısaca rank kodları) tanımlayalım. Rank metrik kodlar iki şekilde ifade edilebilirler: Birincisi matrisin rankı üzerinden, ikincisi vektörlerin rankı üzerindendir. Önce bildiğimiz matris rankı (Tanım 2.5.38) üzerinden olanını tanımlayalım, ardından bir vektörün rankının nasıl hesaplandığını görelim.

Tanım 4.1.7. (Rank Metriği, Rank Kodlar) Girdilerini $\mathbb{F}_q$ sonlu cisminden alan $m \times n$ formundaki tüm matrislerin kümesini ($\mathbb{F}_q^{m \times n}$'yi) $\mathcal{M}$ ile gösterelim. $\mathcal{M}$ kümesinin bir vektör uzayı yapısı belirttiğini biliyoruz. $\mathcal{M}$ uzayının altuzaylarına rank metrik kodlar denir. Bir $M \in \mathcal{M}$ elemanının rankı, bilinen matris rankıdır. $\mathcal{M}$ uzayından aldığımız iki matrisin mesafesi ise bu iki matrisin farklarının rankı olarak tanımlanır ve rank metriği olarak adlandırılır. Bilinen rank gösterimi olan Rk ile gösterilir.

$$\forall\ A, B \in \mathcal{M} \quad \text{için} \quad Rk(A, B) = Rk(A - B)$$

Tanım 4.1.8. (Rank Kodların Minimum Uzaklığı) $\mathcal{A} \subseteq \mathcal{M}$ altuzayını alalım. $\mathcal{A}$ kodunun minimum uzaklığı aşağıdaki şekilde tanımlanır.

$$d(\mathcal{A}) = \min\left\{Rk(M_i, M_j) \mid M_i, M_j \in \mathcal{A}, i \neq j\right\}$$

Şimdi rankın vektörler üzerinden nasıl ifade edildiğini görelim.

Tanım 4.1.9. (Vektör Rankı) $\mathbb{F}_{q^m}$ sonlu cisminin ($\mathbb{F}_q$ üzerinde bir vektör uzayı olarak) bir tabanı $\{\beta_1, \beta_2, \ldots, \beta_m\}$ olsun. O halde $\forall\ v = (v_1, v_2, \ldots, v_n) \in \mathbb{F}_{q^m}^n$ vektörünün girdilerini bu taban elemanlarıyla ifade edebiliriz. $\forall\ 1 \leq i \leq n, 1 \leq j \leq m$ için $c_{ji} \in \mathbb{F}_q$ olmak üzere

$$v_i = c_{1i}\beta_1 + c_{2i}\beta_2 + \cdots + c_{mi}\beta_m = \sum_{\substack{1\leq i\leq n\\ 1\leq j\leq m}} c_{ji}\beta_j.$$

v vektörünün tüm bileşenlerini bu şekilde yazdıktan sonra katsayıları kullanarak $m \times n$ formundaki $V = [c_{ji}]$ matrisi oluşturulur. v vektörünün rankı V'nin matris rankı olarak tanımlanır. Yani $Rk(v) = Rk(V)$'dir.

Not : Bu tezde rank metriği olarak vektörler üzerinden ifade edilen tanımı kullanacağız.

Rank metriğini kullanan kodlar ile Hamming uzaklığını kullanan kodlar arasında bir de dayanak kümesi farkı vardır.

Tanım 4.1.10. (Bir Vektörün Dayanağı) Bir $v = (v_1, v_2, \ldots, v_n) \in \mathbb{F}_{q^m}^n$ vektörünü alalım. $\mathbb{F}_{q^m}$ uzayının v vektörünün bileşenleriyle üretilen altuzayına ($\mathbb{F}_q$ üzerinde bir vektör uzayı) v vektörünün dayanak kümesi denir. Genellikle E ile gösterilir.

$$E = \langle v_1, v_2, \ldots, v_n\rangle \subseteq \mathbb{F}_{q^m}$$

Tanım 4.1.11. (Düşük Ranklı Denklik Kontrol (LRPC) Kodları) Bir $C \subseteq \mathbb{F}_{q^m}$ lineer kodunun rankı d, uzunluğu n ve boyutu k olsun. O halde C'nin denklik kontrol matrisi olan H matrisi $(n-k) \times n$ formundadır. Eğer H matrisinin girdilerinden üretilen ($\mathbb{F}_{q^m}$ üzerinde) altuzayın boyutu en fazla d ise C koduna düşük ranklı denklik kontrol (LRPC)

kodu denir. Genellikle, $H = [h_{ij}]$ matrisinin girdilerinden üretilen altuzay $\mathcal{F}$ ile, tabanı da $\{F_1, F_2, \ldots, F_d\}$ ile gösterilir. $\mathcal{F}$ altuzayının boyutuna H denklik kontrol matrisinin ağırlığı denir.

Aşağıda verilen tanım LRPC kodların özel bir halidir.

Tanım 4.1.12. (QC-LRPC Kodlar) Eğer C kodu bir LRPC kod ve C'nin denklik kontrol matrisi olan H bir yarı devirli matris (Tanım 4.1.3) ise C'ye yarı devirli düşük ranklı denklik kontrol (quasi cyclic low rank parity check) kodu denir.

QC-LRPC kodlar özel olarak H matrisinin devir sayısına göre isimlendirilir. Örneğin H 3-devirli bir matris ise H denklik kontrol matrisine sahip olan C koduna "3-yarı devirli düşük ranklı denklik kontrol kodu" denir.

4.1.1.2.1 LRPC Kodlarla Kodlama

C, $\mathbb{F}_{q^m}$ cismi üzerinde bir LRPC kod ve G de bu kodun üreteç matrisi olsun. C kodunun uzunluğu n ve boyutu k olsun. Bu durumda G matrisi $k \times n$ formundadır.

k uzunluğunda bir m mesajı, $mG = c$ işlemiyle kodlanır ve işlem sonucunda elde edilen c kod sözcüğü alıcıya iletilir.

4.1.1.2.2 LRPC Kodlarla Kod Çözme

Kodlanarak alıcıya gönderilen c kod sözcüğünün bir e hatası ile birlikte iletildiğini ve y halini aldığını varsayalım. Yani $c+e = y$. Dolayısıyla kod çözme algoritmasının amacı e vektörünü bularak $y - e$ işlemi ile birlikte c sözcüğüne ulaşmaktır.

LRPC kodların kod çözme algoritmasındaki temel fikir, H matrisinden bilinen $\mathcal{F}$'nin tabanı ve sendrom vektörünün bileşenlerinden üretilen S uzayından hataların dayanak kümesi E'yi kurtarmaktır. Eğer E elde edilirse, e hata vektörünün tam koordinatları, bir lineer denklem sistemi çözülerek kolaylıkla bulunabilir.

$H = [h_{ij}] \subseteq \mathbb{F}_{q^m}^n$ bir LRPC kodun denklik kontrol matrisi olsun. Tanım 4.1.11'dan bildiğimiz üzere her bir h_{ij} girdisi $\mathcal{F}$ uzayına aittir. $\mathcal{F}$ uzayının tabanı $\{F_1, F_2, \ldots, F_d\}$ olduğundan h_{ij} girdilerini taban elemanlarının lineer kombinasyonu olarak yazabiliriz.

$$h_{ij} = \sum_{v=1}^{d} h_{ijv} F_v$$

İletilen mesajdaki $e = (e_1, e_2, \ldots, e_n)$ vektörü, E uzayında bulunan r ranklı bir hata vektörüdür. E uzayının bir tabanı olarak $\{E_1, E_2, \ldots, E_r\}$ alınabilir ve e vektörünün bileşenleri, bu taban elemanlarının bir lineer kombinasyonu şeklinde yazılabilir.

$$e_j = \sum_{u=1}^{r} e_{ju} E_u$$

$e' = (e_{11}, e_{12}, \ldots, e_{1r}, e_{21}, e_{22}, \ldots, e_{2r}, \ldots, e_{n1}, e_{n2}, \ldots, e_{nr})$ olsun.
$\{F_1E_1, \ldots, F_1E_r, F_2E_1, \ldots, F_2E_r \ldots, F_dE_1, \ldots, F_dE_r\}$ tarafından üretilen çarpım uzayını da $\langle F.E \rangle$ ile gösterelim. $dim(\langle F.E \rangle) = d.r$ olduğunu varsayalım.

$H.e^\top = s^\top$ sendrom eşitliğini $\mathbb{F}_q$ taban cismi üzerinde bir eşitlik olarak dönüştüreceğiz. H matrisi $(n-k) \times n$ ve $e^\top$ vektörü $n \times 1$ formunda olduğundan $s^\top$ vektörü $(n-k) \times 1$ formundadır. Dolayısıyla $s = (s_1, s_2, \ldots, s_{n-k})$ yazabiliriz. s'nin bileşenlerini çarpım uzayının taban elemanlarının bir lineer kombinasyonu olarak yazalım.

$$s_i = \sum_{v=1}^{d} \sum_{u=1}^{d} s_{ivu} F_v E_u$$

s vektörünün tüm bileşenleri için bunu yaparak s vektörünün $\mathbb{F}_q$ cisminin elemanları cinsinden yazımını elde ederiz. Bu yeni yazıma s' diyelim.

$$s' = (s_{111}, \ldots, s_{11r}, s_{121}, \ldots, s_{12r}, \ldots, s_{1d1}, \ldots, s_{1dr}, \ldots, s_{(n-k)11}, \ldots s_{(n-k)dr})^\top$$

Böylelikle sendrom denklemini $\mathbb{F}_q$ üzerinde yazarak aşağıdaki şekle getirmiş oluruz:

$$\sum_{j=1}^{n}\sum_{v=1}^{d}\sum_{u=1}^{r} h_{ijv}e_{ju}F_vE_u = \sum_{u=1}^{r}\sum_{v=1}^{d} s_{ivu}F_vE_u$$

$1 \leq v \leq d$ ve $1 \leq u \leq r$ için F_vE_u çarpanlarının katsayılarını eşitlediğimizde aşağıdaki eşitliği elde ederiz:

$$\sum_{j=1}^{r} h_{ijv}e_{ju} = s_{ivu}$$

Bu denklem sisteminin katsayılar matrisine $A = [a_{ij}]$ dersek, A matrisi $(n-k)rd \times nr$ formunda olur ve denklem sistemini A matrisini kullanarak $A.(e')^\top = s'$ şeklinde yazarız. Ayrıca A matrisinin girdilerinin bir kısmı h_{ijv}'lerden ve bir kısmı da 0'lardan oluşur. Daha açık belirtelim: $1 \leq v \leq d, 1 \leq u \leq r, 1 \leq i \leq n-k$ ve $1 \leq j \leq n$ için

$$a_{u+(v-1)r+(i-1)rd,u+(j-1)r} = h_{ijv}$$

olarak yazılır ve geri kalan girdiler 0 olur. Böylelikle A matrisi aşağıdaki şekilde olur:

$$A = \left[\begin{array}{c|c|c|c}
\begin{matrix} h_{111} & & \\ & \ddots & \\ & & h_{111} \end{matrix} & \begin{matrix} h_{121} & & \\ & \ddots & \\ & & h_{121} \end{matrix} & \cdots & \begin{matrix} h_{1n1} & & \\ & \ddots & \\ & & h_{1n1} \end{matrix} \\ \hline
\begin{matrix} h_{112} & & \\ & \ddots & \\ & & h_{112} \end{matrix} & \begin{matrix} h_{122} & & \\ & \ddots & \\ & & h_{122} \end{matrix} & \cdots & \begin{matrix} h_{1n2} & & \\ & \ddots & \\ & & h_{1n2} \end{matrix} \\ \hline
\vdots & \vdots & \ddots & \vdots \\ \hline
\begin{matrix} h_{(n-k)1d} & & \\ & \ddots & \\ & & h_{(n-k)1d} \end{matrix} & \begin{matrix} h_{(n-k)2d} & & \\ & \ddots & \\ & & h_{(n-k)2d} \end{matrix} & \cdots & \begin{matrix} h_{(n-k)nd} & & \\ & \ddots & \\ & & h_{(n-k)nd} \end{matrix}
\end{array}\right]$$

Bu denklem sistemi nr bilinmeyenli $(n-k)rd$ denklemden oluşur. Bu denklem sistemini çözmek, bize e vektörünün sıfırdan farklı koordinatlarını verir.

Kod çözme algoritmasının işlem süreci biraz karışık olabilir. Bu yüzden algoritmanın daha anlaşılır olması için aşağıdaki şekilde özetleyelim:

1. **Sendrom Uzayı Bulma**

   $H.y^\top = s^\top = (s_1, s_2, \ldots, s_{n-k})^\top$ sendrom vektörü bulunur ve bu vektörün bileşenlerinin ürettiği $S = \langle s_1, s_2, \ldots, s_{n-k} \rangle$ sendrom uzayı bulunur.

2. **Hataların Dayanak Kümesi E'yi Kurtarma**

   S'nin tüm üreteçleri f_i^{-1} ile çarpılarak $S_i = f_i^{-1}S$ altuzayları bulunur. Ardından $E = S_1 \cup S_2 \cup \cdots \cup S_d$ hata kümesinin dayanağı hesaplanır ve E için bir $\{E_1, E_2, \ldots, E_r\}$ tabanı bulunur.

3. **e Hata Vektörünü Kurtarma**

   e hata vektörünün her bir e_j bileşeni, E'nin taban elemanlarının lineer kombinasyonu şeklinde yazılır.

   $$e_j = \sum_{i=1}^{n} e_{ji} E_i$$

   $1 \leq i \leq n-k$ için her bir s_i sendrom vektörleri $F.E$ çarpım uzayının taban elemanlarının lineer kombinasyonu biçiminde yazılır ve ardından $He^\top = s^\top$ denklem sistemi çözülür. (Bu denklem sisteminde nr bilinmeyenli $(n-k)rd$ tane denklem vardır.) Böylelikle e hata vektörü elde edilir.

4. **Asıl Mesajı Elde Etme**

   $xG = y - e$ sistemi çözülerek x mesajı elde edilir.

4.1.1.3 Goppa Kodlar

Bu bölümde Goppa kodlar anlatılmıştır ve bu anlatım yapılırken [21] numaralı kaynaktan yararlanılmıştır.

Bir Sovyet ve Rus Matematikçi olan Valery Denisovich Goppa, Riemann-Roch teoreminden yola çıkarak 1970 yılında, cebirsel geometri ile kodlama teorisi arasında bir köprü inşa etmiştir. Bu fikir, Goppa kodlarına, günümüzde bilinen adıyla, cebirsel-geometrik kodlara kadar ilerlemiştir.

Tanım 4.1.13. (Goppa Kodlar) $g(z) = g_0 + g_1 z + g_2 z^2 + \cdots + g_t z^t$ polinomu $\mathbb{F}_{q^m}$ cismi üzerinde t dereceli bir polinom ve $\forall\ \alpha_i$ için $g(\alpha) \neq 0$ olacak şekilde $L = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ bir nokta kümesi olsun.

$$\left\{ c = (c_1, c_2, \ldots, c_n) \in \mathbb{F}_q^n \mid \sum_{i=1}^{n} \frac{c_i}{z - \alpha_i} \equiv 0 \pmod{g(z)} \right\}$$

şeklinde tanımlanan kümeye $g(z)$ ve L parametreli Goppa kodu denir. Goppa'nın baş harfinin Yunan harflerindeki karşılığı olan "gamma" harfi ve kodun parametreleri kullanılarak $\Gamma(L, g(z))$ ile gösterilir.

$\forall\ 1 \leq i \leq n$ için $g(\alpha_i) \neq 0$ olduğundan $(z - \alpha_i, g(z)) = 1$'dir ve böylece $z - \alpha_i \in L$'nin $\mathbb{F}_{q^m}[z]\big/\langle g(z)\rangle$ bölüm halkasında tersi vardır.

Teorem 4.1.14. $(z - \alpha_i) \in L$'nin $\frac{\mathbb{F}_{q^m}[z]}{\langle g(z)\rangle}$ bölüm halkasındaki çarpımsal tersi $-\left(\frac{g(z)-g(\alpha_i)}{z-\alpha_i}\right) g(\alpha_i)^{-1}$'dir. Bir c vektörünün $\Gamma(L, g)$ kodunun elemanı olabilmesi için gerek ve yeter koşul $\sum_{i=1}^{n} c_i \left(\frac{g(\alpha_i)-g(z)}{z-\alpha_i}\right) g(\alpha^{-1}) \equiv 0 \pmod{g(z)}$ olmasıdır.

Sonuç 4.1.15. Bir c vektörünün $\Gamma(L, g)$ kodunun bir elemanı olması için gerek ve yeter koşul $\sum_{i=1}^{n} c_i \left(\frac{g(\alpha_i)-g(z)}{z-\alpha_i}\right) g(\alpha^{-1})$'nin, $\mathbb{F}_{q^m}[z]$ halkasından bir polinom olarak sıfıra eşit olmasıdır.

Sonuç 4.1.16. Bir $\Gamma(L, g)$ Goppa kodunun $\mathbb{F}_{q^m}$ cismi üzerindeki denklik kontrol matrisi H aşağıdaki gibidir.

$$\begin{bmatrix} g_t g(\alpha_1)^{-1} & g_t g(\alpha_2)^{-1} & \cdots & g_t g(\alpha_n)^{-1} \\ (g_t\alpha_1 + g_{t-1})g(\alpha_1)^{-1} & (g_t\alpha_2 + g_{t-1})g(\alpha_2)^{-1} & \cdots & (g_t\alpha_n + g_{t-1})g(\alpha_n)^{-1} \\ \vdots & \vdots & \ddots & \vdots \\ (g_t\alpha_1^{t-1} + \cdots + g_1)g(\alpha_1)^{-1} & (g_t\alpha_2^{t-1} + \cdots + g_1)g(\alpha_2)^{-1} & \cdots & (g_t\alpha_n^{t-1} + \cdots + g_1)g(\alpha_n)^{-1} \end{bmatrix}$$

H matrisi aşağıdaki gibi üç matrisin çarpımı şeklinde de yazılabilir.

$$\begin{bmatrix} g_t & 0 & 0 & \cdots & 0 \\ g_{t-1} & g_t & 0 & \cdots & 0 \\ g_{t-2} & g_{t-1} & g_t & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_1 & g_2 & g_3 & \cdots & g_t \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \alpha_3 & \cdots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \alpha_3^2 & \cdots & \alpha_n^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{t-1} & \alpha_2^{t-2} & \alpha_3^{t-1} & \cdots & \alpha_n^{t-1} \end{bmatrix} \begin{bmatrix} g(\alpha_1)^{-1} & 0 & \cdots & 0 \\ 0 & g(\alpha_2)^{-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & g(\alpha_n)^{-1} \end{bmatrix}$$

Bu çarpımdaki birinci matrise C, ikinci matrise X ve üçüncü matrise Y diyelim. Yani $H = CXY$ olsun.

Sonuç 4.1.17. Bir c vektörünün $\Gamma(L, g)$ kodunun bir elemanı olması için gerek ve yeter koşul $cH^\top = 0$ olmasıdır. Dolayısıyla $c(CXY)^\top = 0$ olur ve bu da $cY^\top X^\top C^\top = 0$ eşitliğini verir. C matrisi tersinir olduğundan eşitliğin her iki tarafını $(C^\top)^{-1}$ ile çarparak $cY^\top X^\top = 0$ eşitliğini elde ederiz. Buradan, XY matrisi $\Gamma(L, g)$ kodunun denklik kontrol matrisi olarak görülebilir.

$$XY = \begin{bmatrix} g(\alpha_1)^{-1} & g(\alpha_2)^{-1} & \cdots & g(\alpha_n)^{-1} \\ \alpha_1 g(\alpha_1)^{-1} & \alpha_2 g(\alpha_2)^{-2} & \cdots & \alpha_n g(\alpha_n)^{-1} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{t-1} g(\alpha_1)^{-1} & \alpha_2^{t-1} g(\alpha_2)^{-1} & \cdots & \alpha_n^{t-1} g(\alpha_n)^{-1} \end{bmatrix}_{t \times n}$$

$\mathbb{F}_{q^m}$ cisminin elemanlarını $\mathbb{F}_q^m = \mathbb{F}_q \times \cdots \times \mathbb{F}_q$ cisminin elemanlarıyla bire-bir ve örten bir şekilde eşleyebiliriz. Bu eşlemeyi yapıp matrisi tekrar oluşturduğumuzda $\Gamma(L, g)$ kodunun denklik kontrol matrisi $mt \times n$ formunda olur ve bu matrisin satırlarından en az t tanesi $\mathbb{F}_q$ cismi üzerinde lineer bağımsız olur. Dolayısıyla $\Gamma(L, g)$ Goppa kodunun Hamming uzaklığı $d(\Gamma(L, g)) \geq t + 1$ olur. XY matrisinin $\mathbb{F}_q$ cismi üzerinde en fazla mt tane satırı lineer bağımsız olabileceğinden $Rk(XY) \leq mt$ olur ki bu da bize XY'nin dual uzayının (Tanım 2.8.11) boyutunun $n - mt$ den daha büyük veya eşit olduğunu söyler. Bir kodun dualinin duali kendisine eşit (Önerme 2.8.13) olduğundan Goppa kodunun $\mathbb{F}_q$ cismi üzerindeki boyutu $dim(\Gamma(L, g)) \geq n - mt$ olarak bulunur.

Örnek 4.1.18. $x^4 + x + 1$ polinomu $\mathbb{F}_2$ cismi üzerinde, derecesi 4 olan bir primitif (Tanım 2.3.18) polinomdur. Buradan, $\mathbb{F}_{2^4}$ cisminin elemanlarını $\mathbb{F}_2[x]/\langle x^4 + x + 1\rangle$ bölüm halkasının elemanlarıyla bire-bir ve örten bir şekilde eşleyebiliriz. α, $x^4 + x + 1$ polinomunun bir kökü olmak üzere bu eşlemeyi aşağıdaki şekilde yaparak $\mathbb{F}_{2^4}$ cismini oluşturabiliriz.

0		$= (0,0,0,0)$	α^7	$= 1 + \alpha + \alpha^3$	$= (1,1,0,1)$
1		$= (1,0,0,0)$	α^8	$= 1 + \alpha^2$	$= (1,0,1,0)$
α		$= (0,1,0,0)$	α^9	$= \alpha + \alpha^3$	$= (0,1,0,1)$
α^2		$= (0,0,1,0)$	α^{10}	$= 1 + \alpha + \alpha^2$	$= (1,1,1,0)$
α^3		$= (0,0,0,1)$	α^{11}	$= \alpha + \alpha^2 + \alpha^3$	$= (0,1,1,1)$
α^4	$= 1 + \alpha$	$= (1,1,0,0)$	α^{12}	$= 1 + \alpha + \alpha^2 + \alpha^3$	$= (1,1,1,1)$
α^5	$= \alpha + \alpha^2$	$= (0,1,1,0)$	α^{13}	$= 1 + \alpha^2 + \alpha^3$	$= (1,0,1,1)$
α^6	$= \alpha^2 + \alpha^3$	$= (0,0,1,1)$	α^{14}	$= 1 + \alpha^3$	$= (1,0,0,1)$

$g(z) = (z - \alpha)(z + \alpha^{14}) = z^2 + \alpha^7 z + 1$ polinomu ve $L = \{\alpha_i \mid 2 \leq i \leq 13\}$ kümesi ile tanımlanan $\Gamma(L, g)$ Goppa kodunu bulalım.

$\Gamma(L, g)$ kodunun denklik kontrol matrisini bulmak için α değerlerini ve verilen $g(z)$ polinomunu kullanarak XY matrisindeki değerleri hesaplayıp H matrisine ulaşmalıyız. Örneğin, $g(\alpha^2)$'nin $\mathbb{F}_{2^4}$ cismindeki tersi α^{12}'dir. Yani $(0,0,0,1)$ elemanının tersi $(1,1,1,1)$'dir. Bu şekilde her girdiyi tek tek hesaplayarak aşağıdaki matrisi elde ederiz.

$$H = \begin{bmatrix} \alpha^9 & \alpha^{10} & \alpha^9 & \alpha^{14} & \alpha^6 & 0 & \alpha^{10} & \alpha^8 & \alpha^2 & \alpha^7 & \alpha^{14} & \alpha^6 \\ \alpha^{12} & \alpha^6 & \alpha^6 & \alpha & \alpha^{11} & 1 & \alpha^{14} & \alpha^8 & \alpha^{11} & \alpha^{14} & \alpha^{12} & \alpha \end{bmatrix}$$

Matristeki her bir girdiyi eşleme tablomuzdaki karşılıklarıyla yazdığımızda aşağıdaki matrisi elde ederiz.

$$H = \left[\begin{array}{cccccccccccc} 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ \hline 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{array}\right]$$

Son olarak $H.G^{\top} = 0$ eşitliğinden yararlanarak $\Gamma(L, g)$ Goppa kodunun üreteç matrisine ulaşırız.

$$G = \left[\begin{array}{cccccccccccc} 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

4.1.1.3.1 Goppa Kodlarla Kodlama

$der(g(z)) = t$ ve $|L| = n$ olmak üzere $\Gamma(L, g)$ Goppa kodunu alalım. $dim(\Gamma(L, g)) = k$ ve kodun üreteç matrisi $k \times n$ formundaki G matrisi olsun. $\mathbb{F}_q$ üzerinde k uzunluğundaki bir m mesajı $mG = c$ şeklinde kodlanır ve alıcıya gönderilir.

4.1.1.3.2 Goppa Kodlarla Kod Çözme

Alıcı tarafından bir $y = (y_1, y_2, \ldots, y_n)$ mesajı alınmış olsun. Alınan mesajda tam olarak r tane hata olduğunu varsayalım. r sayısı için üst sınır $\frac{d-1}{2}$'dir. Bu durumda alınan y mesajını aşağıdaki şekilde yazabiliriz.

$$y = (y_1, y_2, \ldots, y_n) = (c_1, c_2, \ldots, c_n) + (e_1, e_2, \ldots, e_n)$$

Burada verilen $c = (c_1, c_2, \ldots, c_n)$ vektörü mesajın hatasız hali, $e = (e_1, e_2, \ldots, e_n)$ vektörü de mesajda oluşan hatayı göstermektedir. Buradan anlıyoruz ki, alınan mesajda r tane hata olduğundan e vektörünün tam olarak r tane bileşeni sıfırdan farklıdır. Sıfırdan farklı bileşenlerin sıra numaralarını $B = \{i \mid e_i \neq 0\}$ kümesiyle belirtelim. O halde $|B| = r$'dir.

Eğer e vektörünün hangi bileşenlerinin sıfırdan farklı olduğunu ve bu bileşenlerin yerine hangi değerlerin gelmesi gerektiğini bulabilirsek y mesajındaki hataları düzeltebiliriz. Bunun için önce iki özel polinom tanımlayıp daha sonra kod çözme algoritmasının nasıl çalıştığını göreceğiz.

Tanım 4.1.19. (Hata Polinomları) Hataların konumunu bulmak için "hata konumlandırıcı (error locater) polinom" aşağıdaki gibi tanımlanır. Bu polinomun derecesi r'dir.

$$\sigma(z) = \prod_{i \in B} (z - \alpha_i)$$

Hataların hangi değeri alması gerektiğini bulmak için "hata değerlendirici (error evaluator) polinom" aşağıdaki gibi tanımlanır.

$$\omega(z) = \sum_{i \in B} e_i \prod_{\substack{j \in B \\ j \neq i}} (z - \alpha_j)$$

Önerme 4.1.20. e hata vektörünün ağırlığı (sıfırdan farklı bileşen sayısı) r olsun ve $\sigma(z), \omega(z)$ polinomları Tanım 4.1.19'deki gibi tanımlansın. y'nin sendrom (Tanım 2.8.15) vektörünü de $S(y)$ ile gösterelim. Bu durumda aşağıdakiler sağlanır.

1. $der(\sigma(z)) = r$
2. $der(\omega(z)) \leq r - 1$
3. $(\sigma(z), \omega(z)) = 1$
4. $\forall\, k \in B$ için $e_k = \omega(\alpha_k)/\sigma'(\alpha_k)$, ($\sigma'$: σ'nın türevi)
5. $\sigma(z)S(y) \equiv \omega(z) \pmod{g(z)}$

Kod çözme işleminde ilk adım olarak $S(y)$ hesaplanır.

$$S(y) = \sum_{i=1}^{n} \frac{y_i}{z - \alpha_i}$$

Ardından $\sigma(z)$ ve $\omega(z)$ (Tanım 4.1.19) polinomlarının açık halleri yazılarak aşağıda verilen eşlenik denklem sistemi çözülür. Bu denklem sistemi $2r$ değişkenli t tane denklemden oluşur.
$\sigma(z) = \sigma_0 + \sigma_1 z + \sigma_2 z^2 + \cdots + \sigma_{r-1} z^{r-1} + z^r$
$\omega(z) = \omega_0 + \omega_1 z + \omega_2 z^2 + \cdots + \omega_{r-1} z^{r-1}$

$$\sigma(z)S(y) \equiv \omega(z) \pmod{g(z)}$$

Daha sonra hataların konumlarından oluşan B kümesi belirlenir.

$$B = \{i \mid \sigma(\alpha_i) = 0\}$$

Hataların konumları belirlendikten sonra hataların değerleri aşağıdaki eşitlik ile hesaplanır.

$$\forall\, i \in B \text{ için } e_i = \frac{\omega(\alpha_i)}{\sigma'(\alpha_i)}$$

Son olarak, $y = c + e$ olduğundan $c = y - e$ işlemi yapılarak doğru kod kelimesine (hatasız mesaja) ulaşılır. Böylelikle Goppa kodlarda kod çözme işlemi gerçekleşmiş olur.

Eğer kodlama yapılırken $\mathbb{F}_q$ cismi olarak $\mathbb{F}_2$ kullanılmışsa, kod çözme işlemi için özel bir kod çözme algoritması vardır. Bu algoritma "Patterson'un Algoritması" olarak bilinir. Alınan bir y vektörünün $S(y)$ sendrom vektörünü hesapladıktan sonra $\sigma(z)S(y) \equiv \omega(z) \pmod{g(z)}$ eşlenik denklemini çözerken, $\mathbb{F}_2$ cisminde çalışıldığı için $\omega(z) = \sigma'(z)$ eşitliğinin büyük oranda sağlanışını kullanır. Kullanılan cismin karakteristiği 2 olduğundan, hata konumlandırıcı polinomu $\sigma(z) = a^2(z) + zb^2(z)$ şeklinde çift ve tek kuvvetlerini ayırarak yazabiliriz. Bütün bunları kullanarak Patterson'un algoritmasının işleyişini görelim.

Algoritmanın girdileri y vektörü ve $\Gamma(L, g)$ Goppa kodudur.

1. Adım : $S(y)$ sendrom vektörü hesaplanır.

2. Adım : $T(z) \equiv S(y)^{-1} \pmod{g(z)}$ hesaplanır.

3. Adım : $P(z) = \sqrt{T(z) + z} \pmod{g(z)}$ hesaplanır.

4. Adım : $u(z) \equiv v(z)S(y) \pmod{g(z)}$ olacak şekilde $u(z)$ ve $v(z)$ hesaplanır.

5. Adım : $\sigma(z) = u^2(x) + zv^2(z)$ hata konumlandırıcı polinom hesaplanır.

6. Adım : $\sigma(z)$'nin kökleri bulunur.

7. Adım : Hataların konumu bulunur.

$\mathbb{F}_2$ cisminde olduğumuz için hataların sadece konumunu bulmak yeterlidir çünkü cisimde sadece iki eleman olduğundan hatanın konumundaki eleman cisimdeki diğer elemanla değiştirildiğinde hata düzeltilmiş olur. Dolayısıyla, hataların konumu tespit edildiğinde e vektörünü elde etmiş oluruz.

Kod çözme işlemi yapılınca $c = (c_1, c_2, \ldots, c_n)$ vektörü elde edilir. $c = mG$ şeklinde kodlandığını biliyoruz.

$$(m_1, m_2, \ldots, m_n)G = (c_1, c_2, \ldots, c_n) \quad \Leftrightarrow \quad G^\top \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ m_n \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}$$

Bu denklem sistemini çözdüğümüzde m mesajını elde ederiz.

Örnek 4.1.21. $x^2 - x - 1$ primitif polinomuyla oluşturulmuş $\mathbb{F}_{3^2}$ cismini yazalım. α, verilen polinomun bir kökü olsun. O halde $\alpha^2 = \alpha + 1$ eşitliğinden yararlanarak cismi oluşturabiliriz.

$0 = (0,0)$	$\alpha^2 = 1+\alpha = (1,1)$	$\alpha^5 = -\alpha = (0,-1)$
$1 = (1,0)$	$\alpha^3 = 1-\alpha = (1,-1)$	$\alpha^6 = -1-\alpha = (-1,-1)$
$\alpha = (0,1)$	$\alpha^4 = -1 = (-1,0)$	$\alpha^7 = -1+\alpha = (-1,1)$

$g(z) = z(z - \alpha^7) = z^2 + \alpha^3 z$ polinomu ve $L = \{\alpha^i \mid 0 \leq i \leq 6\}$ kümesi ile oluşturulmuş $\Gamma(L, g)$ Goppa kodunu alalım. Bu durumda $\Gamma(L, g)$ kodunun $\mathbb{F}_3(\alpha)$ cismi üzerindeki denklik

kontrol matrisi

$$H = \begin{bmatrix} \alpha^4 & \alpha^3 & \alpha^2 & \alpha & 1 & \alpha^7 & \alpha^6 \\ \alpha^6 & \alpha^3 & \alpha^6 & \alpha^2 & \alpha^3 & \alpha^5 & \alpha^3 \end{bmatrix}$$

şeklinde olur. Matrisin girdilerini cisimdeki karşılıkları ile yazarak aşağıdaki matrisi elde ederiz.

$$H = \left[\begin{array}{rr|rr|rr|rr} -1 & 1 & 1 & 0 & 1 & -1 & -1 \\ 0 & -1 & 1 & 1 & 0 & 1 & -1 \\ \hline -1 & 1 & -1 & 1 & 1 & 0 & 0 \\ -1 & -1 & -1 & 1 & -1 & -1 & -1 \end{array}\right]$$

$GH^\top = 0$ denkleminin çözüm uzayının tabanını hesaplayarak G üreteç matrisine ulaşırız.

$$G = \begin{bmatrix} -1 & 0 & -1 & 1 & 0 & 0 & 0 \\ -1 & 0 & -1 & 0 & 1 & 1 & 0 \\ -1 & 1 & -1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Göndereceğimiz mesaj $m = (0, 0, 0)$ olsun. Bu mesajı $c = mG = (0, \ldots, 0)$ şeklinde kodlayıp iletelim. Alıcıya ulaşan mesaj $y = (0, 0, 0, 0, 0, 0, -1)$ olsun. Şimdi kod çözme adımlarını uygulayarak hata vektörünü elde edelim.

1. Adım : Sendrom vektörünü hesaplayalım.

$$S(y) = \sum_{i=1}^{6} \frac{y_i}{z - \alpha^i} = \frac{-1}{z - \alpha^6} \equiv \alpha^2 + \alpha z \pmod{g(z)}$$

2. Adım : $\sigma(z) = \sigma_0 + z$ eşitliğini kullanarak $\sigma(z)S(y) \pmod{z^2 + \alpha^3 z}$'yi hesaplayalım.

$$\begin{aligned} \sigma(z)S(y) &= (\sigma_0 + z)(\alpha^2 + \alpha z) \\ &= \sigma_0\alpha^2 + (\alpha^2 + \alpha\sigma_0)z + \alpha z^2 \\ &= \alpha^2\sigma_0 + (\alpha^2 + \alpha\sigma_0\alpha^4)z \\ &= \alpha^2\sigma_0 + (\alpha^7 + \alpha\sigma_0)z \end{aligned}$$

Böylelikle $\omega(z) = \omega_0$ için $\sigma(z)S(y) \equiv \omega(z) \pmod{g(z)}$ eşlenik denkleminin katsayılarını karşılaştırarak aşağıdaki denklem sistemini elde ederiz:

$$\begin{aligned} \omega_0 &= \alpha^2\sigma_0 \\ 0 &= \alpha^7 + \alpha\sigma_0 \end{aligned}$$

Bu denklem sisteminin çözümü $\sigma_0 = \alpha^2$ ve $\omega_0 = \alpha^4$ olarak bulunur ve buradan $\sigma(z) = z+\alpha^2$ ve $\omega(z) = \alpha^4$ polinomları elde edilir.

