T.C.
TRAKYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Gizli Anahtarlı Kriptosistemlerin Tasarımında Cebirsel Yapıların Önemi ve Kriptanaliz

Osman KARAAHMETOĞLU

DOKTORA TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ ANA BİLİM DALI
YRD.DOÇ.DR. ERCAN BULUŞ

2010
EDİRNE

T.C.

TRAKYA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

Gizli Anahtarlı Kriptosistemlerin Tasarımında

Cebirsel Yapıların Önemi ve Kriptanaliz

Osman KARAAHMETOĞLU

DOKTORA TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ ANA BİLİM DALI

Bu tez 04.06.2010 tarihinde Aşağıdaki Jüri Tarafından Kabul Edilmiştir.

Danışman: Yrd.Doç.Dr. Ercan BULUŞ

Yrd.Doç.Dr. Rembiye KANDEMİR

Yrd.Doç.Dr.Akif SABANER

Yrd.Doç.Dr. Muharrem Tolga SAKALLI

Yrd.Doç.Dr. Erdinç UZUN

Doktora Tezi

Trakya Üniversitesi Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Bölümü

ÖZET

Bu tezde, günümüz modern şifreleme algoritmalarına temel oluşturan ve kriptografideki en önemli iki yapıdan biri olan blok şifreleyicileri incelenmektedir. Blok şifreleyicileri verileri döngüler boyunca blok blok şifreler. Bir blok şifreleyicisi birbirini takip eden bir dizi doğrusal ve doğrusal olmayan dönüşümlerden ibarettir. Blok şifreleyicisindeki tek doğrusal olmayan yapı S-kutusudur ve dolayısıyla şifreye gücünü veren yapıdır. Bu nedenledir ki, blok şifreliyicilerine yapılan saldırılar S-kutusunu hedef alır. Bu tezde, modern blok şifreleyicilerin cebirsel yapıları ve bu şifreleyiciler üzerinde yapılan saldırılar incelenmiş, bu şifreleyicilere saldırılar gerçekleştirerek anahtar elde edilmeye çalışılmış ve şifrenin en önemli yapısı olan S-kutusunun cebirsel yapısındaki zaaflar belirlenerek iyileştirilmeye çalışılmıştır.

Tezin giriş bölümünde, blok şifreleyicilerin basit bir tanıtımı yapılmış olup, ilkel şifreleyiciler incelenerek şifreleme sistemlerinin evrimleşme süreci göz önüne serilmiştir. Blok şifreleme algoritmalarına yapılan saldırılar sınıflandırılmış olup, güvenlik modellerinden bahsedilmiştir.

Tezin ikinci bölümünde, teze temel oluşturacak olan matematiksel altyapı verilmiştir. Yapılan analiz ve çalışmalarda kullanılan matematiksel teori ve tanımlar anlatılmıştır.

Tezin üçüncü bölümünde blok şifreleyicileri örneklerle ayrıntılı olarak incelenmiştir. Öncelikle blok şifreleyicilerin mimarisinde temel oluşturan işlem biçimleri anlatılmıştır. Blok şifreleyici mimarilerinden Feistel ve SPN (Substitution Permutation Network) mimarilerinden ayrıntılı olarak bahsedilmiştir. Blok şifreleyicilerinden DES (Data Encryption Standard) ve AES (Advanced Encryption Standard) şifreleme algoritmaları anlatılmıştır.

Tezin dördüncü bölümünde blok şifreleyiciler üzerinde gerçekleştirilen en önemli saldırı olan diferansiyel kriptanaliz ve türevleri olan, kesilmiş (truncated) ve imkansız diferensiyel saldırıdan bahsedilmiştir. Bilinen en önemli blok şifreleyici olan AES S-kutusunun cebirsel yapıları incelenmiştir.

Tezin son bölümü olan beşinci bölümde, tez boyunca yapılan çalışmalar ayrıntılı olarak anlatılmıştır. Feistel ve SPN temelli blok şifreleyicilere yapılan diferansiyel, kesilmiş diferansiyel ve imkansız diferansiyel saldırıların uygulamaları ayrıntılı olarak verilmiştir. Son olarak da, AES şifreleyicisine gücünü veren S-kutusunun cebirsel yapısı ayrıntılı olarak incelenmiş olup, geliştirilen yöntemlerle cebirsel ifadesi elde edilmiş ve bu yöntemlerin başarım karşılaştırmaları verilmiştir. Bu incelemeler sonucunda belirlenen zaaflar ile ilgili iyileştirme önerilerinde bulunulmuştur.

Anahtar Sözcükler: Blok Şifreleyiciler, Cebirsel Yapılar, Blok Şifreleyici Mimarileri, Diferansiyel Kriptanaliz, AES, S-Kutusu

Yıl: 2010

Sayfa: 228

Doctorate Thesis

Trakya University Graduate School of

Natural and Applied Sciences

Department of Computer Engineering

ABSTRACT

In this thesis, the block cipher, which is the foundation of the recent modern ciphers and one of two important components in symetric cryptograghy, is studied. The block ciphers encrypt the plaintext through the rounds. A block cipher is composed of series of linear and nonlinear transformations. The s-box is the only nonlinear component on the block cipher and gives its power to the cipher. Therefore, it is the target of the attacks performed on block ciphers. In this thesis, The algebraic structures of the modern block ciphers and the atacks performed on this ciphers are studied, bits from the key were tried to be extracted by performing attacks on this ciphers and the flows in design of the s-box which is the most important component of any block cipher were tried to be found and improved.

In the introduction section of the thesis, the block ciphers are defined simply and the evoluation of the cyptosystems are introduced by researching the primitive ciphers. In addition to this, the attacks performed on the block cipher are classified.

In the second section, the mathematical background such as theories, definitions, equations etc. is given.

In the third section, the block ciphers are studied in detail by giving examples. Firstly, the operation modes which are the main components of the block ciphers are explained. After that, the two important structures of the block cipher construction models, Feistel and SPN(Substitution Permutation Network), are examined in detail. The block ciphers of DES (Data Encryption Standard) and AES (Advanced Encryption Standard) are studied in detail.

In the fourth section, the diferential cryptanalysis, which is the most important attack performed on the block ciphers, truncated and impossible differentials, which are the derivatives of this, are examined in detail. Besides to that, the algebraic structures of the AES s-box are studied in detail.

Finally, in the fifth section, the study done through thesis were given in detail with experimental results. The practice and the applications of differential crptanalysis, truncated differentials and the impossible differentials analyzing the block ciphers based on Feistel an SPN structures were given in detail. Besides to that, the algebraic structure of the s-box which AES cipher gets the power is examined in detail. The algebraic expression of the s-box is determined with the new methods developed and the performance comparision of this methods is given. The suggestions of impovements are given for the weaknesses found as a consequnce of these analysis.

Keywords: Block Ciphers, Algebraic Structures, Block Cipher Structure, Differential Cryptanalysis, AES, S-Box

Year: 2010

Page 228

TEŞEKKÜR

Bu tezin gerçekleştirilmesi sürecinde gerek araştırmalarımda, gerekse geliştirme süreçlerinde benden desteklerini esirgemeyen tüm herkese teşekkürü bir borç bilirim.

Öncelikle tez çalışmamda karşılaştığım tüm sorunlarda sunduğu fikirler ve verdiği desteklerle önümü açan ve önümüzdeki engellerin birbir aşılmasında destek ve katkılarını esirgemeyen değerli hocam ve danışmanım Sayın Yrd. Doç. Dr. Ercan BULUŞ'a sonsuz teşekkürlerimi sunmak isterim.

Tez sürecinde yapmış olduğum araştırmalarda benden yardımını esirgemeyen, karşılaştığımız teknik sorunlarda üstün zekasıyla sunmuş olduğu çözümlerle darboğazları aşmamızı sağlayan, sevgili arkadaşım ve saygıdeğer hocam Yrd. Doç. Dr. Muharrem Tolga SAKALLI'ya teşekkürü bir borç bilirim.

Doktoraya başlamamda büyük bir pay sahibi olan, tez sürecinde maddi ve manevi tüm desteğini benden esirgemeyen, bu süreçteki tüm kaprislerimi çeken ve beni çalışmaya motive eden hayat arkadaşım, sevgili eşim Ebru EROĞLU KARAAHMETOĞLU'ya teşekkürü bir borç bilirim.

Çalışmakta olduğum şirketten doktora için gerekli izni bana veren saygıdeğer müdürüm Murat PEKMEZYAN'a teşekkürlerimi sunarım.

Sevgili arkadaşlarım ve sayın hocalarım Yrd. Doç. Dr. Tarık YERLİKAYA, Yrd. Doç. Dr. Andaç ŞAHİN MESUT ve Yrd. Doç. Dr. Altan MESUT'a teşekkürlerimi sunarım.

Araştırmalarım sırasında bana katkılarda bulunan Arş. Gör. Fatma BÜYÜKSARAÇOĞLU, Arş. Gör. Derya ARDA, Arş. Gör. H. Nusret BULUŞ ve tüm Trakya Üniversitesi Bilgisayar Mühendisliği Bölümü öğretim üyeleri ve çalışanlarına teşekkürü bir borç bilirim.

Bu tezin izleme komitesinde yer alan ve yine bana verdikleri destek ve değerli katkılarından dolayı Yrd. Doç Dr. Akif SABANER ve Yrd. Doç. Dr. Rembiye KANDEMİR'e sonsuz teşekkürlerimi sunarım.

Bu tezin kabul jürisinde yer alan sevgili arkadaşım ve sayın hocam Yrd. Doç. Dr. Erdinç UZUN'a sonsuz teşekkürlerimi bir borç bilirim.

Son olarak bana bu tez sürecinde ve öncesinde bugünlere gelmemdeki büyük katkılarından dolayı önce babam LÜTFİ KARAAHMETOĞLU, annem Necmiye KARAAHMETOĞLU olmak üzere KARAAHMETOĞLU ve EROĞLU ailesinin tüm fertlerine teşekkürü bir borç bilirim.