3. Adım : $\sigma(z)$ polinomu kullanılarak hataların konumu tespit edilir.

$$\sigma(z) = 0 \;\Rightarrow\; z = \alpha^6 = \alpha^7$$

Dolayısıyla hataların konumları kümesi $B = \{i \mid \sigma(\alpha_i) = 0\} = \{7\}$'dir.

4. Adım : $e_i = \frac{\omega(\alpha_i)}{\sigma'(\alpha_i)}$ eşitliği ile hataların değerleri hesaplanır.

$$e_7 = \frac{\alpha^4}{1} = \alpha^4 = -1$$

5. Adım : $c = y - e$ eşitliği ile hatasız mesaj bulunur.

$$c = y - e = (0,0,0,0,0,0,-1) - (0,0,0,0,0,0,-1) = (0,0,0,0,0,0,0)$$

Son olarak $G^\top m^\top = c^\top$ denklem sistemi çözülür ve $m = (0,0,0)$ mesajına ulaşılır.

4.1.1.4 Reed-Muller Kodlar

Bu bölümde Reed-Muller kodlar anlatılmıştır ve bu anlatım yapılırken [22] numaralı kaynaktan faydalanılmıştır. Reed-Muller kodlar David Eugene Muller (1924-2008) ve Irving Stoy Reed (1923-2012) tarafından ilk olarak 1954 yılında oluşturulmuştur. Hata düzeltme kodları olarak kullanılan bu kod ailesi kablosuz iletişimde de kullanılmaktadır. Diğer sağlam kod aileleri arasında çok üst seviye bir güvenlik sağlamasalar da kod çözme algoritmasının

hızı aradaki açığı kapatmaktadır. Kodlama ve kod çözme algoritmalarına geçmeden önce Reed-Muller kodların tanımını ve nasıl inşa edildiğini görelim.

Tanım 4.1.22. $0 \leq r \leq m$ koşulunu sağlayan r ve m tam sayıları için aşağıdaki parametrelere sahip kod ailesine r. dereceden ikili (r, m)-Reed-Muller kodu denir. $\mathcal{R}(r, m)$ ile gösterilir.

Uzunluk : $n = 2^m$

Boyut : $k = \binom{m}{0} + \binom{m}{1} + \binom{m}{2} + \cdots + \binom{m}{r}$

Minimum Uzaklık : $d = 2^{m-r}$

Bir $\mathcal{R}(r, m)$ kodunu inşa etmek için ilk adım olarak $1 \leq i \leq m$ olmak üzere v_i vektörlerini tanımlayacağız. v_i vektörleri, $\mathcal{R}(r, m)$ kodunun üreteç matrisinin satırları olacaktır. İkili Reed-Muller kodlarını inşa edeceğimiz için v_i vektörlerinin girdileri sıfır ve birlerden oluşacaktır ve $n = 2^m$ olduğundan, girdi sayısı da n olacaktır. Bu vektörleri aşağıdaki şekilde inşa tanımlarız:

$$v_i = \left(\underbrace{00\cdots00}_{2^{i-1}\text{ tane}} \quad \underbrace{11\cdots11}_{2^{i-1}\text{ tane}} \quad \underbrace{00\cdots00}_{2^{i-1}\text{ tane}} \quad \underbrace{11\cdots11}_{2^{i-1}\text{ tane}} \quad \underbrace{00\ldots00}_{2^{i-1}\text{ tane}} \quad \cdots \quad \underbrace{11\cdots11}_{2^{i-1}\text{ tane}} \right)$$

Örnek 4.1.23. $\mathcal{R}(2, 4)$ kodu için v_1, v_2, v_3, v_4 vektörlerini oluşturalım: Uzunluk $2^4 = 16$ olduğundan vektörlerin girdi sayısı da 16 olmalıdır.

$$v_1 = (0101010101010101) \qquad v_2 = (0011001100110011)$$

$$v_3 = (0000111100001111) \qquad v_4 = (0000000011111111)$$

$\mathcal{R}(2, 4)$ kodunun boyutu $k = \binom{4}{0} + \binom{4}{1} + \binom{4}{2} = 1 + 4 + 6 = 11$'dir. Dolayısıyla, üreteç matrisinin satır sayısı 11 olmalıdır. Dört satırını v_1, v_2, v_3, v_4 oluşturur. Şimdi kalan satırları nasıl oluşturacağımızı görelim:

İlk olarak v_0 vektörünün $v_0 = (1111111111111111)$ şeklinde tanımlandığını söyleyelim. Daha sonra, diğer satırlar için $v_i.v_j$ çarpımlarına başvururuz. Bu çarpımı, elimizdeki dört

vektörün her ikili kombinasyonu için yaparız. Buradan da $\binom{4}{2}$ kadar yeni vektör elde ederiz. Bu da bize, kodun boyutunun neden kombinasyonların toplamıyla bulunduğunu açıklar.

İki vektörü bileşensel çarparak yeni vektörü elde ederiz, yani ilk vektörün j. bitiyle ikinci vektörün j. bitinin çarpımı yeni vektörün j. bitini verir. Çarpma işlemini $\mathbb{Z}_2$'deki bilinen çarpma işlemi olarak yaparız. Buradan, $v_1v_2, v_1v_3, v_1v_4, v_2v_3, v_2v_4, v_3v_4$ vektörlerini elde ederiz:

$$\begin{aligned}
v_1v_2 &= (0101010101010101).(0011001100110011) = (0001000100010001)\\
v_1v_3 &= (0101010101010101).(0000111100001111) = (0000010100000101)\\
v_1v_4 &= (0101010101010101).(0000000011111111) = (0000000001010101)\\
v_2v_3 &= (0011001100110011).(0000111100001111) = (0000001100000011)\\
v_2v_4 &= (0011001100110011).(0000000011111111) = (0000000000110011)\\
v_3v_4 &= (0000111100001111).(0000000011111111) = (0000000000001111)
\end{aligned}$$

(Eğer r sayısı 2'den büyük bir sayı olsaydı üçlü, dörtlü, ..., r'li çarpımlar da yapılırdı.)

Böylece, $\mathcal{R}(2,4)$ Reed-Muller kodunun üreteç matrisi aşağıdaki şekilde olur.

$$G = \begin{bmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_1v_2 \\ v_1v_3 \\ v_1v_4 \\ v_2v_3 \\ v_2v_4 \\ v_3v_4 \end{bmatrix} = \begin{bmatrix}
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1\\
0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1\\
0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1\\
0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1\\
0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1\\
0&0&0&0&0&1&0&1&0&0&0&0&0&1&0&1\\
0&0&0&0&0&0&0&0&0&1&0&1&0&1&0&1\\
0&0&0&0&0&0&1&1&0&0&0&0&0&0&1&1\\
0&0&0&0&0&0&0&0&0&0&1&1&0&0&1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1
\end{bmatrix}$$

Reed-Muller kodları üretmenin alternatif bir yolu daha vardır. $\forall\ 1 \leq r \leq m$ için $\mathcal{R}(r,m)$ kodu aşağıdaki şekildedir:

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

Bu yolla birlikte, üreteç matrisimizi ($\mathcal{R}(r'.m')$ kodunun üreteç matrisi $G(r',m')$ üzere) aşağıdaki şekilde yazabiliriz:

$$G(r,m) = \left[\begin{array}{c|c} G(r,m-1) & G(r,m-1) \\ \hline 0 & G(r-1,m-1) \end{array}\right]$$

Tabi ki, bir Reed-Muller kodunu bu şekilde üretebilmek ve üreteç matrisini yazabilmek için kendisinden önce gelen, yani daha küçük parametrelere sahip olan tüm kodları yazmak gerekir. Bu sebeple, bu alternatif yol, Reed-Muller kodlarını oluşturmak için pratik bir yol değildir ama kodun bazı özelliklerini gösterebilmek için kısa yollar sunar.

Önerme 4.1.24. $\mathcal{R}(r,m)$ kodunun minimum uzaklığı 2^{m-r}'dir.

Kanıt. İspatı tümevarımla yapacağız. $\mathcal{R}(0,1)$ ve $\mathcal{R}(1,1)$ kodlarının üreteç matrisleri ve kod kelimeleri sırasıyla aşağıdaki şekildedir:

$$G(0,1) = \begin{bmatrix} 1 & 1 \end{bmatrix} \longrightarrow \mathcal{R}(0,1) = \{00, 11\}$$

$$G(1,1) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \longrightarrow \mathcal{R}(1,1) = \{00, 11, 01, 10\}$$

Dolayısıyla $d(\mathcal{R}(0,1)) = 2 = 2^{1-0}$ ve $d(\mathcal{R}(1,1)) = 1 = 2^{1-1}$'dir. Tümevarımın ikinci adımı olarak, belirli bir m sayısı için ve $\forall\ 0 \leq r \leq m$ için $\mathcal{R}(r,m)$ kodunun minimum uzaklığının 2^{m-r} olduğunu kabul edelim. Tümevarımım son adımı olarak, $\mathcal{R}(r,m+1)$ kodunun minimum uzaklığının 2^{m+1-r} olduğunu gösterelim. Bunun için, biraz önce verdiğimiz alternatif

yolu kullanacağız. $f, f' \in \mathcal{R}(r,m)$ ve $g, g' \in \mathcal{R}(r-1,m)$ olsun. Bu durumda $c_1 = (f, f+g)$ ve $c_2 = (f', f'+g')$ vektörleri $\mathcal{R}(r, m+1)$ kodunun kelimeleridir.

(i) Eğer $g = g'$ ise $d(c_1, c_2) = 2.d(f, f') \geq 2.2^{m-r} = 2^{m-r+1}$ olur.

(ii) Eğer $g \neq g'$ ise $d(c_1, c_2) = w(f - f') + w((f + g) - (f' + g'))$ olur. Ayrıca, $\forall\ x, y$ için $w(x + y) \geq w(x) - w(y)$ olduğundan aşağıdaki eşitsizliği yazabiliriz.

$$d(c_1, c_2) \geq w(f - f') + w(g - g') - w(f - f') = w(g - g')$$

$g - g' \in \mathcal{R}(r-1, m)$ olduğundan $w(g - g') \geq 2^{m-r+1}$'dir.

Böylece, **(i)** ve **(ii)**'den $d(\mathcal{R}(r, m+1)) = 2^{m-r+1}$ olarak bulunur. □

Önerme 4.1.25. **($\mathcal{R}(r,m)$ Kodunun Duali) :** Bir $\mathcal{R}(r,m)$ kodunun duali $\mathcal{R}(m-r-1, m)$ Reed-Muller kodudur.

Kanıt. İspatı 2 adımda yapacağız. Birinci adımda iki kodun boyutlarının toplamının üst uzayın boyutuna, yani $\mathbb{F}_2^{2^m}$'nin boyutuna eşit olduğunu göstereceğiz. ($dim(\mathbb{F}_2^{2^m}) = 2^m$) İkinci adımda ise iki uzaydan da alınan keyfi elemanların birbirine dik olduğunu, yani iç çarpımlarının sıfır olduğunu göstereceğiz.

1. Adım : $\mathcal{R}(r,m)$ kodunun boyutu $\binom{m}{0} + \binom{m}{1} + \cdots + \binom{m}{r}$'dir. $\mathcal{R}(m-r-1, m)$ kodunun boyutu ise $\binom{m}{0} + \binom{m}{1} + \cdots + \binom{m}{m-r-1}$'dir. Kombinasyon hesabında, $l < k$ birer doğal sayı olmak üzere $\binom{k}{l} = \binom{k}{k-l}$ olduğunu biliyoruz:

$$\binom{k}{l} = \frac{k!}{(k-l)!\,(l)!} = \frac{k!}{(k-l)!\,(k-(k-l))!} = \frac{k!}{(k-(k-l))!\,(k-l)!} = \binom{k}{k-l}$$

Dolayısıyla aşağıdaki eşitlikler sağlanır.

$$\begin{array}{ccc}
\binom{m}{0} & = & \binom{m}{m} \\
\binom{m}{1} & = & \binom{m}{m-1} \\
\binom{m}{2} & = & \binom{m}{m-2} \\
\vdots & & \vdots \\
\binom{m}{m-r-2} & = & \binom{m}{r+2} \\
\binom{m}{m-r-1} & = & \binom{m}{r+1} \\
\vdots & & \vdots \\
\binom{m}{m-1} & = & \binom{m}{1} \\
\binom{m}{m} & = & \binom{m}{0}
\end{array}$$

Buradan, $\mathcal{R}(m-r-1,m)$ kodunun boyutu için aşağıdaki eşitliği elde ederiz.

$$\binom{m}{0}+\binom{m}{1}+\cdots+\binom{m}{m-r-1}=\binom{m}{m}+\binom{m}{m-1}+\cdots+\binom{m}{r+1}$$

Böylece $\mathcal{R}(r,m)$ ve $\mathcal{R}(m-r-1,m)$ kodlarının boyutları toplamı aşağıdaki şekilde yazılabilir.

$$\binom{m}{0}+\binom{m}{1}+\cdots+\binom{m}{r}+\binom{m}{r+1}+\cdots+\binom{m}{m-1}+\binom{m}{m}$$

Kombinasyon hesabından biliyoruz ki $\sum_{i=0}^{m}\binom{m}{i}=2^m$'dir. Böylece ispatın ilk adımı tamamlanır.

Şimdi ikinci adıma geçelim. $a\in\mathcal{R}(r,m)$ ve $b\in\mathcal{R}(m-r-1,m)$ olsun. $\mathcal{R}(r,m)$ kodunun taban elemanları $v_0,v_1,\ldots,v_m$ vektörlerinin en fazla r'li çarpımlarından oluştuğu için a vektörünü, taban elemanlarının bir lineer kombinasyonu cinsinden yazmak, aslında derecesi en fazla r olan, $v_0,v_1,\ldots,v_m$ değişkenlerine bağlı olan ve katsayılarını $\mathbb{F}_2$ cisminden olan bir polinom olarak yazmakla aynıdır. Dolayısıyla a vektörüne en fazla r dereceli bir polinom gözüyle bakabiliriz. Aynı durum b vektörü için de geçerlidir, yani b vektörü, derecesi en fazla $m-r-1$ olan ve $v_0,v_1,\ldots,v_m$ değişkenlerine bağlı bir polinomu ifade eder. Dolayısıyla, a ve b vektörlerinin çarpımının ifade ettiği polinom, derecesi en fazla $(m-r-1)+r=m-1$ olan ve $v_1,v_2,\ldots,v_m$ değişkenlerine bağlı olan bir polinomdur. Yani $ab\in\mathcal{R}(m-1,m)$'dir.

Ayrıca Reed-Muller kodlarında bir vektörün barındırdığı 1 sayısı çift olmak zorundadır. Yani ab vektörünün ağırlığı çift sayıdır. Böylece a ile b'nin iç çarpımı çift sayıda 1'in toplamına eşittir ki bu toplamın $\mathbb{F}_2$'deki karşılığı 0'dır. Yani $a \perp b$'dir ve böylece ispat tamamlanmış olur. □

Önerme 4.1.26. $k < r < m$ olmak üzere $\mathcal{R}(k, m) \subset \mathcal{R}(r, m)$'dir.

Kanıt. Kodların üreteç matrisleri göz önünde bulundurulduğunda, önermenin doğruluğu aşikardır. Dolayısıyla ispatı açmaya gerek yoktur. □

Şimdi Reed-Muller kodlarla kodlama ve kod çözme işlemlerinin nasıl gerçekleştiğini bir örnek üzerinden görelim.

4.1.1.4.1 Reed-Muller Kodlarla Kodlama

Kolaylık açısından, üreteç matrisini daha önce oluşturduğumuz $\mathcal{R}(2, 4)$ kodunu kullanalım.

$$G = \begin{bmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_1v_2 \\ v_1v_3 \\ v_1v_4 \\ v_2v_3 \\ v_2v_4 \\ v_3v_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Kodun boyutu 11 olduğundan iletilecek mesajın uzunluğu 11 olarak seçilir. Mesaj, $M = (m_1, m_2, \ldots, m_{11})$ olsun. Kodlama işlemi MG çarpımı ile yapılır. Bu çarpım sonucunda

1×16 formunda bir çıktı elde edilir.

$$MG = \begin{bmatrix} m_1 & m_2 & \cdots & m_{11} \end{bmatrix} \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_2 v_4 \\ v_3 v_4 \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & \cdots & c_{16} \end{bmatrix} = C$$

Bu yolla $M = 10010100100$ mesajını kodlarsak çıktı olarak

$$C = MG = 1110\ 0010\ 1110\ 0010$$

vektörü bulunur. $\mathcal{R}(2,4)$ kodunun minimum uzaklığı $2^{m-r} = 2^2 = 4$ olduğundan en fazla 1 hataya kadar düzeltme yapabilir. Elde ettiğimiz C vektörünün alıcıya 1 hata ile gittiği senaryo üzerinden kod çözme işleminin nasıl gerçekleştiğini görelim.

4.1.1.4.2 Reed-Muller Kodlarla Kod Çözme

Alıcı 16 girdili $y = (y_1, y_2, \ldots, y_{16})$ mesajını $1110\ 0010\ 1010\ 0010$ şeklinde almış olsun. y mesajının hatasız halinin MG şeklinde hesaplanarak elde edildiğini bildiğimizden aşağıdaki denklemleri oluşturarak hata arayışına başlarız.

$$\begin{aligned}
y_1 &= m_1 \\
y_2 &= m_1 + m_2 \\
y_3 &= m_1 + m_3 \\
y_4 &= m_1 + m_2 + m_3 + m_6 \\
\\
y_5 &= m_1 + m_4 \\
y_6 &= m_1 + m_2 + m_4 + m_7 \\
y_7 &= m_1 + m_3 + m_4 + m_9 \\
y_8 &= m_1 + m_2 + m_3 + m_4 + m_6 + m_7 + m_9
\end{aligned}$$

$$
\begin{aligned}
y_9 &= m_1 + m_5 \\
y_{10} &= m_1 + m_2 + m_5 + m_8 \\
y_{11} &= m_1 + m_3 + m_5 + m_{10} \\
y_{12} &= m_1 + m_2 + m_3 + m_5 + m_6 + m_8 + m_{10}
\end{aligned}
$$

$$
\begin{aligned}
y_{13} &= m_1 + m_4 + m_5 + m_{11} \\
y_{14} &= m_1 + m_2 + m_4 + m_5 + m_7 + m_8 + m_{11} \\
y_{15} &= m_1 + m_3 + m_4 + m_5 + m_9 + m_{10} + m_{11} \\
y_{16} &= m_1 + m_2 + m_3 + m_4 + m_5 + m_6 + m_7 + m_8 + m_9 + m_{10} + m_{11}
\end{aligned}
$$

Dikkat edersek y mesajının girdilerini dörtlü bloklara ayırıp herbir bloğu kendi içerisinde topladığımızda her toplam m_6 sonucunu vermektedir. (Toplama işlemini $\mathbb{F}_2$ cisminde yaptığımızdan m_6 hariç hepsi silinmektedir.)

$$
\begin{aligned}
&\textbf{1. Blok :} && y_1 + y_2 + y_3 + y_4 && \equiv && m_6 && \pmod 2 \\
&\textbf{2. Blok :} && y_5 + y_6 + y_7 + y_8 && \equiv && m_6 && \pmod 2 \\
&\textbf{3. Blok :} && y_9 + y_{10} + y_{11} + y_{12} && \equiv && m_6 && \pmod 2 \\
&\textbf{4. Blok :} && y_{13} + y_{14} + y_{15} + y_{16} && \equiv && m_6 && \pmod 2
\end{aligned}
$$

Dolayısıyla y mesajındaki herbir dörtlü bloğun toplamının aynı sonucu vermesi beklenir. Fakat y mesajında bir adet hata olduğundan bir blok diğer üç bloktan farklı bir sonuç verecektir. Bu da bize hatanın hangi blokta olduğunu söyler. Alınan y mesajında bunu kontrol edelim:

1. Blok : $1 + 1 + 1 + 0 \equiv 1 \pmod 2$ **2. Blok :** $0 + 0 + 1 + 0 \equiv 1 \pmod 2$

3. Blok : $1 + 0 + 1 + 0 \equiv 0 \pmod 2$ **4. Blok :** $0 + 0 + 1 + 0 \equiv 1 \pmod 2$

Buradan anlıyoruz ki alınan mesajdaki hata 3. blokta, yani $y_9, y_{10}, y_{11}, y_{12}$ girdilerinden birindedir, diğer bloklar hatasız ulaşmıştır. Şimdi farklı dörtlü blokların toplamını kullanarak hatanın konumunu ayrıntılandıralım, yani 3. bloğun neresinde olduğu hakkında daha

ayrıntılı bilgi edinelim.

$$\begin{aligned}
\textbf{1. Blok :} \quad & y_1 + y_2 + y_5 + y_6 \equiv m_7 \pmod 2 \\
\textbf{2. Blok :} \quad & y_3 + y_4 + y_7 + y_8 \equiv m_7 \pmod 2 \\
\textbf{3. Blok :} \quad & y_9 + y_{10} + y_{13} + y_{14} \equiv m_7 \pmod 2 \\
\textbf{4. Blok :} \quad & y_{11} + y_{12} + y_{15} + y_{16} \equiv m_7 \pmod 2
\end{aligned}$$

Bu denklemleri, alınan y mesajında uyguladığımızda aşağıdaki sonuçları elde ederiz.

1. Blok : $1 + 1 + 0 + 0 \equiv 0 \pmod 2$ **2. Blok :** $1 + 0 + 1 + 0 \equiv 0 \pmod 2$

3. Blok : $1 + 0 + 0 + 0 \equiv 1 \pmod 2$ **4. Blok :** $1 + 0 + 1 + 0 \equiv 0 \pmod 2$

Buradan anlıyoruz ki hata 3. blokta, yani $y_9, y_{10}, y_{13}, y_{14}$ girdilerinden birindedir, diğer bloklar hatasız ulaşmıştır. Bir önceki blok denkleminden hatanın konumunun $y_9, y_{10}, y_{11}, y_{12}$ girdilerinden birinde olduğunu belirlemiştik, şimdi ise $y_9, y_{10}, y_{13}, y_{14}$ girdilerinden birinde olduğunu saptadık. İki bloğun ortak girdileri y_9, y_{10} olduğundan alınan mesajdaki hatanın bu iki girdiden birinde olduğunu söyleyebiliyoruz. Hatanın konumunu tam olarak belirlemek için son bir blok ayrımı yapacağız.

$$\begin{aligned}
\textbf{1. Blok :} \quad & y_1 + y_3 + y_5 + y_7 \equiv m_9 \pmod 2 \\
\textbf{2. Blok :} \quad & y_2 + y_4 + y_6 + y_8 \equiv m_9 \pmod 2 \\
\textbf{3. Blok :} \quad & y_9 + y_{11} + y_{13} + y_{15} \equiv m_9 \pmod 2 \\
\textbf{4. Blok :} \quad & y_{10} + y_{12} + y_{14} + y_{16} \equiv m_9 \pmod 2
\end{aligned}$$

Bu denklemleri, alınan y mesajında uyguladığımızda aşağıdaki sonuçları elde ederiz:

1. Blok : $1 + 1 + 0 + 1 \equiv 1 \pmod 2$ **2. Blok :** $1 + 0 + 0 + 0 \equiv 1 \pmod 2$

3. Blok : $1 + 1 + 0 + 1 \equiv 1 \pmod 2$ **4. Blok :** $0 + 0 + 0 + 0 \equiv 0 \pmod 2$

Bu denklemler bize hatanın 4. blokta, yani $y_{10}, y_{12}, y_{14}, y_{16}$ girdilerinden birinde olduğunu söyler. Önceki denklemlerden, hatanın y_9 veya y_{10}'da olduğunu belirlemiştik. Son denklem sisteminden elde ettiğimiz sonuçla birleştirerek, alınan y mesajındaki hatanın, sonuçlardaki ortak girdi olan y_{10}'da olduğunu söyleriz. y mesajındaki 10. girdiyi değiştirerek hatasız

mesaja, yani C'ye ulaşırız.

$$C = y - (0000\ 0000\ 0100\ 0000) = 1110\ 0010\ 1110\ 0010$$

Son olarak $MG = C$ denklem sistemi çözülerek $M = 10010100100$ mesajına ulaşılır.

4.1.1.5 Reed-Solomon Kodlar

Bu kısımda Reed-Solomon kodlar anlatılmıştır ve bu anlatım yapılırken [23] numaralı kaynaktan faydalanılmıştır. Reed-Solomon kodlar 1960 yılında Irving Stoy Reed ve Gustave Solomon tarafından oluşturulmuştur. Bu kod ailesi ikili olmayan (non-binary) kodlar sınıfındadır. Reed-Solomon kodları CD, DVD, Barkod ve QR kodlardan başlayıp kablosuz uydu iletişimi, yüksek hızlı ADSL modemler ve mobil bağlantılar gibi devam eden birçok alanda hata düzeltme kodları olarak kullanılmıştır. Bu kod sınıfının birçok alanda uygulaması vardır fakat bizim anlatacağımız kısım, kodlama ve kod çözme algoritmalarının nasıl işlediğidir. Kodlama algoritmasından başlayalım.

4.1.1.5.1 Reed-Solomon Kodlarla Kodlama

Reed-Solomon kodları ikili olmayan kodlar olduğundan cismimiz $\mathbb{F}_2$ değildir, bu yüzden ilk adım olarak bir sonlu cisim belirlenir.

$$\mathbb{F}_q = \left\{0, 1, \alpha, \alpha^2, \ldots, \alpha^{q-2}\right\}$$

Daha sonra bu cisim içerisinden belirli N tane eleman seçilir.

$$A = \{\alpha_1, \alpha_2, \ldots, \alpha_N\}$$

Burada seçilen noktalar ve cisim herkes tarafından görülebilecek şekilde yayımlanır. Ardından, uzunluğu $n < N$ olacak şekilde iletilecek mesaj belirlenir.

$$\forall\, i = 0, \ldots, n-1 \ \text{için} \ \ m_i \in \mathbb{F}_q \ \ \text{olmak üzere} \ \ M = (m_0, m_1, m_2, \ldots, m_{n-1})$$

Son olarak, katsayıları M mesajının girdileri olan $n-1$ dereceli bir polinom belirlenir.

$$f(x) = m_0 + m_1 x + m_2 x^2 + \cdots + m_{n-1} x^{n-1}$$

Kodlama işlemi, A kümesinin elemanlarının sırasıyla $f(x)$ polinomundaki değerleri hesaplanarak yapılır.

$$\begin{array}{rcl} f(\alpha_1) & = & c_1 \\ f(\alpha_2) & = & c_2 \\ & \vdots & \\ f(\alpha_N) & = & c_N \end{array} \qquad \Longrightarrow \qquad \text{Kodlanmış Mesaj :} \quad C = (c_1, c_2, \ldots, c_N)$$

Kodlama işlemi burada son bulur. İşlemden açıkça görüleceği üzere $q > N > m$ eşitsizliği vardır. Ayrıca Reed-Solomon kodlarının hata düzeltme sınırı aşağıdaki eşitsizlikte verildiği şekildedir.

$$t \quad \text{hata sayısı olmak üzere} \quad N - n \geq 2t$$

Bu eşitsizliğin sebebini kod çözme algoritmasının işleyişinden anlayacağız.

4.1.1.5.2 Reed-Solomon Kodlarla Kod Çözme

Mesajın kodlanmış hali olan C'nin alıcıya t adet hatayla ulaştığını varsayalım.

$$C' = (c_1, c_2', c_3, c_4, \ldots, c_{N-1}', c_N)$$

Burada, işaretli olan girdiler hatalı girdilerdir olduğunu varsayalım.

$\alpha_1, \alpha_2, \ldots, \alpha_N$ elemanlarının görüntüsü sırasıyla C''nün girdileri olacak şekilde yeni bir

polinom belirlenir. Bu polinomu açıkça yazmaya gerek yoktur, sadece sırasıyla hangi girdilerde hangi çıktıları verdiğini bilmek yeterlidir.

$$\begin{aligned} R(\alpha_1) &= c_1 \\ R(\alpha_2) &= c_2' \\ &\vdots \\ R(\alpha_N) &= c_N \end{aligned}$$

$R(x)$ polinomunun derecesi de $f(x)$ ile aynıdır, yani $n-1$'dir. Şimdi bir hata polinomu belirleyeceğiz. Bu polinomun kökleri, çıktısı hatalı olan girdiler olacaktır. Hangi girdilerin çıktısının hatalı olduğunu tabi ki bilmiyoruz, zaten algoritmanın amacı, bu polinomu belirleyerek hatanın yerini tespit etmektir. Hata polinomuna $E(x)$ diyelim. Varsayalım ki çıktısı hatalı olan girdiler $\alpha_{i_1}, \alpha_{i_2}, \ldots, \alpha_{i_t}$ elemanları olsunlar. Bu durumda hata polinomumuz aşağıdaki şekilde yazılabilir.

$$E(x) = (x - \alpha_{i_1})(x - \alpha_{i_2}) \cdots (x - \alpha_{i_t})$$

$E(x)$ polinomunun kökleri sadece çıktıları hatalı olan girdiler olduğundan diğer girdilerde sıfırdan farklı değerler alır. Ayrıca çıktıları hatasız olan girdiler için $f(x)$ ile $R(x)$ polinomları aynı sonucu vermektedir. Dolayısıyla aşağıdaki eşitliği yazabiliriz.

$$\forall\, i = 1, 2, \ldots, N \text{ için } f(\alpha_i)E(\alpha_i) = R(\alpha_i)E(\alpha_i)$$

$f(x)$ polinomunun derecesi $n-1$ ve $E(x)$ polinomunun derecesi t olduğundan $f(x)E(x)$ çarpımının derecesi $n+t-1$ olur. Böylece, $n+t$ adet katsayıya sahip olan ve $n+t-1$ dereceli genel bir polinom yazıp denklem sistemi oluşturabiliriz. Çarpım polinomunu

$$f(x)E(x) = d_0 + d_1 x + d_2 x^2 + \cdots + d_{n+t-1} x^{n+t-1}$$

şeklinde yazalım. Bu durumda denklemimiz aşağıdaki şekli alır. $\forall\, i = 1, 2, \ldots, N$ için

$$d_0 + d_1\alpha_i + d_2\alpha_i^2 + \cdots + d_{n+t-1}\alpha_i^{n+t-1} = R(\alpha_i)E(\alpha_i)$$

Toplamda N tane α_i olduğu için denklem sistemimizde N tane denklem vardır. Denklem sisteminin sol tarafında $n+t$, sağ tarafında t tane olmak üzere toplamda $n+2t$ tane değişken vardır. Sonuç olarak $n + 2t$ değişkenli N tane denklemimiz var. Bu denklem sisteminin tek çözümünün olabilmesi için $N \geq n + 2t$ eşitsizliğin sağlanması gerekmektedir.

$$N \geq n + 2t \quad \Rightarrow \quad N - n \geq 2t$$

Bu durum, algoritmanın kaç hata çözebileceğine ilişkin eşitsizliğin sebebini açıklar.

Denklem sistemini çözdüğümüzde hem hangi girdilerin çıktılarının hatalı hesaplandığını hem de tüm d_i katsayılarını elde etmiş oluruz. Böylelikle hem $f(x)E(x)$ çarpım polinomunu hem de $E(x)$ polinomunu yazabiliriz. Son olarak

$$\frac{f(x)E(x)}{E(x)}$$

bölme işlemini yaparak $f(x)$ polinomunu elde ederiz. Geriye sadece hatalı çıktıya sahip olan girdilerin $f(x)$ polinomu altındaki görüntülerini bularak kodlanmış mesajın doğru halini yazmak kalır. Böylelikle kod çözme işlemi tamamlanır.

Şimdi anlattıklarımızı bir örnek üzerinden görelim.

Örnek 4.1.27. $\mathbb{F}_{11} = \{0, 1, 2, \ldots, 10\}$ cismi üzerinden rastgele $N = 5$ eleman seçelim.

$$\alpha_1 = 2 \qquad \alpha_2 = 3 \qquad \alpha_3 = 5 \qquad \alpha_4 = 6 \qquad \alpha_5 = 7$$

Göndereceğimiz (kodlayacağımız) mesaj $n = 3$ uzunluğundaki $M = (3, 7, 8)$ olsun. Bu durumda kullanacağımız polinom aşağıdaki şekilde oluşur.

$$f(x) = m_0 + m_1x + m_2x^2 = 3 + 7x + 8x^2$$

Herbir α_i'nin $f(x)$ polinomu altındaki görüntüsünü hesaplayarak kodlama işlemini bitirelim.

$$\begin{array}{rcl} f(\alpha_1) = f(2) = 49 & & \\ f(\alpha_2) = f(3) = 96 & & \\ f(\alpha_3) = f(5) = 238 & \Longrightarrow & \text{Kodlanmış Mesaj : } C = (49, 96, 238, 333, 444) \\ f(\alpha_4) = f(6) = 333 & & \\ f(\alpha_5) = f(x) = 444 & & \end{array}$$

Normalde görüntülerin $(\mathrm{mod}\ 11)$'deki değerlerini yazmamız gerekir fakat amacımız denklem sistemi çözmek olduğu için modüler aritmetiğe girmeden de halledebiliriz. Böylece işlem adımlarını biraz kısaltmış oluruz.

$N = 5$ ve $n = 3$ olduğundan çözebileceğimiz hata sayısı en fazla $\frac{N-n}{2} = \frac{5-3}{2} = 1$ kadar olabilir.