Temmuz 2010 **Osman KARAAHMETOĞLU**

İÇİNDEKİLER

TABLOLAR LİSTESİ

ŞEKİLLER LİSTESİ

BÖLÜM 1

1 GİRİŞ

Kriptografi, kelime kökeni olarak Yunanca gizli/saklı anlamına gelen kryptós ve yazmak anlamına gelen gráphein kelimesinden türetilmiştir. Kriptografi, gizlilik, kimlik denetimi, bütünlük gibi bilgi güvenliği kavramlarını sağlamak için çalışan matematiksel yöntemler bütünü olarak tanımlanabilir. Bir başka deyişle kriptografi, okunabilir durumdaki bir bilginin istenmeyen taraflarca okunamayacak bir hale dönüştürülmesinde kullanılan tekniklerin tümüdür.

Simetrik anahtarlı kriptosistemlerin bir türü olan blok şfreleyicileri gücünü, tek doğrusal olmayan yapısı olan S-kutusundan almaktadır. Bu nedenle S-kutuları blok şifreleyicilere gerçekleştirilen saldırıların hedef noktasıdır. DES (Data Encryption Standard) şifreleyicisinin anahtarlarının diferansiyel kriptanaliz ile elde edilmesi, kriptoloji uzmanlarını doğrusal ve diferansiyel kriptanalize karşı dayanaklı bir şifre arayışına sokmuştur. NIST'in gerçekleştirmiş olduğu yarışmayı kazanan Rijndael şifreleme algoritması AES (Advanced Encryption Standard) olarak isimlendirilmiş olup, dünya genelinde iletişim güvenliğini sağlamak amaçlı olarak kullanılmaktadır. AES S-kutularının doğrusal ve diferansiyel kriptanalize karşı dayanaklı olması, kriptanalizcileri farklı saldırı türleri geliştirmeye yöneltmiştir. Yapılan araştırmalar göstermektedir ki; cebirsel saldırılar AES şifreleyicisinin çeşitli döngülerinde etkili olabilmektedir.

Bu tezde, blok şifreleyicilerinin cebirsel yapıları ve bu şifreleyicilere gerçekleştirilen saldırılar irdelenmiştir. Blok şifreleyicilerdeki tek doğrusal olmayan

yapı olan S-kutularının cebirsel yapıları incelenmiş, geliştirilen yöntemlerle cebirsel ifadeleri elde edilmiş ve belirlenen zaaflarla ilgili iyiliştirme önerileri sunulmuştur.

1.1 Kriptografik Sistem

Mao kriptografik sistemle ilgili şu tanımı yapmıştır [1]:

Tanım 1.1. (Kriptografik Sistem) : Bir kriptografik sistemin içerdiği elemanlar şunlardır;

- alfabetik karakterler dizisinden oluşan P açık metin uzayı;
- şifreli metin mesajları kümesinden oluşan C şifreli metin uzayı;
- olası şifreleme anahtarları kümesinden oluşan K şifreleme anahtar uzayı ve olası deşifreleme anahtarları kümesinden oluşan K' deşifreleme anahtar uzayı;
- etkili bir anahtar planlama algoritması: $\gamma : N \to K \times K'$;
- etkili bir şifreleme algoritması: $\varepsilon : P \times K \to C$;
- etkili bir deşifreleme algoritması: $\varepsilon' : C \times K' \to P$.

1.2 Blok Şifreleyiciler

Blok şifreler, Shannon'un önerdiği karıştırma (confusion) ve yayılma (diffusion) tekniklerine dayanır [2]. Karıştırma şifreli metin ve açık metin arasındaki ilişkiyi gizlemeyi amaçlarken, yayılma açık metindeki izlerin şifreli metinde sezilmesini engellemek için kullanılır. Karıştırma ve yayılma, sırasıyla yer değiştirme ve doğrusal dönüşüm işlemleri ile gerçeklenir. Feistel Ağları ve Yer Değiştirme Permütasyon Ağları (Substitution-Permutation Network - SPN) [3] olmak üzere iki ana blok şifreleme mimarisi vardır. Her ikisi de yer değiştirme ve doğrusal dönüşümü kullanır. Ayrıca her iki mimari de çarpım (product) şifrelerinin örneklerindendir. Yani birden fazla şifreleme işleminin birleşmesi ile oluşturulurlar. Tekrarlanan şifreler yine çarpım şifreleridir ve aynı şifreleme adımının döngü olarak isimlendirilen tekrarlanan uygulamasından ibarettir. Bir döngü birden fazla şifreleme adımı içerebilir. Genellikle her döngüde bir

ana anahtardan elde edilen ve alt anahtar olarak isimlendirilen farklı anahtar değerleri kullanılır [4].

A. Menezes, P. Van Oorschot, ve S. Vanstone blok şifreleyicinin temel kavramını şu şekilde tanımlamıştır [3]:

Tanım 1.2. (Simetrik Anahtarlı Blok Şifreleyici): n-bitlik simetrik anahtarlı bir şifreleyici $\forall k \in \{0,1\}^l$ için $\{0,1\}^n$ 'den $\{0,1\}^l$ 'ye tersi alınabilir bir haritalamadan ibaret olan bir fonksiyondur ve $e:\{0,1\}^n \times \{0,1\}^l \to \{0,1\}^n$ şeklinde gösterilir. Ters haritalama $d_k(c)$ olarak gösterilen deşifreleme fonksiyonudur. Burada $c = e_k(p)$ ifadesi açık metin p'nin k anahtarı ile şifrelenmesi ile elde edilen şifreli metin c'dir.

1.3 Blok Şifreleme Algoritmalarının Özellikleri

1.3.1 Anahtar

Blok şifreleme algoritmalarında anahtarın uzunluğu ya da bit sayısı en temel saldırı olan geniş anahtar arama (exhaustive search) saldırısına karşın güçlü olmalıdır. Örneğin DES [5] şifreleyicisi 56-bit anahtar kullanırken AES [6] şifreleyicisi DES'in bu zaafını örter niteliktedir ve 128, 192, 256 bit anahtar seçenekleri mevcuttur. Ayrıca anahtarın rastlantısal olması gerekmektedir.

1.3.2 Döngü Sayısı

Blok şifreleme algoritmalarında döngü sayısı iyi seçilmek zorundadır. Çünkü doğrusal dönüşüm ve yer değiştirmelerin bu seçilen değerle algoritmaya yeterli gücü vermesi gerekmektedir. Ayrıca yapılan saldırıların başarısız olmasını sağlayan en önemli şartlardan biridir. Bu sayı için herhangi bir teorik hesaplama olmamasına rağmen Lars Knudsen'e göre kabaca döngü sayısı [4],

$$r \geq \frac{dn}{w} \tag{1.1}$$

ifadesindeki gibi olmalıdır. Burada r döngü sayısını, d yer değiştirme durumuna bir sözcük almak için gerekli maksimum döngü sayısını, n blok genişliğini, w ise tüm

şifrede yer değiştirme durumuna giriş olan minimum sözcük genişliğini temsil etmektedir. (1.1) ifadesinde yayılma tekniği ihmal edilmiştir. Lars Knudsen'e göre [4] bazı algoritmaların döngü sayılarının neler olması gerektiği Tablo 1.1'de gösterilmektedir.

Tablo 1.1. Bazı Şifreleme Algoritmaları için Döngü Sayıları

Algoritma	**Döngü sayısı**	[4]**'e göre olması gereken döngü sayısı**
DES	16	21
IDEA	8	8
Blowfish	16	16
AES(Rijndael)	10	16

1.3.3 S-kutuları

S-kutuları bir blok şifreleme algoritmasının en önemli elemanıdır. Çünkü algoritmadaki tek doğrusal olmayan yapıdır ve dolayısıyla algoritmaya gücünü vermektedir. S-kutuları için üç önemli nokta vardır. Bunların belirlenmesinde doğrusal kriptanaliz [7] , diferansiyel kriptanaliz [7] [8] ve Davies [9] saldırıları etkili olmuştur. Bunlar;

i. *SAC (Strict Avalance Criteria)*: 1 bit giriş değişimi sonucunda tüm çıkış bitlerinin değişme olasılığı $\frac{1}{2}$ olur.

ii. *S-kutularının genişliği*: Kriptanaliz saldırıları düşünüldüğünde büyük bir kutu küçüğüne oranla daha iyi olacaktır. Ayrıca diferansiyel saldırılardan korunmak için büyük sayıda çıkış bitleri ve doğrusal saldırılardan korunmak için büyük sayıda giriş bitleri gereklidir.

iii. *S-kutusu gereksinimleri*: Çıkışların dağılımları Davies [9] saldırısına karşın kontrol edilmeli, çıkışlar girişe göre doğrusal olmamalı ve S-kutusunun her sırasındaki değerler tek olmalıdır [4].

1.4 Çarpım(Product) Kriptosistem

Bilgisayar sistemlerinin hesaplama güçlerinde meydana gelen hızlı artış nedeniyle, kriptosistemler saldırılara karşı dayanıklı olabilmek için daha karmaşık şifreleme fonksiyonları ve geniş anahtar uzayları kullanmalıdırlar. Shannon tarafından bulunan ve çarpım kriptosistem olarak isimlendirilen yöntem, modern kriptosistemler için önemli bir fikir teşkil etmiştir [10]. Bu sayede küçük kriptosistemlerden büyük kriptosistemler oluşturmak mümkün olmuştur.

Tanım 1.3. İki veya daha fazla dönüşümün kendilerinden daha güçlü bir şifreleyici elde edecek şekilde birleştirilmesi sonucu elde edilen şifreleme yöntemine çarpım kriptosistem denir.

$S_1 = (P_1, C_1, K_1, \varepsilon_1, D_1)$, $S_2 = (P_2, C_2, K_2, \varepsilon_2, D_2)$ olan iki kriptosisteme sahip olalım. $C_1 = P_2$ ise S_1 ve S_2 çarpım kriptosistemi $(S_1 \times S_2)$,

$$(K_1, K_2) \in \kappa_1 \times \kappa_2 \text{ için,}$$

$$(P_1, C_2, \kappa_1 \times \kappa_2, \varepsilon, D)$$

$$e_{(K_1,K_2)}(x) = e_{K_2}\left(e_{K_1}(x)\right)$$

$$d_{(K_1,K_2)}(y) = d_{K_1}\left(d_{K_2}(y)\right)$$

olarak tanımlanır.