C mesajının alıcıya $C' = (49, 96, \underline{110}, 333, 444)$ şeklinde, içinde 1 hata ile gittiği senaryoyu düşünelim ve hatanın konumunu ve doğru halini bulalım. Çıktıları hatalı mesajın girdileri olan $R(x)$ polinomunun değerlerini yazalım.

$$\begin{array}{rcl} R(\alpha_1) = R(2) = 49 \\ R(\alpha_2) = R(3) = 96 \\ R(\alpha_3) = R(5) = 110 \\ R(\alpha_4) = R(6) = 333 \\ R(\alpha_5) = R(7) = 444 \end{array}$$

Şimdi hata polinomumuzu tanımlayalım. Toplamda $t = 1$ tane hata olduğu için hata polinomumuzu, hatalı olan girdinin ön görüntüsü e olmak üzere $E(x) = x - e$ şeklinde yazabiliriz.

f polinomunun derecesi $n - 1 = 3 - 1 = 2$ ve E polinomunun derecesi 1 olduğundan $f(x)E(x)$ polinomunun derecesi $2 + 1 = 3$ olur. Dolayısıyla $f(x)E(x)$ polinomunu aşağıdaki şekilde genel bir polinom olarak yazabiliriz:

$$f(x)E(x) = d_0 + d_1x + d_2x^2 + d_3x^3$$

Şimdi denklem sistemimizi, $\forall\, i = 1, 2, 3, 4, 5$ için

$$d_0 + d_1\alpha_i + d_2\alpha_i^2 + d_3\alpha_i^3 = R(\alpha_i)E(\alpha_i)$$

eşitliğini kullanarak oluşturabiliriz.

$$\begin{array}{lcccccccccccc}
x = \alpha_1 = 2 & : & d_0 & + & 2d_1 & + & 4d_2 & + & 8d_3 & = & 49(2-e) & = & 98 - 49e \\
x = \alpha_2 = 3 & : & d_0 & + & 3d_1 & + & 9d_2 & + & 27d_3 & = & 96(3-e) & = & 288 - 96e \\
x = \alpha_3 = 5 & : & d_0 & + & 5d_1 & + & 25d_2 & + & 125d_3 & = & 110(5-e) & = & 550 - 110e \\
x = \alpha_4 = 6 & : & d_0 & + & 6d_1 & + & 36d_2 & + & 216d_3 & = & 333(6-e) & = & 1998 - 333e \\
x = \alpha_5 = 7 & : & d_0 & + & 7d_1 & + & 49d_2 & + & 343d_3 & = & 444(7-e) & = & 3108 - 444e
\end{array}$$

Denklem sisteminin sağ tarafındaki e bilinmeyenine sahip terimleri sol tarafa attığımızda aşağıdaki denklem sistemini elde ederiz:

$$\begin{array}{ccccccccccc}
d_0 & + & 2d_1 & + & 4d_2 & + & 8d_3 & + & 49e & = & 98 \\
d_0 & + & 3d_1 & + & 9d_2 & + & 27d_3 & + & 96e & = & 288 \\
d_0 & + & 5d_1 & + & 25d_2 & + & 125d_3 & + & 110e & = & 550 \\
d_0 & + & 6d_1 & + & 36d_2 & + & 216d_3 & + & 333e & = & 1998 \\
d_0 & + & 7d_1 & + & 49d_2 & + & 343d_3 & + & 444e & = & 3108
\end{array}$$

Bu denklem sistemini çözdüğümüzde aşağıdaki sonuçları elde ederiz:

$$d_0 = -15 \qquad d_1 = -32 \qquad d_2 = -33 \qquad d_3 = 8 \qquad e = 5$$

$e = 5 = \alpha_3$ olduğundan kodlanmış mesajdaki hatanın 3. girdide olduğunu anlıyoruz. Ayrıca bu sonuçlar bize

$$f(x)E(x) = -15 - 32x - 33x^2 + 8x^3 \quad \text{ve} \quad E(x) = x - 5$$

olduğunu söyler. Polinom bölmesi yaparak $f(x)$ polinomunu elde ederiz.

$$\frac{f(x)E(x)}{E(x)} = \frac{8x^3 - 33x^2 - 32x - 15}{x - 5} = 8x^2 + 7x + 3 = f(x)$$

Son olarak $f(\alpha_3) = f(5) = 8(5)^2 + 7(5) + 3 = 238$ hesabını yaparak hatalı girdinin doğru şeklini buluruz. Sonuç olarak M mesajının doğru kodlanmış haline, yani $C = (49, 96, 238, 333, 444)$'e ulaşırız.

Aslında M mesajına ulaşmak için mesajın doğru kodlanmış halini bulamıza gerek yoktu. $f(x)$ polinomunu bulduğumuz zaman M mesajını doğrudan yazabilirdik, çünkü M mesajı $f(x)$ polinomunun katsayılarıdır.

$$f(x) = 3 + 7x + 8x^2 \qquad \Rightarrow \qquad M = (3, 7, 8)$$

4.1.2 Kodlama Teorisindeki Zor Problemler

Bu kısımda kod-tabanlı kriptografik algoritmaların kullandığı matematiksel problemler anlatılmıştır. Bu anlatım yapılırken [21] numaralı kaynaktan faydalanılmıştır.

1. **Genel Kod Çözme Problemi**

   Bir $C \subset \mathbb{F}_q^n$ lineer kodu verilsin. Bir t tam sayısı ve bir $c \in \mathbb{F}_q^n$ vektörü verildiğinde $d(x, c) \leq t$ olacak şekilde bir $x \in C$ kod kelimesinin bulunması problemine genel kod çözme problemi (general decoding problem) denir.

2. **Sendrom Kod Çözme Problemi**

   $\mathbb{F}_q$ cismi üzerinde bir H matrisi, bir s vektörü ve negatif olmayan bir t tam sayısı verilsin. Hamming ağırlığı $w(x) = t$ olan ve $Hx^\top = s^\top$ eşitliğini sağlayan bir $x \in \mathbb{F}_q^n$ vektörünün bulunması problemine sendrom kod çözme problemi (syndrome decoding problem) denir.

3. **Goppa Parametreli Sendrom Kod Çözme Problemi**

   $2^m \times r$ formunda, ikilik tabanda bir H matrisi ve bir s sendrom vektörü verilsin. Ağırlığı $\frac{r}{m}$ olan ve $Hx^\top = s^\top$ eşitliğini sağlayan bir x kod kelimesinin var olup olmadığına karar verme problemine Goppa parametreli sendrom kod çözme problemi (Goppa parameterized syndrome decoding problem) denir.

4. **Goppa Kodu Ayırt Etme Problemi**

   $r \times n$ formunda verilen bir H matrisinin, bir Goppa kodunun denklik kontrol matrisi olup olmadığını bulma problemine Goppa kodu ayırt etme problemi (Goppa code distinguishing problem) denir.

4.1.3 Klasik McEliece

Bu kısımda Klasik McEliece kriptosistemi anlatılmıştır ve bu anlatım yapılırken [21] ve [24] numaralı kaynaklardan faydalanılmıştır. Klasik McEliece algoritması McEliece ve Niederreiter algoritmaları baz alınarak geliştirilmiştir. Bu sebeple, algoritmanın işleyişini tam olarak anlayabilmek için öncelikle McEliece ve Niederreiter algoritmalarını öğrenmekte fayda vardır.

4.1.3.1 McEliece

McEliece kriptosistemi 1978 yılında Robert J. McEliece tarafından geliştirilmiştir. İkili Goppa kodları baz alan bu algoritmanın güvenliği rastgele bir lineer kodun çözülmesinin zorluğuna dayanmaktadır. Mantıklı parametreler seçildiğinde şimdiye kadar hiç kimse tarafından kırılamamıştır. McEliece algoritmasının en büyük sorunu anahtar boyutlarının çok büyük olmasıyken sunduğu güvenlik seviyesi sebebiyle bu sorun göz ardı edilebilmektedir. Teknolojinin henüz yeterli ilerleme kaydedemediği bir zamanda ortaya çıktığı için anahtar boyutları sebebiyle diğer algoritmaların elde ettiği üne kavuşamayan McEliece algoritması, Shor algoritmasının diğer kriptosistemleri tehdit etmesiyle geniş yankı uyandırmıştır. Şimdi McEliece kriptosisteminin işleyişini görelim:

Kullanılan Goppa kodunun parametreleri n (kodun uzunluğu), k ($\mathbb{F}_2$ cismi üzerindeki kodun boyutu) ve t (Goppa kodunu üretirken kullanılan polinomun derecesi)'dir. Bu parametreler açık anahtarlar olarak kullanılır. Gizli anahtarlar ise L (Goppa kodunu üretirken kullanılan nokta kümesi), g (kodu üretirken kullanılan polinom), P (permütasyon matrisi) ve S (singüler olmayan matris)'tir. Parametreleri verdiğimize göre artık algoritmanın adımlarına geçebiliriz. Algoritmanın adımlarını temel iletişim senaryomuzdaki Arif ve Buse üzerinden anlatalım.

1. Adım : İlk olarak Arif, anahtarları belirlemek için güvenlik seviyesi bakımından uygun olacak şekilde parametreler belirler.

$$n, k, t, L, g, P, S$$

Burada S matrisi $k \times k$ ve P matrisi $n \times n$ formundadır.

2. Adım : Arif seçtiği parametrelere göre bir $[n, k]$-Goppa kodu ve kodun üreteç matrisi olan G'yi belirler. Kodun parametreleri dolayısı ile G matrisi $k \times n$ formunda olur.

3. Adım : Ardından $\widehat{G} = S.G.P$ matrisini hesaplar. Bu matris $k \times n$ formundadır.

4. Adım : Elde ettiği değerlerden $\widehat{G}$ ve t'yi açık anahtarlar olarak gönderir. S, P, G matrislerini ise gizli anahtarlar olarak tutar.

5. Adım : Buse, Arif'e göndereceği M açık metnini belirler. Bu mesaj k uzunluğundadır.

6. Adım : Daha sonra, Buse n uzunluğunda olan ve Hamming ağırlığı t olan bir Z vektörü seçer.

7. Adım : Ardından, Buse $c = M.\widehat{G} + Z$ şifreli metnini hesaplayarak Arif'e gönderir.

8. Adım : Arif, gizli anahtarı olan P permütasyon matrisini kullanarak P^{-1} matrisini hesaplar.

9. Adım : Son olarak Arif, c şifreli metnini P^{-1} ile çarparak M mesajına ulaşır. Çarpma ve sadeleştirmeleri yaptıktan sonra birkaç farklı M mesajına ulaşılabilir. Burada, Goppa

kodları için kod çözme algoritması (4.1.1.3.2) kullanılır.

$$\begin{aligned} c.P^{-1} &= (M.S.G.P + Z).P^{-1} \\ &= M.S.G.P.P^{-1} + Z.P^{-1} \\ &= M.S.G + Z.P^{-1} \end{aligned}$$

Burada $Z.P^{-1}$ vektörünün ağırlığı ve seçilen Goppa kodunun üreteç polinomunun derecesi t olduğundan kod çözme algoritması başarılı sonuç verir ve böylelikle M açık metnine ulaşılır.

4.1.3.2 Niederreiter

Niederreiter algoritması 1986 yılında Harald Niederreiter tarafından ortaya atılmıştır. Yapı olarak McEliece algoritmasının bir çeşidi olarak görülebilir. Her iki algoritmada da Goppa kodları kullanılır. Dolayısıyla McEliece şifreleme sistemiyle neredeyse aynı güvenlik seviyesine sahiptir. Farklılaştıkları kısım ise açık anahtar üretim işlemleri, kodlama ve kod çözme adımlarıdır. Şimdi algoritmanın adımlarını yine temel iletişim senaryomuz üzerinden görelim:

1. Adım : Arif $n \times n$ formunda bir P permütasyon matrisi ve $(n-k) \times (n-k)$ formunda singüler olmayan bir S matrisi belirler.

2. Adım : Ardından t dereceli bir Goppa polinomu ve $\mathbb{F}_q$ cismi üzerinde n elemanlı bir L nokta kümesi seçerek $\Gamma(L, g)$ Goppa kodunu belirler ve bu kodun $(n-k) \times n$ formundaki H eşlik denetim matrisini oluşturur.

3. Adım : Arif, $\widehat{H} = S.H.P$ çarpımını yapar. $\widehat{H}$, matrisini açık anahtar olarak verir ve S, H ve P matrislerini gizli anahtarlar olarak saklar.

4. Adım : Ardından Buse, Arif'e göndermek istediği M mesajını belirler. Bu mesajın Hamming ağırlığı t ve uzunluğu n'dir.

5. Adım : Daha sonra $c = \widehat{H}.M^{\top}$ işlemini yaparak mesajı şifreler ve Arif'e gönderir.

6. Adım : Buse'den şifreli c mesajını alan Arif kendi gizli anahtarlarını da kullanarak $H.Z^{\top} = S^{-1}.c$ eşitliğini sağlayan bir Z vektörü bulur.

7. Adım : Goppa kod çözme algoritmasını (4.1.1.3.2) kullanarak $Z - M.P^{\top}$ kod kelimesini ve $M.P^{\top}$ hata vektörünü elde eder. Arif P matrisini bildiği için $M.P^{\top}$ vektörünü kullanarak M mesajına kolaylıkla geçebilir.

McEliece ve Niederreiter algoritmalarının işleyişini öğrendiğimize göre şimdi Klasik McEliece algoritmasına geçebiliriz.

4.1.3.3 Klasik McEliece

Klasik McEliece algoritması esasında bir anahtar kapsülleme algoritmasıdır. Anahtar kapsülleme algoritmalarını basitçe, klasik kriptosistemlerdeki anahtar değişim algoritmalarının yerini tutuyor olarak tanımlayabiliriz. Anahtar kapsülleme algoritmaları üç aşamadan oluşur:

1. Anahtar üretimi
2. Kapsülleme
3. Kapsülden Çıkarma

Anahtar üretim aşaması, girdi olarak bit cinsinden güvenlik parametrelerini alan ve çıktı olarak gizli ve açık anahtar veren olasılıksal bir algoritmadır. Kapsülleme aşaması ise açık anahtarı alıp bir ortak anahtar ve bir şifreli metin çifti veren algoritmadır. Kapsülden çıkarma aşaması ise gizli anahtarı ve kapsülleme algoritmasından çıkan şifreli metin çiftinin birini alıp bir ortak anahtar veren veya hata ile sonuçlanan bir algoritmadır. Bu anlattıklarımızı algoritma adımlarında daha net göreceğiz. Şimdi temel iletişim senaryomuz üzerinden algoritma adımlarını görelim:

Önce Buse, aşağıdaki adımları uygulayarak kendi anahtar çiftini üretir:

1. Adım : İlk olarak Buse $\mathbb{F}_{2^m}[z]$ polinom halkasından t dereceli, monik, indirgenemez bir $g(z)$ polinomu seçer.

2. Adım : Ardından bütün elemanları farklı olacak şekilde bir $\mathcal{A} = \{\alpha_1, \alpha_2, \ldots, \alpha_n\} \subseteq \mathbb{F}_{2^m}$ noktalar kümesi belirler.

3. Adım : $\forall\, i = 1, 2, \ldots, t$ ve $\forall\, 1, 2, \ldots, n$ için $h_{ij} = \alpha_j^{i-1} g(\alpha_i)^{-1}$ olacak şekilde $t \times n$ formundaki $\widehat{H} = [h_{ij}]$ matrisini hesaplar.

4. Adım : Daha sonra, $\mathbb{F}_{2^m}$ ve $\mathbb{F}_2^m$ uzayları arasındaki izomorfizmayı kullanarak $\widehat{H}$ matrisinin her bir girdisini $\mathbb{F}_2^m$ uzayındaki vektörlerle değiştirerek $mt \times n$ formundaki $\widetilde{H}$ matrisini oluşturur.

5. Adım : Eğer mümkünse, elde ettiği $\widetilde{H}$ matrisine satır indirgeme işlemlerini uygulayarak $H = [I_{mt} \mid T_{mt\times(n-mt)}]$ matrisini oluşturur.

6. Adım : Son olarak n bit uzunluğunda rastgele bir s dizisi oluşturur.

7. Adım : T matrisini açık anahtar olarak verir ve $s, g(z), \alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n$ parametrelerini gizli anahtar olarak tutar.

Buse'den T'yi alan Arif, aşağıdaki adımları uygulayarak anahtar kapsülleme sürecini yürütür:

8. Adım : İlk olarak Arif, Hamming ağırlığı t olan bir $e \in \mathbb{F}_2^n$ vektörü seçer.

9. Adım : Kodlama Adımı: Buse'nin açık anahtarı olan T'yi kullanarak önce H matrisini, ardından da $C_0 = H.e = [I \mid T].e$ vektörünü hesaplar.

10. Adım : $C_1 = \mathbf{H}(2, e)$ vektörünü hesaplar. (Buradaki **H**, Busenin anahtarı değil, SHAKE256 adındaki hash algoritmasıdır.) Sonra C_0 ile C_1 vektörlerini birleştirerek $C = (C_0, C_1)$ şifreli metnini üretir. Burada C'nin uzunluğu $mt + 256$ bittir.

11. Adım : Son olarak Arif, yine SHAKE256 algoritmasını kullanarak $K = \mathbf{H}(1, e, C)$ ortak anahtarını hesaplar.

Buse, Arif'in elde ettiği C şifreli metnini alır ve bir K' ortak anahtar adayına ulaşmak için aşağıdaki adımları uygulayarak anahtarı kapsülden çıkarma sürecini yürütür:

12. Adım : İlk olarak Buse, aldığı C şifreli metnini $C_0 \in \mathbb{F}_2^{mt}$ ve $C_1 \in \mathbb{F}_2^{256}$ olarak ayırır.

13. Adım : Kod Çözme Adımı:

(i) : $K = \mathbf{H}(1, e, C)$ ortak anahtarındaki 1 sayısı b olarak ayarlanır.

(ii) : Girdi olarak C_0 şifreli metni ve $s, g(z), \alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n$ gizli anahtarı kullanılır.

(iii) : C_0 vektörünün sonuna $n - mt$ tane sıfır eklenerek $v = (C_0, 0, 0, \ldots, 0)$ vektörüne

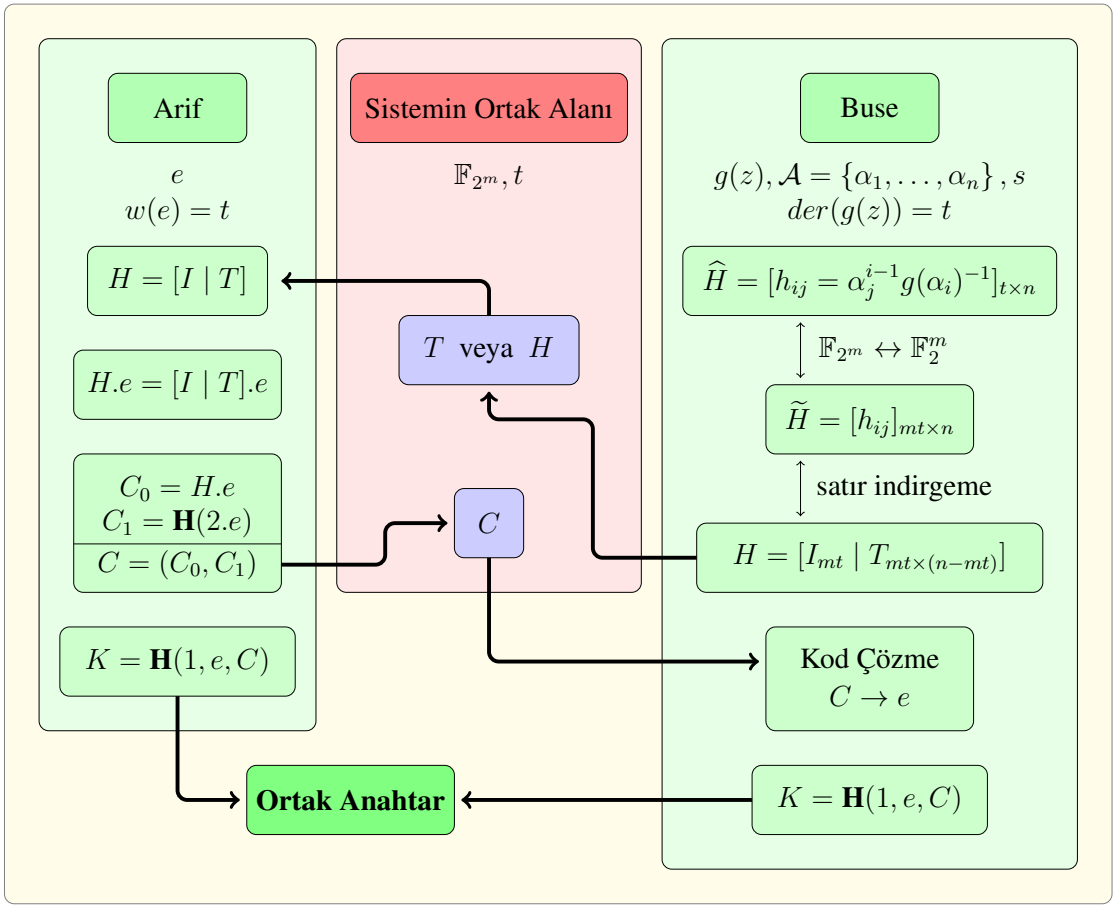
genişletilir.

(iv) : Kod çözme algoritması kullanılarak, $\Gamma(g(z), \mathcal{A})$ Goppa kodunda $dist(c, v) \leq t$ olacak şekildeki tek türlü c kod kelimesi bulunur. Eğer böyle bir c bulunuyorsa, $e = v + c$ olarak yazılır. Eğer $w(e) = t$ ve $C_0 = H.e$ ise kod çözme adımı e olarak sonuçlanır, aksi halde adım **(v)**'e geçilir.

(v) : Adım **(iv)** negatif sonuçlandıysa, s vektörü e olarak ve 0 sayısı da b olarak ayarlanır ve adım **(vi)**'dan devam edilir.

(vi) : $C_1' = \mathbf{H}(2, e)$ hesaplanır ve $C_1' = C_1$ olup olmadığı kontrol edilir.

(vii) : $K' = \mathbf{H}(b, e, C)$ ortak anahtar adayı hesaplanır. Eğer kapsülden çıkarma adımlarında hiçbir hata yapılmadıysa C_1' ile C_1 eşleşir ve K' ile K anahtarları özdeş olurlar. Yani, tüm bu süreç sonunda Arif ile Buse'nin ortak anahtarı K olarak bulunur.



Şekil 4.2. Klasik McEliece algoritması

Algoritmanın daha iyi anlaşılması için küçük parametreler kullanarak temel iletişim senaryomuz üzerinden bir örnek verelim.

Örnek 4.1.28. İlk olarak algoritmada kullanılacak parametreler belirlenir. Üzerinde çalışılacak cisim

$$\mathbb{F}_{2^4} = \frac{\mathbb{F}_2[x]}{\langle x^4 + x^3 + 1 \rangle} = \mathbb{F}_2(\beta)$$

olsun. Burada β, $\mathbb{F}_{2^4}$ cisminin bir ilkel elemanı olup $x^4 + x^3 + 1$ polinomunun bir köküdür. Böylece cismimizi aşağıdaki şekilde yazabiliriz:

$$\mathbb{F}_{2^4} = \left\{0, 1, \beta, \beta^2, \ldots, \beta^{14}\right\} \quad , \quad \beta^{15} = 1$$

Hata düzeltme kapasitemiz, yani t parametremiz 2 olsun.

Parametreler belirlendikten sonra Buse, $t = 2$ dereceli indirgenemez polinom olarak $g(z) = z^2 + z + \beta$ polinomunu seçmiş olsun.

Şimdi $\mathbb{F}_{2^4}$ cisminin elemanlarının ikilik sistemdeki karşılıklarını bulup ardından $\widehat{H}$ matrisinin girdilerini üreterek matrisi oluşturalım.

$$\begin{array}{lllllllllll}
0 & = & & & & & & & & = & (0,0,0,0) \\
1 & = & 1 & & & & & & & = & (1,0,0,0) \\
\beta & = & & & \beta & & & & & = & (0,1,0,0) \\
\beta^2 & = & & & & & \beta^2 & & & = & (0,0,1,0) \\
\beta^3 & = & & & & & & & \beta^3 & = & (0,0,0,1) \\
\beta^4 & = & 1 & & & & & + & \beta^3 & = & (1,0,0,1) \\
\beta^5 & = & 1 & + & \beta & & & + & \beta^3 & = & (1,1,0,1) \\
\beta^6 & = & 1 & + & \beta & + & \beta^2 & + & \beta^3 & = & (1,1,1,1) \\
\beta^7 & = & 1 & + & \beta & + & \beta^2 & & & = & (1,1,1,0) \\
\beta^8 & = & & & \beta & + & \beta^2 & + & \beta^3 & = & (0,1,1,1) \\
\beta^9 & = & 1 & & & + & \beta^2 & & & = & (1,0,1,0)
\end{array}$$

$$
\begin{array}{rcccccccl}
\beta^{10} & = & & \beta & & & + & \beta^3 & = (0,1,0,1) \\
\beta^{11} & = & 1 & & + & \beta^2 & + & \beta^3 & = (1,0,1,1) \\
\beta^{12} & = & 1 & + & \beta & & & & = (1,1,0,0) \\
\beta^{13} & = & & \beta & + & \beta^2 & & & = (0,1,1,0) \\
\beta^{14} & = & & & & \beta^2 & + & \beta^3 & = (0,0,1,1)
\end{array}
$$

Matrisin girdilerini $h_{ij} = \beta_j^{i-1} g(\beta_i)^{-1}$ şeklinde hesaplayacağımızı algoritmayı anlatırken belirtmiştik.

$$
\begin{array}{rcllll}
h_{1,1} & = & (1+1+\beta)^{-1} & = \beta^{-1} & = \beta^{14} & \\
h_{1,2} & = & (0+0+\beta)^{-1} & = \beta^{-1} & = \beta^{14} & \\
h_{1,3} & = & (\beta^2+\beta+\beta)^{-1} & = \beta^{-2} & = \beta^{13} & \\
h_{1,4} & = & (\beta^4+\beta^2+\beta)^{-1} & = (\beta^3+\beta^2+\beta+1)^{-1} & = (\beta^6)^{-1} & = \beta^9 \\
h_{1,5} & = & (\beta^6+\beta^3+\beta)^{-1} & = \beta^6 & & \\
\vdots & & & & & \\
h_{2,1} & = & 0 \cdot (1+1+\beta)^{-1} & = 0 & & \\
h_{2,2} & = & 1 \cdot (0+0+\beta)^{-1} & = \beta^{-1} & = \beta^{14} & \\
h_{2,3} & = & \beta \cdot (\beta^2+\beta+\beta)^{-1} & = \beta \cdot \beta^{-2} & = \beta^{-1} & = \beta^{14} \\
h_{2,4} & = & \beta^2 \cdot (\beta^4+\beta^2+\beta)^{-1} & = \beta^2 \cdot \beta^9 & = \beta^{11} & \\
h_{2,5} & = & \beta^3 \cdot \beta^6 & = \beta^9 & & \\
\vdots & & & & &
\end{array}
$$

Tüm girdiler hesaplandığında $\widehat{H}$ matrisi aşağıdaki şekilde elde edilir:

$$
\widehat{H} = \begin{bmatrix}
\beta^{14} & \beta^{14} & \beta^{13} & \beta^9 & \beta^6 & \beta^6 & \beta^3 & \beta^7 & \beta^{11} & \beta^7 & \beta^9 & \beta^3 & \beta^{12} & \beta^{13} & \beta^{11} & \beta^{12} \\
0 & \beta^{14} & \beta^{14} & \beta^{11} & \beta^9 & \beta^{10} & \beta^8 & \beta^{13} & \beta^3 & 1 & \beta^3 & \beta^{13} & \beta^8 & \beta^{10} & \beta^9 & \beta^{11}
\end{bmatrix}
$$

$\mathbb{F}_{2^4}$ ile $\mathbb{F}_2^4$ arasındaki izomorfizmayı kullanarak $\widetilde{H}$ matrisini oluşturalım:

$$\widetilde{H} = \left[\begin{array}{cccccccccccccccc} 0&0&0&1&1&1&0&1&1&1&1&0&1&0&1&1 \\ 0&0&1&0&1&1&0&1&0&1&0&0&1&1&0&1 \\ 1&1&1&1&1&1&0&1&1&1&1&0&0&1&1&0 \\ 1&1&0&0&1&1&1&0&1&0&0&1&0&0&1&0 \\ \hline 0&0&0&1&1&0&0&0&0&1&0&0&0&0&1&1 \\ 0&0&0&0&0&1&1&1&0&0&0&1&1&1&0&0 \\ 0&1&1&1&1&0&1&1&0&0&0&1&1&0&1&1 \\ 0&1&1&1&0&1&1&0&1&0&1&0&1&1&0&1 \end{array}\right]$$

Elde ettiğimiz $\widetilde{H}$ matrisine satır indirgeme işlemlerini uygulayarak aşağıdaki H matrisini elde ederiz:

$$H = \left[\begin{array}{cccccccc|cccccccc} 1&0&0&0&0&0&0&0&0&1&0&0&0&1&1&1 \\ 0&1&0&0&0&0&0&0&0&1&1&1&0&0&0&1 \\ 0&0&1&0&0&0&0&0&1&1&1&1&1&0&1&1 \\ 0&0&0&1&0&0&0&0&0&1&0&1&1&1&0&1 \\ 0&0&0&0&1&0&0&0&0&0&0&1&1&1&1&0 \\ 0&0&0&0&0&1&0&0&1&1&0&0&1&1&1&0 \\ 0&0&0&0&0&0&1&0&1&0&1&1&0&1&0&0 \\ 0&0&0&0&0&0&0&1&0&1&1&0&0&1&1&0 \end{array}\right] = \left[I_8 \mid T\right]$$

Son olarak Buse $n = 2^4$ bit uzunluğunda rastgele bir s vektörü seçer. Seçtiği vektör $s = (0000000000000000)$ olsun. Ardından T matrisini herkese açık bir şekilde Arif'e gönderir.

Buse'den T matrisini alan Arif, anahtar kapsülleme sürecini başlatır. Bunun için ilk olarak Hamming ağırlığı $w(e) = t = 2$ olan rastgele bir e vektörü seçer. Seçtiği vektör $e = 1100000000000000$ olsun.

Daha sonra Arif Buse'den aldığı T matrisinin sol tarafına 8×8 formundaki birim matrisi ekleyerek H matrisini oluşturur ve $C_0 = He$ vektörünü hesaplar.

$$C_0 = He = (11000000)$$

Ardından SHAKE256 hash algoritmasını kullanarak $C_1 = \mathbf{H}(2, e)$ değerini hesaplar.

$$C_1 = 26fe36f811ac8fe9f19ba997a39d3682ef06b29509cca1903ffe4a0b247c833f$$

Bu değer 16'lık tabanda yazılmıştır. Bu değerin ikilik tabandaki yazımı da şu şekildedir:

$$\begin{aligned} C_1 = \mathbf{H}(2, e) \quad = \quad & 00100110111111100011011011111000 \\ & 00010001101011001000111111101001 \\ & 11110001100110111010100110010111 \\ & 10100011100111010011011010000010 \\ & 11101111000001101011001010010101 \\ & 00001001110011001010000110010000 \\ & 00111111111111100100101000001011 \\ & 00100100011111001000001100111111 \end{aligned}$$

Ardından C_0 ile C_1 vektörlerini birleştirerek $C = (C_0, C_1)$ şifreli metnini yazar ve herkese açık olacak şekilde Buse'ye gönderir.

$$\begin{aligned} C = (C_0, C_1) = \quad & 11000000001001101111111000110 1101 \\ & 11110000001000110101100100011 1111 \\ & 10100111110001100110111010100 1100 \\ & 10111101000111001110100110110 1000 \\ & 00101110111100000110101100101 0010 \\ & 10100001001110011001010000110 0100 \\ & 00001111111111111001001010000 0101 \\ & 10010010001111100100000110011 1111 \end{aligned}$$

Arif son olarak tekrardan SHAKE256 hash algoritmasını kullanarak ortak anahtar olan $K = \mathbf{H}(1, e, C)$'yi hesaplar.

$$K = 90d7c9dccc4689f6894b1b6e58ee9b38328e4df9937536eb9b5715a38ee4e1be$$

Hash algoritması sonucunda elde edilen bu değer 16'lık tabandadır ve bu değerin ikilik tabandaki karşılığı şu şekildedir:

$$\begin{aligned} K = \mathbf{H}(1, e, C) = \quad & 10010000110101111100100111011100 \\ & 11001100010001101000100111110110 \\ & 10001001010010110001101101101110 \\ & 01011000111011101001101100111000 \\ & 00110010100011100100110111111001 \\ & 10010011011101010011011011101011 \\ & 10011011010101110001010110100011 \\ & 10001110111001001110000110111110 \end{aligned}$$

Böylelikle hedeflenen ortak anahtarı Arif herkesten gizli bir şekilde elde etmiş oldu. Şimdi de yine herkesten gizli bir şekilde anahtarı Buse'nin elde etmesi gerekmektedir. Bunun için Arif'ten aldığı C şifreli metnini kullanarak anahtarı kapsülden çıkarma sürecini başlatır.

Önce Arif'ten aldığı C vektöründen yararlanarak 16 bit uzunluğundaki

$$v = (C_0, 00000000) = (1100000000000000)$$

vektörünü yazar ve bu $g(z)$ polinomu kullanılarak üretilen Goppa kodunda bu vektöre en yakın vektörü bulmak için kod çözme algoritmasını başlatır.

İlk adım olarak $S(v)$ sendrom vektörünü hesaplar:

$$S(v) = \frac{1}{z} + \frac{1}{z+1} \pmod{g(z)} \qquad \Rightarrow \qquad S(v) = \beta^{13}$$

Daha sonra $S(z)\sigma(z) \equiv w(z) \pmod{g(z)}$ denkliğini uygular:

$$(z^2 + (\alpha_1 + \alpha_2+)z + \alpha_1\alpha_2)\beta^{13} = \alpha_1 + \alpha_2 \pmod{g(z)}$$

Bu denklikte her iki tarafta aynı terimlerin katsayıları eşitlendiğinde aşağıdaki denklemlere ulaşılır:

$$\alpha_1 + \alpha_2 = 0 \quad \text{ve} \quad \alpha_1\alpha_2 = 0$$

Bu iki denklemden $\alpha_1 = 0$ ve $\alpha_2 = 0$ sonuçları elde edilir. Bunun anlamı, kullanılan Goppa kodunda v vektörüne en yakın vektörün (1100000000000000) olduğudur. Yani $e = (1100000000000000)$ olarak bulunur. Ayrıca $e = v+c$ eşitliğinden $c = (0000000000000000)$ kod kelimesi de elde edilir.

Sonuç olarak Buse, kod çözme işlemiyle Arif'in seçtiği e vektörünü elde etmiştir. Buna ek olarak C şifreli metnine de sahiptir. Yani SHAKE256 hash algoritmasını kullanarak $\mathbf{H}(1, e, C)$ değerini hesaplayabilecek kadar veriyi elde etmiştir. Biliyoruz ki bu değer Arif'in de elde ettiği K anahtarıdır. Böylelikle hem Arif hem de Buse herkesten gizli olarak bir ortak anahtara sahip olmuşlardır.

4.1.4 McNie

Bu kısımda McNie algoritması anlatılmıştır ve bu anlatım yapılırken [20] ve [25] numaralı kaynaktan yararlanılmıştır. Öncelikle belirtmeliyiz ki McNie algoritması, McEliece ve Niederreiter algoritmalarının bir kombinasyonudur. McNie algoritmasının güvenilirlik seviyesi, kodlama teorisindeki zor problemlerden biri olan Sendrom kod çözme problemine dayanır. Ayrıca, McNie algoritmasında üretilen açık anahtar, gizli anahtarı maskelemek için de kullanılan rastgele bir üreteç matrisini de içermektedir. Böylelikle, McNie algoritması, bilinen ataklara karşı alışılmışın dışında bir direnç göstermektedir.

McNie algoritmasının kullandığı kod ailesi, QC-LRPC (4.1.1.2) kodlardır.

McNie Algoritması bir açık anahtarlı şifreleme algoritmasıdır. Kuantum-sonrası algoritmalarda açık anahtarlı şifreleme algoritmaları üç aşamadan oluşur:

1. Anahtar üretimi

2. Şifreleme

3. Deşifreleme

McNie şifreleme algoritmasının anahtar üretim aşamasında 3 tane gizli ve 2 tane açık olmak üzere toplamda 5 anahtar üretilir. Şifreleme aşamasında c_1 ve c_2 olmak üzere iki adet şifreli metin oluşturulup ardından birleştirilerek tek bir metin haline getirilir. Deşifreleme aşamasında c_1 ve c_2'nin birlikte bulunduğu bir denklem çözülerek açık metin elde edilir.

McNie algoritması, temelinde LRPC kod ailesiyle çalışır. Fakat anahtar boyutlarını küçültmek için McNie algoritmasının QC-LRPC kod ailesiyle çalışan versiyonu yapılmıştır. Algoritmanın farklı versiyonları arasında çok büyük farklar yoktur fakat anahtar üretim aşamasındaki fark göz ardı edilecek kadar küçük değildir. Bu sebeple, önce McNie algoritmasının LRPC kodlarla çalışan versiyonunu görelim, ardından QC-LRPC kodlarla anahtar üretiminin nasıl yapıldığını anlatalım.