Çarpım kriptosistemler kriptosistemlerin kombinasyonu olarak da isimlendirilir. İki kriptosistem ancak ve ancak, birinci kriptosistemin şifreli metni, ikinci kriptosistemin açık metni ise, çarpım kriptosistem olarak isimlendirilir. İki kriptosistemin çarpımı her zaman yeni bir kriptosistem oluşturmaz. Örneğin, Vigenere şifreleyicisi ile Kaydırma (Shift) şifreleyicisinin çarpımı yine Vigenere şifreleyicisidir.

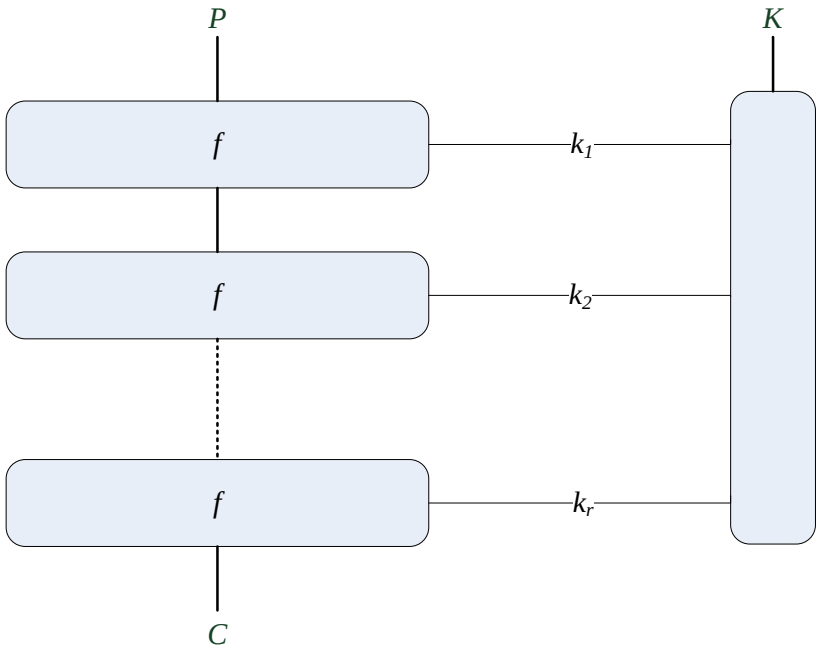
Bir kriptosistemin yeni bir kriptosistem elde etmek için kendisi ile birleştirildiği durumda aynı şifreleme algoritması iki kez uygulanır. Eğer $S \times S = S$ ise, S şifreleyicisi eşgüçlü öğe (idempotent) olarak isimlendirilir. Yer değiştirme şifreleyicisi, Vigenere şifreleyicisi, Hill şifreleyicisi ve Permütasyon şifreleyicisi eşgüçlüdür. Eğer S sistemi eşgüçlü değilse, aşağıdaki gibi sistemler oluşturmak mümkün olur:

$$\underbrace{S \times S \times \ldots \times S}_{n} = S^n$$

Böyle sistemler döngülü kriptosistemler olarak isimlendirilir. Modern blok şifreleyicilerin mimarisinde döngü yapısı kullanılır.

Döngüsel blok şifreleyicisi, döngü fonksiyonu olarak adlandırılan f fonksiyonunun sıradüzensel tekrarını içeren bir blok şifreleyicisidir. Fonksiyonun parametreleri, döngü sayısı r, blok bit uzunluğu n, K anahtarının bit uzunluğu k 'dan oluşur. K anahtarından r alt anahtar (k_i - döngü anahtarları) türetilir.

f fonksiyonunun tersinin alınabilmesi için, döngü fonksiyonu f, k_i'nin her bir değeri için döngü giriş değerleri ile birebir örten (bijection)' dir.



Şekil 1.1. Döngülü Blok Şifreleme Yapısı

1.5 İlkel Şifreleyiciler

1.5.1 Sezar Şifreleyicisi (Kaydırma Şifreleyicisi)

Sezar şifreleyicisinde açık metin harflerin sayısal karşılıkları ile ifade edilir. Sayısal olarak gösterimi yapılan açık metni oluşturan harflere karşılık gelen değere, belli bir değer eklenerek şifreli metin elde edilir. Eklenen değer anahtar değeridir. Açık

metin şifreli metindeki harflere karşılık gelen sayısal değerden, anahtar değerinin çıkarılması ile elde edilir.

$0 \le K \le 25$ için

$$E_K(x) = (x + K) \bmod 26$$

$$D_K(x) = (x - K) \bmod 26$$

Algoritma 1.1. Sezar Şifreleyicisi Algoritması

Latin alfabesinde toplam 26 karakter olduğu için bu işlemler mod 26 işlem uzayında gerçekleştirilir. Şifreleme ve deşifreleme süreçlerinin algoritmik gösterimi Algoritma 1.1'de verilmiştir.

Örnek 1.1. Açık metin değerimiz "ebruyuseviyorum" olsun. İlgili metnin sayısal olarak ifadesi aşağıdaki gibidir:

e b r u y u s e v i y o r u m

04 01 17 20 24 20 18 04 21 08 24 14 17 20 12

Anahtar değerimiz 10 olsun. Bu durumda her harfin sayısal karşılığına K=10 anahtar değerinin eklenmesi ile elde edilecek şifreli metin aşağıdaki gibidir:

14 11 01 04 08 04 02 14 05 18 08 24 01 04 22

o l b e i e c o f s i y b e w

Sezar şifreleyicisinde deşifreleme ise toplama (şifreleme) işleminin tersi olan çıkarma ile gerçekleştirilir. Yine mod 26 uzayında gerçekleştirilen işlem sonucunda elde edilen açık metin değeri aşağıdaki gibidir:

04 01 17 20 24 20 18 04 21 08 24 14 17 20 12

e b r u y u s e v i y o r u m

Güvenli bir kirptosistem için, anahtar uzayının yoğun anahtar arama yöntemi ile anahtar değeri elde edilemeyecek kadar geniş olması gerekir. 26 yeterli bir değer değildir.

1.5.2 Yer Değiştirme (Substitution) Şifreleyicisi

Sezar şifreleyicisinin genelleştirilmiş hali olup, harfler yine sayılarla ifade edilir. Sonlu bir X kümesinde tanımlanan bir birebir örten $\pi : X \to X$ permütasyonu aracılığıyla şifreleme gerçekleştirilir. Her bir π permütasyonunun π^{-1} ters permütasyonu vardır ve bu iki permütasyon şu kuralı sağlar:

$$\pi(x) = x' \text{ ancak ve ancak } \pi^{-1}(x') = x. \tag{1.2}$$

Tanım 1.4. $P, C \in Z_{26}, \kappa$ 26 sembolün tüm permütasyonlarını içerir. $\pi \in \kappa$ için

$$e_\pi(x) = \pi(x)$$
$$d_\pi(y) = \pi^{-1}(y)$$

olur.

Örnek 1.2. Açık metin değerimiz yine "ebruyuseviyorum" olsun. Şifreleme işleminde kullanılacak olan permütasyon aşağıda gösterilmiştir:

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

C G H W Z Q T N M L S X V R Y E O F D J I K U P B A

Yukarıdaki permütasyonla açık metin değerinin şifrelenmesi sonucu elde edilen açık metin değeri aşağıdadır:

e b r u y u s e v i y o r u m

Z G F I B I D Z K M B Y F I V

Deşifrelemede kullanılacak olan ters permütasyon, asıl permütasyondaki satırların yer değiştirmesi ile elde edilir ve aşağıda gösterilmiştir:

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

Z Y A S P R B C U T V J I H Q X F N K G W M D L O E

İlgili ters permütasyonun kullanılması ve şifreli metinden açık metnin elde edilmesi aşağıda gösterilmiştir:

Z G F I B I D Z K M B Y F I V

e b r u y u s e v i y o r u m

1.5.3 Permütasyon Şifreleyicisi

Permütasyon şifreleyicisi monoalfabetik[1] olmayan bir şifreleyicidir.

Tanım 1.5. m sabit bir tamsayı, $P = C = (Z_{26})^m$ olsun ve $\kappa \{1,2,\ldots,m\}$ kümesinin tüm permütasyonlarını içersin. $\pi \in \kappa$ anahtarı için (π^{-1} π permütasyonunun tersidir.),

$$e_\pi(x_1,\ldots,x_m) = (x_{\pi(1)}\ldots,x_{\pi(m)})$$

$$d_\pi(y_1,\ldots,y_m) = (y_{\pi(1)}\ldots,y_{\pi(m)}).$$

Örnek 1.3. Açık metin değerimiz "ebruyuseviyorumben" olsun. $m = 6$ ve $\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 3 & 1 & 6 & 2 & 5 \end{pmatrix}$ ve $\pi^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 3 & 5 & 2 & 1 & 6 & 4 \end{pmatrix}$ olsun.

Öncelikle açık metin 6 karakter uzunluğunda bloklara bölünür.

Blok 1	Blok 2	Blok 3
e b r u y u	s e v i y o	r u m b e n
u r e u b y	i v s o e y	b m r n u e

Deşifreleme için π permütasyonunun tersi olan π^{-1} permütasyonu kullanılır.

u r e u b y	i v s o e y	b m r n u e
e b r u y u	s e v i y o	r u m b e n

Permütasyon şifreleyicileri şifreli metin saldırılarına karşı güvenlidir. Fakat bilinen açık metin saldırılarında, açık metin ve şifreli metin kullanılarak blok uzunluğu ve anahtarı bulmak zor olmayacaktır.

1.5.4 Vigenere Şifreleyicisi

Vigenere şifreleyicisi de monoalfabetik olmayan bir şifreleyici örneğidir.

[1] monoalfabetik : Şifreleme sonucunda bir harf hep aynı harfe dönüşüyorsa, bu şifreleyiciye monoalfabetik şifreleyici denir.