Not : LRPC kodlar için kod çözme algoritmasını Kısım 4.1.1.2.2'de gördük. Bu sebeple McNie algoritmasının işlem sürecini anlatırken kod çözme algoritmasını tekrar açmadan kullanacağız ve Φ_H ile göstereceğiz.

4.1.4.1 LRPC Kodlarla McNie

Algoritmanın işlem sürecini temel iletişim senaryomuz üzerinden anlatalım.

Birinci Aşama : (Anahtar Üretimi) İlk olarak Arif, uygun parametreler seçer ve bu parametrelere göre anahtarları üretir:

- Sonlu $\mathbb{F}_{q^m}$ cismi belirlenir.
- $\mathbb{F}_{q^m}$ cismi üzerinde uzunluğu n olan, k boyutlu bir C LRPC kodu seçilir.

- C kodunun denklik kontrol matrisi H oluşturulur. Bu matris $(n-k) \times n$ formundadır. (Burada işlemi kısaltmak için, C kodunu seçip denklik kontrol matrisini bulmak yerine doğrudan bir H denklik kontrol matrisi seçilerek de başlanabilir.)
- Ardından, $n \times n$ formunda bir P permütasyon matrisi keyfi seçilir.
- Daha sonra $(n-k) \times (n-k)$ formunda ve tersinir olan keyfi bir S matrisi seçilir.
- $l \times n$ formunda keyfi bir G' matrisi seçilir.
- Son olarak $F = G'P^{-1}H^{\top}S$ çarpımı hesaplanır. F matrisi $l \times (n-k)$ formundadır.

Gerekli seçimleri ve hesaplamaları yaptıktan sonra Arif, G' ve F matrislerini herkese açık bir biçimde paylaşır. H, S ve P matrislerini ise bir anahtar paylaşma algoritması kullanarak gizli olarak Buse'ye iletir.

İkinci Aşama : (Şifreleme) Arif, Buse'ye göndermek istediği m mesajını belirler. Mesaj l uzunluğunda olmalıdır. Ardından aşağıdaki adımları izleyerek mesajı şifreler:

- n uzunluğunda bir e hata vektörü seçilir. e vektörünün rankı, kullanılan kodun kod çözme kapasitesini geçmemeli.
- $c_1 = mG' + e$ vektörü hesaplanır. Bu vektör n uzunluğundadır.
- $c_2 = mF$ vektörü hesaplanır. Bu vektör $n-k$ uzunluğundadır.
- Son olarak c_1 ve c_2 art arda eklenerek $c = (c_1, c_2)$ şifreli metni elde edilir.

Şifreleme aşamasını tamamlayan Arif, elde ettiği c şifreli metnini Buse'ye gönderir.

Üçüncü Aşama : (Deşifreleme) Arif'ten şifreli metni alan Buse aşağıdaki adımları izleyerek m açık metnine ulaşır:

- İlk olarak c metni c_1 ve c_2 şeklinde tekrar iki parçaya ayrılır.

- Daha sonra s' sendrom vektörü aşağıdaki işlemle hesaplanır:

$$\begin{aligned} s &= c_1 P^{-1} H^\top - c_2 S^{-1} \\ &= (mG' + e)P^{-1}H^\top - (mG'P^{-1}H^\top S)S^{-1} \\ &= eP^{-1}H^\top \end{aligned}$$

- Φ_H kod çözme algoritması kullanılarak s vektöründen eP^{-1} elde edilir. Yani $\Phi_H(s) = eP^{-1}$'dir.

- Elde edilen eP^{-1} vektörünü sağdan P ile çarparak e hata vektörüne ulaşılır.

- Son olarak Buse, $c_1 = mG' + e$ eşitliğini $mG' = c_1 - e$ şeklinde düzenleyip bir denklem sistemi elde eder ve bu denklem sistemini çözerek m açık metnine ulaşır.

Algoritmanın çalışma prensibini gördük, şimdi McNie Algoritmasında anahtar üretiminin QC-LRPC kodlar ile nasıl yapıldığını görelim:

4.1.4.2 QC-LRPC Kodlarla McNie

Anahtar üretim aşamasının önce 3-yarı devirli LRPC kodlarla, ardından 4-yarı devirli LRPC kodlarla olan versiyonunu görelim:

4.1.4.2.1 3-Yarı Devirli LRPC Kodlarla McNie

- İlk olarak sonlu bir $\mathbb{F}_{q^m}$ cismi belirlenir.

- Ardından $\mathbb{F}_{q^m}$ üzerinde n uzunluğunda bir $(3, 2)$-yarı devirli LRPC kodu seçilir. Seçilen kod $(3, 2)$-yarı devirli olduğundan n sayısı 3'ün bir tam katı olmalıdır.
  Algoritmayı kısaltmak için doğrudan 3-yarı devirli bir denklik kontrol matrisi seçilerek başlanabilir. Bunun için öncelikle $\mathbb{F}_{q^m}$ üzerinde $\frac{n}{3}$ uzunluğunda 3 vektör belirlenir. Bu vektörlere h_1, h_2 ve h_3 diyelim ve $\forall\ i = 1, 2, 3$ için $h_i = (h_{i1}, h_{i2}, \ldots, h_{i\frac{n}{3}})$ olsun.

Ardından bu vektörleri ilk satırları kabul eden $\frac{n}{3} \times \frac{n}{3}$ formunda üç tane devirli matris yazılır. Bu matrislere H_1, H_2 ve H_3 diyelim:

$$H_1 = \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1\frac{n}{3}} \\ h_{1\frac{n}{3}} & h_{11} & \cdots & h_{1(\frac{n}{3}-1)} \\ \vdots & \vdots & \ddots & \vdots \\ h_{12} & h_{13} & \cdots & h_{11} \end{bmatrix}$$

$$H_2 = \begin{bmatrix} h_{21} & h_{22} & \cdots & h_{2\frac{n}{3}} \\ h_{2\frac{n}{3}} & h_{21} & \cdots & h_{2(\frac{n}{3}-1)} \\ \vdots & \vdots & \ddots & \vdots \\ h_{22} & h_{23} & \cdots & h_{21} \end{bmatrix}$$

$$H_3 = \begin{bmatrix} h_{31} & h_{32} & \cdots & h_{3\frac{n}{3}} \\ h_{3\frac{n}{3}} & h_{31} & \cdots & h_{3(\frac{n}{3}-1)} \\ \vdots & \vdots & \ddots & \vdots \\ h_{32} & h_{33} & \cdots & h_{31} \end{bmatrix}$$

Bu devirli matrisler yan yana eklenerek H denklik kontrol matrisi oluşturulur:

$$H = \begin{bmatrix} H_1 & H_2 & H_3 \end{bmatrix}_{\frac{n}{3} \times n}$$

- Sıradaki adım G' matrisini üretmektir. G' matrisini üretmek için önce $\mathbb{F}_{q^m}$ cisminden $\frac{n}{3}$ uzunluğunda iki vektör seçilir ve bu vektörleri ilk satırları kabul eden devirli matrisler oluşturulur. Seçilen matrisler $g_1 = (g_{11}, g_{12}, \ldots, g_{1\frac{n}{3}})$ ve $g_2 = (g_{21}, g_{22}, \ldots, g_{2\frac{n}{3}})$ olsun ve bu matrislerden ürtilen devirli matrislere G_1 ve G_2 diyelim:

$$G_1 = \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1\frac{n}{3}} \\ g_{1\frac{n}{3}} & g_{11} & \cdots & g_{1(\frac{n}{3}-1)} \\ \vdots & \vdots & \ddots & \vdots \\ g_{12} & g_{13} & \cdots & g_{11} \end{bmatrix}$$

$$G_2 = \begin{bmatrix} g_{21} & g_{22} & \cdots & g_{2\frac{n}{3}} \\ g_{2\frac{n}{3}} & g_{21} & \cdots & g_{2(\frac{n}{3}-1)} \\ \vdots & \vdots & \ddots & \vdots \\ g_{22} & g_{23} & \cdots & g_{21} \end{bmatrix}$$

Bu matrisler kullanılarak G' matrisi aşağıdaki şekilde oluşturulur:

$$G' = \left[\begin{array}{cc|c} I_{\frac{n}{3}} & & G_1 \\ & I_{\frac{n}{3}} & G_2 \end{array}\right]_{\frac{2n}{3}\times n}$$

- Daha sonra $n \times n$ formunda P permütasyon matrisi üretilir. Bu permütasyon matrisi için koşul P^{-1} matrisinin 9 tane $\frac{n}{3} \times \frac{n}{3}$ formunda devirli matrisin birleşmesiyle oluşmasıdır, yani P^{-1} matrisi aşağıdaki şekilde olmalıdır:

$$P^{-1} = \begin{bmatrix} P_1 & P_2 & P_3 \\ P_4 & P_5 & P_6 \\ P_7 & P_8 & P_9 \end{bmatrix} \text{ öyle ki } \forall\, i = 1, 2, \ldots, 9 \text{ için } P_i \text{ devirli matris}$$

- Daha sonra $\frac{n}{3} \times \frac{n}{3}$ formunda keyfi bir tersinir ve devirli S matrisi üretilir.

- $F = G'P^{-1}H^{\top}S$ hesabı yapılarak $\frac{2n}{3} \times \frac{n}{3}$ formunda F matrisi oluşturulur. Ardından F matrisine sütun indirgeme işlemleri uygulanır. Eğer F matrisinin sütunca indirgenmiş hali bir birim matris ve bir devirli matrisin üst üste eklenmesiyle oluşturulmuş gibi durmuyorsa h_i ve g_i vektörleri yeniden seçilerek işlemlere tekrar başlanır.

$$
\begin{aligned}
F &= G'P^{-1}H^\top S \\
&= \left[\begin{array}{cc|c} I_{\frac{n}{3}} & & G_1 \\ & I_{\frac{n}{3}} & G_2 \end{array}\right]_{\frac{2n}{3}\times n} \left[\begin{array}{ccc} P_1 & P_2 & P_3 \\ P_4 & P_5 & P_6 \\ P_7 & P_8 & P_9 \end{array}\right]_{n\times n} \left[\begin{array}{c} H_1^\top \\ H_2^\top \\ H_3^\top \end{array}\right]_{n\times\frac{n}{3}} S_{\frac{n}{3}\times\frac{n}{3}} \\
&= \left[\begin{array}{ccc} P_1+G_1P_7 & P_2+G_1P_8 & P_3+G_1P_9 \\ P_4+G_2P_7 & P_5+G_2P_8 & P_6+G_2P_9 \end{array}\right]_{\frac{2n}{3}\times n} \left[\begin{array}{c} H_1^\top \\ H_2^\top \\ H_3^\top \end{array}\right]_{n\times\frac{n}{3}} S_{\frac{n}{3}\times\frac{n}{3}} \\
&= \left[\begin{array}{c} (P_1+G_1P_7)H_1^\top + (P_2+G_1P_8)H_2^\top + (P_3+G_1P_9)H_3^\top \\ (P_4+G_2P_7)H_1^\top + (P_5+G_2P_8)H_2^\top + (P_6+G_2P_9)H_3^\top \end{array}\right]_{\frac{2n}{3}\times\frac{n}{3}} S_{\frac{n}{3}\times\frac{n}{3}}
\end{aligned}
$$

Burada parametrelerin uygun seçildiğini ve elde edilen F matrisinin sütun indirgenmiş eşelon formunun istenilen formatta sonuçlandığını varsayarak devam edeceğiz.

$$
F \xrightarrow{E_1} E_1(F) \xrightarrow{E_2} E_2(E_1(F)) \xrightarrow{E_3} \cdots \xrightarrow{E_k} E_k(E_{k-1}(\ldots(E_1(F)))) = \left[\begin{array}{c} I_{\frac{n}{3}} \\ \hline F' \end{array}\right]
$$

Daha önce de söylediğimiz gibi, F matrisinin sütunca indirgenmiş eşelon formu bir birim matris ve bir devirli matrisin birleşmesiyle oluşmalıdır. Dolayısıyla, buradaki F' matrisi devirli bir matristir.

- Son olarak, F' ve G' matrisleri açık anahtarlar olarak yayımlanırken S, H ve P matrisleri gizli anahtarlar olarak tutulur.

4.1.4.2.2 4-Yarı Devirli LRPC Kodlarla McNie

- İlk olarak bir $\mathbb{F}_{q^m}$ sonlu cismi belirlenir.

- Daha sonra $\mathbb{F}_{q^m}$ cismi üzerinde n uzunluğunda $(4,2)$-yarı devirli LRPC kodu seçilir. Seçilen kod $(4,2)$-yarı devirli olduğundan n sayısı 4'ün bir tam katı olmalıdır.

  Daha önce de yaptığımız gibi, algoritmanın işlem sürecini kısaltmak için $(4,2)$-yarı devirli bir kod bulup denklik kontrol matrisini hesaplamak yerine doğrudan bir $(4,2)$-yarı devirli bir denklik kontrol koduyla başlayabiliriz. Bunun için öncelikle $\mathbb{F}_{q^m}$ cismi üzerinde $\frac{n}{4}$ uzunluğunda 8 tane vektör seçilir ve ardından bu vektörleri ilk satırı kabul eden devirli matrisler üretilir. Daha sonra bu matrisler birleştirilerek H denklik kontrol matrisi elde edilir.

  Seçilen vektörlere $h_1, h_2, \ldots, h_8$ diyelim ve $\forall\, i = 1, \ldots, 8$ için $h_i = (h_{i1}, h_{i2}, \ldots, h_{i\frac{n}{4}})$ olsun. Bu vektörleri ilk satırı kabul eden devirli matrislere $H_1, \ldots, H_8$ diyelim.

$$\forall\, i = 1, \ldots, 8 \text{ için } H_i = \begin{bmatrix} h_{i1} & h_{i2} & \cdots & h_{i\frac{n}{4}} \\ h_{i\frac{n}{4}} & h_{i1} & \cdots & h_{i(\frac{n}{4}-1)} \\ \vdots & \vdots & \ddots & \vdots \\ h_{i2} & h_{i3} & \cdots & h_{i1} \end{bmatrix}$$

  Böylelikle H matrisi aşağıdaki şekilde oluşturulur:

$$H = \begin{bmatrix} H_1 & H_2 & H_3 & H_4 \\ H_5 & H_6 & H_7 & H_8 \end{bmatrix}_{\frac{2n}{4} \times n}$$

- Ardından G' matrisi üretilir. G' matrisini üretmek için öncelikle $\mathbb{F}_{q^m}$ cismi üzerinde $\frac{n}{4}$ uzunluğunda 3 tane vektör seçilir ve bu vektörleri ilk satırı kabul eden devirli matrisler yazılır. Seçilen vektörlere g_1, g_2, g_3 diyelim ve $\forall\, i = 1, 2, 3$ için $g_i = (g_{i1}, g_{i2}, \ldots, g_{i\frac{n}{4}})$ olsun. Bu vektörlerle üretilen devirli matrislere G_1, G_2, G_3 diyelim.

$$\forall\, i = 1, 2, 3 \text{ için } G_i = \begin{bmatrix} g_{i1} & g_{i2} & \cdots & g_{i\frac{n}{4}} \\ g_{i\frac{n}{4}} & g_{i1} & \cdots & g_{i\frac{n}{4}-1} \\ \vdots & \vdots & \ddots & \vdots \\ g_{i2} & g_{i3} & \cdots & g_{i1} \end{bmatrix}$$

Bu devirli matrisler ve $\frac{n}{4} \times \frac{n}{4}$ formundaki birim matrisler birleştirilerek G' matrisi aşağıdaki şekilde oluşturulur:

$$G' = \left[\begin{array}{ccc|c} I_{\frac{n}{4}} & & & G_1 \\ & I_{\frac{n}{4}} & & G_2 \\ & & I_{\frac{n}{4}} & G_3 \end{array}\right]_{\frac{3n}{4} \times n}$$

- Daha sonra $\frac{2n}{4} \times \frac{2n}{4}$ formunda tersinir ve devirli olan keyfi bir S matrisi seçilir.

- $\mathbb{F}_q$ üzerinde $n \times n$ formunda bir P permütasyon matrisi seçilir. Bu permütasyon matrisi için koşul, matrisin tersinin $\frac{n}{4} \times \frac{n}{4}$ formunda 16 tane devirli ve tersinir matristen oluşuyor olmasıdır:

$$P^{-1} = \begin{bmatrix} P_1 & P_2 & P_3 & P_4 \\ P_5 & P_6 & P_7 & P_8 \\ P_9 & P_{10} & P_{11} & P_{12} \\ P_{13} & P_{14} & P_{15} & P_{16} \end{bmatrix} \text{ öyle ki } \forall\, i = 1, \ldots, 16 \text{ için } P_i \text{ devirli ve tersinir}$$

- $F = G' P^{-1} H^\top S$ hesabı yapılarak $\frac{3n}{4} \times \frac{2n}{4}$ formundaki F matrisi bulunur ve bu matrise sütun indirgeme işlemleri yapılarak birim matrisler ve devirli matrislerin birleşmesiyle oluşmuş bir matris elde edilir. Eğer sütun indirgeme işleminin sonucunda istenilen formatta bir matris elde edilemiyorsa h_i ve g_i vektörleri değiştirilerek işlemlere baştan başlanır.

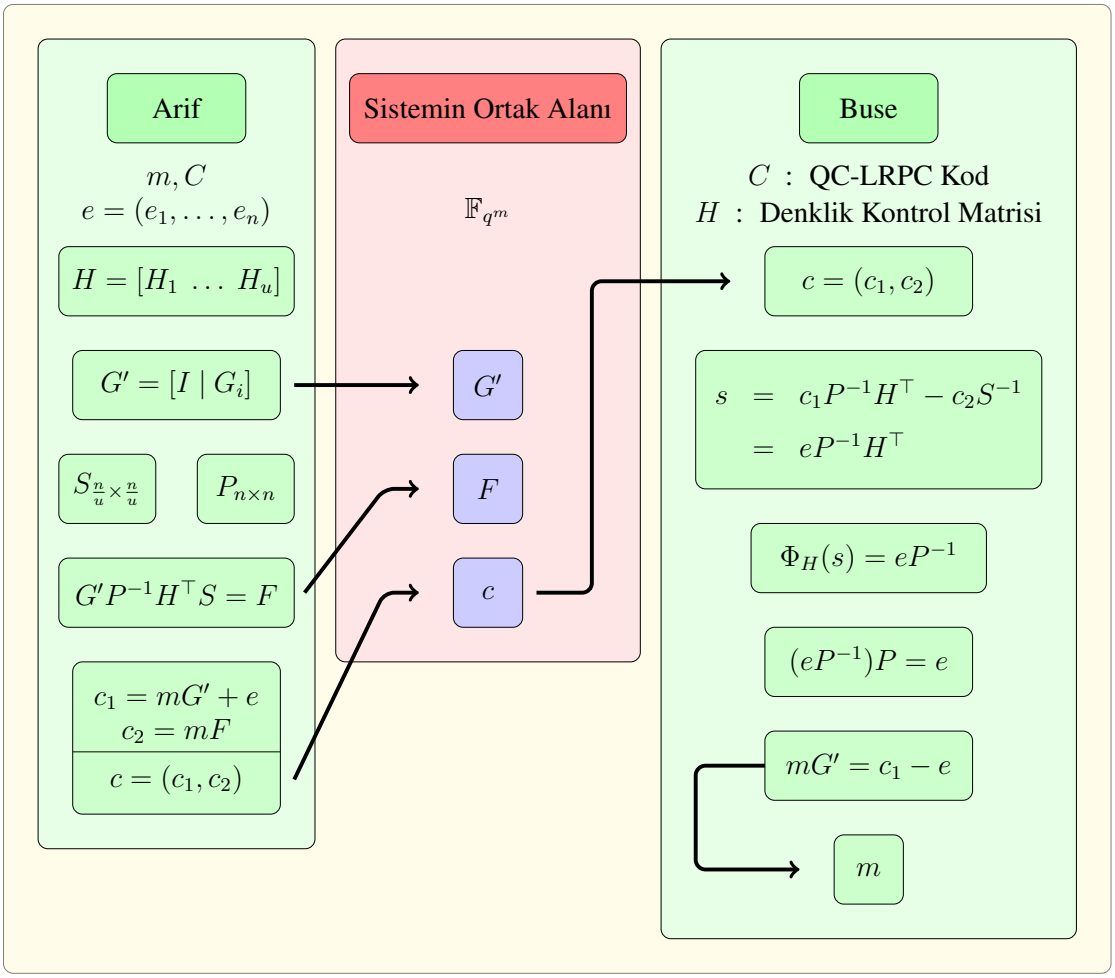
$$
\begin{aligned}
F &= G' P^{-1} H^{\top} S \\
&= \left[\begin{array}{ccc|c} I_{\frac{n}{4}} & & & G_1 \\ & I_{\frac{n}{4}} & & G_2 \\ & & I_{\frac{n}{4}} & G_3 \end{array}\right] \begin{bmatrix} P_1 & P_2 & P_3 & P_4 \\ P_5 & P_6 & P_7 & P_8 \\ P_9 & P_{10} & P_{11} & P_{12} \\ P_{13} & P_{14} & P_{15} & P_{16} \end{bmatrix} \begin{bmatrix} H_1^{\top} & H_5^{\top} \\ H_2^{\top} & H_6^{\top} \\ H_3^{\top} & H_7^{\top} \\ H_4^{\top} & H_8^{\top} \end{bmatrix} S \\
&= \begin{bmatrix} \sum_{i=1}^{4}(P_i + G_1 P_{i+12}) H_i^{\top} & \sum_{i=1}^{4}(P_i + G_1 P_{i+12}) H_{i+4}^{\top} \\ \sum_{i=1}^{4}(P_{i+4} + G_2 P_{i+12}) H_i^{\top} & \sum_{i=1}^{4}(P_{i+4} + G_2 P_{i+12}) H_{i+4}^{\top} \\ \sum_{i=1}^{4}(P_{i+8} + G_3 P_{i+12}) H_i^{\top} & \sum_{i=1}^{4}(P_{i+8} + G_3 P_{i+12}) H_{i+4}^{\top} \end{bmatrix} S
\end{aligned}
$$

Hesaplanan F matrisine sütun indirgeme işlemleri uygulandığında istenilen form elde edildiğini varsayalım. Aksi halde, daha önce de söylediğimiz gibi işlemlere en baştan başlamak gerekir.

$$
F \xrightarrow{E_1} E_1(F) \xrightarrow{E_2} E_2(E_1(F)) \xrightarrow{E_3} \cdots \xrightarrow{E_k} E_k(E_{k-1}(\ldots(E_1(F)))) = \left[\begin{array}{cc} I_{\frac{n}{4}} & \\ & I_{\frac{n}{4}} \\ \hline F' & F'' \end{array}\right]
$$

- Son olarak, F', F'' ve G' matrisleri açık anahtarlar olarak yayımlanırken S, H ve P matrisleri gizli anahtarlar olarak tutulur.

McNie algoritmasının işleyişini aşağıdaki şekil ile özetleyebiliriz:



Şekil 4.3. McNie algoritması

McNie algoritmasının nasıl işlediğini ve QC-LRPC kodları nasıl kullandığını öğrendiğimize göre bir örnek vererek daha anlaşılır hale getirebiliriz:

Örnek 4.1.29. Örneğimizi 3-yarı devirli LRPC kodlarla verelim.
İlk olarak parametreleri belirleyelim:

- Örneğimizin amacı algoritmanın daha iyi anlaşılması olduğu için işlemleri basit düzeyde tutmak daha iyi olur. Bu sebeple $q = 2$ ve $m = 1$ alalım. Yani kullanacağımız sonlu cisim $\mathbb{F}_{q^m} = \mathbb{F}_2$ olsun.

- Kullanacağımız kod $(3, 2)$-yarı devirli olsun. Uzunluğu $n = 6$ alalım. Bu durumda G üreteç matrisi 4×6 formunda $(3, 2)$-yarı devirli bir matris olur. İşlemleri biraz

kısaltmak adına kodun üreteç matrisiyle başlamak yerine, doğrudan H denklik kontrol matrisini üreterek başlayabiliriz. G üreteç matrisi $(3,2)$-yarı devirli olduğuna göre H denklik kontrol matrisi 2×6 formunda $(3,1)$-yarı devirli bir matris olur.

Parametreleri belirlediğimize göre anahtar üretim aşamasına geçebiliriz:

Anahtar üretim Aşaması

- H matrisi 2×6 formunda $(3,1)$ yarı devirli bir matris olduğundan bu matrisi üretmek için öncelikle $\mathbb{F}_2$ cismi üzerinde 3 tane 2 uzunluğunda vektör seçilir. $h_1 = (1,1)$, $h_2 = (1,0)$ ve $h_3 = (0,1)$ olsun. Bu vektörleri sırasıyla ilk satır kabul eden devirli matrisler aşağıdakilerdir:

$$H_1 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \qquad H_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad H_3 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

  Bu devirli matrisleri birleştirerek H matrisini elde ederiz:

$$H = \begin{bmatrix} H_1 & H_2 & H_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

- G' matrisini üretmek için $\mathbb{F}_2$ cismi üzerinde 2 uzunluğunda 2 tane vektör seçilir. $g_1 = (1,0)$ ve $g_2 = (1,1)$ olsun. Bu vektörleri sırasıyla ilk satırları olarak kabul eden devirli matrislere G_1 ve G_2 diyelim:

$$G_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad G_2 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

Bu matrisler kullanılarak G' matrisi aşağıdaki şekilde üretilir:

$$G' = \left[\begin{array}{cc|c} I_2 & & G_1 \\ & I_2 & G_2 \end{array}\right] = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

- $\frac{n}{3} \times \frac{n}{3} = 2 \times 2$ formunda tersinir S matrisi belirlenir. İşlemlerin kolaylığı açısından bu matrisi $S = I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ şeklinde seçelim.
- $n \times n = 6 \times 6$ formundaki P permütasyon matrisini de işlemlerde kolaylık olması açısından $P = I_6$ şeklinde seçelim. Böylece, $P^{-1} = P$ kolaylığı da sağlanır.
- $F = G'P^{-1}H^\top S$ çarpımı yapılarak F matrisi aşağıdaki şekilde bulunur:

$$F = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Normalde, elde edilen F matrisine sütun indirgeme işlemleri uygulanarak üst kısmı birim matris olacak şekilde ayarlamak gerekirdi fakat ulaştığımız F matrisinin üst kısmı zaten birim matris olarak bulundu. Bu yüzden sütun indirgeme işlemi yapmamıza gerek kalmadı.

$$F = \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \\ \hline 0 & 1 \\ 1 & 0 \end{array}\right] = \left[\begin{array}{c} I_2 \\ F' \end{array}\right]$$

Böylece F' matrisi $F' = \left[\begin{array}{cc} 0 & 1 \\ 1 & 0 \end{array}\right]$ şeklinde elde edilir.

- F' ve G' matrisleri açık anahtarlarımız, S, H ve P matrisleri de gizli anahtarlarımızdır.

Parametreler belirlendikten ve anahtarlar hesaplandıktan sonra şifrelenecek olan mesaj seçilir ve şifreleme aşamasına geçilir.

Şifrelemek istediğimiz mesajın uzunluğu G' matrisinin satır sayısı ile aynı olmalıdır. Mesajımız $m = (1, 0, 0, 0)$ olsun.

Şifreleme Aşaması

- İlk olarak $n = 6$ uzunluğunda bir e hata vektörü belirlenir. $e = (0, 0, 0, 0, 0, 1)$ olsun.
- $c_1 = mG' + e$ işlemi yapılarak şifreli metnin ilk parçası bulunur:

$$\begin{aligned} c_1 &= mG' + e \\ &= \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \end{array}\right] \left[\begin{array}{cccccc} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{array}\right] + \left[\begin{array}{cccccc} 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right] \\ &= \left[\begin{array}{cccccc} 1 & 0 & 0 & 0 & 1 & 0 \end{array}\right] + \left[\begin{array}{cccccc} 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right] \\ &= \left[\begin{array}{cccccc} 1 & 0 & 0 & 0 & 0 & 1 \end{array}\right] \end{aligned}$$

- $c_2 = mF$ işlemi yapılarak şifreli metnin ikinci parçası bulunur:

$$c_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

- Son olarak c_1 ve c_2 şifreli metin parçaları art arda eklenerek c şifreli metni elde edilir:

$$c = (c_1, c_2) = (1, 0, 0, 0, 0, 1, 1, 0)$$

Şifreli metin alıcıya gönderildikten sonra alıcı, Φ_H kod çözme algoritmasını ve gizli anahtarları kullanarak şifreli metinden açık metni elde eder.

Deşifreleme Aşaması

- Önce şifreli metin n ve $n - k$ uzunluğunda iki parçaya ayrılır:

$$c = (1, 0, 0, 0, 0, 1)(1, 0)$$

$$c_1 = (1, 0, 0, 0, 0, 1) \quad \text{ve} \quad c_2 = (1, 0)$$

- Daha sonra S ve H gizli anahtarları kullanılarak $s = c_1 P^{-1} H^\top - c_2 S^{-1}$ hesaplanır ve s sendrom vektörü bulunur:

$$
\begin{aligned}
s &= c_1 P^{-1} H^\top - c_2 S^{-1} \\
&= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} I_6 \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} - \begin{bmatrix} 1 & 0 \end{bmatrix} I_2 \\
&= \begin{bmatrix} 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 1 \end{bmatrix}
\end{aligned}
$$

Sendrom vektörünü bulduğumuz eşitliği açık halde yazıp tekrar sadeleştirdiğimizde aşağıdaki eşitliğe ulaşırız:

$$
\begin{aligned}
s &= c_1 P^{-1} H^\top - c_2 S^{-1} \\
&= (mG' + e) P^{-1} H^\top - (mG' P^{-1} H^\top S) S^{-1} \\
&= mG' P^{-1} H^\top + eP^{-1} H^\top - mG' P^{-1} H^\top \\
&= eP^{-1} H^\top
\end{aligned}
$$

Buradan $s = (1, 1) = eP^{-1} H^\top$ eşitliğini elde ederiz.

- Bulduğumuz sendrom vektörünü Φ_H kod çözme algoritmasına girdi olarak verdiğimizde çıktı olarak $eP^{-1} = (0, 0, 0, 0, 0, 1)$ vektörünü elde ederiz. Ardından eşitliğin her iki tarafını, gizli anahtarımız olan P ile çarparak e hata vektörünü elde ederiz:

$$
(eP^{-1})P = (0, 0, 0, 0, 0, 1)P = (0, 0, 0, 0, 0, 1) \;\; \Rightarrow \;\; e = (0, 0, 0, 0, 0, 1)
$$

- $mG' + e = c_1$ eşitliği $mG' = c_1 - e$ şeklinde düzenlenerek aşağıdaki denklem sistemi elde edilir:

$$mG' = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$m = (m_1, m_2, m_3, m_4)$ deyip denklem sistemini çözdüğümüzde $m_1 = 1$, $m_2 = 0$, $m_3 = 0$, $m_4 = 0$ değerlerini buluruz. Böylece $m = (1, 0, 0, 0)$ açık metnini, yani mesajın deşifrelenmiş halini elde ederiz.

4.1.5 ROLLO

Bu kısımda ROLLO algoritması anlatılmıştır ve bu anlatım yapılırken [26] ve [27] numaralı kaynaklardan faydalanılmıştır.

ROLLO algoritmasının tam adı "Rank-Ouroboros, Lake and Locker"dır. Buradan da tahmin edebileceğimiz üzere, ROLLO algoritması, bu üç algoritmanın bir derlemesidir. Bu algoritmalar NIST'in yarışmasında aday gösterilen ve rank metrik kodlardan aynı LRPC kodu için kod çözme algoritmasını (4.1.1.2.2) kullanan algoritmalardır. Bu üç algoritma geliştirilerek ROLLO algoritmasının üç temel versiyonunu oluşturmuştur:

1. Lake $\longrightarrow$ ROLLO-I
2. Locker $\longrightarrow$ ROLLO-II
3. Rank-Ouroboros $\longrightarrow$ ROLLO-III

Bu üç algoritmanın işlem süreçleri neredeyse aynıdır fakat bunlardan ikisi anahtar kapsülleme, diğeri ise şifreleme algoritmasıdır. Şimdi algoritmaların işlem süreçlerine geçelim:

4.1.5.1 ROLLO-I (Lake)

ROLLO-I algoritması bir anahtar kapsülleme algoritmasıdır. Daha önce de belirttiğimiz gibi, kuantum-sonrası kriptografideki anahtar kapsülleme algoritmaları, klasik kriptografideki

anahtar değişim algoritmalarıyla benzerdir. Anahtar kapsülleme algoritmalarının işlem süreçleri üç aşamadan oluşur:

1. Anahtar üretimi
2. Kapsülleme
3. Kapsülden çıkarma

Şimdi bu aşamaların ayrıntılarını görelim:

1. Anahtar Üretimi

- İlk olarak q, m, n, r, d parametreleri belirlenir ve bir indirgenemez $P(x) \in \mathbb{F}_{q^m}[x]$ polinomu bulunur.
- $\mathbb{F}_{q^m}$ cisminin vektörel anlamda d boyutlu keyfi bir $\mathcal{F}$ altuzayı seçilir.
- Daha sonra $\mathcal{F}^n \times \mathcal{F}^n$ çarpım kümesinden bir (x, y) ikilisi seçilir. Burada x vektörü $(\mathrm{mod}\ P)$'ye göre tersinir ve $rank(x) = rank(y) = d$ olmalıdır.
- Ardından $h = x^{-1}y\ (\mathrm{mod}\ P)$ çarpımı hesaplanır. (Burada çarpım yapılırken, x ve y vektörlerini, birer polinomun katsayıları olarak kullanıyoruz ve polinom çarpması yapıyoruz. Daha sonra $(\mathrm{mod}\ P)$'ye göre modunu alıp elde edilen polinomun katsayılarını h vektörü olarak yazıyoruz.)
- Son olarak h vektörü açık anahtar olarak yayımlanırken x ve y vektörleri gizli anahtarlar olarak tutulur.