Tanım 1.6. m bir tamsayı ve $P = C = \kappa = (Z_{26})^m$ olsun. $K = (k_1, k_2, \ldots, k_m)$ anahtarı için,

$$e_K(x_1, \ldots, x_m) = (x_1 + k_1, \ldots, x_m + k_m)$$

$$d_K(y_1, \ldots, y_m) = (y_1 - k_1, \ldots, y_m - k_m).$$

Tüm bu işlemler Z_{26} uzayında gerçekleştirilir.

Örnek 1.4. Açık metin değerimiz "ebruyuseviyorum" , $m = 5$ ve anahtar değeri *"OSMAN"* olsun.

e b r u y	*u s e v i*	*y o r u m*
OSMAN	*OSMAN*	*OSMAN*
t u d v m	*j k r w v*	*m g e v z*

Deşifreleme işlemi de aynı anahtar kullanılarak gerçekleştirilir. Şifreleme işleminde kullanılan toplama işleminin yerine toplamanın tersi olan çıkarma işlemi kullanılır.

t u d v m	*j k r w v*	*m g e v z*
OSMAN	*OSMAN*	*OSMAN*
e b r u y	*u s e v i*	*y o r u m*

Vigenere şifreleyicisine saldırmak için blok uzunluğunu ve gizli anahtarı bilmek gerekir. Wolfe Friedman 1920 yılında rastlantı indisi (index of coincidence) isimli bir yöntem sunmuştur [11].

Tanım 1.7. Rastlantı İndisi (Index of Coincidence). $x = x_1 x_2 \ldots x_n$ n alfabetik karakterden oluşan bir dize olsun. $A, B, \ldots, Z$ karakterlerinin oluşma frekansları ise $f_0\, f_1, \ldots, f_{25}$ olarak tanımlansın. x karakterinin rastlantı indisi aşağıdaki gibi tanımlanır:

$$I_c(x) = \frac{\sum_{i=0}^{25} f_i(f_i - 1)}{n(n-1)}. \tag{1.3}$$

$I_c(x)$ rastgele seçilen iki x elemanının eşit olma olasılığıdır. x monoalfabetik bir şifreleme sonucunda elde edilen bir şifreli metin ise , $I_c(x) \approx 0{,}065$ 'dir. Eğer x rastgele bir dize ise $I_c(x) \approx 0{,}038$ 'dir.

1.5.5 Hill Şifreleyicisi

Hill şifreleyicisi 1929 yılında Lester S. Hill tarafından sunulmuştur [12]. Vigenere şifreleyicisinin benzeridir. $P = C = (Z_{26})^m$, κ anahtar elemanları Z_{26} uzayında olan $m \times m$ bir matristir [13].

Tanım 1.8. $P = C = (Z_{26})^m$ ve κ anahtar elemanları Z_{26} uzayında tersi alınabilir tüm $m \times m$ matrisleri içersin. $K \in \kappa$ $x, y \in (Z_{26})^m$ ve tüm işlemler Z_{26} uzayında yapılmak şartıyla,

$$e_K(x) = xK$$
$$d_K(y) = yK^{-1}$$

olarak tanımlanır.

Örnek 1.5: m =2 ve $K = \begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix}$ ve açık metin "ebruyuseviyorumm" olsun.

eb	ru	yu	se	vi	yo	ru	mm
(4,2)	(17,20)	(24,20)	(18,4)	(21,8)	(24,14)	(17,20)	(12,12)

(4,2) K = (4×11 + 2×3 , 4×8 + 2×7) = (24,20) yu

(17,20) K = (17×11 + 20×3 , 17×8 + 20×7) = (13,16) nq

(24,20) K = (24×11 + 20×3 , 24×8 + 20×7) = (12,20) mu

(18,4) K = (18×11 + 4×3 , 18×8 + 4×7) = (2,16) cq

(21,8) K = (21×11 + 8×3 , 21×8 + 8×7) = (21,16) vq

(24,14) K = (24×11 + 14×3 , 24×8 + 14×7) = (20,4) ue

(17,20) K = (17×11 + 20×3 , 17×8 + 20×7) = (13,16) nq

(12,12) K = (12×11 + 12×3 , 12×8 + 12×7) = (12,24) my

eb ru yu se vi yo ru mm

yu nq mu cq vq ue nq my

Deşifreleme için K matrisinin tersi olan $K^{-1} = \begin{pmatrix} 7 & 18 \\ 23 & 11 \end{pmatrix}$ matrisi kullanılır.

(24,20) K^{-1} = (24×7 + 20×23 , 24×18 + 20×11) = (4,2)

(13,16) K^{-1} = (13×7 + 16×23 , 13×18 + 16×11) = (17,20)

(12,20) K^{-1} = (12×7 + 20×23 , 12×18 + 20×11) = (24,20)

(2,16) K^{-1} = (2×7 + 16×23 , 2×18 + 16×11) = (18,4)

(21,16) K^{-1} = (2×7 + 16×23 , 2×18 + 16×11) = (18,4)

(20,4) K^{-1} = (20×7 + 4×23 , 20×7 + 4×11) = (24,14)

(13,16) K^{-1} = (13× 7 + 16×23 , 13×7 + 16×11) = (17,20)

(12,24) K^{-1} = (12×7 + 12×23 , 12×7 + 12×23) = (12,12)

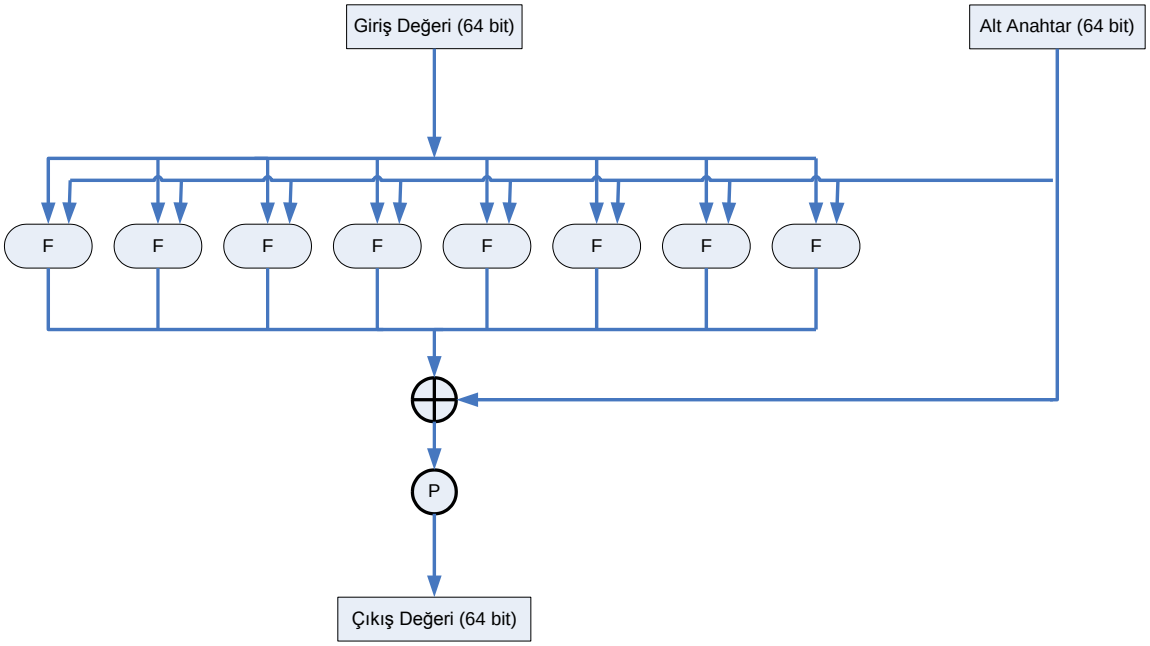
yu nq mu cq vq ue nq my

eb ru yu se vi yo ru mm

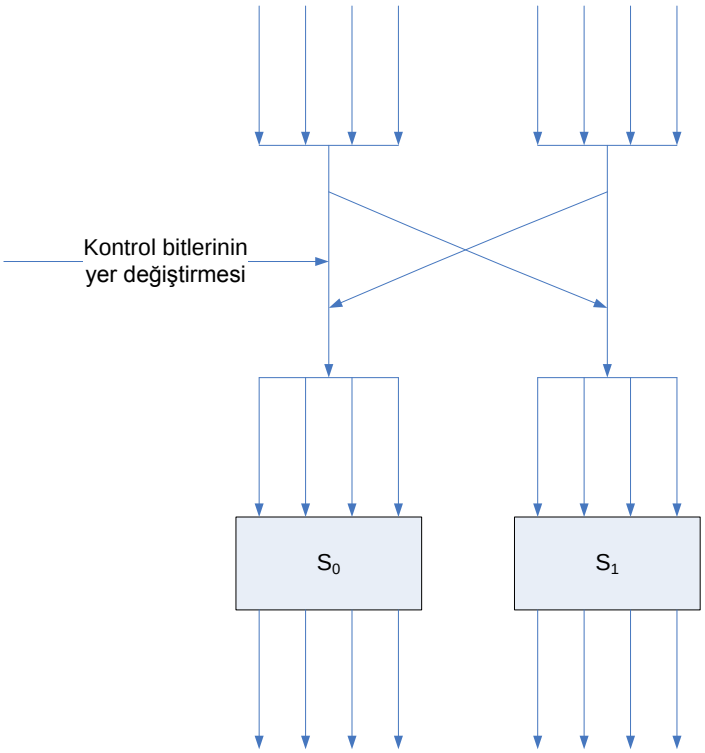
Açık metin ve şifreli metin arasında doğrusal bir ilişki varsa, Hill şifreleyicisi güvenli bir kriptosistem değildir.

1.6 Lucifer (İlk Blok Şifreleyici)

Feistel mimarisini kullanan DES benzeri bir blok şifreleyici olup, bir döngüsünde veri bloğunun yarısı üzerinde işlem yapar. Anahtar ile birlikte işleme tabi tutulan yarı blok, döngü sonunda diğer yarı blok ile XOR işlemine tabi tutulduktan sonra, sonraki döngünün giriş değerinin yarı bloğu elde edilir. Diğer yarı blok işleme tabi tutulmadan bir sonraki döngüye geçer. İki yarı blok diğer döngünün girişlerine verilmeden önce yer değiştirilir. Böylelikle diğer döngüde önceki döngüde işleme tabi tutulmayan blok işleme tabi tutulur.