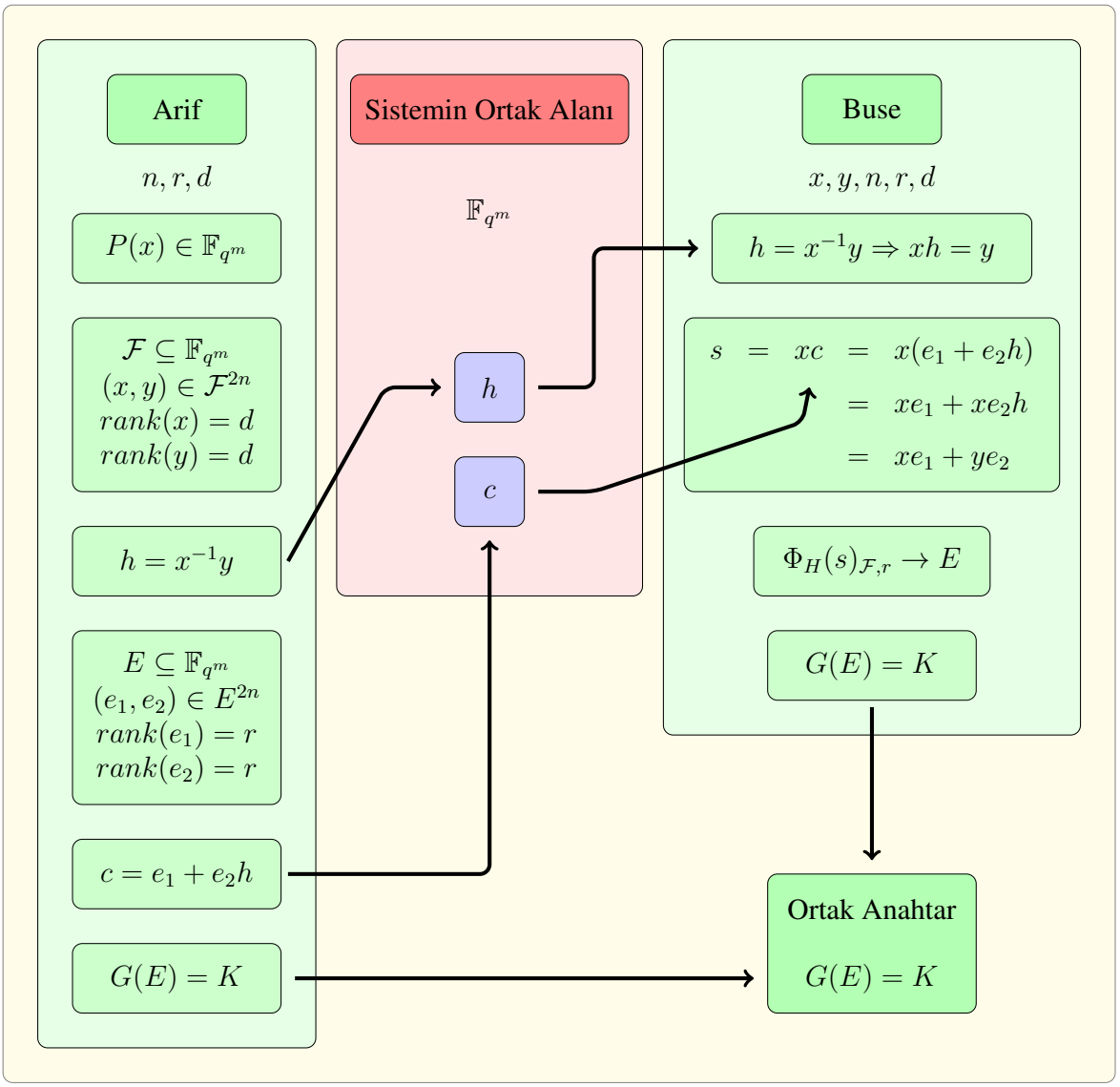
2. Kapsülleme

- $\mathbb{F}_{q^m}$ cisminin keyfi bir E altuzayı seçilir. Bu altuzayın boyutu r olmalıdır.
- $rank(e_1) = rank(e_2) = r$ olacak şekilde $(e_1, e_2) \in E^n \times E^n$ sıralı ikilisi seçilir.
- $c = e_1 + e_2 h\ (\mathrm{mod}\ P)$ şifreli metni hesaplanır. (Buradaki işlemler de anahtar üretim aşamasındaki katsayı polinom eşlemesi şeklinde yapılır.)

- Bir G hash fonksiyonu belirlenir ve $G(E)$ hesaplanır. $K = G(E)$ anahtardır.

3. Kapsülden Çıkarma

- $s = xc = xe_1 + ye_2 \pmod{P}$ hesaplanır.
- s vektörü, $\mathcal{F}$ altuzayı ve r parametresi ile LRPC kod çözme algoritması (4.1.1.2.2) kullanılarak E kümesi elde edilir.
- G hash fonksiyonu ile $G(E)$ hesaplanır. Böylece $K = G(E)$ anahtarına ulaşılır.



Şekil 4.4. ROLLO-I algoritması

4.1.5.2 ROLLO-II (Locker)

ROLLO-II algoritması bir şifreleme algoritmasıdır. Daha önce de belirttiğimiz gibi, şifreleme algoritmaları üç temel aşamadan oluşur:

1. Anahtar üretimi
2. Şifreleme
3. Deşifreleme

ROLLO-II için bu aşamaların ayrıntılarını görelim:

1. Anahtar Üretimi

ROLLO-II algoritmasının anahtar üretim aşaması ROLLO-I ile aynıdır.

- q, m, n, r, d parametreleri belirlenir ve bir indirgenemez $P(x) \in \mathbb{F}_{q^m}[x]$ polinomu bulunur.
- d boyutlu keyfi $\mathcal{F} \subseteq \mathbb{F}_{q^m}$ (vektörel anlamda) altuzayı seçilir.
- Ardından, x tersinir ve $rank(x) = rank(y) = d$ olacak şekilde $(x, y) \in \mathcal{F}^n \times \mathcal{F}^n$ sıralı ikilisi seçilir.
- $h = x^{-1}y$ hesaplanır.
- Son olarak h açık anahtar olarak yayımlanırken x ve y gizli anahtarlar olarak tutulur.

2. Şifreleme

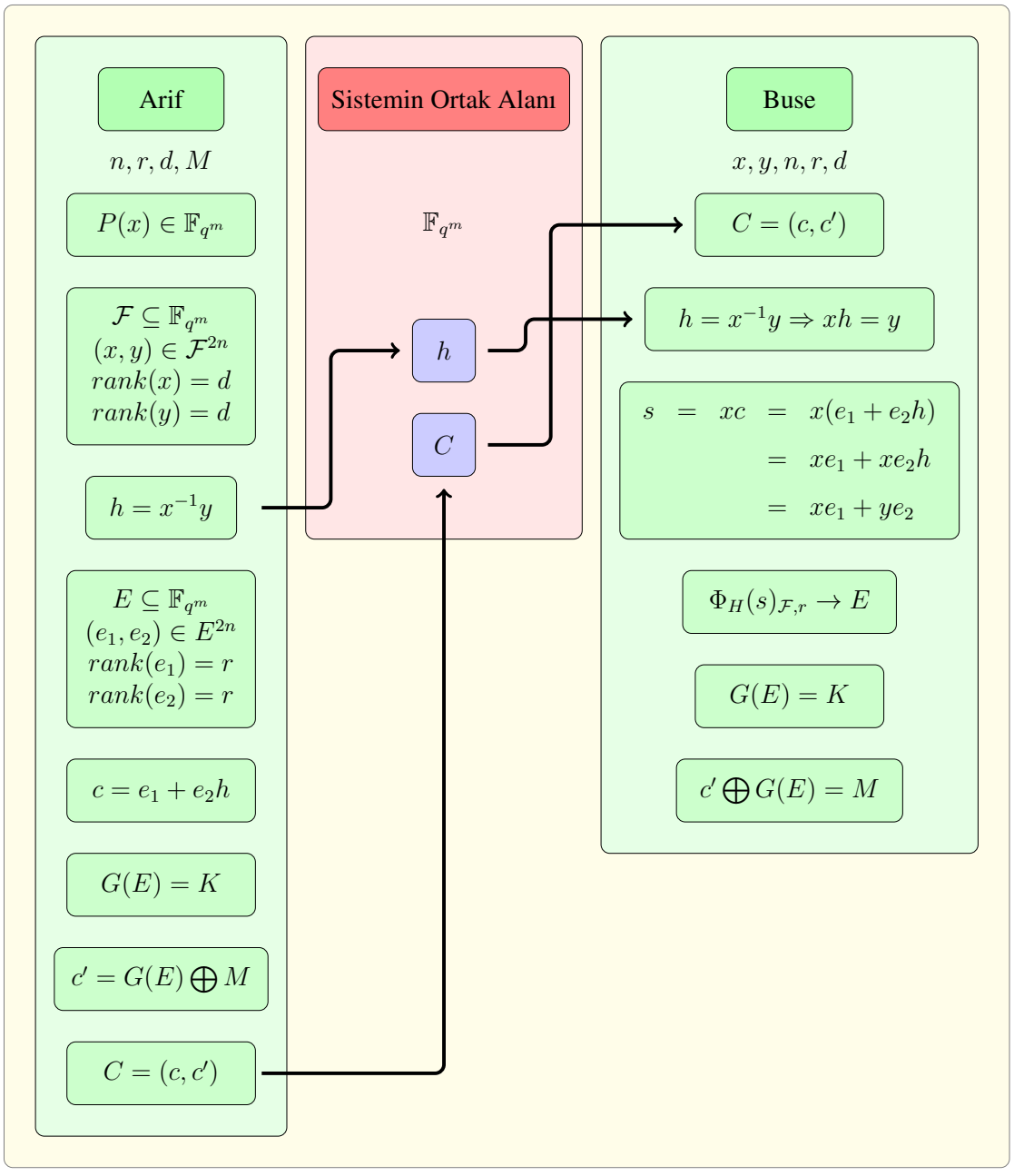
- $\mathbb{F}_{q^m}$ cisminin r boyutlu keyfi bir E altuzayı seçilir.
- $E^n \times E^n$ uzayından $rank(e_1) = rank(e_2) = r$ olacak şekilde bir (e_1, e_2) sıralı ikilisi seçilir.
- $c = e_1 + e_2 h \pmod{P}$ hesaplanır.
- Bir hash fonksiyonu belirlenir ve $G(E)$ hesaplanır.

- n uzunluğunda olan, şifrelenecek M mesajı belirlenir.
- $c' = G(E) \oplus M$ hesaplanır. (Buradaki toplama işlemi XOR (3.2.24) toplamadır.)
- c ve c' vektörleri arka arkaya eklenerek C şifreli metni hesaplanır. $C = (c, c')$

3. Deşifreleme

- C şifreli metni c ve c' şeklinde iki parçaya ayrılır.
- $s = xc = xe_1 + ye_2 \pmod{P}$ sendrom vektörü hesaplanır.
- s vektörü, $\mathcal{F}$ altuzayı ve r parametresi ile LRPC kodlar için kod çözme algoritması (4.1.1.2.2) kullanılarak E kümesi bulunur.
- $G(E)$ hesaplanır.
- $c' \oplus G(E) = M$ işlemi yapılarak M açık metnine, yani asıl mesaja ulaşılır.

ROLLO-II algoritmasını aşağıdaki şekil ile özetleyebiliriz:



Şekil 4.5. ROLLO-II algoritması

4.1.5.3 ROLLO-III (Rank-Ouroboros)

ROLLO-III algoritması bir anahtar kapsülleme algoritmasıdır. Daha önce de belirttiğimiz gibi, ROLLO-I algoritması da bir anahtar kapsülleme algoritmasıdır. İki algoritma, işlem süreci yönünden birbirlerine benzemektedirler fakat kullanılan anahtarların boyuna göre sağladıkları güvenlik seviyesi ve çıktı uzunlukları değişmektedir. ROLLO-III algoritmasının

işlem sürecini anlatmadan önce iki algoritmanın, kullanılan parametrelere göre sağladıkları güvenlik seviyelerinin ve çıktı uzunluklarının bulunduğu tabloyu verelim:

Algoritma Adı	q	m	n	d	r	$P(x)$	Gizli Anahtar	Açık Anahtar	Şifreli Metin	Güvenlik Seviyesi
ROLLO-I	2	79	47	6	5	$x^{47}+x^5+1$	320 bit	3720 bit	3720 bit	128 bit
	2	89	53	7	6	$x^{53}+x^6+x^2+x1$	320 bit	4720 bit	4720 bit	192 bit
ROLLO-III	2	101	47	6	5	$x^{47}+x^5+1$	320 bit	5072bit	9504 bit	128 bit
	2	107	59	8	6	$x^{59}+x^7+x^4+x^2+1$	320 bit	6640 bit	12640 bit	192 bit

Şekil 4.6. ROLLO-I ve ROLLO-III parametreler

Şimdi ROLLO-III algoritmasının işlem sürecine geçebiliriz:

1. Anahtar Üretimi

- Öncelikle q, m, d, r, n parametreleri belirlenir ve indirgenemez bir $P(x) \in \mathbb{F}_{q^m}[x]$ polinomu bulunur.
- $\mathbb{F}_{q^m}$ cisminin (vektörel anlamda) d boyutlu bir $\mathcal{F}$ altuzayı seçilir.
- $\mathcal{F}^n \times \mathcal{F}^n$ çarpım kümesinden bir (x, y) sıralı ikilisi ve $\mathbb{F}_{q^m}^n$ uzayından keyfi bir h vektörü seçilir.
- $s = x + hy \pmod P$ vektörü hesaplanır.
- Son olarak h ve s açık anahtarlar olarak yayımlanırken x ve y gizli anahtarlar olarak tutulur.

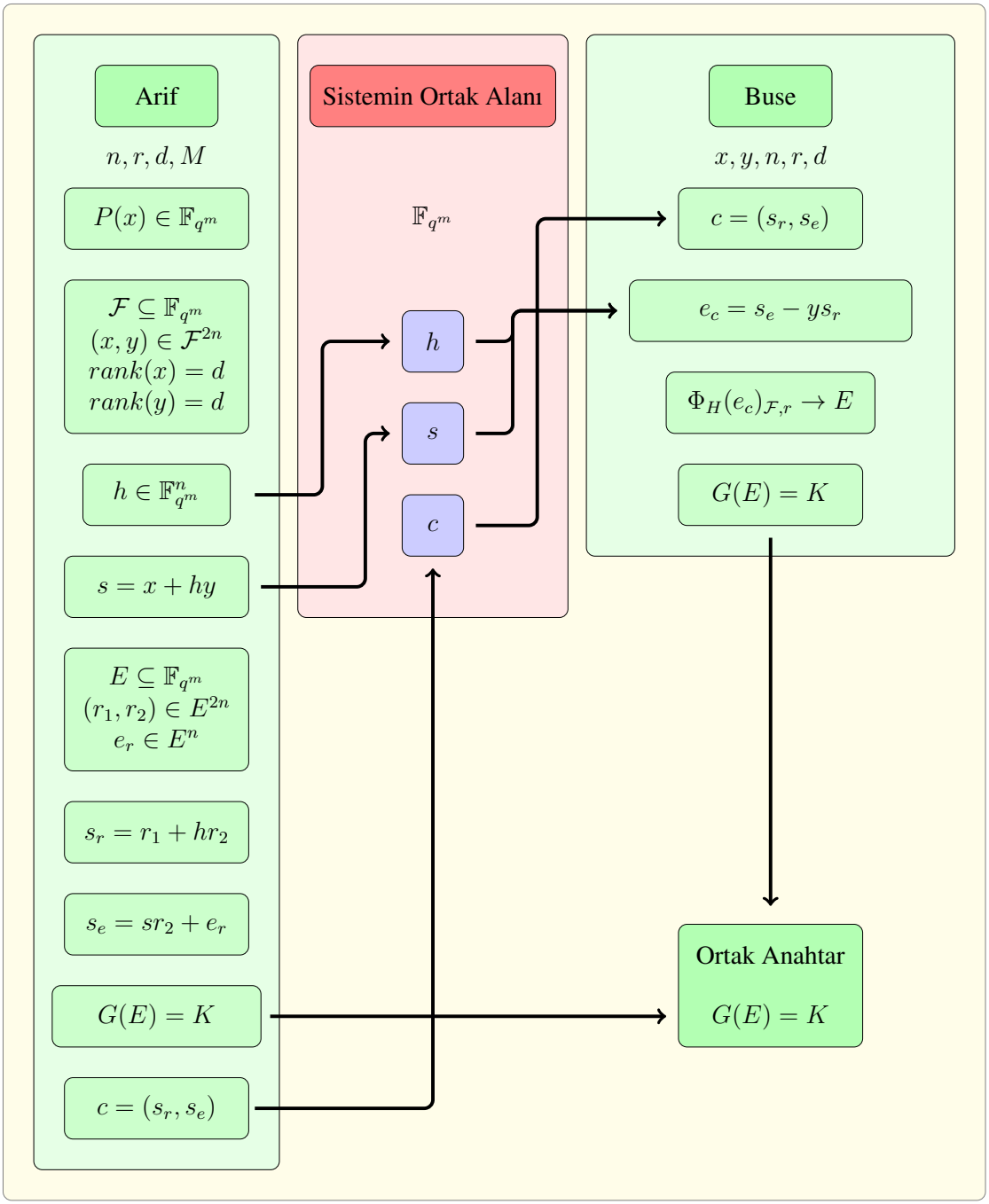
2. Kapsülleme

- $\mathbb{F}_{q^m}$ cisminin vektörel anlamda r boyutlu bir E altuzayı seçilir.
- E^n uzayından keyfi r_1, r_2 ve e_r vektörleri seçilir.
- $s_r = r_1 + hr_2 \pmod P$ hesaplanır.

- $s_e = sr_2 + e_r \pmod{P}$ hesaplanır.
- Bir G hash fonksiyonu belirlenir ve $G(E) = K$ hesaplanır.
- Son olarak s_r ve s_e vektörleri art arda eklenerek $c = (s_r, s_e)$ şifreli metni oluşturulur ve gönderilir.

3. Kapsülden Çıkarma

- c şifreli metni, uzunluk bakımından iki eş parçaya ayrılarak s_r ve s_e bulunur.
- $e_c = s_e - ys_r \pmod{P}$ hesaplanır.
- LRPC kodlar için kod çözme algoritmasına (4.1.1.2.2) girdi olarak $\mathcal{F}$, e_c ve r verilerek E kümesi bulunur.
- G hash fonksiyonu kullanılarak $G(E) = K$ bulunur. Ortak anahtar K'dır.



Şekil 4.7. ROLLO-III algoritması

4.1.6 BIKE

Bu bölümde BIKE algoritması anlatılmıştır ve bu anlatım yapılırken [19] numaralı kaynaktan faydalanılmıştır. BIKE algoritması, yarı-devirli orta yoğunluklu denklik kontrol (4.1.1.1)

kodlarını ve bu kodlar üzerinde bit çevirerek kod çözme (bit flipping decoding) algoritmasını kullanan bir anahtar kapsülleme algoritmasıdır. Bu sebeple, BIKE algoritmasının işleyişine geçmeden önce bit çevirerek kod çözme işleminin nasıl gerçekleştiğini görmekte fayda vardır.

4.1.6.1 Bit Çevirerek Kod Çözme (Bit Flipping Decoding)

Bit flipping algoritması $\mathbb{F}_2$ cismi üzerinde kurulan kodlar için kod düzeltmeye yarayan bir algoritmadır. Biliyoruz ki, $\mathbb{F}_2$ üzerinde iletilen bir sözcükte hata varsa, hatanın konumunu tespit edip, o konumdaki biti $\mathbb{F}_2$ cismindeki diğer elemanla değiştirmek yeterlidir. Bu işleme bit çevirme denir.

Biz bu bölümde, QC-MDPC kodlar üzerinde bit çevirme algoritmasının nasıl işlediğini basit düzeyde göreceğiz.

C, uzunluğu n ve boyutu k olan bir QC-MDPC kod olsun. Bu durumda C kodunun G üreteç matrisi, $k \times n$ formunda olan bir yarı-devirli matristir. (Bit flipping algoritmasının sağlıklı işleyebilmesi için G matrisini oluşturan devirli matrislerin sütun vektörlerinin ağırlıklarının eşit olması gerekir.) C kodunun denklik kontrol matrisi H ise $(n-k)\times n$ formunda, yine yarı-devirli bir matristir. (G'nin sütun vektörlerinin ağırlıkları eşit olduğu için H matrisinin de sütun vektörlerinin ağırlıkları eşittir.) H matrisinin sütun vektörlerinin ağırlığına d diyelim.

İletilmek istenen mesaj k uzunluğundaki $m \in \mathbb{F}_2^k$ vektörü olsun. Bu mesaj mG işlemi ile kodlanır ve alıcıya gönderilir. Bu işlem sonucunda n uzunluğunda bir vektör elde edilir. Alıcıya ulaşmış olan bu vektöre $e = (e_1, \ldots, e_n)$ diyelim.

Alıcı, ilk olarak $eH^\top$ işlemini yapar. Eğer işlemin sonucu sıfır ise kod sözcüğünün doğru iletildiği sonucuna varır. Sözcük doğru iletildiğine göre $mG = e$ denklem sistemini çözerek m mesajına ulaşır.

Eğer işlemin sonucu sıfıra eşit değilse, alıcı iletilen sözcükte hata olduğunu anlar ve aşağıdaki süreci izler. (Tabi ki iletilen sözcükteki hata sayısının, kodun hata düzeltme kapasitesini aşmadığını varsayıyoruz.)

1. Adım : H matrisinin sütunlarını 1'den n'ye kadar, $h_1, h_2, \ldots, h_n$ şeklinde numaralandırır.

2. Adım : $eH^\top = s$ işleminin sonucuna s dersek, $\forall\, i = 1, \ldots, n$ için $h_i \cdot s$ işlemini yapar.

3. Adım : Eğer $h_i \cdot s$ vektörünün ağırlığı (2.8.7) h_i vektörünün ağırlığından küçük ise hiçbir şey yapılmaz fakat büyük veya eşit ise e vektörünün i. bileşenindeki bit $\mathbb{F}_2$ cismindeki diğer elemanla değiştirilir. Bunu matematiksel olarak aşağıdaki şekilde yazabiliriz:

$$w(h_i \cdot s) \geq w(h_i) \quad \Rightarrow \quad e = (e_1, e_2, \ldots e_{i-1}, e_i + 1, e_{i+1}, \ldots, e_n) \pmod 2$$

Algoritmanın daha iyi anlaşılması için bir örnek verelim:

Örnek 4.1.30. C kodunun denklik kontrol matrisi H, aşağıdaki şekilde olsun:

$$H = \left[\begin{array}{ccccc|ccccc} 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

Alıcıya iletilmek istenen sözcük $E = (1, 1, 1, 1, 1, 0, 0, 0, 0, 0)$ olsun fakat iletilirken vektörün beşinci bitinde bir bozulmanın meydana geldiğini ve alıcıya ulaşan vektörün $e = (1, 1, 1, 1, \underline{0}, 0, 0, 0, 0, 0)$ olduğunu varsayalım. (Dikkat edelim: H matrisinin her sütununun ağırlığı 2'dir. Yani $\forall\, i = 1, \ldots, 10$ için $w(h_i) = 2$'dir.)

Alıcı e vektörünü aldıktan sonra $eH^\top$ işlemini yaparak

$$eH^\top = s = (0, 1, 0, 0, 1) \neq (0, 0, 0, 0, 0, 0) = 0$$

sonucuna ulaşır ve bir hatanın olduğunu anlar. Ardından $\forall\ i = 1, \ldots, 10$ için $w(h_i \cdot s)$ ve $w(h_i)$ ağırlıklarını karşılaştırarak düzeltmesi gereken bitleri tespit eder.

$$
\begin{array}{lcccccccl}
i = 1 \text{ için} & : & w(h_1 \cdot s) & = & 0 & < & w(h_1) & \Rightarrow & e_1 \text{ çevrilmez} \\
i = 2 \text{ için} & : & w(h_2 \cdot s) & = & 1 & < & w(h_2) & \Rightarrow & e_2 \text{ çevrilmez} \\
i = 3 \text{ için} & : & w(h_3 \cdot s) & = & 1 & < & w(h_3) & \Rightarrow & e_3 \text{ çevrilmez} \\
i = 4 \text{ için} & : & w(h_4 \cdot s) & = & 0 & < & w(h_4) & \Rightarrow & e_4 \text{ çevrilmez} \\
i = 5 \text{ için} & : & w(h_5 \cdot s) & = & 2 & \geq & w(h_5) & \Rightarrow & e_5 \ \underline{\textbf{çevrilir!}} \\
i = 6 \text{ için} & : & w(h_6 \cdot s) & = & 1 & < & w(h_6) & \Rightarrow & e_6 \text{ çevrilmez} \\
i = 7 \text{ için} & : & w(h_7 \cdot s) & = & 1 & < & w(h_7) & \Rightarrow & e_7 \text{ çevrilmez} \\
i = 8 \text{ için} & : & w(h_8 \cdot s) & = & 1 & < & w(h_8) & \Rightarrow & e_8 \text{ çevrilmez} \\
i = 9 \text{ için} & : & w(h_9 \cdot s) & = & 0 & < & w(h_9) & \Rightarrow & e_9 \text{ çevrilmez} \\
i = 10 \text{ için} & : & w(h_{10} \cdot s) & = & 1 & < & w(h_{10}) & \Rightarrow & e_{10} \text{ çevrilmez}
\end{array}
$$

Bu işlem sonucunda alıcıya ulaşan vektörde beşinci bitin hatalı olduğu tespit edilmiş olur ve beşinci biti $\mathbb{F}_2$ cismindeki diğer elemanla değiştirilerek hata düzeltilir. Sonuç olarak $E = (1, 1, 1, 1, 1, 0, 0, 0, 0, 0)$ vektörüne ulaşılır.

BIKE algoritmasının güvenliği, tamamen geçici anahtarlara dayanır. Bunun anlamı, her anahtar değişiminde yeni bir anahtar çiftinin oluşturulmasıdır. Bu cümleyi, algoritmaların işlem sürecini gördüğümüzde daha iyi anlayacağız.

BIKE algoritmasının birçok çeşidi vardır, biz bu bölümde üç belirgin çeşidini inceleyeceğiz ve bunları basitçe BIKE-1, BIKE-2, BIKE-3 şeklinde isimlendireceğiz. Algoritmaları incelerken göreceğiz ki bu üç algoritma, çok belirgin farklar barındırsalar da ya McEliece ya da Niederreiter algoritmalarına benzemektedirler.

Algoritmanın üç çeşidinin de kullandığı ortak temel parametreleri, tekrar tekrar yazmamak için önceden söyleyelim:

Öncelikle, $x^r - 1 \big/ x - 1 \in \mathbb{F}_2[x]$ bir indirgenemez polinom olacak şekilde bir r asal sayısı belirlenir. Ardından, uzunluğu $n = 2r$, boyutu r ve hata düzeltme kapasitesi t olan bir ikili

kod seçilir. Bu kodun üreteç matrisinin sütun ağırlığına d_v diyelim ve d_v sayısı bir tek tam sayı olsun.

Bu parametrelere ek olarak, bir de BIKE algoritmasının hem anahtar kapsülleme hem de kapsülden çıkarma aşamalarında kullanılacak olan bir hash fonksiyonu belirlenir. Bu fonksiyona **K** diyelim ve

$$\mathbf{K} \ : \ \mathbb{F}_2^n = \{0,1\}^n \quad \longrightarrow \quad \mathbb{F}_2^{l_\mathbf{K}} = \{0,1\}^{l_\mathbf{K}}$$

şeklinde tanımlayalım. Buradan da anlayacağımız üzere, n uzunluğundaki anahtarların özetlerinin uzunluğu $l_\mathbf{K}$ olarak adlandırılmıştır. (Genelde bu uzunluk 256 bit olarak belirlenir.)

Şimdi algoritmalara geçebiliriz:

4.1.6.2 BIKE-1

BIKE algoritmasının bu çeşidinin özelliği, McEliece algoritmasının bir benzerini kullanarak çok hızlı bir anahtar üretimi sağlamasıdır.

Önce, BIKE-1 algoritmasının McEliece'ten farkını söyleyelim: McEliece algoritmasında, gizli anahtarlardan biri olan P permütasyon matrisinin tersi alıp diğer devirli matrisler ile çarpıyorduk. BIKE-1 algoritmasında bu ters alma ve çarpma işlemi yoktur.

BIKE-1 algoritması bir anahtar kapsülleme algoritması olduğundan, daha önce de belirttiğimiz gibi üç aşamadan oluşur. Bunlar, anahtar üretimi, kapsülleme ve kapsülden çıkarmadır:

Anahtar Üretimi

1. r asal sayısı ve $P = {x^r - 1}/{x - 1} \in \mathbb{F}_2[x]$ indirgenemez polinomu belirlenir.
2. Bir QC-MDPC kodu seçilir. Bu kodu kısaca, genel kullanım olan C harfiyle gösterelim. C kodunun denklik kontrol matrisi olan H matrisinin sütun vektörlerinin ağırlığına w diyelim.

3. $\mathbb{F}_2[x]/\langle x^r - 1\rangle$ polinom halkasından iki tane polinom seçilir. Bu polinomlara h_0 ve h_1 diyelim. $w(h_0) = \frac{w}{2}$ ve $w(h_1) = \frac{w}{2}$ olmalıdır.

4. $\mathbb{F}_2[x]/\langle x^r - 1\rangle$ halkasından, ağırlığı $w(g) \approx \frac{r}{2}$ olan bir g polinomu seçilir.

5. $gh_0 = f_1$ ve $gh_1 = f_0$ çarpımları hesaplanır.

6. Hesaplamalardan da anlaşılacağı üzere h_0 ve h_1 polinomlarının ağırlıkları f_0 ve f_1 polinomlarının ağırlıklarından daha düşüktür. Özel olarak h_0 ve h_1 polinomlarına seyrek anahtarlar denir ve bunlar gizli anahtarlardır, f_0 ve f_1 polinomlarına ise yoğun anahtarlar denir ve bunlar açık anahtarlardır.

Kapsülleme

1. $\mathbb{F}_2[x]/\langle x^r - 1\rangle$ halkasından e_0 ve e_1 polinomları seçilir. e_0 ve e_1 polinomlarının vektör karşılıklarının ağırlıkları toplamı, C kodunun hata düzeltme kapasitesi kadar olmalıdır. $w(e_0) + w(e_1) = t$

2. $\mathbb{F}_2[x]/\langle x^r - 1\rangle$ halkasından bir m polinomu seçilir.

3. $mf_0 + e_0 = c_0$ ve $mf_1 + e_1 = c_1$ işlemleri yapılır.

4. **K** hash fonksiyonu kullanılarak $\mathbf{K}(e_0, e_1) = K$ hesaplanır.

Kapsülden Çıkarma

1. $c_0h_0 + c_1h_1 = s$ işlemi yapılarak s sendrom vektörü bulunur.

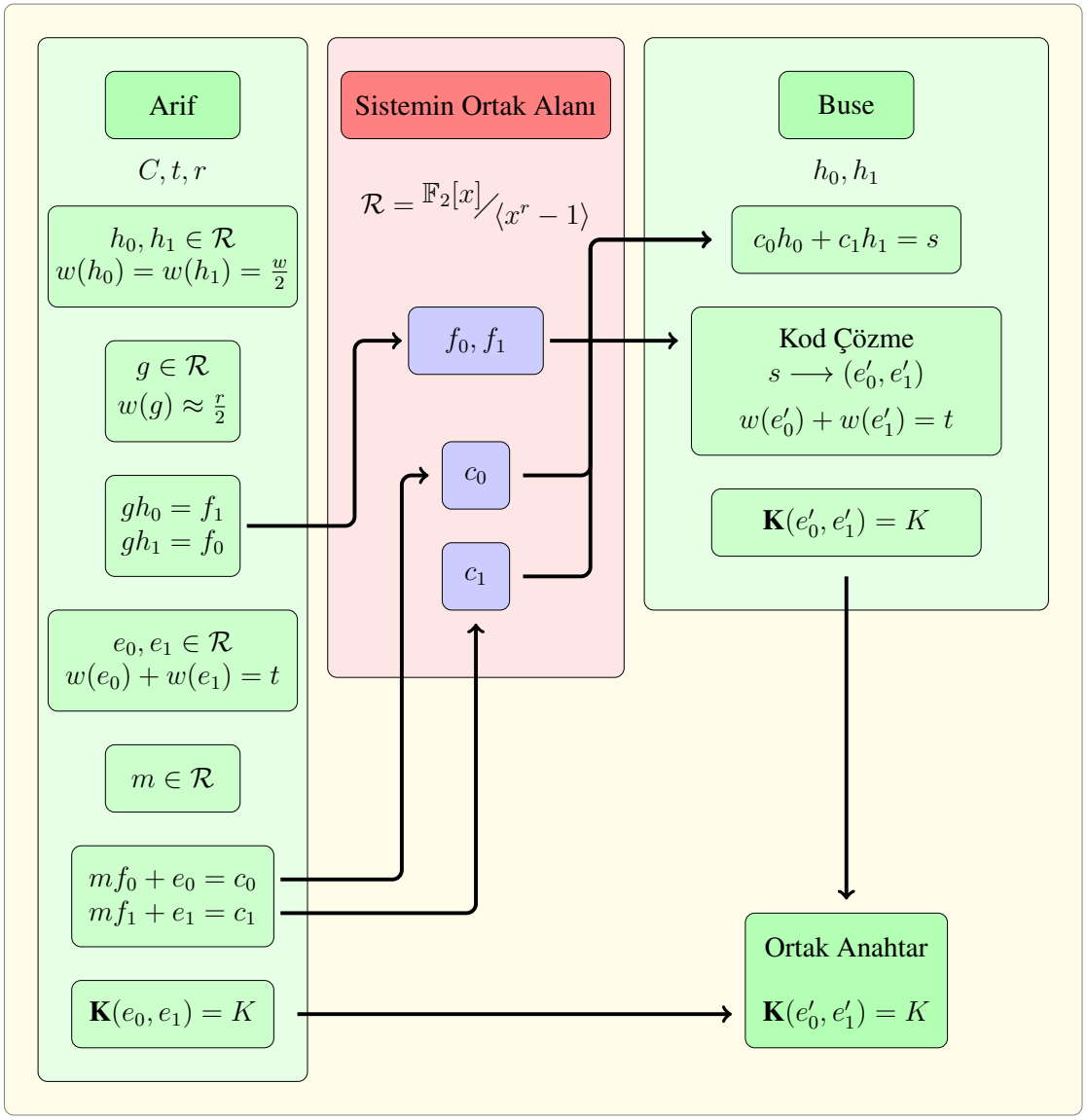
2. s vektörüne QC-MDPC kodlar için bir kod çözme algoritması uygulanarak (e'_0, e'_1) vektörü elde edilir. QC-MDPC kodlar için bir kod çözme algoritması örneği bit çevirerek kod çözme (4.1.6.1) algoritmasıdır.

3. Eğer kod çözme işlemi yapıldığında $w(e'_0, e'_1) \neq t$ sonucuna ulaşılırsa kod çözme algoritması tekrar uygulanarak ağırlıkları toplamı t olan farklı (e'_0, e'_1) vektörleri elde edilir.

Ağırlıklar toplamı t olmadığı zaman özet fonksiyonu altındaki görüntüleri (e_0, e_1) vektörlerinin görüntülerinden farklı olur ki bu da ortak anahtar oluşturma durumunu engeller.

4. **K** hash fonksiyonu kullanılarak $\mathbf{K}(e'_0, e'_1) = K$ hesaplanır.

BIKE-1 algoritmasını, temel iletişim senaryomuz üzerinden aşağıdaki şekil ile özetleyebiliriz:



Şekil 4.8. BIKE-1 algoritması

4.1.6.3 BIKE-2

BIKE algoritmasının bu çeşidinde Niederreiter (4.1.3.2) algoritmasının sistematik denklik kontrol matrisli bir hali baz alınır. Bunun sağladığı en büyük avantaj, algoritmadaki tüm girdiler için tek bir r uzunluklu blokun yeterli olmasıdır. Böylelikle, oldukça minimize edilmiş bir format ortaya çıkar. Bu da veri tabanında kaplanan alandan tasarruf sağlar.

Diğer yönden, BIKE-1'den farklı olarak, algoritmanın bu çeşidinde bir polinomun tersini bulma işlemi bulunmaktadır ki bu işlem anahtar üretimi kısmında yer aldığı için anahtar üretim aşamasının şifreleme aşamasından biraz daha yavaş işlemesine sebebiyet verir.

Şimdi algoritmanın işlem sürecini anlatalım:

Anahtar Üretimi

1. $P = {}^{x^r - 1}\!/_{x - 1} \in \mathbb{F}_2[x]$ bir indirgenemez polinom olacak şekilde bir r asal sayısı belirlenir.
2. Bir QC-MDPC kodu seçilir. Bu kodun denklik kontrol matrisinin sütun ağırlıklarına w diyelim ve seçilen kodun hata düzeltme kapasitesi t olsun.
3. ${}^{\mathbb{F}_2[x]}\!/_{\langle x^r - 1 \rangle}$ halkasından iki polinom seçilir. Bu polinomlara h_0 ve h_1 diyelim. Burada h_0 polinomu tersinir ve $w(h_0) = w(h_1) = \frac{w}{2}$ olmalıdır.
4. h_0 polinomunun tersi bulunur.
5. $h = h_1 h_0^{-1}$ hesaplanır.
6. h_0 ve h_1 gizli anahtarlar olarak tutulurken h açık anahtar olarak gönderilir.

Kapsülleme

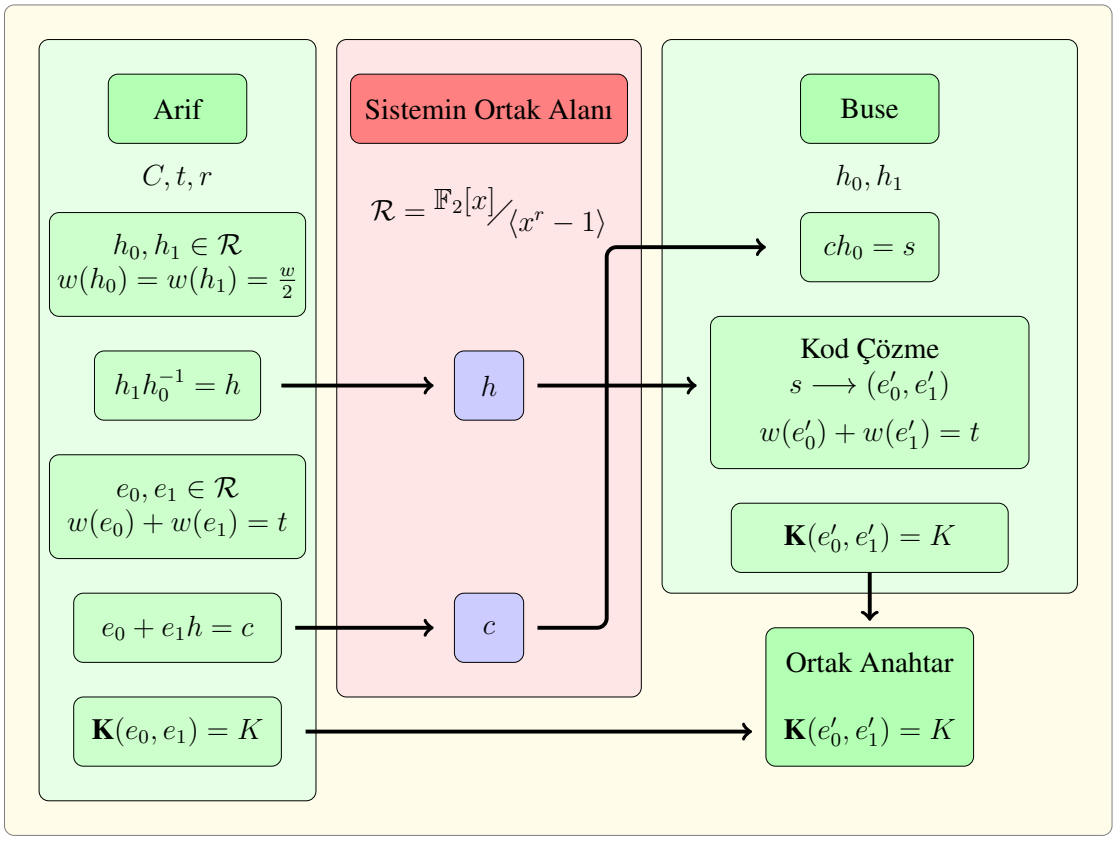
1. ${}^{\mathbb{F}_2[x]}\!/_{\langle x^r - 1 \rangle}$ halkasından iki polinom seçilir. Bu polinomlara e_0 ve e_1 diyelim. Burada $w(e_0) + w(e_1) = t$ olmalı.
2. $e_0 + e_1 h = c$ hesaplanır.

3. **K** hash fonksiyonu kullanılarak $\mathbf{K}(e_0, e_1) = K$ hesaplanır.

Kapsülden Çıkarma

1. $ch_0 = s$ sendrom vektörü hesaplanır.
2. s vektörüne QC-MDPC kodlar için bir kod çözme algoritması uygulanarak (e'_0, e'_1) vektörü elde edilir.
3. Eğer kod çözme işlemi sonucunda elde edilen (e'_0, e'_1) vektörü için $w(e'_0) + w(e'_1) \neq t$ ise eşitlik sağlanana kadar kod çözme işlemi yeniden yapılır.
4. **K** hash fonksiyonu kullanılarak $\mathbf{K}(e'_0, e'_1) = K$ hesaplanır.

BIKE-2 algoritmasını, temel iletişim senaryomuz üzerinden aşağıdaki şekil ile özetleyebiliriz:



Şekil 4.9. BIKE-2 algoritması

4.1.6.4 BIKE-3

Algoritmanın bu çeşidi ROLLO-III (4.1.5.3) algoritmasını baz alır. BIKE-3 algoritmasının işlemleri, birazdan göreceğimiz üzere, BIKE-1 algoritmasına çok benzemektedir. Örneğin, BIKE-3'te de anahtarın tersini bulma işlemi yoktur ve anahtar üretim aşamasında 2 adet açık anahtar bulunmaktadır. İki algoritma arasındaki temel fark, kapsülden çıkarma şamasındaki sendrom vektörünün kod çözme işleminde çıkardığı zorluktadır. BIKE-3 algoritmasında bu işlem çok daha rahat yapılmaktadır. Ayrıca, parametre seçimine bağlı olarak sağlanılan güvenlik seviyesi bakımından da BIKE-3 algoritması BIKE-1 ve BIKE-2 algoritmalarının önüne geçmektedir.

Şimdi algoritmanın işlem sürecine geçebiliriz:

Anahtar Üretimi

1. $P = {x^r - 1}/{x - 1} \in \mathbb{F}_2[x]$ bir indirgenemez polinom olacak şekilde bir r asal sayısı belirlenir.

2. Bir QC-MDPC (4.1.1.1) kodu seçilir. Seçilen kodun denklik kontrol matrisinin sütun ağırlıklarına w diyelim ve hata düzeltme kapasitesi t olsun.

3. ${\mathbb{F}_2[x]}/{\langle x^r - 1\rangle}$ halkasından keyfi h_0 ve h_1 polinomları seçilir.
   Burada $w(h_0) = w(h_1) = \frac{w}{2}$ olmalıdır.

4. ${\mathbb{F}_2[x]}/{\langle x^r - 1\rangle}$ halkasından $w(g) \approx \frac{r}{2}$ olacak şekilde bir g polinomu seçilir.

5. $h_1 + gh_0 = f_0$ işlemi yapılır ve diğer yandan $g = f_1$ olarak işaretlenir.

6. h_0 ve h_1 gizli anahtarlar olarak tutulur ve f_0 ile f_1 açık anahtarlar olarak gönderilir.

Kapsülleme

1. ${\mathbb{F}_2[x]}/{\langle x^r - 1\rangle}$ halkasından $w(e) = \frac{t}{2}$ ve $w(e_0) + w(e_1) = t$ olacak şekilde e, e_0, e_1 polinomları seçilir.

2. $e + e_1 f_0 = c_0$ ve $e_0 + e_1 f_1 = c_1$ işlemleri yapılır ve ardından sonuçlar uç uca eklenerek $c = (c_0, c_1)$ vektörü oluşturulur.

3. **K** hash fonksiyonu kullanılarak $\mathbf{K}(e_0, e_1) = K$ hesaplanır.

Kapsülden Çıkarma

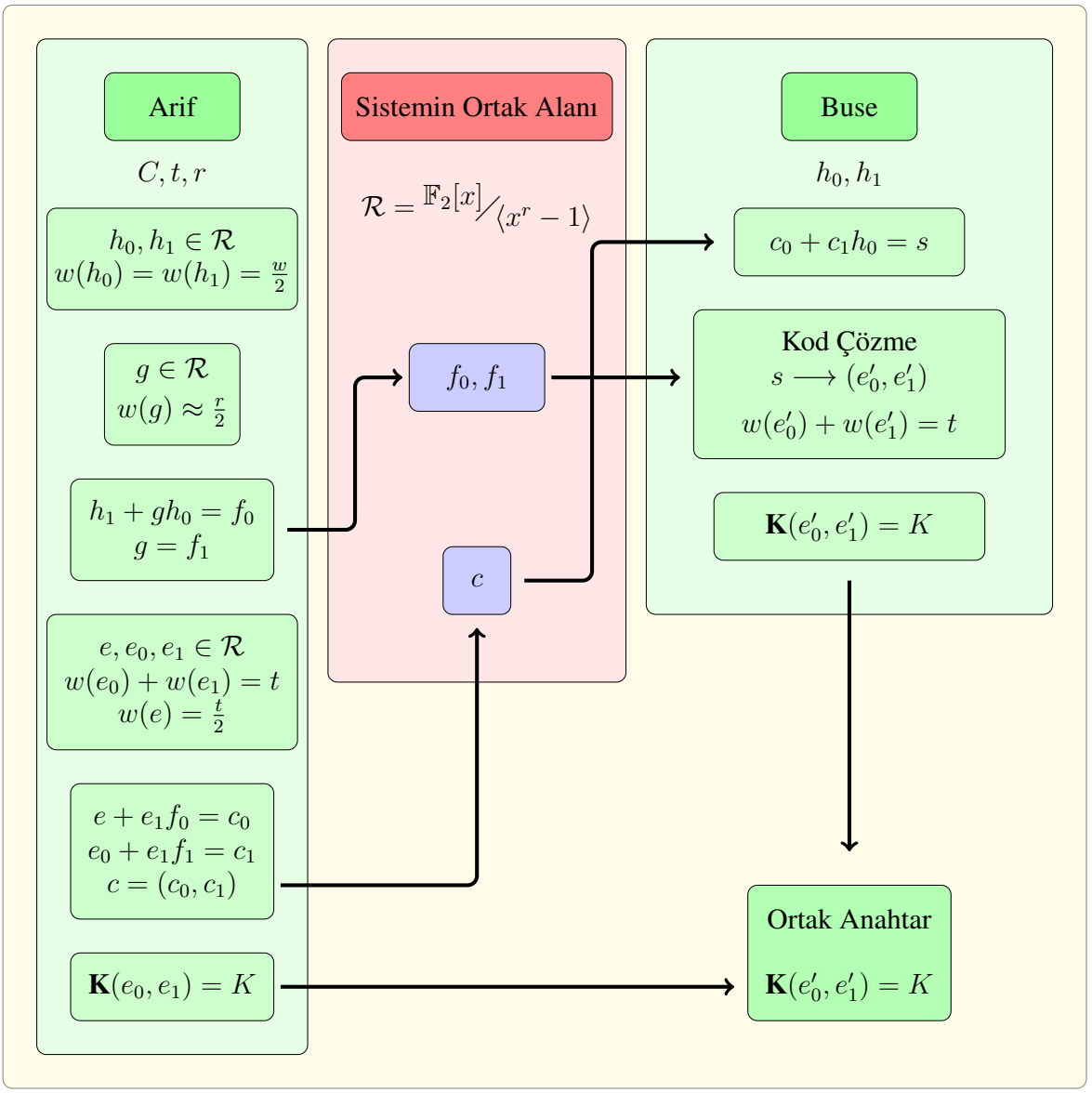
1. $c_0 + c_1 h_0 = s$ sendrom vektörü hesaplanır.

2. s vektörüne QC-MDPC kodlar için bir kod çözme algoritması olarak kullanılabilen bit çevirme ile kod çözme (4.1.6.1) algoritması uygulanarak (e'_0, e'_1) vektörü bulunur.

3. Eğer $w(e'_0, e'_1) \neq t$ ise kod çözme algoritması yeniden uygulanarak eşitliği sağlayan bir (e'_0, e'_1) çifti bulunur.

4. $\mathbf{K}(e'_0, e'_1) = K$ hesaplanır.

BIKE-3 algoritmasını, temel iletişim senaryomuz üzerinden aşağıdaki şekil ile özetleyebiliriz:



Şekil 4.10. BIKE-3 algoritması

4.1.7 HQC

Bu bölümde HQC kriptosistemi anlatılmıştır ve bu anlatım yapılırken [28] numaralı kaynaktan faydalanılmıştır.

HQC kriptosisteminin tam adı "Hamming Quasi-Cyclic" kriptosistemidir. Adından da tahmin edilebileceği üzere, HQC algoritması yarı devirli üreteç matrisleri (4.1.3) ve kod

sözcüklerinin Hamming ağırlıkları (2.8.7) ile içli dışlı olan bir algoritmadır. HQC algoritmasının birkaç farklı versiyonu vardır, hatta algoritmadaki kod çözme aşamasının daha hızlı olması amacıyla Reed-Muller (4.1.1.4) ve Reed-Solomon (4.1.1.5) kodları birlikte kullanan bir versiyonu da geliştirilmiştir. İki kodun birlikte kullanıldığı duruma birleşik kodlar adı verilmektedir. HQC algoritmasının bu versiyonuyla ilgilenenler için tanımından bahsedelim.

Tanım 4.1.31. (Birleşik Kodlar - Concatenated Codes) Birleşik kodlar, harici (external) ve dahili (internal) olarak adlandırılan iki kodun bir arada kullanılmasıyla oluşturulur. Harici kod $\mathbb{F}_q$ üzerinde, dahili kod ise $\mathbb{F}_2$ üzerinde seçilir. Harici ve dahili kodların parametrelerine sırasıyla $[n_e, k_e, d_e]$ ve $[n_i, k_i, d_i]$ diyelim. Burada $q = 2^{k_i}$ olmalıdır.

$\mathbb{F}_q^{n_e}$ ile $N = n_e n_i$ olmak üzere $\mathbb{F}_2^N$ arasında bire-bir ve örten bir dönüşüm vardır.

$$\mathbb{F}_q^{n_e} \longrightarrow \mathbb{F}_2^N$$

Bu dönüşüm kullanılarak harici kod, $n = n_e n_i$, $k = k_e k_i$ ve $d \geq d_e d_i$ parametrelerine sahip bir ikili koda dönüştürülebilir.

HQC algoritmasında kullanılan harici kod, $\mathbb{F}_{256}$ üzerinde oluşturulan ve boyutu 32 olan bir Reed-Solomon kodu, dahili kod ise $[128, 8, 64]$ parametrelerine sahip Reed-Muller kodudur. Hatta, dahili kod 5 tekrarlı olacak şekilde kullanılarak kod uzunluğu 5 katına çıkarılmıştır.

Tanım 4.1.32. (Kesme Fonksiyonu) Bir elemanın bir fonksiyon altındaki görüntüsü bazen istenilenden uzun olabilir. Kesme fonksiyonu, girdi olarak belirli uzunlukta bir sözcüğü ve bir i tam sayısını alır ve çıktı olarak sözcüğün ilk i harfini verir. Biz, anlatım sırasında kesme fonksiyonu kullanacağımız zaman, ingilizce adı olan "truncate"nin kısaltılmış halini, yani "**tc**" harflerini kullanacağız.

$$\mathbf{tc}(a = (a_1, \ldots, a_n), i) = (a_1, \ldots, a_i, \cancel{a_{i+1}}, \cancel{a_{i+2}}, \ldots, \cancel{a_n}) = (a_1, a_2, \ldots, a_i)$$

HQC kriptosisteminin, hem bir açık anahtarlı şifreleme hem de bir anahtar kapsülleme algoritması olarak kullanılabilen versiyonları vardır. Önce açık anahtarlı şifreleme versiyonunu, ardından da anahtar kapsülleme algoritmasının işleyişlerini verelim.

4.1.7.1 HQC Açık Anahtarlı Şifreleme (HQC-PKE)

HQC kriptosisteminin açık anahtarlı şifreleme için kullandığı iki tip kod vardır. Bunlardan birisi lineer kodların temel halidir. Bildiğimiz üzere bir C lineer kodunu, üç parametresini kullanarak $[n, k, d]$ şeklinde gösterebiliriz. Bu C kodunun üreteç matrisi olan G, $k \times n$ formundadır ve bu kodun hata düzeltme kapasitesi, iyi bir kod çözme algoritması kullanıldığında $\frac{d-1}{2}$'ye kadar çıkmaktadır. HQC'nin kullandığı diğer kod tipi ise 2-yarı devirli kodlardır. Bu tip kodların denklik kontrol matrisi H ise bir devirli matristir.

Şimdi algoritmanın işlem sürecine geçelim:

Anahtar Üretimi

1. İlk olarak n, k, t, w parametreleri belirlenir.

   n : kodun uzunluğu
   k : kodun boyutu
   t : kodun hata düzeltme kapasitesi
   w : denklik kontrol matrisinin sütun ağırlığı

2. $[n, k, d]$ parametrelerine sahip bir C kodu, yani $k \times n$ formunda bir G üreteç matrisi seçilir.
3. $\mathbb{F}_2[x]/\langle x^n - 1\rangle$ halkasından h, x ve y polinomları seçilir. Polinomların ağırlıkları w olmalıdır.
4. $x + h \cdot y = s$ hesaplanır.
5. x ve y gizli anahtarlar olarak tutulurken h ve s açık anahtarlar olarak gönderilir.

Şifreleme

1. $\mathbb{F}_2[x]/\langle x^n - 1\rangle$ halkasından e, r_1 ve r_2 polinomları seçilir. Polinomların ağırlıkları w olmalıdır.

2. $r_1 + h \cdot r_2 = u$ hesaplanır.

3. Şifrelenecek m mesajı seçilir. Mesajın uzunluğu k olmalıdır.

4. $\mathbf{tc}(mG + s \cdot r_2 + e, l) = v$ hesaplanır. (4.1.32)

5. $c = (u, v)$ şifreli metni oluşturulur.

Deşifreleme

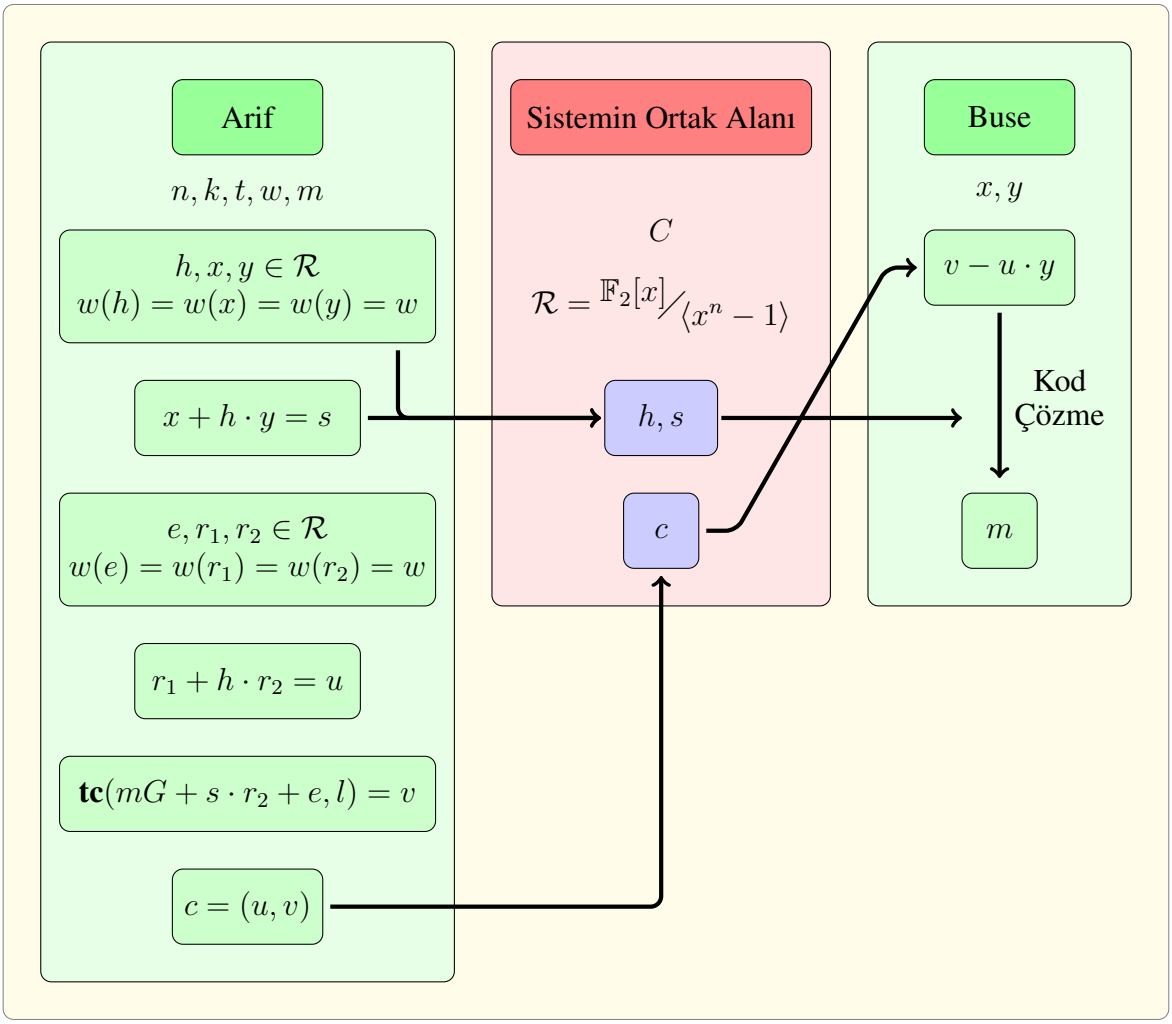
1. x, y gizli anahtarları ve c şifreli metni kullanılarak $v - u \cdot y$ hesaplanır.

2. Uygun kod çözme algoritması kullanılarak $v - u \cdot y$ çözülür ve m şifreli mesajına ulaşılır.

Not : Algoritmada kullanılan G üreteç matrisi açık olarak biliniyor. Bu sebeple, algoritmanın güvenliği, kullanılan C koduna değil, x ve y anahtarlarının gizliliğine bağlıdır.

Daha önce de belirttiğimiz gibi, HQC algoritmasında kullanılan kod ailesi, Reed-Muller ve Reed-Solomon kodlarının birleşik halidir. Kullanılan Reed-Muller kodunun uzunluğuna n_1 ve Reed-Solomon kodunun uzunluğuna n_2 dersek, algoritmadaki üreteç matrisinin sütunları $\mathbb{F}_2^{n_1 n_2}$ uzayına aittir. Seçilen h polinomunun vektörel gösterimi ise $\mathbb{F}_2^n$ uzayının bir elemanıdır. Buradaki n sayısı $n_1 n_2$'den büyük olan en küçük asal sayıdır. Algoritmadaki kesme fonksiyonundaki l uzunluğu ise $l = n - n_1 n_2$ işlemi ile hesaplanır.

HQC açık anahtarlı şifreleme algoritmasını, temel iletişim senaryomuz üzerinden aşağıdaki şekil ile özetleyebiliriz:



Şekil 4.11. HQC açık anahtarlı şifreleme

4.1.7.2 HQC Anahtar Kapsülleme Mekanizması (HQC-KEM)

HQC'nin anahtar kapsülleme algoritmasının güvenliği, sendrom kod çözme problemine dayanmaktadır. Ayrıca, içerisinde iki adet hash fonksiyonu vardır ve böylece algoritmanın barındırdığı rastgelelik oranı yüksektir. Dolayısıyla, kırılması neredeyse imkansızdır.

HQC-KEM algoritmasının kullandığı anahtarların boyları ve içerdiği işlem yükü sebebiyle biraz yavaş çalışmaktadır ama sağladığı güvenlik seviyesi bu açığı rahatlıkla kapatabilmektedir. Şimdi algoritmanın işlem sürecine geçebiliriz. Bildiğimiz üzere, anahtar kapsülleme algoritmaları üç aşamadan oluşur: Anahtar üretimi, kapsülleme ve kapsülden çıkarma.

Anahtar Üretimi

1. Önce kullanılacak C kodu ve C'nin üreteç matrisi G belirlenir. C kodu $\mathbb{F}_2$ üzerinde yarı-devirli bir koddur ve parametreleri n, k, d'dir. Dolayısıyla G matrisi de $\mathbb{F}_2$ üzerinde $k \times n$ formunda yarı-devirli bir matristir. C'nin denklik kontrol matrisinin sütun ağırlıkları w'dir.

2. ${}^{\mathbb{F}_2[x]}\!/_{\langle x^n - 1 \rangle}$ halkasından x, y ve h polinomları, $\mathbb{F}_2^k$ uzayından da bir σ vektörü seçilir. Burada, x, y, h polinomlarının vektör karşılıklarının ve σ vektörünün Hamming ağırlıkları w olmalıdır.

$$w(x) = w(y) = w(h) = w(\sigma) = w$$

3. $x + h \cdot y = s$ hesaplanır.

4. x, y, σ gizli anahtarlar olarak tutulurken h ve s açık anahtarlar olarak gönderilir.

Kapsülleme

1. $\mathbb{F}_2$ uzayından bir m vektörü seçilir. $w(m) = w$ olmalıdır.

2. Rastgeleliği artırmak için bir $\mathcal{G}$ özet fonksiyonu belirlenir (HQC-KEM için bu özet fonksiyonu SHAKE256 olarak belirlenmiştir.) ve $\mathcal{G}(m, h, s, T) = \theta$ hesaplanır. Burada, T vektörü, $\mathbb{F}_2^{128}$ uzayından seçilen keyfi bir vektördür.

3. SHAKE256-PRNG (Pseudo-Random Number Generator) fonksiyonuna girdi olarak θ verilir ve çıktı olarak üç vektör alınır. Bu vektörlere e, r_1, r_2 diyelim. Ağırlıkları w olmalıdır.

$$w(e) = w(r_1) = w(r_2) = w$$

4. $r_1 + h \cdot r_2 = u$ hesaplanır.

5. Kesme fonksiyonu (4.1.32) kullanılarak $\mathbf{tc}(mG + s \cdot r_2 + e, l) = v$ hesaplanır.

6. u ve v vektörleri uç uca eklenerek $c = (u, v)$ şifreli metni oluşturulur.

7. Rastgeleliği artırmak için bir $\mathcal{K}$ özet fonksiyonu daha belirlenir (HQC-KEM için burada kullanılan özet fonksiyonu da SHAKE256 olarak belirlenmiştir.) ve $\mathcal{K}(m, c) = K$ hesaplanır.

Kapsülden Çıkarma

1. Uygun bir kod çözme algoritması belirlenir. Girdi olarak x, y, σ ve c verilir, çıktı olarak bir aday mesaj alınır. Bu adaya m' diyelim.

2. $\mathcal{G}$ hash fonksiyonu kullanılarak $\mathcal{G}(m', h, s, T) = \theta'$ hesaplanır.

3. SHAKE256-PRNG algoritmasına girdi olarak θ' verilir ve çıktı olarak üç vektör alınır. Bu vektörlere e', r_1', r_2' diyelim. Ağırlıkları w olmalıdır.

$$w(e') = w(r_1') = w(r_2') = w$$

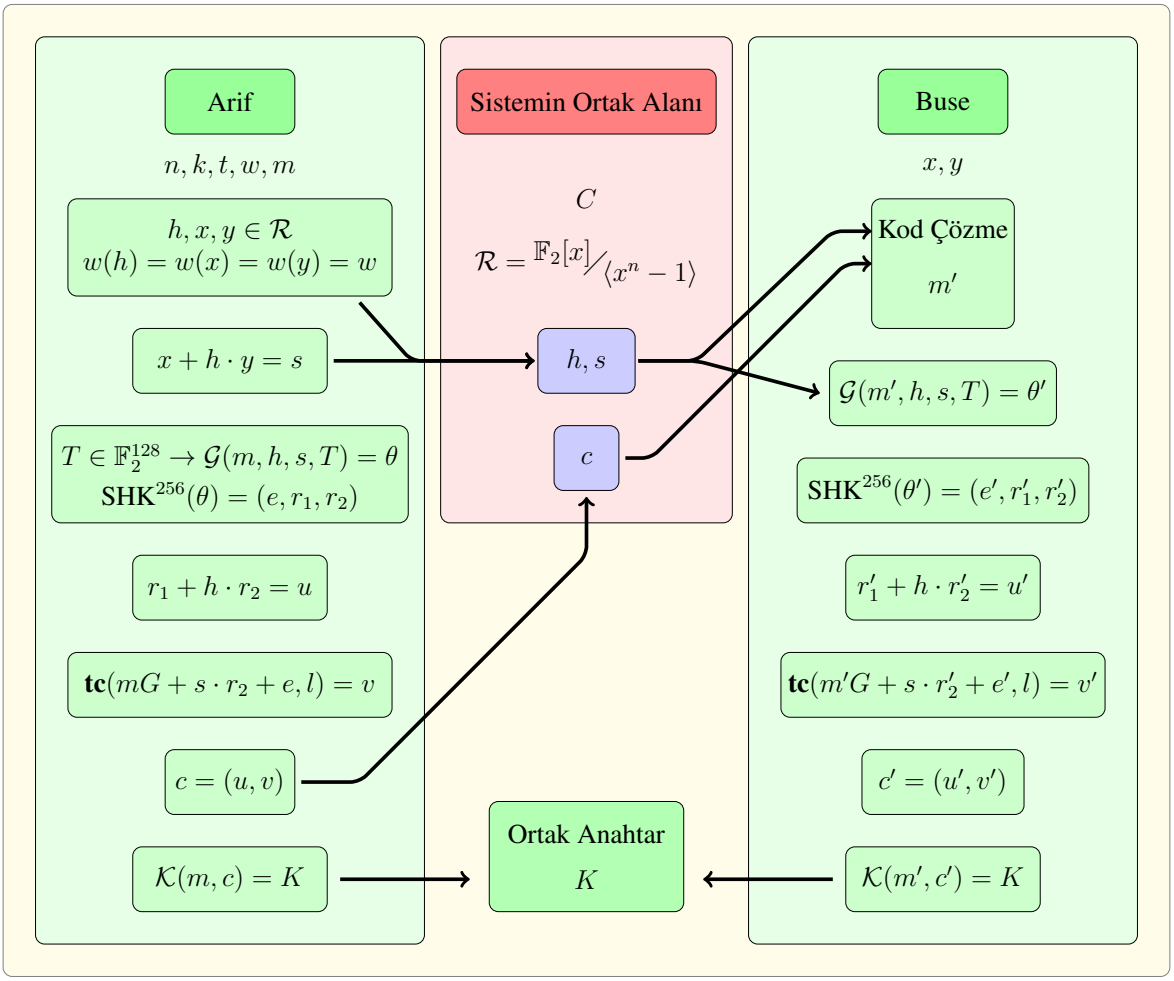
4. $r_1' + h \cdot r_2' = u'$ hesaplanır.

5. Kesme fonksiyonu (4.1.32) kullanılarak $\mathbf{tc}(m'G + s \cdot r_2' + e', l) = v'$ hesaplanır.

6. u' ve v' vektörleri uç uca eklenerek $c' = (u', v')$ oluşturulur.

7. Eğer $c = c'$ ise $\mathcal{K}$ özet fonksiyonu kullanılarak $\mathcal{K}(u', v') = K$ hesaplanır. Eğer $c \neq c'$ ise kapsülden çıkarma adımlarına yeniden başlanır. Kod çözme algoritmasıyla yeni bir m'' adayı bulunarak $c = c'$ olana kadar adımlar tekrarlanır.

HQC anahtar kapsülleme şifreleme algoritmasını, temel iletişim senaryomuz üzerinden aşağıdaki şekil ile özetleyebiliriz:



Şekil 4.12. HQC anahtar kapsülleme mekanizması

4.2 Kafes-Tabanlı Kriptografi

Kafes-tabanlı kriptografi, adından da anlaşılacağı üzere, matematiksel kafes yapıları üzerine kurulmuş kriptosistemlerden oluşur. Kafes-tabanlı kriptosistemlerin güvenlikleri, kafes yapılarındaki işlemlerin uzun uğraşlar gerektirmesi ve kafesler üzerinde çözülmesi zor olan problemlere dayanır. Ayrıca, kod-tabanlı kriptosistemler kadar olmasa da kuantum bilgisayarlar üzerinde oluşturulabilecek ataklara karşı çok iyi direnç gösterebilmektedirler. NIST'in yaptığı yarışmada ön elemeyi geçip 1. aşamaya aday gösterilen algoritmalardan 26 tanesi kafes tabanlıdır. Biz algoritmaları incelerken NIST'te yapılan kriptoataklara dayanma derecelerinden bağımsız olarak çalışacağız.

Kafes tabanlı algoritmalara geçmeden önce kafes yapıları üzerinde çözülmesi zor problemleri verelim.

4.2.1 Kafesler Üzerindeki Zor Problemler

Bu bölümde kafes-tabanlı algoritmaların güvenliklerinin sağlandığı, kafes yapıları üzerindeki zor problemler anlatılmıştır ve bu anlatım yapılırken [29] ve [30] numaralı kaynaklardan faydalanılmıştır.

4.2.1.1 En Kısa Vektör Problemi

Daha önce de belirttiğimiz gibi kafesler, $\mathbb{R}^n$'in ayrık elemanlara sahip altuzaylarıdır (2.7.1). Kriptografide kullanılan kafes yapıları tam kapasiteli kafeslerdir (full-rank lattices). Yani üst uzay olan $\mathbb{R}^n$ ile altuzay olan $\mathcal{L}$ kafesinin boyutları aynıdır (Yani n'dir). Ayrıca, kriptografide sadece $\mathbb{Z}^n$'in altkümesi olan kafesler kullanılır. Bunun anlamı, kullanılacak olan sonlu cisimlerin eleman sayısının bir asal sayı olduğudur.

q bir asal sayı ve $a_1, a_2, \ldots, a_n \in \mathbb{Z}_q^n$ vektörleri lineer bağımsız olsunlar. $\mathcal{B} = \{a_1, \ldots, a_n\}$ kümesini taban kabul eden kafesi aşağıdaki şekilde yazabiliriz:

$$\mathcal{L} = \{c_1a_1 + c_2a_2 + \cdots + c_na_n \mid c_1, \ldots, c_n \in \mathbb{Z}\}$$

$\mathcal{L}$ kafesinin elemanlarından boyu en kısa olanını bulma problemine "en kısa vektör problemi" (shortest vector problem) denir. İlk bakışta zor gibi görünmeyebilir fakat kafes-tabanlı kriptosistemlerin kullandıkları kafes yapılarında q ve n sayıları çok büyük olduğundan bu problemin çözümü oldukça fazla zaman almaktadır. Kaldı ki, küçük sayılarda bile bu problemi çözmek kolay değildir. Problemin zorluğunu daha iyi sezebilmemiz için bir örnek verelim ve çözümü bilgisayar yardımı olmadan bulmaya çalışalım.

Örnek 4.2.1. $q = 113$ ve $n = 2$ olsun. $a_1 = (13, 17), a_2 = (55, 75) \in \mathbb{Z}_q^n = \mathbb{Z}_{113}^2$ vektörlerini alalım. Bu vektörlerle oluşturulan

$$\mathcal{L} = \{c_1(13, 17) + c_2(55, 75) \mid c_1, c_2 \in \mathbb{Z}\}$$

kafesinin en kısa elemanını bulalım.

İlk aklımıza gelen durum, $c_1 = 1$ ve $c_2 = 0$ olduğu durum olabilir, çünkü a_1 vektörünün boyu daha kısadır. Fakat $\mathcal{L}$ kafesindeki en kısa vektör a_1 değildir. Örneğin, $c_1 = -8$ ve $c_2 = 2$ olduğu durumda

$$\begin{aligned} c_1(13, 17) + c_2(55, 75) &= -8(13, 17) + 2(55, 75) \\ &= (-104, -146) + (110, 150) \\ &= (6, 14) \end{aligned}$$

vektörü elde edilir ki $(6, 14)$ vektörünün boyu $(13, 17)$ vektörünün boyunun neredeyse $\frac{2}{3}$'ü kadardır. Fakat $(6, 14)$ vektörü de $\mathcal{L}$ kafesindeki en kısa vektör değildir. Örneğin $c_1 = -4$ ve $c_2 = 1$ için baktığımızda

$$\begin{aligned} c_1(13, 17) + c_2(55, 75) &= -4(13, 17) + 1(55, 75) \\ &= (-52, -68) + (55, 75) \\ &= (3, 7) \end{aligned}$$

vektörünü elde ederiz ki bu vektör $(6, 14)$ vektöründen daha kısadır.