Şekil 1.2. Lucifer Şifreleyicisinin Bir Döngüsü



Şekil 1.3. Lucifer Şifreleyicisinin S-Kutusu

Lucifer şifreleyicisi 128 bitlik blokları 128 bitlik anahtarla şifrelemekte olup, döngü fonksiyonu simetrik özelliklere sahiptir ve tüm döngülerinde veri bloğunun sağ

tarafını işleme tabi tutar. 16 döngüden oluşan Lucifer şifreleyicisi diferansiyel kriptanalize karşı zayıf bir şifreleyicidir [13].

Lucifer şifreleyicisinin her döngüsü 72-bitlik alt anahtarlar kullanır. İlk döngünün alt anahtarı, anahtarın birinci sekizlisinin iki kez tekrarlanması ve takip eden 7 sekizliden oluşur. Sonraki döngünün anahtarı, anahtarın 7 sekizli döndürülmesiyle elde edilir.

Yerine koyma işlemi 8 adet benzer 4-bit S-kutusu çifti $(S_0 \,\&\, S_1)$ içerir (bknz. Tablo 1.2). Bu S-kutuları anahtar değerine göre $(S_0|S_1)$ veya $(S_1|S_0)$ sıralamasında kullanılır.

Tablo 1.2. Lucifer Şifreleyicisinin S-Kutusu

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

1.7 Sonlu Cisimler

Teorem 1.1. p asal sayı, n pozitif tamsayı olma şartı ile $m = p^n$ ise m. dereceden bir sonlu cisim vardır.

p^n elemanlı bir sonlu cisim Galois alanı olarak tanımlanır ve $GF(p^n)$ ile ifade edilir. p asal sayısı için $GF(p)$ Z_p'ye eşittir. $GF(p^n), n>1$ için $GF(p)$'den $GF(p)$ uzayındaki n. dereceden indirgenemez polinom kullanılarak elde edilir.

$GF(p)$ uzayında n. dereceden bir polinom şu şekilde ifade edilir:

$$0 \le a_i \le p-1, a_n \ne 0 \text{ (veya } a_i \in GF(p), a_n \ne 0\text{) için}$$

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 = \sum_{i=0}^{n} a_i x^i \,. \tag{1.4}$$

Tanım 1.9. $P(x)$ polinomu n' den küçük derecede iki polinomun çarpımı olarak yazılamıyorsa indirgenemez polinom olarak isimlendirilir.

Örnek 1.6. $P_1(x) = x^3 + x^2 + 1, P_2(x) = x^3 + x + 2$ $GF(2^3)$ uzayında tanımlanmış iki polinom olsun.

$$P_1(x) + P_2(x) = (2x^3 + x^2 + x + 3) \bmod 3 = 2x^3 + x^2 + x$$
$$P_1(x)P_2(x) = (x^3 + x^2 + 1)(x^3 + x + 2) = x^6 + x^4 + 2x^3 + x^5 + x^3 + 2x^2 + x^3 + x + 2$$
$$= x^6 + x^5 + x^4 + x^3 + 2x^2 + x + 2$$

$GF(p)$ üzerinde tanımlanan bir polinom çarpma ve toplama işlemine göre komütatif halkadır.

1.8 Modüler Polinom Aritmetiği

Teorem 1.2. $m(x)$ $GF(p)$ üzerinde tanımlanan bir polinom olsun. $GF(p)$ üzerinde tanımlı $P(x)$ polinomu, eğer $P(x) = q(x)m(x) + r(x)$ ve $r(x)$ polinomunun derecesi *m(x)* polinomunun derecesinden küçükse, $P(x) = r(x) \bmod m(x)$ olarak ifade edilebilir ve *P(x)* polinomu *r(x) mod m(x)* ifadesi ile kongruenttir.

Teorem 1.3. Sıfırdan farklı olan bir $P(x)$ polinomu için, eğer $P(x)f(x) \equiv 1 \bmod m(x)$ olacak şekilde bir $f(x)$ polinomu varsa, $f(x)$ polinomu modüler polinom aritmetiğinde $P(x)$ polinomunun çarpma işlemine göre tersidir ve $f(x) = P^{-1}(x)$ şeklinde ifade edilir.

Modüler polinom aritmetiği komütatif halkadır. Eğer $m(x)$ indirgenemez bir polinom ise sıfırdan farklı tüm polinomların modüler polinom aritmetiğinde çarpma işlemine göre tersi vardır. Bu sayede p^n. dereceden sonlu cisim oluşturmak mümkün olacaktır.

AES şifreleme algoritmasında kullanılan $GF(2^8)$ uzayını oluşturalım. Sonlu cismi oluşturmak için Z_2[2] uzayında tanımlanan ve aşağıda gösterilen $m(x)$ indirgenemez polinomu kullanılacaktır:

$$m(x) = x^8 + x^4 + x^3 + x + 1.$$

[2] Z_2 ikili taban uzayını göstermektedir.

$GF(2^8)$ uzayındaki tüm elemanlar polinom olarak veya 8 bitlik ikili dize olarak ifade edilebilir. Örneğin $x^6+x^4+x^2+x+1$ (01010111) olarak ifade edilir. Diğer bir polinom ise x^7+x^4+x (10010010) olsun. Sonlu cisimdeki toplama işlemi ise şu şekilde gerçekleştirilir:

$$(x^6+x^4+x^2+x+1)+(x^7+x^4+x)=x^7+x^6+x^2+1.$$

İkili gösterimde aynı toplama işlemi şu şekilde gerçekleştirilir:

$$(01010111)\oplus(10010010)=(11000101).$$

Çarpma işlemi toplama işlemine göre biraz karmaşık bir işlem olup aşağıda matematiksel olarak ifade edilmiştir:

$$x^8 \equiv x^4+x^3+x+1 \bmod m(x).$$

$$P(x)=\sum_{i=0}^{7} b_i x^i, b_i = 0 \text{ veya } 1,\ i=0,1,\cdots,7.$$

$$xP(x)=\sum_{i=0}^{7} b_i x^{i+1}$$

$$\equiv \begin{cases} \sum_{i=0}^{6} b_i x^{i+1} & b_7=0 \\ \left(\sum_{i=0}^{6} b_i x^{i+1}\right)+x^4+x^3+x+1 & b_7=1 \end{cases}$$

$$xP(x) \equiv \begin{cases} b_6b_5b_4b_3b_2b_1b_0 & b_7=0 \\ b_6b_5b_4b_3b_2b_1b_0\,0\oplus(00011011) & b_7=1 \end{cases}$$

$xP(x)$ işleminin sonucunu hesaplamak için kaydırma ve XOR işlemlerine ihtiyaç vardır. Bu işlemlerin i kez yapılması durumunda $x^iP(x)$ değeri elde edelir. Sonuç olarak $GF(2^8)$'de çarpma işlemini yapmak için kaydırma ve XOR işlemlerine ihtiyaç vardır.

1.9 Karıştırma (Confusion) ve Yayılma (Diffusion)

Bir kriptografik sistem için iki temel yapı bloğu vardır;

i. karıştırma,

ii. yayılma.

Karıştırma (Confusion), açık metin ve şifreli metin arasındaki ilişkiyi belirsizleştirmek için, şifreli metin istatistikleri ile şifreleme anahtarının değeri arasındaki ilişkiyi, mümkün olduğu kadar karmaşık hale getirir. Bu karmaşık alt anahtar oluşturma algoritması ile başarılır. Böylelikle şifreli metin incelenerek istatistiksel modellere erişilmesi zorlaşır. Basit bir yer değiştirme şifresi olan Ceasar şifresinde, her bir harf başka bir harfle yer değiştirmekteyken, modern yer değiştirme şifrelerinin yapısı daha karmaşıktır. Uzun bir açık metin bloğunda yer değişikliği yapılarak şifreli metin bloğu oluşturulur, ve açık metindeki veya anahtardaki tüm bitler için yer değiştirme teknikleri kullanılır. Bu tip bir yer değiştirmenin yeterli olmadığı, German Enigma karmaşık yer değiştirme algoritmasının ikinci dünya savaşında kırılması ile ispatlanmıştır [4].

Permütasyon ile sağlanan yayılma (diffusion), açık metnin artığını (redundancy) şifreli metin üzerine dağıtır. Böylece, açık metnin istatistiksel yapısı şifreli metnin uzun aralıklı istatistiklerine dağıtılır. Bir şifreli metin rakamının birçok açık metin rakamı tarafından etkilenmesi, artıkları bulmaya çalışan bir kriptanalistin oldukça zaman harcamasına neden olacaktır. Yayılma yapmayı sağlamanın en basit yolu doğrusal dönüşümdür (permütasyon da denir). Sütun yer değiştirme şifresi gibi basit bir yer değiştirme şifresi, basitçe açık metindeki harflerin yerlerini değiştirir. Modern şifreler bu tip permütasyonlara ek olarak tüm mesajın içinden mesajın bölümlerini yayılan yayılma çeşitlerini de kullanırlar.

Bazı geri besleme şemaları yayılmaya eklense de akış şifreleri sadece karıştırmaya güvenirler. Blok şifreleme algoritmaları hem karıştırma hem de yayılmayı kullanırlar. Genel bir kural olarak, yayılma yalnız başına kolayca kırılır.

1.10 Karmaşıklık Teorisi

Karmaşıklık teorisi, değişik kriptografik teknik ve algoritmaların hesapsal karmaşıklıklarını analiz etmeye yarar. Kriptografik teknikleri ve algoritmaları karşılaştırır ve güvenliklerini belirler.

Bir algoritmanın karmaşıklığı algoritmayı çalıştırmak için gereken hesap gücüyle ölçülür. Karmaşıklık teorisinin asıl amacı, hesapsal problemleri, onları çözmek için gerek duyulan kaynaklara göre sınıflandırmak için teknikler sağlamaktır. Sınıflandırma belirli bir hesapsal modele bağlı olmamalıdır; fakat esas olarak problemin gerçek zorluğunu ölçmelidir. Bir algoritmanın hesapsal karmaşıklığı genellikle iki değişkenle ölçülür: T (zaman karmaşıklığı için) ve S (uzay karmaşıklığı veya bellek gereksinimi için). Hem T, hem S genellikle n (girdinin büyüklüğü)'in fonksiyonları olarak ifade edilirler.

Genellikle bir algoritmanın hesapsal karmaşıklığı büyük-O ile gösterilir. Hesapsal karmaşıklığın büyüklük derecesi (order) n büyüdükçe, karmaşıklık fonksiyonunun en hızlı büyüyen terimidir; bitin düşük sıradaki terimleri önemsenmez. Örneğin bir algoritmanın zaman karmaşıklığı $4n^2 + 7n + 12$ ise, hesapsal karmaşıklık n^2 derecesindedir ve $O(n^2)$ olarak ifade edilir.

Girdi büyüklüğü, zaman ve boşluk gereksinimini etkiler. Örneğin eğer $T = O(n)$ ise, girdi büyüklüğünü 2 katına çıkarmak algoritmanın çalışma zamanını 2 katına çıkarır. Eğer $T = O(2^n)$ ise girdi büyüklüğüne 1 bit eklemek algoritmanın çalışma zamanını 2 katına çıkarır [14].

1.11 Blok Şifreleyicilerine Karşı Yapılan Saldırılar

1.11.1 Saldırının Sonucu (Outcome of an Attack)

Knudsen saldırı sonuçlarını, saldırı sırasında elde edilen bilgiye göre aşağıdaki gibi sınıflandırmıştır [15]:

- *Tamamen kırma (Total break)*: Saldırgan gizli anahtar k 'yı elde eder.
- *Evrensel Türetim (Global Deduction)*: Saldırgan k anahtarının asıl değerini bilmeden $e_k(.)$ veya $d_k(.)$ fonksiyonuna eşdeğer bir algoritma bulur. Evrensel türetim blok şifreleyicilerin blok yapıları içerdiği durumlarda olasıdır. Örneğin şifreli metnin belirli bir bölümünün, açık metnin belirli bölümünden tamamen bağımsız olduğu durumlarda uygulanabilir. Böyle bir

blok şifreleyici anahtar uzunluğu önemsenmeksizin bilinen açık metin saldırılarında evrensel türetim yöntemine karşı zayıftır. Başka bir saldırıda döngü alt anahtarları evrensel türetim ile elde edilebilir. Bu durumda anahtar planlama algoritmasında tek yönlü fonksiyonların kullanılması gereklidir.

- *Yerel Türetim (Local Deduction)*: Saldırganın elde ettiği şifreli metinden açık metni, açık metni elde ettiyse şifreli metni bulmasıdır. Açık metin veya şifreli metin sayısı azsa yerel türetim, tümüyle kırma kadar tehlikelidir.
- *Ayırt Edici Saldırı (Distinguishing Attack)*: Saldırgan saldırılan blok şifreleyicinin düzgün (uniform) olarak rastgele seçilmiş bir permütasyon mu yoksa gizli anahtar tarafından belirtilmiş permütasyonlardan biri mi olduğunu söyleyebilir. Ayırt edici saldırı çok tehlikeli gibi görünmese de bazen tümüyle kırmayla sonuçlanan anahtar elde etme saldırılarına dönüşebilir [16].
- *Bilgi Türetim Saldırısı (Information Deduction Attack)*: Saldırgan önceden bilmediği gizli anahtar, şifreli metin veya açık metin hakkında bazı bilgiler elde edebilir. Bilgi türetim açık metin ve şifreli metin düşük entropiye sahipse önemli bir sorundur [2].

1.11.2 Blok Şifreleyicilerinde Saldırı Tipleri

- *Sadece şifreli metin saldırı*: Bu pasif saldırı tipinde, saldırgan sadece şifreli metinleri inceleyerek anahtar hakkında bilgi elde etmeye çalışır.
- *Bilinen açık metin saldırı*: Bu pasif saldırı tipinde, saldırgan bir veya daha fazla açık metin ve karşılığında şifreli metne sahiptir. Bu türdeki saldırının amacı anahtarı bulmaktır. Bilinen açık metin saldırısının tipik örneği doğrusal kriptanalizdir [7] [17].
- *Seçilen açık metin saldırı*: Bu aktif saldırı tipini gerçekleştirmek için, saldırgan açık metni seçebilmekte ve karşılığı olan şifreli metni elde edebilmektedir. Açık metin elde edilen şifreli metne bağlı olmak zorunda değildir. Saldırgan anahtarı ve daha önce görülmeyen bir şifreli metne ait

açık metni elde etmek için türetilmiş tüm verileri kullanır. Bu tür saldırının tipik bir örneği diferansiyel kriptanalizdir [7] [18] [19].

- *Seçilen şifreli metin saldırı*: Saldırgan istediği deşifreleme için işleme tabi tutulan şifreli metin ve karşılığı olan açık metne erişebilmektedir.
- *Uyarlanabilir seçilen açık metin saldırı*: Bu saldırı seçilen açık metin saldırısı olup açık metinler, elde edilen şifreli metinlere göre seçilir.
- *Uyarlanabilir şifreli metin saldırı*: Saldırgan deşifrelenmek üzere şifreli metni, sonraki seçeneklerini, önceki yaydığı verilerin sonuçlarını temel alarak yayma özgürlüğünü kullanarak, yayabilir.
- *Birleştirilmiş seçilen açık metin ve şifreli metin saldırı*: Uyarlanabilir saldırılarının çok güçlü bir türü olan bu saldırıda, saldırgan keyfi olarak seçilmiş metinlerden istediğini şifreleyelebilir ve deşifreleyebilir. Böyle bir saldırının en tipik örneği Wagner'in boomerang saldırısıdır [20] [21].
- *İlişkili anahtar (related-key) saldırı*: Bu saldırı türünde saldırgan şifreleme ve deşifrelemede kullanılan anahtarlar arasında matematiksel bir ilişkiyi bilmekte veya seçebilmekte olup anahtarların değerlerini bilmemektedir.

Winternitz ve Hellman ilişkili anahtar saldırı ile ilgili şu kavramı sunmuştur [22]:

"Eğer iki farklı sistem yaklaşık olarak aynı karmaşıklıkta ise ve sistemlerden biri seçilen anahtar kriptanalitik saldırısına karşı zayıf kalırken diğer sistem güçlü ise, güçlü olan seçilmelidir."

1.11.3 Saldırı Modelleri

Pascal Junod blok şifreleyicilerin iç yapısı ile ilgili bilgiler ve uygulama ayrıntılarını incelerken topladığı bilgilerden faydalanarak, saldırı tiplerini şu şekilde sınıflandırmıştır [23]:

- *Kapalı kutu saldırılar*: Açık metin ve anahtarı girdi olarak alıp, şifreli metni çıktı olarak veren blok şifreleyicileri tehdit eden genel bir saldırı türüdür. Bu tür saldırılar blok şifreleyicinin iç yapısına bağlı olmayıp, tüm blok

şifreleyicilerine uygulanabilir. Bu saldırının zaman karmaşıklığı anahtar uzunluğu ve blok şifreleyicinin blok uzunluğuna göre değişir. Yoğun anahtar arama ve genel zaman-bellek seçimi kapalı kutu saldırılarına örnek olarak verilebilir.

- *Kısayol (Shortcut) saldırılar*: Kapalı kutu saldırılardan farklı olarak, kısayol saldırılar blok şifreleyicilerin iç ayrıntılarının matematiksel analizinden elde edilen bilgileri temel alırlar.
- *Yan-Kanal (Side-Channel) Saldırı*: Blok şifreleyicilerin temel yapısı yazılımsal veya donanımsal olarak gerçekleştirilir. Yan-kanal saldırılar gerçekleştirmeden kaynaklanan fiziksel olguları kötüye kullanırlar. Örneğin zamanlama (timing) [24] [25] [26] [27] saldırıları algoritmanın çalışma zamanının veri veya anahtar değerine bağlı olduğu durumlarda uygulanabilir. Blok şifreleyicilerin fiziksel gerçekleştirmesindeki güçsüzlükten faydalanmanın diğer bir yöntemi ise kurcalamalara karşı dayanıklı (tamper-proof) donanımların güç tüketimini ölçmektir. Bu ölçümlerden anahtar ile ilgili bazı bilgiler elde edilir. Son olarak, hata analizi ile Biham ve Shamir tarafından sunulan aşağıdaki fikir istismar edilmiştir [28]:

  "Saldırgan blok şifreleyicilerin çalışması sırasında algoritma davranışı üzerindeki etkilerini incelemek ve anahtar hakkında bazı bilgiler elde etmek amacıyla, fiziksel aletlerle oynayarak hatalar oluşturur."

1.11.4 Saldırının Parametreleri

- *Zaman karmaşıklığı*: Zaman karmaşıklığı saldırının başarıyla gerçekleştirilmesi için gerekli hesaplama miktarıdır. Hesaplama birimi saldırının yoğun anahtar aramayla karşılaştırılmasına göre seçilir.
- *Veri karmaşıklığı*: Saldırının gerçekleştirilmesi için gerekli açık metin, şifreli metin gibi veri miktarıdır. Bu veri anahtar sahibinden elde edileceğine göre, bu karmaşıklığın iletişim karmaşıklığına doğrudan etkisi söz konusudur.

- *Başarı olasılığı*: Saldırının başarı olasılığı istatistiksel olarak bağımsız şekilde belirli sayıda tekrar edildiğinde saldırının başarılı olma frekansını gösterir.
- *Bellek karmaşıklığı*: Saldırı gerçekleştirmek için gerekli önceden hesaplanmış veri ve üzerinde çalışılan tehdit modelinden elde edilen veriyi saklamak için gerekli bellek miktarını ölçer.