Daha kısa bir vektör elde edebilmek için uygun c_1 ve c_2 katsayılarını bulmamız gerekiyor. Bu örnekten de anlayacağımız üzere deneyerek yapmaya kalkarsak, en kısa vektörü bulduğumuzdan emin olabilmemiz için $2^{113} - 1$ deneme yapmamız gerekmektedir.

En kısa vektör probleminin çözümünü doğrudan vermese de çözümü kısaltabilecek birkaç yol vardır. Bunlardan ikisi "Lagrange indirgeme (Lagrange reduction)" ve "LLL (Lenstra-Lenstra-Lovasz)" yöntemleridir. Lagrange indirgeme yöntemi, boyutu 2 olan kafes yapılarında kullanılır. Bu yöntem, taban vektörlerinden kısa olanının, uzun olandan

çıkarılarak uzun vektörden daha kısa olan bir vektör elde edilmesi mantığını temel alır. LLL yöntemi ise daha yüksek boyutlu kafes yapılarında kullanılır. Verilen taban elemanlarının değiştirilerek, daha kısa ve birbirine dik olan vektörlerin elde edilmesi mantığına dayanan bir yöntemdir. Bu yöntemler hakkında daha ayrıntılı bilgilere [31] numaralı kaynaktan ulaşılabilir.

4.2.1.2 En Yakın Vektör Problemi

Tam sayı girdili ve n-boyutlu bir kafesin $\mathbb{Z}^n$'in bir altuzayı olduğunu söylemiştik. $\mathcal{L} \subseteq \mathbb{Z}^n$ olmak üzere bir $b \in \mathbb{Z}^n$ vektörünü alalım ($b \in \mathcal{L}$ olması gerekmez). $\mathcal{L}$ kafesinin elemanlarından b'den farklı olup b'ye en yakın olanını bulma problemine "en yakın vektör problemi" (closest vector problem) denir.

İlk bakışta bu problem de basit gibi görünüyor fakat küçük sayılar kullanıldığında bile oldukça zordur. Bu zorluğu daha iyi sezebilmek için küçük sayıların kullanıldığı bir örnek verelim ve bilgisayar yardımı olmadan çözmeye çalışalım.

Örnek 4.2.2. Örnek 4.2.1 üzerinden gidelim. $a_1 = (13, 17), a_2 = (55, 75) \in \mathbb{Z}_{113}^2$ vektörlerinin ürettiği

$$\mathcal{L} = \{c_1(13, 17) + c_2(55, 75) \mid c_1, c_2 \in \mathbb{Z}\}$$

kafesinde $b = (90, 14)$ vektörüne en yakın elemanı bulmaya çalışalım.

İlk aklımıza gelen, a_1 ve a_2 vektörlerinin x koordinatlarını, b vektörünün x koordinatına en yakın olacak şekilde ayarlayıp c_1 ve c_2 katsayılarını bulmak olabilir (Bu bile ancak deneyerek elde edebileceğimiz bir yoldur). $2 \cdot 55 - 2 \cdot 13 = 84$ değeri 90 sayısına oldukça yakındır (Burada $c_1 = -2$ ve $c_2 = 2$ seçilmiştir). Bu katsayıları y koordinatı için kullandığımızda $2 \cdot 75 - 2 \cdot 17 = 116 \equiv 3 \pmod{113}$ sayısını elde ederiz. Yani $c_1 = -2$ ve $c_2 = 2$ için

$$c_1 \cdot a_1 + c_2 \cdot a_2 = -2(13, 17) + 2(55, 75) \equiv (84, 3) \pmod{113}$$

vektörü elde ediliyor. Bu vektöre b_1 diyelim. $b_1 = (84, 3)$

Fakat, x koordinatının yakınlığından biraz taviz verip y koordinatını yaklaştırmak istersek sonuç farklı çıkacaktır. Bunun için $c_1 = -1$ ve $c_2 = 2$ katsayılarını kullanabiliriz:

$$c_1 \cdot a_1 + c_2 \cdot a_2 = -1(13, 17) + 2(55, 75) \equiv (97, 20) \pmod{113}$$

Bu vektöre de b_2 diyelim. $b_2 = (97, 20)$

Görüldüğü üzere,

$$b - b_1 = (90, 14) - (84, 3) = (6, 11)$$

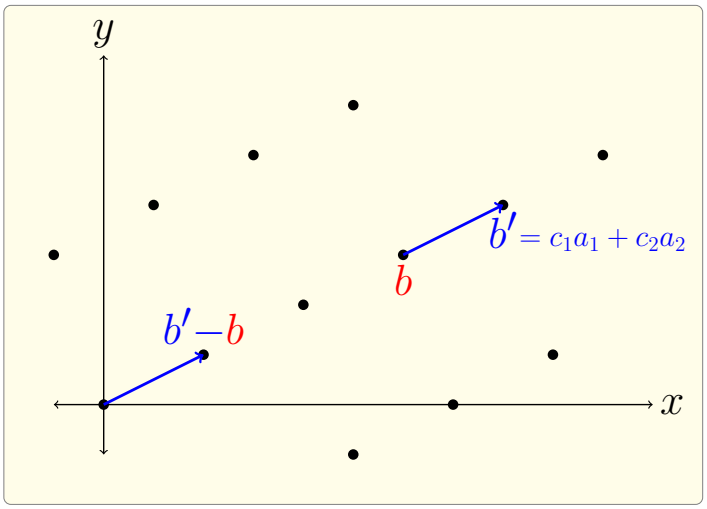
olmasına rağmen

$$b - b_2 = (90, 14) - (97, 20) = (-7, -6)$$

oldu. Buradan anlıyoruz ki b_2 vektörü, b_1 vektörüne kıyasla, b vektörüne daha yakındır.

Daha farklı katsayılar kullanarak daha yakın vektörler elde edebiliriz fakat bunun için oldukça fazla durumu hesaplamamız gerekmektedir.

NOT : Eğer verilen b vektörü $\mathcal{L}$ kafesinin elemanı ise, ve $\mathcal{L}$ kafesinde b'den farklı olup, b'ye en yakın olan vektörü bulabildiysek, yani kısacası en yakın vektör problemini çözebildiysek, aynı zamanda en kısa vektör problemini de çözmüş oluruz. b'ye en yakın olan vektöre b' dersek, $\mathcal{L}$ kafesindeki en kısa vektör $b' - b$ vektörüdür. Kriptografide kullanılan kafes yapıları sonlu cisimler üzerinde inşa edildiği için kafesin merkezini, yani orijin noktasını, kafesteki herhangi bir nokta olarak seçebiliriz. Dolayısıyla, orijini b noktasına taşıyarak en kısa vektörün, $b' - b$ olduğunu rahatlıkla görebiliriz. Bunu aşağıdaki şekil ile daha rahat anlayabiliriz:



Şekil 4.13. En kısa ile en yakın vektör problemleri arasındaki ilişki

4.2.1.3 En Kısa Taban Problemi

Kriptografi kullanılan kafes yapıları, birer altuzay olduklarından, tek bir tabana bağlı olarak yazılmak zorunda değillerdir. Bir sürü farklı taban ile ifade edilebilirler. Bir $\mathcal{L}$ kafesinin tüm tabanlarının en uzun vektörlerini kıyaslayıp, bunlardan en kısa olana sahip tabanı bulma problemine "en kısa taban problemi" (shortest basis problem) denir.

En kısa taban bulma problemi, diğer iki problem gibi ilk bakışta kolay görünmüyor. Hatta problemin tanımından bile, deneme-yanılma yoluna başvurulması gerektiği açıkça görülüyor. En kısa taban bulma probleminin çözümü için Gram-Schmidt ve Lagrange indirgeme yöntemleri kullanılarak yaklaşık sonuçlar elde edilebilir fakat en doğru sonuç için deneme-yanılma yolundan başka bir çare henüz bulunamamıştır.

Problemin zorluğunun daha iyi anlaşılabilmesi için küçük sayılar üzerinden, bilgisayar yardımı olmadan bir örnek yapalım:

Örnek 4.2.3. $\mathbb{Z}_{113}^2$ üzerinde $a_1 = (13, 43)$ ve $a_2 = (41, 97)$ vektörleri ile üretilen

$$\mathcal{L} = \{c_1 a_1 + c_2 a_2 \mid c_1, c_2 \in \mathbb{Z}\}$$

kafesinin en kısa tabanını bulmaya çalışalım.

Elimizde bulunan ilk aday taban $\mathcal{B}_1 = \{a_1 = (13, 43), a_2 = (41, 97)\}$ kümesidir. Fakat bu tabanın, aranan en kısa taban olmadığı aşikardır. Çünkü verilen vektörlerden uzun olanından kısa olanını çıkardığımızda daha kısa bir vektöre ulaşabiliriz.

$$a_2 - a_1 = (41, 97) - (13, 43) = (28, 54)$$

Elde edilen bu vektöre a_3 diyelim. $a_3 = (28, 54)$.

Bu durumda yeni taban adayımız $\mathcal{B}_2 = \{a_1 = (13, 43), a_3 = (28, 54)\}$ kümesi olur. En kısa taban problemindeki "kısalık" tanımı düşünüldüğünde $\mathcal{B}_2$ tabanının $\mathcal{B}_1$ tabanından daha kısa olduğu açıkça görülür. Fakat $\mathcal{L}$ kafesi için en kısa taban $\mathcal{B}_2$ de değildir. Çünkü bu tabandan daha kısasının bulunabileceği açıktır.

$\mathcal{B}_2$ tabanındaki vektörlerden uzun olanından kısa olanını çıkardığımızda

$$a_4 := a_3 - a_2 = (28, 54) - (13, 43) = (15, 13)$$

vektörünü ve $\mathcal{B}_3 = \{a_1 = (13, 43), a_4 = (15, 13)\}$ tabanını elde ederiz. Fakat bu taban da en kısa olamaz, çünkü $a_5 := a_1 - a_4$ vektörü a_1 vektöründen daha kısadır ve böylece $\mathcal{B}_4 = \{a_5, a_4\}$ kümesi $\mathcal{L}$ kafesi için $\mathcal{B}_3$ kümesinden daha kısa bir taban belirtir.

En kısa tabanı bulmak için bu işlem sürecine devam edilmesi gerekiyor ve en doğru sonuç için, neredeyse bütün kombinasyonların denenmesi gerekiyor ve bu da problemin zorluğunun seviyesini daha net görmemizi sağlıyor. (Bu örnek için uyguladığımız yöntem Lagrange indirgeme yönteminin biraz farklı bir halidir.)

Problemin zorluğunu anladığımıza göre, örneğe devam etmeye gerek yoktur. Kaldı ki en doğru sonucu bulmamız sayfalarca sürebilir.

4.2.1.4 Gürültülüyken Çözme Problemi (Learning With Errors)

Gürültülüyken çözme problemi kriptografik anlamda güvenliği sağlayan en önemli problemlerden biridir. 2005 yılında Oded Regev tarafından ortaya konulmuştur.

Bir eşlenik denklem sistemini çözmenin, küçük bir değişim karşısında ne kadar zor olabileceğini gösteren bu problemi önce cebirsel olarak açıklayıp ardından problemin kafeslerle olan bağlantısını anlatalım.

q bir asal sayı olmak üzere tüm girdilerini $\mathbb{Z}_q$ cisminden alan bir eşlenik denklem sistemi aşağıdaki gibi yazılabilir:

$$\begin{array}{ccccccccc}
a_{11}s_1 & + & a_{21}s_2 & + & \cdots & + & a_{n1}s_n & = & b_1 \pmod q \\
a_{12}s_1 & + & a_{22}s_2 & + & \cdots & + & a_{n2}s_n & = & b_2 \pmod q \\
a_{13}s_1 & + & a_{23}s_2 & + & \cdots & + & a_{n3}s_n & = & b_3 \pmod q \\
\vdots & & \vdots & & \vdots & & \vdots & & \vdots \\
a_{1m}s_1 & + & a_{2m}s_2 & + & \cdots & + & a_{nm}s_n & = & b_m \pmod q
\end{array}$$

Bu denklem sistemini aşağıdaki gibi matris formunda yazabiliriz:

$$A_{m\times n}\cdot s_{n\times 1} = \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{n1} \\ a_{12} & a_{22} & \cdots & a_{n2} \\ a_{13} & a_{23} & \cdots & a_{n3} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1m} & a_{2m} & \cdots & a_{nm} \end{bmatrix} \cdot \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ \vdots \\ s_n \end{bmatrix} \equiv \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_m \end{bmatrix} = b_{m\times 1} \pmod q$$

Bu sistemin ilaveli katsayılar matrisini (2.5.29) yazıp Gauss metoduyla, yani satır indirgenmiş eşelon formuna (2.5.35) getirerek sistemin çözümünün olup olmadığını anlarız ve varsa tüm çözümlerini buluruz. Fakat bu denklem sistemine, aşağıdaki şekilde bir hata vektörü (gürültü) eklersek çözüm neredeyse imkansız hale gelir; diğer bir deyişle, deneme-yanılmadan başka bir yol kalmaz:

$$\begin{bmatrix} a_{11} & a_{21} & \cdots & a_{n1} \\ a_{12} & a_{22} & \cdots & a_{n2} \\ a_{13} & a_{23} & \cdots & a_{n3} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1m} & a_{2m} & \cdots & a_{nm} \end{bmatrix} \cdot \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ \vdots \\ s_n \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ \vdots \\ e_m \end{bmatrix} = b + e \equiv \begin{bmatrix} b'_1 \\ b'_2 \\ b'_3 \\ \vdots \\ b'_m \end{bmatrix} \pmod{q}$$

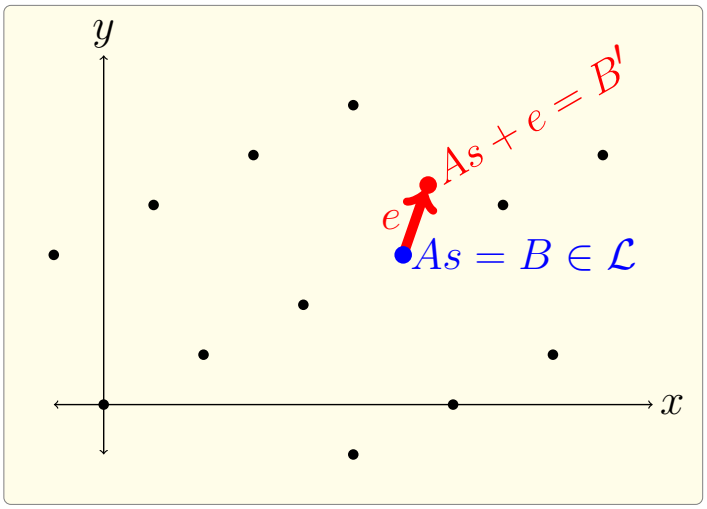
Tanım 4.2.4. Yukarıda verilen gürültülü denklem sisteminin çözülüp s vektörünün bulunması problemine "gürültülüyken çözme problemi" (learning with errors problem, veya kısaca LWE) denir.

Şimdi bu problemin kafes yapılarıyla olan bağlantısını görelim:

Denklem sisteminde verilen A matrisinin sütunlarını bir $\mathcal{L}$ kafesinin baz vektörleri olarak kabul edersek, $B := A \cdot s$ çarpımı $\mathcal{L}$ kafesinin bir elemanı olur. Yani, $\forall\ i = 1, \ldots, n$ için $a_i = (a_{i1}, a_{i2}, \ldots, a_{im})$ olmak üzere

$$B := \sum_{i=1}^{n} s_i a_i \in \mathcal{L} = \left\{ c_1 a_1 + c_2 a_2 + \ldots + c_n a_n \mid c_i \in \mathbb{Z} \right\}.$$

Bu çarpıma bir de e gürültüsünü ekleyince $B' := B + e = A \cdot s + e$ noktası elde edilir ki bu nokta $\mathcal{L}$ kafesindeki B noktasına e vektörünün boyu kadar uzaklıkta bir noktadır. Eğer e vektörü yeterince küçük seçilirse gürültülüyken çözme ve en yakın vektör problemleri eşdeğer olurlar. Bu bağlantıyı, aşağıdaki şekil ile daha iyi anlayabiliriz:



Şekil 4.14. Gürültülüyken çözme ve en yakın vektör problemleri bağlantısı

Problemin genel halinde A matrisi $m \times n$ formundadır. Bu yüzden çözümün varlığı ve tekliği garanti edilemez. Fakat problemin kafesler için olan versiyonunda bu sorun yaşanmayacaktır, çünkü kafes tabanlı kriptografide tam kapasiteli kafesler (full-rank lattices) kullanılmaktadır. Yani burada A matrisi $n \times n$ formundadır. Dolayısıyla n değişkenli, n tane lineer bağımsız denklemden oluşan bir sistemimiz vardır. Bu nedenle çözüm tek türlü olarak bulunur.

Şimdi kafes tabanlı algoritmaların işleyişlerine geçip, bu zor problemleri nasıl kullandıklarını görelim. İlk algoritmamız, anahtar kapsülleme mekanizması ve açık anahtarlı şifreleme sistemi olarak kullanılabilen ve aynı zamanda, anahtar kapsülleme kategorisinde, NIST'in yaptığı yarışmada rakiplerini eleyerek şampiyonluğu kazanan KYBER algoritmasıdır.

Biz bu tezde KYBER algoritmasının sadece açık anahtarlı şifreleme versiyonunu inceleyeceğiz, anahtar kapsülleme mekanizması olan versiyonu, burada anlatabileceğimizden daha karmaşık bir yapıya sahiptir.

4.2.2 KYBER Açık Anahtarlı Şifreleme (KYBER-PKE)

Bu kısımda KYBER algoritması [32] ve [33] numaralı kaynaklardan faydalanılarak anlatılmıştır.

KYBER algoritmasının güvenliği gürültülüyken çözme (4.2.1.4) problemine dayanmaktadır. Algoritmanın standart halindeki asal sayı $q = 3329$, anahtar uzunluğu $n = 256$'dır ve kullanılan kafesin boyutuna göre üç farklı versiyonu vardır. Bu versiyonlar aşağıdakilerdir:

Versiyon Adı	**n**	**k**	**q**
KYBER512	256	2	3329
KYBER768	256	3	3329
KYBER1024	256	4	3329

Biz, algoritmanın kendi parametrelerinden ziyade, işleyişini öğrenmek istediğimiz için genel parametreler üzerinden anlatacağız.

Algoritmaya geçmeden önce birkaç notasyon verelim:

- $\mathcal{R}_q := {}^{\mathbb{Z}_q[x]}\!/_{\langle x^n + 1 \rangle}$
- $\mathcal{S}_n := \mathcal{R}_q$'nun katsayıları $[-n, n]$ aralığında olan elemanları.
- $\{0,1\}^n = \mathbb{Z}_2^n :=$ Açık metin uzayı.
  Bir $m \in \{0,1\}^n$ açık metni $\mathcal{R}_q$'dan katsayıları 0 ve 1 olan polinomlarla eşleştirilebilir:
  Örneğin, $n = 5$ ve $m = 10110$ ise $m \leftrightarrow 1 + x^2 + x^3$ eşlemesi yapılabilir. (2.1.)
- $\lfloor c \rfloor := c$'ten küçük olan en büyük tam sayı.
- $\lceil c \rceil := c$'ten büyük olan en küçük tam sayı.
- $\lceil c \rfloor := c$'e en yakın tam sayı.
  Örneğin, $\lceil 13,2 \rfloor = 13, \ \lceil 13,7 \rfloor = 14, \ \lceil 13,5 \rfloor = 14$

Son olarak bir de algoritmanın kullandığı "çevirme (rounding) fonksiyonu"nu gösterelim:

q asal sayısı bir tek sayı olsun ve $[0, q-1]$ aralığından keyfi bir a tam sayısı alalım.

$\mathbb{Z}_q$'nun elemanlarını $0, 1, \ldots, q-1$ yerine aşağıdaki şekilde sıralayalım:

$$-\frac{q-1}{2}, \ldots, -2, -1, 0, 1, 2, \ldots, \frac{q-1}{2}$$

Şimdi yukarıda verdiğimiz notasyonları kullanarak bu sonlu cismi aşağıdaki şekilde üç parçaya ayıralım:

$$\{-\tfrac{q-1}{2}, -\tfrac{q-3}{2}, \ldots, \lfloor -\tfrac{q}{4} \rfloor\} \cup \{\lceil -\tfrac{q}{4} \rceil, \ldots, -2, -1, 0, 1, 2, \ldots, \lfloor \tfrac{q}{4} \rfloor\} \cup \{\lceil \tfrac{q}{4} \rceil, \ldots, \tfrac{q-3}{2}, \tfrac{q-1}{2}\}.$$

$\mathbb{Z}_q$ sonlu cismi üzerinden $\mod$ alırken kalanları bu üç kümenin birinden alacağız: Yani, $a \in \mathbb{Z}_q$ için $a \equiv a' \pmod q$ gösteriminde a' bu üç kümeden birinde olacak.

Tanım 4.2.5. Yukarıda verilen notasyonlarla birlikte çevirme (rounding) fonksiyonu aşağıdaki şekilde tanımlanır:

$$Rnd_q(a) := \begin{cases} 0 & , \quad -\frac{q}{4} < a' < \frac{q}{4} \\ 1 & , \quad \text{diğer} \end{cases}$$

Yani, seçilen a sayısı, cismin üç parçaya ayırdığımız halinin orta kısmına denk geliyorsa fonksiyonun görüntüsü 0'dır ve diğer durumlarda 1'dir.

Not : Algoritmanın işleyişinde bir polinomun çevirme fonksiyonu altındaki görüntüsü bulunurken, fonksiyon sadece katsayılara uygulanır monomlar değişmez.

Şimdi algoritmanın işleyişine geçebiliriz:

Açık anahtarlı şifreleme sistemleri, daha önce de belirttiğimiz gibi üç aşamadan oluşur. Bunlar, anahtar üretim aşaması, şifreleme aşaması ve deşifreleme aşamasıdır. KYBER-PKE için bu aşamaları temel iletişim senaryomuz üzerinden sadeleştirilmiş haliyle görelim:

Anahtar Üretim Aşaması

1. Adım : Önce Arif parametreleri belirler: q, n, k, n_1, n_2

2. Adım : Bir $A \in \mathcal{R}_q^{k \times k}$ matrisi belirler.

3. Adım : Bir $s \in \mathcal{S}_{n_1}^k$ ve bir de $e \in \mathcal{S}_{n_2}^k$ seçer.

4. Adım : $As + e = t$'yi hesaplar.

5. Adım : A ve t'yi açık anahtarlar olarak gönderirken s'yi gizli anahtar olarak tutar.

Şifreleme Aşaması

1. Adım : Buse şifrelemek istediği $m \in \{0,1\}^n$ mesajını belirler.

2. Adım : Arif'in herkese açık olarak gönderdiği A ve t anahtar çiftini alır.

3. Adım : Bir $r \in \mathcal{S}_{n_1}^k$, bir $e_1 \in \mathcal{S}_{n_2}^k$ ve bir de $e_2 \in \mathcal{S}_{n_2}$ seçer.

4. Adım : $A^\top r + e_1 = u$ ve $t^\top r + e_2 + \lceil \frac{q}{2} \rfloor m = v$ hesaplarını yapar.

5. Adım : u ve v'yi art arda ekleyerek $c = (u, v)$ şifreli metnini elde eder ve Arif'e gönderir. Burada $c \in \mathcal{R}_q^k \times \mathcal{R}_q$'dur.

Deşifreleme Aşaması

1. Adım : Buse'den c şifreli metnini alan Arif, metni tekrardan iki parçaya ayırarak u ve v'yi elde eder.

2. Adım : $v - s^\top u$'yu hesaplar.

3. Adım : Elde ettiği sonucun çevirme fonksiyonu (4.2.5) altındaki görüntüsünü hesaplayıp m mesajına ulaşır:

$$m = Rnd_q(v - s^\top u)$$

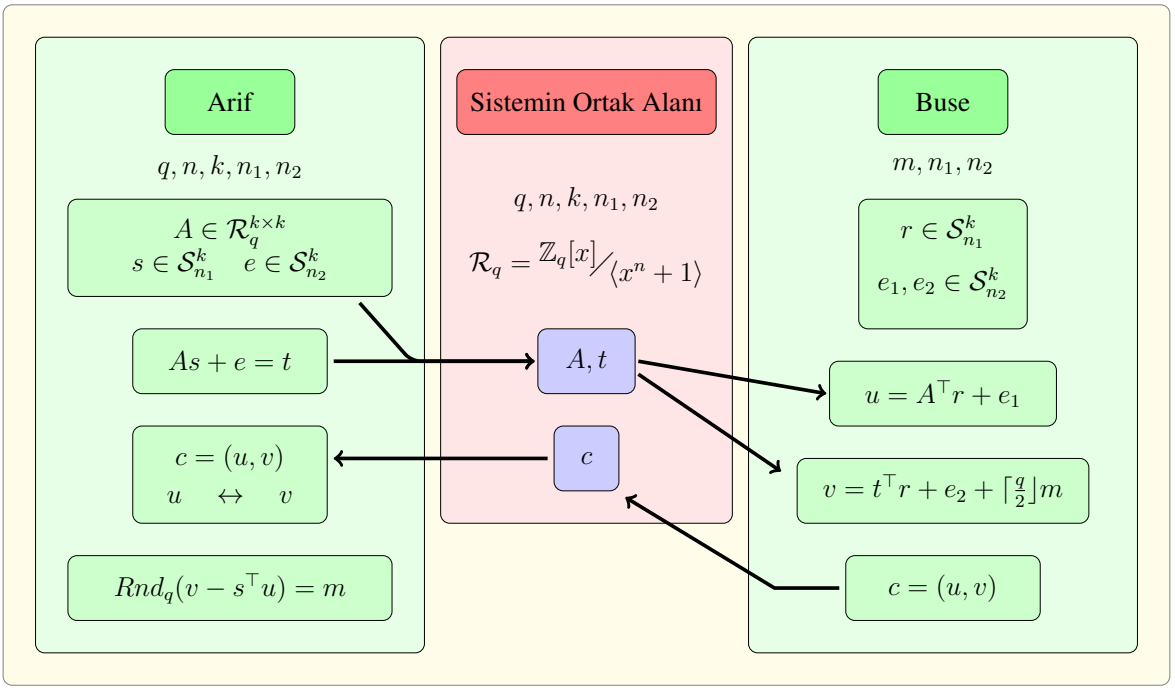
Son adımda, $v - s^\top u$ ifadesinin çevirme fonksiyonu altındaki görüntüsüyle mesajın doğrudan nasıl bulunabildiği kısmı bize şaşırtıcı gelebilir. Aslında bu sonuç, algoritma sürecinde

seçilen parametrelerin özel oluşuyla ilgilidir. Açıklayalım; Öncelikle $v - s^\top u$ ifadesini aşağıdaki şekilde açıp sadeleştirelim:

$$\begin{aligned} v - s^\top u &= (t^\top r + e_2 + \lceil \tfrac{q}{2} \rfloor m) - s^\top u \\ &= (s^\top A^\top + e^\top) r + e_2 + \lceil \tfrac{q}{2} \rfloor m - s^\top (A^\top r + e_1) \\ &= s^\top A^\top r + e^\top r + e_2 + \lceil \tfrac{q}{2} \rfloor m - s^\top A^\top r - s^\top e_1 \\ &= e^\top r + e_2 - s^\top e_1 + \lceil \tfrac{q}{2} \rfloor m \end{aligned}$$

Son aşamada elde edilen kısımdaki $\lceil \frac{q}{2} \rfloor m$ kısmı mesajın doğrudan kendisidir, çünkü $Rnd_q(\frac{q}{2}) = 1$'dir ve $1 \cdot m = m$'dir. Geriye kalan $e^\top r + e_2 - s^\top e_1$ ifadesindeki e, r, e_1, e_2 ve s değişkenleri ise $Rnd_q(e^\top r + e_2 - s^\top e_1) = 0$ değerini verecek şekilde seçilir. Böylelikle algoritmanın son adımındaki ifadenin çevirme fonksiyonu altındaki görüntüsü bize doğrudan m mesajını verir.

KYBER-PKE algoritmasını aşağıdaki şekil ile özetleyebiliriz:



Şekil 4.15. KYBER-PKE algoritması

Algoritmanın daha iyi anlaşılabilmesi için küçük sayılar kullanarak bir örnek verelim:

Örnek 4.2.6. Örneğimizi temel iletişim senaryomuz üzerinden verelim:

Anahtar Üretim Aşaması

1. Adım : Arif'in seçtiği parametreler $q = 113, n = 4, k = 2, n_1 = 2, n_2 = 2$ olsun.

2. Adım : Arif, $A \in \mathcal{R}_{113}^{2\times 2}$ olacak şekilde matrisi seçmiş olsun:

$$A = \begin{bmatrix} 17 + 23x + 90x^2 + 37x^3 & 15 + 49x + 77x^2 + 105x^3 \\ 98 + 47x + 15x^2 + 100x^3 & 56 + 71x + 98x^2 + 4x^3 \end{bmatrix}$$

3. Adım : Arif, $s \in \mathcal{S}_2^2$ ve $e \in \mathcal{S}_2^2$ olacak şekilde aşağıdaki vektörleri seçmiş olsun:

$$s = \begin{bmatrix} 2 + x + x^2 + x^3 \\ 2x + x^3 \end{bmatrix}, \qquad e = \begin{bmatrix} 1 - x + x^2 + x^3 \\ 1 + x + x^2 \end{bmatrix}$$

4. Adım : Arif seçimlerini tamamladığına göre $As + e = t$ hesabını yaparak aşağıdaki sonuca ulaşır:

$$t = \begin{bmatrix} 78 + x + 64x^2 + 35x^3 \\ 69 + 92x + 101x^2 + 47x^3 \end{bmatrix}$$

5. Adım : Son olarak Arif, A ve t'yi herkese açık bir şekilde Buse'ye gönderir. Yani A ve t Arif'in açık anahtarlarıdır. s'yi ise gizli tutar.

Şifreleme Aşaması

1. Adım : Buse, şifrelemek istediği $m \in \{0, 1\}^4$ mesajını 1111 şeklinde seçmiş olsun. O halde Buse'nin mesajının polinom hali $1 + x + x^2 + x^3$ şeklindedir.

2. Adım : Buse, $r \in \mathcal{S}_2^2$, $e_1 \in \mathcal{S}_2^2$ ve $e_2 \in \mathcal{S}_2$ olacak şekilde aşağıdakileri seçmiş olsun:

$$r = \begin{bmatrix} 2 - 2x + 2x^2 + x^3 \\ 1 + 2x + x^2 \end{bmatrix}, \quad e_1 = \begin{bmatrix} 1 + 2x + 2x^2 + x^3 \\ 1 - 2x - 2x^2 + x^3 \end{bmatrix}, \quad e_2 = \begin{bmatrix} 1 + x + x^2 + x^3 \end{bmatrix}$$

3. Adım : Buse, $A^\top r + e_1 = u$ ve $t^\top r + e_2 + \lceil \frac{113}{2} \rfloor m = v$ hesaplarını yaparak aşağıdaki sonuçlara ulaşır:

$$u = \begin{bmatrix} 15 + 106x + x^2 + 22x^3 \\ 101 + 71x + 49x^2 + 102x^3 \end{bmatrix}, \qquad v = \begin{bmatrix} 29 + 66x + 94x^2 + 82x^3 \end{bmatrix}$$

4. Adım : Buse, bulduğu u ve v değerlerini uç uca ekleyerek $c = (u, v)$ şifreli metnini elde eder ve herkese açık bir şekilde Arif'e gönderir.

Deşifreleme Aşaması

1. Adım : Arif, Buse'den aldığı c şifreli metnini tekrar iki parçaya ayırarak u ve v'yi elde eder.

2. Adım : Arif, $v - s^\top u$'yu hesaplayarak aşağıdaki sonuca ulaşır:

$$64 + 48x + 66x^2 + 56x^3$$

3. Adım : Son olarak Arif elde ettiği polinomun Rnd_{113} çevirme fonksiyonu altındaki görüntüsünü hesaplayarak m mesajına ulaşır. Bunun için öncelikle $\mathbb{Z}_{113}$ cismini, algoritmanın istediği şekilde parçalamalıyız:

$$\mathbb{Z}_{113} = \{-56, \ldots, -29\} \cup \{-28, \ldots, -1, 0, 1, \ldots, 28\} \cup \{29, \ldots, 56\}$$

Arif'in hesapladığı polinomun bu parçalamadaki karşılıkları hangi kümenin içerisinde kalıyorsa çevirme fonksiyonu altındaki görüntüleri de ona göre değer alır.
$\{-28, \ldots, 0, \ldots, 28\}$ kümesinin içinde kalanların Rnd_{113} fonksiyonu altındaki görüntüleri 0, diğer kümelerde kalanların görüntüleri 1 olacaktır: Dolayısıyla

$$Rnd_{113}(56) = Rnd_{113}(48) = 1$$

olur. $66 \equiv -47 \pmod{113}$ ve $64 \equiv -49 \pmod{113}$ olduğundan

$$Rnd_{113}(66) = Rnd_{113}(64) = 1$$

olur. Nihai sonuç olarak, Arif'in bulduğu polinomun Rnd_{113} çevirme fonksiyonu altındaki görüntüsü $1 + x + x^2 + x^3$ olarak bulunur. Bu görüntünün $\{0, 1\}^4$ kümesindeki karşılığı da 1111'dir. Yani deşifreleme sonucunda elde edilen mesaj $m = 1111$ olup, Buse'nin mesajıyla aynı olarak bulunmuştur.

4.2.3 CRYSTALS-Dilithium

Bu kısımda CRYSTALS-Dilithium algoritması, [34] ve [35] numaralı kaynaklardan faydalanılarak anlatılmıştır.

CRYSTALS-Dilithium algoritması, kafes yapıları üzerindeki gürültülüyken çözme problemini (4.2.1.4) kullanan bir dijital imza algoritmasıdır. Kısaca Dilithium olarak adlandırılan algoritma, NIST'in yaptığı standartlaştırma yarışmasının dijital imzalama dalındaki rekabeti kazanarak tahtın sahibi olmuştur. Dilithium algoritmasını ön plana çıkaran en önemli sebep, barındırdığı büyük imza boyutlarına rağmen diğer algoritmalara kıyasla çok düşük gecikme süresi sunmasıdır. Ayrıca kullanım alanı olarak da çok geniş bir yelpazeye sahiptir. Güvenli iletişim protokolleri (TLS, VPN), yüksek boyutlu sistemlerdeki yazılım güncellemelerinin doğrulanması ve bulut depolama sistemlerinin cihazlarla güvenli alışveriş yapmasını sağlayan ioT sistemleri gibi birçok alanda kullanılmaktadır.

Dilithium algoritmasının, kabul gören ve kullanımda olan üç temel versiyonu vardır: Bunlar Dilithium II, Dilithium III ve Dilithium V olarak adlandırılmaktadır. Bu versiyonlarda

kullanılan parametreler aşağıdaki tabloda verilmiştir:

Parametre	**Dilithium II**	**Dilithium III**	**Dilithium V**
Kafes boyutu	4	6	8
Asal sayı	8380417	8380417	8380417
Gürültünün boyu	2	4	2
İmza uzunluğu	~ 19660 bit	~ 26210 bit	~ 35220 bit
Açık anahtar uzunluğu	~ 10650 bit	~ 15560 bit	~ 21300 bit
Gizli anahtar uzunluğu	~ 20480 bit	~ 32770 bit	~ 42590 bit

Dilithium II algoritması, genelde daha küçük imzalara ihtiyaç duyan sistemler için kullanılır. Bu alanlara, mobil cihaz haberleşmeleri ve sosyal medya uygulamaları örnek verilebilir. Dilithium III algoritması ise biraz daha büyük imza ve güvenlik seviyesi gerektiren uygulamalar için elverişlidir. Örneğin, mobil bankacılık ve kurumsal hesapları koruyan yazılımlar. Dilithium V algoritması ise resmi ve askeri haberleşmeler gibi en üst seviye güvenlik önlemleri gerektiren iletişim süreçleri için kullanışlıdır.