Bir blok şifreleyicisi, anahtar uzunluğunun (l) iki katından önemli ölçüde az zaman karmaşıklığına $(2l)$ sahip bir saldırı düzenlenmesi durumunda kırılmış olarak nitelendirilir. Blok şifreleyici, açık metnin bazı bitleri yoğun anahtar aramadan daha hızlı bir şekilde elde edilirse kısmi olarak kırılmış sayılır.

Tanım 1.10. Bir blok şifreleyici, herhangi bir genel saldırının gerektirdiği karmaşıklıktan hem zaman hem veri karmaşıklığı açısından daha az karmaşıklıkta bir saldırı ile kırılamıyorsa güvenli olarak nitelendirilir.

1.11.5 Kapalı Kutu Saldırılar

Kapalı kutu saldırılar yoğun anahtar arama, çoklu şifreleme, anahtar çakışma saldırıları ve zaman bellek seçimi saldırılarıdır:

1.11.5.1 Yoğun Anahtar Arama

Bu saldırı yönteminde saldırgan açık metin ve şifreli metne (p,c) sahiptir. Saldırgan anahtar uzayındaki tüm anahtarlar (2^l adet) ile p açık metnini şifreleyerek c şifreli metnini elde etmeye çalışır. Doğru anahtarı tekil olarak tanıyabilmek için, ek olarak küçük sayıda açık metin şifreli metin çifti daha gerekebilir [29].

Yoğun anahtar arama diğer saldırı yöntemleri için gösterge (benchmark) olarak düşünülür. Blok şifreleme üzerinde teorik kırma veya akademik kırma yoğun anahtar aramanın zaman karmaşıklığından (2^l) daha az zaman karmaşıklığına sahip saldırıdır.

Yoğun anahtar aramanın önemli bir özelliği [23], saldırının anahtar uzayının farklı alt kümeleriyle ilgilenen birçok işlemcide veya adanmış makinelerde paralel

olarak çalıştırılan iş parçacıklarından oluşmasıdır. Yoğun anahtar aramanın başarı olasılığı araştırılan anahtar uzayının oranına eşittir. Eğer saldırgan anahtar uzayının onda birinde arama yaparsa, başarı oranı kabaca %10 olur. Başka bir deyişle, sabit anahtar uzunluğu (l) blok şifreleyicinin güvenliğinin üst sınırını tanımlar. Bu nedenle, güvenli bir blok şifreleyeci için, sabit anahtar uzunluğu yoğun anahtar aramayı engelleyecek kadar büyük olmalıdır.

1.11.5.2 Anahtar Çakışma Saldırıları

Anahtar çakışma saldırısı Biham tarafından sunulmuştur [30]. Saldırının yöntemi çok basit olup doğum günü çelişkisini (birthday paradox) temel alır.

Tanım 1.11 (Doğumgünü Çelişkisi – Birthday Paradox). Anahtar uzunluğu (l), blok uzunluğundan küçük $(n < l)$ olan bir blok şifreleyicisine saldırdığımızı düşünelim. Ayrıca bilinen açık metin (p) birçok farklı anahtar ile şifrelensin. p açık metnini $2^{\frac{l}{2}}$ farklı anahtar ile şifrelediğimizde, doğum günü çelişkisine göre $2^{\frac{l}{2}}$ şifreli metni inceledikten sonra, anahtarlardan en az biri aradığımız anahtar olmalıdır.

1.11.5.3 Çoklu Şifreleme

1977 yılında Diffie ve Hellman [31] çift şifreleme üzerinde ortadaki adam (meet-in-the-middle) saldırısında, çoklu şifreleme senaryosunda en az üç kat şifreleme kullanılmasını önermiştir. Bu saldırı şu şekilde çalışır:

- Aynı anahtarla şifrelenmiş birkaç tane (p_i, c_i) açık metin şifreli metin çiftine sahip olalım.

$$c_i = e_{k_2}\left(e_{k_1}(p_i)\right) \tag{1.5}$$

- Verilen (p_1, c_1) çifti için, 2^l olası anahtar s ile $m_s = e_s(p_1)$ ifadesi hesaplanır ve (m_s, s) çifti m_s ile dizinlenen bir tabloda saklanır.

- Sonra c_1 değeri 2^l olası anahtar değeri t ile deşifrelenir ve her bir $m_t = d_t(c_1)$ çifti için, ilk tabloda $m_t = m_s$ ifadesinin eşitliğine bakılır.
- Her bir çözüm (s,t) anahtar çifti için olası bir çözümü açıkça tanımlar. Diğer birkaç bilinen açık metin ve şifreli metin çiftini kullanarak doğru anahtar (k_1, k_2) ayıklanabilir.
- Bu saldırı l-bit anahtar kullanan çift şifreleme modunda çalışan bir blok şifreleyicisini $O(2^l)$ zaman karmaşıklığında kırar ve $O(2^l)$ bellek hücresi kullanır.

1.11.5.4 Zaman-Bellek Seçimi (Time-Memory Tradeoff)

Hellman yoğun anahtar aramaya uygulanabilecek zaman-bellek seçimi saldırısını sunmuştur [32]. Bu saldırıdaki fikir bazı bilgileri önceden hesaplamayı ve anahtar aramayı hızlandırmak için kullanmayı içerir. Bu saldırı $O\left(2^{\frac{2l}{3}}\right)$ bellek sözcüğü kullanarak $O\left(2^{\frac{2l}{3}}\right)$ şifreleme ile l-bit anahtarı elde edebilir.

1.11.6 İstatistiksel Saldırılar

Blok şifreleyicilerin istenmeyen olasılıksal özelliklerini kullanan saldırılar hakkında kısaca bilgi vereceğiz:

1.11.6.1 Doğrusal Kriptanaliz

1993 yılında Matsui tarafından teorik bir saldırı olarak keşfedilmiştir [17]. Daha sonra DES algoritmasına karşı başarı ile uygulanmıştır. Doğrusal kriptanaliz, şifreli metin bitleri ile açık metin bitleri arasındaki yüksek olasılıkta doğrusal ifadelerin meydana gelme avantajını kullanır. Bunun yolu da S-kutularından geçer. Saldırganın algoritmayı bildiği (Kerckhoffs kuralı) ve belli sayıda açık metin ve şifreli metinlere

sahip olduğu varsayılır. S-kutularının büyüklüğü, aktif S-kutularının (doğrusal ifade içinde olan) sayısının artışı ve doğrusal sapması ($\frac{1}{2}$ 'den + veya -) küçük S-kutularının tasarımı doğrusal kriptanalizin uygulanmasını engelleyici faktörlerdir [29].

Tanım 1.12. (Kerckhoffs Kuralı). Bir kriptosistem, sistemle ilgili anahtar dışındaki bilinmesi gereken herşey bilinmesine rağmen güvenlidir. Kerckhoffs kuralı 19. yüzyılda Auguste Kerckhoffs tarafından belirlenmiştir.

1.11.6.2 Diferansiyel Kriptanaliz

Diferansiyel kriptanaliz günümüzde bilinen en önemli saldırılardan birisidir. Diferansiyel kriptanaliz ile DES'in anahtarları teorik olarak, tüm anahtar uzayında denemeyle beklenen çalışma gücünden daha az bir güçle elde edilebilmektedir [19].

Diferansiyel Kriptanaliz [7] [18] [19], kriptosistemlerin yeniden gözden geçirilmesine ve yeni sistemlerinin bu saldırıya karşı dayanıklı tasarlanmalarına neden olmuştur. Bu kriptanaliz yöntemi açık metin ikilileri farkının bunlara karşılık gelen şifreli metin ikilileri üzerindeki etkisini kullanarak analiz yapar. Bu farklar olası anahtarlara ihtimal atamak ve ihtimali en yüksek anahtarları belirlemek için kullanılır. Aynı farka sahip olan bir çok açık metin ikilisi ve bunlara karşılık gelen şifreli metin ikililerini kullanır.

1.11.6.3 İmkansız Diferansiyeller (Impossible Differentials)

İmkansız diferansiyel kriptanaliz blok şifreleyiciler üzerindeki diferansiyel kriptanalizin bir çeşididir. Klasik diferansiyel kriptanaliz döngüler boyunca ilerletilen beklenen olasılığın üzerindeki farkları izlerken, imkansız diferansiyel kriptanaliz algoritmanın ara aşamalarında oluşan imkansız farklara odaklanır [33].

İmkansız diferansiyel kriptanaliz blok şifreleyicilerin döngüleri boyunca imkansız olayların bulunmasına bağlıdır. Bu sayede tüm olası gizli anahtarlar tahmin edilebilir ve belirlenen imkansız olayı oluşturan tüm anahtarlar, yanlış anahtarlar olarak işaretlenir. Çünkü, doğru anahtarlar böyle bir imkansız olayı hiçbir zaman oluşturmazlar.

İmkansız olayları oluşturmanın en verimli yöntemi Biham tarafından bulunan ortada kaybet (miss-in-the-middle) tekniğidir. Bu teknik sürekli gerçekleşen iki olay üzerine odaklanır. Çelişkiye (contradiction) neden olan bu iki olay birleştirilerek, imkansız olay oluşturulur [34].

1.11.6.4 Yüksek Dereceli Diferansiyel Kriptanaliz

Yüksek dereceli diferansiyel kriptanaliz mantık haritalama türevinin kavramını kullanır. $B:\{0,1\}^d \to \{0,1\}^d, a \in \{0,1\}^d$ ise, B'nin a noktasındaki türevi $\Delta_a B:\{0,1\}^d \to \{0,1\}^d$

$$\Delta_a B(x) \overset{def}{=} B(x \oplus a) \oplus B(x) \tag{1.6}$$

olarak verilir.