Dilithium algoritmasının bu üç versiyonunu da ayrı ayrı incelemek yerine üçünün de çalışma prensibini anlayabileceğimiz bir genel versiyonunu göreceğiz. Daha öncesinde, Dilithium algoritmasının ilham kaynağı olan ve Dilithium'a kıyasla anlaşılabilirliği daha yüksek olan Schnorr algoritmasını görelim:

4.2.3.1 Schnorr

Şu ana kadar ele aldığımız tüm dijital imza algoritmalarında olduğu gibi, Schnorr imza algoritması da üç aşamadan oluşur: anahtar üretimi, imzalama, imza doğrulama aşamaları. Şimdi, algoritmada kullanacağımız parametreleri belirleyip temel iletişim senaryomuz üzerinden adımlara geçelim:

Parametreler

- Bir $n \in \mathbb{Z}$ tam sayısı

- n mertebeli bir çarpımsal devirli grup G
- G grubunun bir üreteci g
- Bir **H** özet fonksiyonu

G grubunu örneğin, p bir asal olmak üzere $\mathbb{Z}_p^* = \{1, 2, 3, 4, \ldots, p-1\}$ şeklinde alabiliriz.

Anahtar Üretimi Aşaması

1. Adım : Arif bir $a \in [1, n-1]$ tam sayısı seçer.

2. Adım : g^a'yı hesaplar.

3. Adım : a sayısını gizli anahtar olarak tutarken, g^a sayısını açık anahtar olarak gönderir.

Burada kullanılan matematiksel problem, kolayca anlaşılabileceği üzere ayrık logaritma problemidir.

İmzalama Aşaması

1. Adım : Arif, imzalaması gereken $M \in \{0,1\}^n$ mesajını alır.

2. Adım : Arif bir $y \in [1, n-1]$ tam sayısı seçer.

3. Adım : Ardından $w = g^y$ sayısını hesaplar.

4. Adım : Daha sonra, **H** özet fonksiyonunu kullanarak $\mathbf{H}(M, w) = c$ değerini bulur.

5. Adım : Son olarak, $z = y + ca$ sayısını hesaplar ve (c, z) imza çiftini oluşturur.

İmza Doğrulama Aşaması

1. Adım : Buse, Arif'in oluşturduğu (c, z) imza çiftini alıp c ve z olarak ayırır.

2. Adım : Daha sonra, $w' = g^z (g^a)^{-c}$ hesabını yapar.

3. Adım : Son olarak, **H** özet fonksiyonunu kullanarak $\mathbf{H}(M, w') = c'$ değerini hesaplar. Eğer $c = c'$ ise imza doğru, aksi halde hatalıdır.

Burada, $c = c'$ iken neden imzanın doğru olduğunu merak ediyorsak aşağıdaki eşitliğe bakabiliriz:

$$w' = g^z(g^a)^{-c} = g^z g^{-ca} = g^{y+ca} g^{-ca} = g^{y+ca-ca} = g^y = w$$

Şimdi Dilithium algoritmasının işleyişine geçebiliriz:

4.2.3.2 Dilithium

Adımlardan önce notasyonları verelim:

- q bir asal sayı
- $\mathbb{Z}_q$ sonlu cisim.
- $\mathcal{R}_q := {}^{\mathbb{Z}_q[x]}\!/_{\langle x^n + 1 \rangle}$
- $\eta := q \approx 2^k$ iken $k - n = \eta$ olacak şekildeki tam sayı.
- $\mathcal{S}_\eta := \mathcal{R}_q$'daki, katsayıları $[-\eta, \eta]$ aralığında olan polinomlar kümesi.
- $\lambda_1 := 2^{19}$
- $\tilde{\mathcal{S}}_{\lambda_1} := \mathcal{R}_q$'daki, katsayıları $(-\lambda_1, \lambda_1]$ aralığında olan polinomlar kümesi.
- $(k, l) := k > l$ olacak şekildeki tam sayı ikilisi.
- $\mathcal{B}_\tau := \mathcal{S}_1$'deki, tam olarak τ tane katsayısı ± 1 (diğer katsayıları 0) olan polinomlar kümesi.
- $\beta := \tau \cdot \eta$
- $\lambda_2 := \frac{\alpha}{2}$
- $\lambda :=$ oluşacak imza çiftinin bit uzunluğunun yarısı.
- $\mathbf{H} :=$ özet fonksiyonu

Algoritmanın standartlaştırılmış halindeki parametreler $\lambda_1 = 2^{19}$, $\eta = 2$, $(k, l) = (8, 7)$, $\tau = 60$, $\beta = 120$, $n = 256$'dır ve kullanılan özet fonksiyonu, McEliece ve HQC kriptosistemlerinde de kullanılan **SHAKE256** adındaki özet fonksiyonudur.

Algoritmanın işleyişine geçmeden önce son olarak iki fonksiyonun tanımını verelim: Bunlar Yüksek-Bit (HighBits) ve Düşük-Bit (LowBits) fonksiyonlarıdır.

Tanım 4.2.7. (Yüksek-Bit ve Düşük-Bit Fonksiyonları) İki fonksiyonun tanımlarını birlikte veriyor oluşumuzun sebebi bu fonksiyonların aslında bir işlemin sonucunu paylaşmalarıdır. Öncelikle bir q asal sayısı belirlenir. (Tabi ki algoritmaların güvenlik seviyelerini yüksek tutmak amacıyla bu asal sayı çok büyük olacağı için aynı zamanda bir tek sayıdır.) Ardından bir $r \in [0, \ldots, q-1]$ sayısı belirlenir ve son olarak bir de $q-1$ sayısını tam bölecek şekilde bir α sayısı seçilir. Yani bir m sayısı için $\alpha \cdot m = q - 1$'dir. Daha sonra $\frac{-a}{2} \leq r_0 \leq \frac{\alpha}{2}$ ve $0 \leq r_1 \leq m$ olacak şekilde $r = r_1 \cdot \alpha + r_0$ eşitliği yazılır.

Yüksek-bit fonksiyonunu $\mathcal{YB}$ ve düşük-bit fonksiyonunu $\mathcal{DB}$ ile gösterirsek fonksiyonların girdi ve çıktılarını aşağıdaki şekilde verebiliriz:

$$\begin{aligned} \mathcal{YB}(r, \alpha) &= r_1 \\ \mathcal{DB}(r, \alpha) &= r_0 \end{aligned}$$

Bu fonksiyonların daha iyi anlaşılması için bir örnek verelim:

Örnek 4.2.8. $q = 17$ ve $\alpha = 4$ olsun. r değişkenine bağlı olarak çıktıları yazalım:

$r = n$	için :	$n = r_1' \cdot \alpha + r_0'$	$\Rightarrow$	(r_1, r_0)	$= (r_1', r_0')$
$r = 0$	için :	$0 = 0 \cdot 4 + 0$	$\Rightarrow$	(r_1, r_2)	$= (0, 0)$
$r = 1$	için :	$1 = 0 \cdot 4 + 1$	$\Rightarrow$	(r_1, r_2)	$= (0, 1)$
$r = 2$	için :	$2 = 0 \cdot 4 + 2$	$\Rightarrow$	(r_1, r_2)	$= (0, 2)$
$r = 3$	için :	$3 = 1 \cdot 4 - 1$	$\Rightarrow$	(r_1, r_2)	$= (1, -1)$
$r = 4$	için :	$4 = 1 \cdot 4 + 0$	$\Rightarrow$	(r_1, r_2)	$= (1, 0)$
$\vdots$					
$r = 16$	için :	$16 = 4 \cdot 4 + 0$	$\Rightarrow$	(r_1, r_2)	$= (4, 0)$

Not : Yüksek-bit ve düşük-bit fonksiyonlarına girdi olarak bir polinom verildiğinde, fonksiyon polinomun katsayılarına ayrı ayrı uygulanır ve görüntüler aynı yerlerine katsayı olarak yazılır.

Şimdi algoritmanın adımlarını temel iletişim senaryomuz üzerinden en sade haliyle görelim:

Anahtar Üretim Aşaması

1. Adım : Arif ilk olarak parametreleri belirler: q, n, k, l, η, λ.

2. Adım : Arif bir $A \in \mathcal{R}_q^{k \times l}$ matrisi seçer. (Matrisin sütun vektörleri bir kafesin taban vektörleridir.)

3. Adım : Ardından bir $s_1 \in \mathcal{S}_\eta^l$ ve bir de $s_2 \in \mathcal{S}_\eta^k$ vektörü seçer.

4. Adım : $As_1 + s_2 = t$ hesabını yapar.

5. Adım : A ve t'yi açık anahtarlar olarak gönderirken s_1 ve s_2'yi gizli anahtarlar olarak tutar.

Burada, A ve t'yi kullanarak s_1 ve s_2'yi bulmak, gürültülüyken çözme problemini çözmek demektir ki bunun neredeyse imkansız olduğunu gösterdik.

İmzalama Aşaması

1. Adım : Arif ilk olarak imzalayacağı $M \in \{0,1\}^n$ mesajını alır.

2. Adım : Arif bir $y \in \tilde{\mathcal{S}}_{\lambda_1}^l$ seçer.

3. Adım : Ardından $Ay = w$ hesabını yapar.

4. Adım : Bulduğu w polinomunu ve $\alpha = 2 \cdot \lambda_2$ sayısını yüksek-bit (4.2.7) fonksiyonunun girdileri olarak kullanarak $\mathcal{YB}(w, \alpha) = w_1$ hesabını yapar.

5. Adım : **H** özet fonksiyonunu kullanarak $\mathbf{H}(M, w_1) = c$ hesabını yapar.

6. Adım : Ardından $y + cs_1 = z$ polinomunu hesaplar.

7. Adım : Son olarak c ve z sonuçlarını art arda ekleyerek (c, z) imza çiftini oluşturup gönderir.

İmza Doğrulama Aşaması

1. Adım : Arif'ten (c, z) çiftini alan Buse ilk olarak imza çiftini c ve z olarak ayırır.

2. Adım : Daha sonra Arif'in açık anahtarları olan A, t'yi ve imza çiftinden elde ettiği c ve z'yi kullanarak $Az - ct = w_2$ hesabını yapar.

3. Adım : Yüksek-bit fonksiyonunu kullanarak $\mathcal{YB}(w_2, \alpha) = w_3$ görüntüsünü elde eder.

4. Adım : **H** özet fonksiyonunu kullanarak $\mathbf{H}(M, w_3) = c'$ hesabını yapar. Eğer $c = c'$ ise imza doğrudur.

"Son adımdaki eşitliğin sağlanması imzayı nasıl doğrulayabildi?" açıklayalım:

İmzanın doğrulanabilmesi için $\mathbf{H}(M, w_1) = \mathbf{H}(M, w_3)$ eşitliğinin sağlanması gerekir. Bunun için de özet fonksiyonu altındaki görüntüsü w_1 ile aynı olan bir w_3 elde etmemiz gereklidir. w_1 polinomunu $\mathcal{YB}(w, \alpha) = w_1$ hesabıyla elde ettiğimize göre, yüksek bit fonksiyonu altındaki görüntüsü w_1 olan bir ifade bulmamız yeterli olacaktır. Bu ifade de tam olarak $Az - ct$'dir. Aşağıdaki işlemleri takip edersek bunun nasıl mümkün olduğunu kolaylıkla anlayabiliriz:

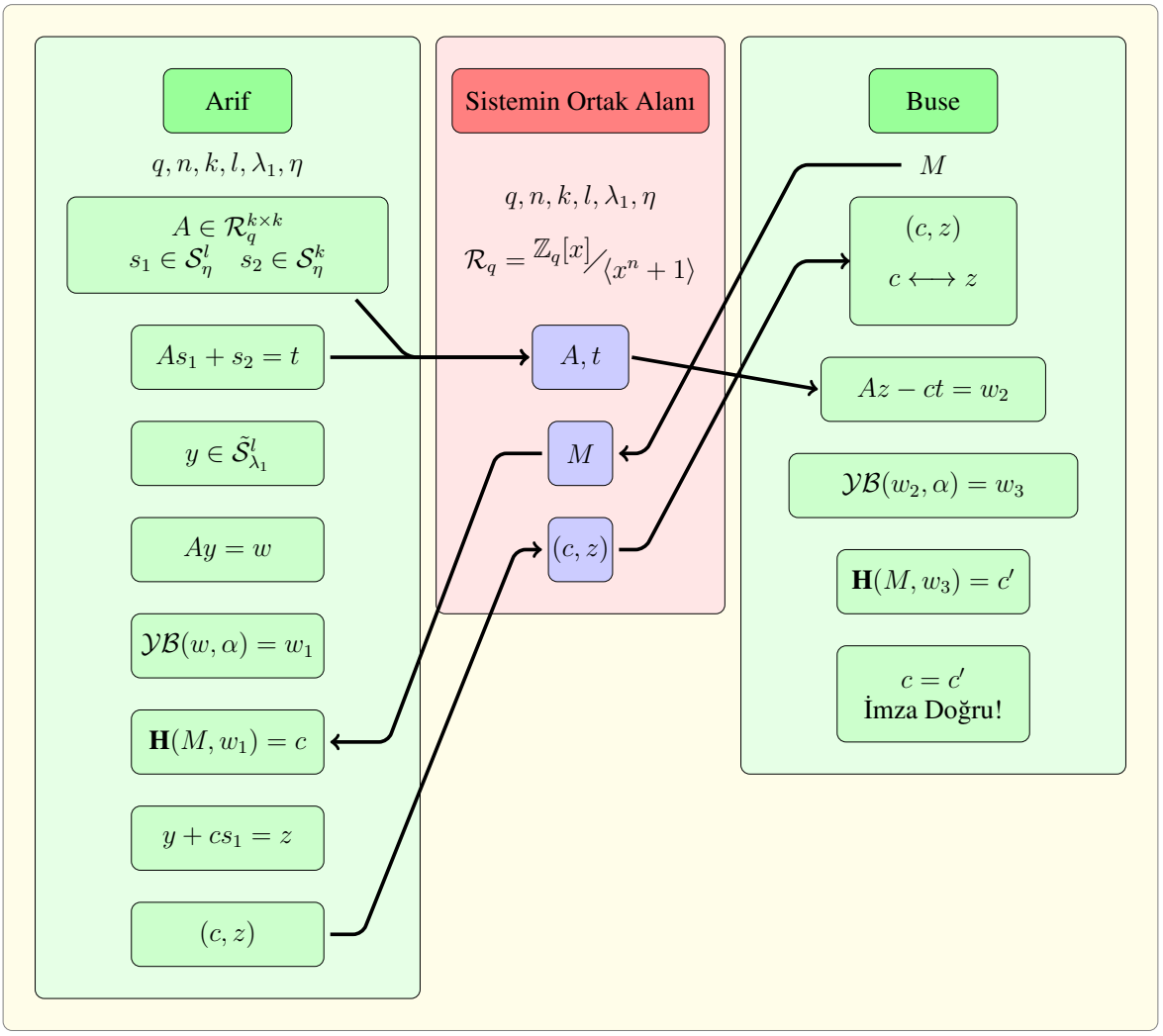
$$\begin{aligned} Az &= A(y + cs_1) \\ &= Ay + Acs_1 \\ &= Ay + cAs_1 \\ &= w + c(t - s_2) \\ &= w + ct - cs_2 \end{aligned}$$

Buradan da ct terimini eşitliğin sol tarafına atarak $Az - ct = w - cs_2$ eşitliğini elde ederiz. Bu eşitlikteki $w - cs_2$ polinomunun yüksek-bit fonksiyonu altındaki görüntüsü alınırken cs_2 ifadesinin görüntüsü sıfırlanır ve

$$\mathcal{YB}(w - cs_2, \alpha) = \mathcal{YB}(w, \alpha) = w_1$$

eşitliği elde edilir. Bu da $\mathcal{YB}(Az - ct) = w_1$ olduğu anlamına gelir.

Dilithium Algoritmasını aşağıdaki şekil ile özetleyebiliriz:



Şekil 4.16. Dilithium algoritması

Algoritmanın daha iyi anlaşılabilmesi için küçük parametreler kullandığımız bir örnek verelim:

Örnek 4.2.9. Örneğimizi temel iletişim senaryomuz üzerinden verelim:

Anahtar Üretim Aşaması

1. Adım : Arif'in belirlediği parametreler $q = 16417$, $n = 4$, $(k, l) = (3, 2)$, $\eta = 10$, $\lambda = 1024$, $\alpha = 1026$ olsun.

2: Adım : Arif'in seçtiği $A \in \mathcal{R}_q^{k \times l}$ matrisi aşağıdaki matris olsun:

$$A = \begin{bmatrix} 15196 + 7926x + 8057x^2 + 13612x^3 & 14303 + 5x + 12257x^2 + 2347x^3 \\ 9765 + 10436x + 10983x^2 + 5860x^3 & 5393 + 311x + 5144x^2 + 10841x^3 \\ 11868 + 7995x + 10716x^2 + 3121x^3 & 9390 + 2055x + 6505x^2 + 2440x^3 \end{bmatrix}$$

3. Adım : Arif'in seçtiği $s_1 \in \mathcal{S}_\eta^l$ ve $s_2 \in \mathcal{S}_\eta^k$ vektörleri aşağıdaki vektörler olsun:

$$s_1 = \begin{bmatrix} 2 + 5x - 10x^2 - 5x^3 \\ 2 + 3x - 4x^2 - 4x^3 \end{bmatrix} \qquad s_2 = \begin{bmatrix} -8 + 3x + 4x^2 + 4x^3 \\ 8 + 10x + x^2 + 8x^3 \\ -3 - 8x + 6x^2 + 6x^3 \end{bmatrix}$$

4. Adım : Arif seçtiği matris ve vektörler ile $A \cdot s_1 + s_2 = t$ hesabını yaparak aşağıdaki sonuca ulaşır:

$$t = \begin{bmatrix} 5384 + 8401x + 14221x^2 + 11425x^3 \\ 4587 + 1291x + 5976x^2 + 9841x^3 \\ 3959 + 14751x + 15381x^2 + 14072x^3 \end{bmatrix}$$

5. Adım : Son olarak Arif, A ve t'yi herkese açık bir şekilde Buse'ye gönderirken s_1 ve s_2'yi gizli anahtarları olarak tutar.

İmzalama Aşaması

1. Adım : Arif Buse'den imzalaması gereken M mesajını alır. (Bu örnekte M mesajını işleme sokmayacağımız için, herhangi bir mesaj seçebiliriz.)

2. Adım : Arif'in seçtiği $y \in \tilde{\mathcal{S}}_{\lambda_1}^l$ vektörü aşağıdaki vektör olsun:

$$y = \begin{bmatrix} 707 - 155x + 357x^2 - 822x^3 \\ 474 - 566x - 869x^2 - 542x^3 \end{bmatrix}$$

3. Adım : Arif, $A \cdot y = w$ hesabını yaparak aşağıdaki sonuca ulaşır:

$$w = \begin{bmatrix} 7023 + 3184x + 4074x^2 + 5566x^3 \\ 9495 + 7254x + 7431x^2 + 13483x^3 \\ 4189 + 7420x + 13635x^2 + 4161x^3 \end{bmatrix}$$

4. Adım : Bulduğu w vektörünü ve $\alpha = 1026$ sayısını yüksek-bit fonksiyonuna girdi olarak vererek aşağıdaki sonuçlara ulaşır ve bu sonuçları kullanarak w_1 vektörünü oluşturur:

$$\begin{array}{rcrcrcl}
7023 & = & 7 \cdot 1026 - 159 & \Rightarrow & \mathcal{YB}(7023, \alpha) & = & 7 \\
3184 & = & 3 \cdot 1026 + 106 & \Rightarrow & \mathcal{YB}(3184, \alpha) & = & 3 \\
4074 & = & 4 \cdot 1026 - 30 & \Rightarrow & \mathcal{YB}(4074, \alpha) & = & 4 \\
5566 & = & 5 \cdot 1026 + 436 & \Rightarrow & \mathcal{YB}(5566, \alpha) & = & 5 \\
9495 & = & 9 \cdot 1026 + 261 & \Rightarrow & \mathcal{YB}(9495, \alpha) & = & 9 \\
7254 & = & 7 \cdot 1026 + 72 & \Rightarrow & \mathcal{YB}(7254, \alpha) & = & 7 \\
7431 & = & 7 \cdot 1026 + 249 & \Rightarrow & \mathcal{YB}(7431, \alpha) & = & 7 \\
13483 & = & 13 \cdot 1026 + 145 & \Rightarrow & \mathcal{YB}(13483, \alpha) & = & 13 \\
4189 & = & 4 \cdot 1026 + 85 & \Rightarrow & \mathcal{YB}(4189, \alpha) & = & 4 \\
7420 & = & 7 \cdot 1026 + 238 & \Rightarrow & \mathcal{YB}(7420, \alpha) & = & 7 \\
13635 & = & 13 \cdot 1026 + 297 & \Rightarrow & \mathcal{YB}(13635, \alpha) & = & 13 \\
4161 & = & 4 \cdot 1026 + 57 & \Rightarrow & \mathcal{YB}(4161, \alpha) & = & 4
\end{array}$$

$$w_1 = \begin{bmatrix} 7 + 3x + 4x^2 + 5x^3 \\ 9 + 7x + 7x^2 + 13x^3 \\ 4 + 7x + 13x^2 + 4x^3 \end{bmatrix}$$

5. Adım : Arif, bu adımda bir özet fonksiyonu kullanarak $\mathbf{H}(M, w_1) = c$ özet değerini hesaplar. Algoritmanın kullandığı özet fonksiyonunun SHAKE256 olduğunu söylemiştik fakat bu örnek için SHAKE256 algoritmasını kullanmak gereksiz yere adımı uzatır. Bu sebeple $\mathbf{H}(M, w_1) = c$ özet değerini $c = -1 - x + x^2 - x^3$ alarak devam edelim.

6. Adım : Arif daha sonra $y + c \cdot s_1 = z$ değerini hesaplayarak aşağıdaki sonuca ulaşır:

$$z = \begin{bmatrix} 715 - 167x + 359x^2 - 804x^3 \\ 475 - 571x - 870x^2 - 533x^3 \end{bmatrix}$$

7. Adım : Arif son olarak, bulduğu c ve z vektörlerini art arda uç uca ekleyerek (c, z) imza çiftini oluşturur ve herkese açık bir şekilde Buse'ye gönderir.

İmza Doğrulama Aşaması

1. Adım : Arif'ten imza çiftini alan Buse ilk olarak imza çiftini tekrar c ve z olacak şekilde iki parçaya ayırır.

2. Adım : Daha sonra Arif'in açık anahtarları olan A ve t ile imza çiftinden elde ettiği c ve z'yi kullanarak $Az - ct = w_2$ hesabını yapar ve aşağıdaki sonuca ulaşır:

$$w_2 = \begin{bmatrix} 7012 + 3179x + 4085x^2 + 5563x^3 \\ 9486 + 7273x + 7426x^2 + 13490x^3 \\ 4194 + 7409x + 13630x^2 + 4178x^3 \end{bmatrix}$$

3. Adım : Buse, bulduğu w_2 vektörünün girdilerini yüksek-bit fonksiyonuna girdi olarak vererek aşağıdaki sonuçlara ulaşır ve bu sonuçları kullanarak w_3 vektörünü oluşturur:

$$\begin{array}{lcllcl}
7012 & = & 7 \cdot 1026 - 170 & \Rightarrow & \mathcal{YB}(7012, \alpha) & = & 7 \\
3179 & = & 3 \cdot 1026 + 101 & \Rightarrow & \mathcal{YB}(3179, \alpha) & = & 3 \\
4085 & = & 4 \cdot 1026 - 19 & \Rightarrow & \mathcal{YB}(4085, \alpha) & = & 4 \\
5563 & = & 5 \cdot 1026 + 433 & \Rightarrow & \mathcal{YB}(5563, \alpha) & = & 5 \\
9486 & = & 9 \cdot 1026 + 252 & \Rightarrow & \mathcal{YB}(9486, \alpha) & = & 9 \\
7273 & = & 7 \cdot 1026 + 91 & \Rightarrow & \mathcal{YB}(7273, \alpha) & = & 7 \\
7426 & = & 7 \cdot 1026 + 244 & \Rightarrow & \mathcal{YB}(7426, \alpha) & = & 7 \\
13490 & = & 13 \cdot 1026 + 152 & \Rightarrow & \mathcal{YB}(13490, \alpha) & = & 13 \\
4194 & = & 4 \cdot 1026 + 90 & \Rightarrow & \mathcal{YB}(4194, \alpha) & = & 4
\end{array}$$

$$
\begin{array}{lcllcl}
7409 & = & 7 \cdot 1026 + 227 & \Rightarrow & \mathcal{YB}(7409, \alpha) & = & 7 \\
13630 & = & 13 \cdot 1026 + 292 & \Rightarrow & \mathcal{YB}(13630, \alpha) & = & 13 \\
4178 & = & 4 \cdot 1026 + 74 & \Rightarrow & \mathcal{YB}(4178, \alpha) & = & 4
\end{array}
$$

$$
w_3 = \begin{bmatrix} 7 + 3x + 4x^2 + 5x^3 \\ 9 + 7x + 7x^2 + 13x^3 \\ 4 + 7x + 13x^2 + 4x^3 \end{bmatrix}
$$

4. Adım : Buse bu adımda M mesajını ve bulduğu w_3 vektörünü **H** özet fonksiyonuna girdi olarak verdiğinde $\mathbf{H}(M, w_3) = c'$ değerini elde eder. Ayrıca kolaylıkla görüleceği üzere, Arif'in fonksiyona verdiği girdi olan w_1 vektörü ile Buse'nin verdiği w_3 vektörü aynıdır. Dolayısıyla her ikisinin de fonksiyondan elde edecekleri çıktılar da aynı olacaktır. Yani;

$$c = \mathbf{H}(M, w_1) = \mathbf{H}(M, w_3) = c'$$

elde edilir ve imza doğrulanmış olur.

5. SONUÇLAR

Klasik McEliece algoritması halihazırda bir cebirsel-geometrik kod olan Goppa kodlarını kullanmaktadır. Diğer kod ailelerine kıyasla Goppa kodlarda daha az veri kullanarak daha büyük boyutlu kodlar üretilebiliyor. Bu sebeple, McEliece ile yapılan bir şifrelemede kullanılan Goppa kodunu bulmak için daha az sayıda deneme yapmak yeterli olur. Diğer yandan, cebirsel-geometrik kodların bir de dezavantajı vardır. Kodların yapısı gereği işlem yükü çok ağırdır ve uzun zaman alır. Bu yüzden, daha az sayıda deneme yapmak, daha kısa zaman harcamak anlamına gelmez! Klasik McEliece'in diğer kod tabanlı algoritmalara kıyasla daha güvenilir olmasının sebebi de budur.

McNie, ROLLO ve BIKE algoritmalarının en büyük avantajı yarı devirli kodlar kullandıkları için üreteç matrislerinin tek bir satır kullanılarak üretilebiliyor olmasıdır. Bu durum algoritmaların işleyiş hızını çok artırsa da güvenlikten ödün verdikleri anlamına da gelir. Yine de, bu algoritmasının bu kadar yaygın bir isim yapmasının sebebi, sahip olduğu yüksek hızları kullanarak daha büyük parametreli kodlarda işlem yapabiliyor olmalarıdır. Kullandıkları kodlar cebirsel-geometrik kodlar kadar üst seviye güvenlik sağlamasalar da yüksek parametreli olmaları, kırılmalarını biraz olsun zorlaştırmaktadır. Bu durum, yarı devirli kodlar kullanan neredeyse tüm algoritmalar için geçerlidir.

HQC algoritmasında kullanılan kod ailesinin kod çözme kapasitesi barındırdığı vektörlerin Hamming ağırlıklarına bağlıdır. Fakat algoritmada kullanılan kodun uzunluğuna kıyasla gizli anahtarlarının ağırlıkları çok düşüktür. Bu sebeple HQC algoritmasının güvenlik seviyesi diğer algoritmalara kıyasla biraz daha aşağıda kalmaktadır. Fakat çalışma hızı da aynı oranda yüksektir.

KYBER ve Dilithium algoritmaları güvenilirliklerini gürültülüyken çözme problemine borçludur. Seçilen gürültü vektörleri, doğrudan algoritmadaki ana polinom halkasından seçilmek yerine bu polinom halkasının katsayıları büyük oranda daraltılmış halinden seçilmektedir. Bu durumu, güvenlik açısından bir kayıp olarak görebilme ihtimalimiz var fakat KYBER ve Dilithium'da kullanılan halkaların genişleme katsayıları çok büyük

olduğundan, gürültü vektörü uzayları da oldukça geniş olur. Bu sebeple, kafes tabanlı algoritmalar daha güvenilirdirler. Ayrıca, çalışma hızları da kod tabanlı olanlara göre oldukça yüksektir. NIST'in yaptığı yarışma sonuçları da bunu doğrulamaktadır.

KAYNAKÇA

[1] David Kahn. *The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet*. Scribner, **1996**. ISBN 9780684831305.

[2] Simon Singh. *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography*. Anchor Books, **1999**. ISBN 9780385495325.

[3] Roger S. Bagnall. *Reading Papyri, Writing Ancient History*. Routledge, **1995**. ISBN 9780415237043.

[4] Mark Collier and Bill Manley. *How to Read Egyptian Hieroglyphs*. University of California Press, **2003**. ISBN 9780520239494.

[5] Peter Jensen. *Writing and Its Relation to Speech*. J.C. Hinrichs, **1907**.

[6] Julius Caesar. *Commentarii de Bello Gallico*. Loeb Classical Library. Harvard University Press, **1917**.

[7] Al-Kindi. A manuscript on deciphering cryptographic messages (risala fi istikhraj al-mu'amma), **ca. 850**. Original Arabic manuscript, 9th century. English translation in: The Origins of Cryptology: The Arab Contributions by Ahmad Y. al-Hassan.

[8] James S. Kraft and Lawrence C. Washington. *An Introduction to Number Theory with Cryptography*. CRC Press, **2018**. ISBN 9781138063471.

[9] Tom M. Apostol. Explicit formulas for euler's totient function and the number of divisors. *The American Mathematical Monthly*, 80(8):749–751, **1973**.

[10] Davender S. Malik, John M. Mordeson, and Mridul K. Sen. *Fundamentals of Abstract Algebra*. International Series in Pure and Applied Mathematics. McGraw Hill College, **1996**. ISBN 978-0070400351.

[11] Fethi Çallıalp. *Cebir*. Sakarya Üniversitesi Matbaası, Adapazarı, **1994**.

[12] Max Neunhoeffer. Finite fields - course notes 2013, **2013**. Accessed: 2025-01-22.

[13] Bülent Saraç. Kodlama teorisi ders notu, **2024**. Erişim: 24 Ocak 2025.

[14] Richard Bronson. *Matrix Methods: An Introduction*. Academic Press, **1991**. ISBN 978-0121352516.

[15] Theodore Shifrin and Malcolm R. Adams. *Linear Algebra: A Geometric Approach*. W. H. Freeman and Company, **2011**. ISBN 978-1-4292-1521-3.

[16] Oded Regev. Lattice-based cryptography. In *Annual International Cryptology Conference*, pages 131–141. Springer, **2006**.

[17] Yağmur Çakıroğlu. *Ağırlıklı Projektif Uzaylardaki Parametrik Kodlar*. Master's thesis, Hacettepe Üniversitesi Fen Bilimleri Enstitüsü, **2019**.

[18] Wade Trappe and Lawrence Washington. *Introduction to Cryptography with Coding Theory*. Pearson, **2005**. ISBN 978-0131862395.

[19] Nicolas Aragon, Paulo L. Barreto, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Santosh Ghosh, Shay Gueron, Tim Güneysu, Carlos Aguilar Melchor, Rafael Misoczki, Edoardo Persichetti, Jan Richter-Brockmann, Nicolas Sendrier, Jean-Pierre Tillich, Valentin Vasseur, and Gilles Zémor. Bike: Bit flipping key encapsulation, **2017**. `https://bikesuite.org/files/v5.0/BIKE_Spec.2022.10.10.1.pdf`.

[20] Jon-Lark Kim, Young-Sik Kim, Lucky Galvez, Myeong Jae Kim, and Nari Lee. Mcnie: A code-based public-key cryptosystem. *CoRR*, abs/1812.05008, **2018**. `https://arxiv.org/pdf/1812.05008`.

[21] Harshdeep Singh. Code based cryptography: Classic mceliece. *CoRR*, abs/1907.12754, **2019**. `https://arxiv.org/pdf/1907.12754`.

[22] Emmanuel Abbe, Ori Sberlo, and Amir Shpilka. *Reed-Muller Codes*. Now Publishers, **2023**. ISBN 978-1638281443.

[23] Stephen B. Wicker and Vijay K.Bhargava. *Reed-Solomon Codes and Their Application*. I.E.E.E. Press, **1994**.

[24] Classic McEliece Team. Classic McEliece: Conservative Code-Based Cryptography—Cryptosystem Specification. Technical Report spec-20221023, Classic McEliece, classic.mceliece.org, **2022**. Defining the Classic McEliece KEM functions KeyGen, Encap, Decap.

[25] Sevde Kara. *Kod Tabanlı Kuantum Sonrası Bazı Şifreleme Algoritmaları Ve Anahtar Kapsülleme Mekanizmalarının İncelenmesi*. Master's thesis, TOBB Ekonomi ve Teknoloji Üniversitesi Fen Bilimleri Enstitüsü, **2019**.

[26] Carlos Aguilar Melchor, Nicolas Aragon, Magali Bardet, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Adrien Hauteville, Ayoub Otmani, Olivier Ruatta, Jean-Pierre Tillich, and Gilles Zémor. Rollo - rank-ouroboros, lake & locker, **2020**. `https://pqc-rollo.org/doc/rollo-specification_2020-04-21.pdf`.

[27] Carlos Aguilar-Melchor, Nicolas Aragon, Emanuele Bellini, Florian Caullery, Rusydi H. Makarim, and Chiara Marcolla. Constant time algorithms for ROLLO-i-128. Cryptology ePrint Archive, Paper 2020/1066, **2020**.

[28] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jurjen Bos, Jean-Christophe Deneuville, Arnaud Dion, Philippe Gaborit, Jérôme Lacan, Edoardo Persichetti, Jean-Marc Robert, Pascal Véron, and Gilles Zémor. Hamming quasi-cyclic (hqc) - fourth round version. Technical report, NIST PQC Standardization Process, **2024**. Updated version 23/02/2024.

[29] Dong Pyo Chi, Jeong Woon Choi, Jeong San Kim, and Taewan Kim. Lattice based cryptography for beginners. Cryptology ePrint Archive, Paper 2015/938, **2015**.

[30] Miklós Ajtai. Generating hard instances of lattice problems. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC)*, pages 99–108. ACM, Philadelphia, PA, USA, **1996**. doi:10.1145/237814.237838.

[31] Steven D. Galbraith. Mathematics of public key cryptography. In *Chapter 17: Bilinear Pairings*. Cambridge University Press, **2012**.

[32] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Kyber algoritma spesifikasyonları ve destekleyici dokümantasyon (sürüm 3.01), **2021**.

[33] Alfred Menezes. Kyber and dilithium, **2024**. Erişim: Nisan 2025.

[34] Léo Ducas, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: Digital signatures from module lattices. `https://eprint.iacr.org/2017/633`, **2017**. Cryptology ePrint Archive, Report 2017/633.

[35] Alfred Menezes. Kyber and dilithium. `https://cryptography101.ca/wp-content/uploads/2025/01/V3-slides-Kyber-and-Dilithium.pdf`, **2025**. Lecture slides from Cryptography 101.