B'nin $(a_i \ldots a_1)$ noktalarındaki $i.$ türevi tekrarlamalı (recursive) olarak

$$\Delta^{(i)}_{a_i,\ldots,a_1} B \overset{def}{=} \Delta_{a_i}\left(\Delta^{(i-1)}_{a_i,\ldots,a_1} B\right) \tag{1.7}$$

ifadesi ile gösterilir. Cebirsel dereceyi ilgilendiren aşağıdaki ifadedeki eşitsizlik Lai [35] tarafından ispatlanmıştır.

$$\deg(\Delta_a B) \le \deg(B) - 1 \tag{1.8}$$

Eğer $\deg(B) = m$ ise, B' nin $(m+1).$ türevi 0' dır. Yüksek dereceli diferansiyel kriptanaliz, parça haritalaması düşük cebirsel dereceye sahip şifreleyicileri kırmak için bu gözlemden faydalanmıştır. Saldırı türevler ve belirli alt anahtarların bitlerinin birleştirilmesini içeren, bir dizi denklemlerin oluşturulması ve alt anahtar bitlerinin doğru değerlerinin belirlenmesi için yoğun arama tekniklerinin kullanılmasını içerir. Geleneksel diferansiyel kriptanaliz yöntemlerine karşı güvenli olan şifreleme algoritmaları, yüksek dereceli diferansiyel kriptanaliz yöntemine karşı zayıf kalabilir [29].

1.11.6.5 Kesilmiş Diferansiyeller (Truncated Differentials)

$2n$ bitlik bir Feistel şifreleyicisi üzerindeki geleneksel diferansiyel saldırıda, diferansiyel belirli bir sayıda döngüden sonra, şifreli metnin n bitini tahmin edebilmek için kullanılan bir araçtır. i döngü kadar şifrelemeden sonra a giriş farkı, b çıkış farkını oluşturuyorsa, (a,b) i döngülük diferansiyel olarak isimlendirilir.

Her zaman n bitlik değerin tamamının tahmin edilmesi gerekli değildir. Bazı durumlarda 1 bitlik değer bile yeterli olabilmektedir. n bit değerin sadece bir kısmını tahmin eden diferansiyel, kesilmiş diferansiyel olarak isimlendirilir [36].

Tanım 1.13. (a,b) i döngülük diferansiyel olsun. a' a'nın alt dizisi, b' b'nin alt dizisi olsun. (a',b') i döngülük kesilmiş diferansiyel olarak tanımlanır [36].

1.11.6.6 Boomerang Saldırısı

Diferansiyel saldırı gibi uyarlanabilir bir saldırı olan boomerang saldırısı Wagner tarafından sunulmuştur [37]. Bu saldırı klasik diferansiyel saldırıdan farklı olarak bütün şifreleyicinin tek bir diferansiyelce geçilmesini gerektirmez ve yüksek olasılıklı diferansiyelleri kullanır. Takip eden paragrafda saldırının ayrıntısı verilmiştir:

f bir blok şifreleyici olsun ve $f = f_1 \circ f_0$ olarak ifade edilebilsin. $(\delta,\delta^{'})$ diferansiyel karakteristiğinin f_0 fonksiyonuna göre oluşma olasılığı $\pi_{(\delta,\delta^{'})}$ olsun: $\delta = p_1 \oplus p_2$ ise, $\delta^{'} = f_0(p_1) \oplus f_0(p_2)$ 'dir. $c_1 = f(p_1)$ ve $c_2 = f(p_2)$ olsun. $\lambda = c_1 \oplus c_3 = c_2 \oplus c_4$ eşitliği, f_1 fonksiyonunun $(\lambda,\lambda^{'})$ diferansiyel karakteristiğinin $\pi_{(\lambda,\lambda^{'})}$ olasılıkla oluşması şartı ile tanımlansın. Bu durumda,

$$\lambda^{'} = f_1^{-1}(c_1) \oplus f_1^{-1}(c_3) = f_1^{-1}(c_2) \oplus f_1^{-1}(c_4)$$

olur.

Saldırı şu şekilde devam eder:

– $\delta = p_1 \oplus p_2$ şartı ile (p_1, p_2) açık metin çifti seçilir ve şifreleyerek (c_1, c_2) şifreli metin çifti bulunur.

– $(c_1 \oplus \lambda, c_2 \oplus \lambda)$ çiftini deşifreleyerek (p_3, p_4) açık metin çifti elde edilir ve $\lambda = p_3 \oplus p_4$ eşitliğini sağlayıp sağlamadığı kontrol edilir.

– Wagner terminolojisine göre sağ dörtlü (p_1, p_2, c_3, c_4) dörtlü değişkenler grubu (tuple) olarak tanımlanır. $\delta = p_3 \oplus p_4$ bumerangını gözlemleyebilme olasılığı,

$$\pi_b = \pi^2_{(\delta,\delta')} \times \pi^2_{(\lambda,\lambda')}$$

şeklinde ifade edilir.

1.11.6.7 İlişkili-anahtar (Related-key) Kriptanaliz

İlişkili anahtar kriptanaliz [38] [39] [40] [41] [42] x_1 ve x_2 açık metinlerinin (eşit olabilir) k_1 ve k_2 gibi farklı anahtarlarla şifrelenmesi durumundaki farkın ilerletilmesi zayıflığından yararlanır.

Gösterim

r döngülü ilişkili-anahtar diferansiyel (α, β, δ) üçlüsüdür.

α : Blok şifreleme algoritmasının giriş değerlerinin farkıdır.

β : $r.$ döngünün çıkışlarının farkıdır.

δ : Anahtarların farkıdır.

r döngülü ilişkili-anahtar diferansiyelin olasılığı $r.$ döngünün çıkış farkının β, giriş farkının α, anahtar farkının δ ve açık metin x_1 ve anahtar k_1 'in düzgün (uniform) olarak rastgele seçilmiş olması olasılığına eşittir.

İlişkili-anahtar diferansiyel saldırısı şu şekilde çalışır:

– Yüksek derecede olası $R-1$ döngü ilişkili-anahtar diferansiyel bulunur (R: Blok şifreleyicinin döngü sayısı).

– Rastgele bir açık metin x_1 değeri seçilir ve k_1 anahtarı ile şifrelenerek y_1 şifreli metin değeri elde edilir. $x_2 = x_1 + \alpha$ ifadesi ile hesaplanan x_2 değeri

$k_2 = k_1 + \delta$ ifadesi ile hesaplanan k_2 değeri ile şifrelenerek y_2 değeri elde edilir.

– Tüm olası $\left(k_1^R, k_2^R\right)$ son döngü anahtar çiftleri $d_{k_1^R}(y_1)$ ve $d_{k_2^R}(y_2)$ değerleri arasındaki fark β olması koşulu altında bulunur. $d_k(y)$ değeri, y giriş değeri ve k anahtarı için şifreleme algoritmasının ilk döngüsünün çıkış değeridir. Daha önce hesaplanan anahtar çiftlerinden birine karşılık gelen her bir sayaca bir eklenir.

– Önceki iki aşama bir veya daha fazla son döngü anahtarı diğerlerinden önemli ölçüde fazla sayılana kadar tekrarlanır. Bu anahtarların doğru olup olmadığı kontrol edilir.

1.11.6.8 Toplam (Integral) Saldırı

Toplam kriptanaliz [43] [44] birçok değerin toplamlarını analiz eder. Bu saldırının en eski ve genel hali kare (Square) [45] [46] saldırısıdır. Saldırının temelindeki kavram toplamadır. Bir toplama saldırısında belirli sayıda döngüden sonra toplamların değerleri tahmin edilmeye çalışılır.

1.11.7 Cebirsel Saldırılar

Shannon bir blok şifreleyicininin kırılması ile ilgili şu ifadede bulunmuştur [10]:

"Bir blok şifreleyicinin kırılması için, karmaşık tipte çok sayıda bilinmeyenleri olan eşzamanlı eşitlikler sistemini çözecek kadar güç gereklidir."

1.11.7.1 Interpolasyon Saldırısı

Blok şifreleyicileri kırmak için kullanılan tamamen cebirsel bir yöntem olan interpolasyon saldırısı Knudsen ve Jakobsen tarafından sunulmuştur [47]. Bu saldırı Lagrange formülünü temel alır.

Tanım 1.14. (Lagrange İnterpolasyonu). $p(x)$ $(n-1)$. dereceden tek bir polinom olsun. $p(x_i)=y_i$ şartını sağlayan n adet (x_i, y_i) açık metin şifreli metin çifti için $p(x)$ polinomu (1.9)'daki ifade ile gösterilebilir.

$$p(x)=\sum_{i=1}^{n} y_i \prod_{\substack{1\le j\le n \\ j\neq i}} \frac{x-x_j}{x_i-x_j} \tag{1.9}$$

1.11.7.2 Curtois-Pieprzyk Saldırısı

Saldırıdaki ilk aşama, verilen blok şifreleyiciyi sistem eşitlikleri ile ifade etmektir. Bu bilgilerin toplanması durumunda tüm blok şifreleyiciyi matematiksel olarak tanımlayan büyük bir sisteme sahip olunur. Eğer bu sistemi yoğun anahtar aramadan daha hızlı bir şekilde çözmek mümkün olursa, şifreleyici kırılmış demektir. Ferguson, Schroeppel ve Whiting Rijndael şifreleyicisini 2^{50} terimden oluşan tek bir eşitlikle ifade etmiştir [48].

Curtois-Pieprzyk, Rijndael ve Serpent şifreleyicilerinin S-kutularını $GF(2)$'de aşırı tanımlı (overdefined) cebirsel eşitlikler sistemi olarak ifade etmiştir [49].

1.12 Güvenlik Modeli

Güvenlik modelini açıkça tanımlayabilmek için ilkellerin güvenliğini ispatlamak gereklidir. Blok şifreleyicinin güvenliğinin sezgisel tanımı Daemen ve Rijmen'in *K-güvenlik* kavramıdır [50]. Matematiksel açıdan bu tanımın gerçeklenmesi çok zordur.

Tanım 1.15. Bir blok şifreleyici tüm olası saldırı stratejilerine karşı, aynı tipteki blok şifreleyicilerinin çoğu gibi aynı hesaplama gücü ve bellek gereksinimlerine sahipse *K-güvenli* demektir.

1.12.1 Mükemmel Güvenlik

Mükemmel güvenlik modeli Shannon tarafından tanıtılmıştır [10]. Bu modelde saldırganın sonsuz bir güce sahip olduğu varsayılır. Fakat sadece şifreli metin saldırıları ile sınırlıdır.

Tanım 1.16. Bir blok şifreleyici açık metin X ve şifreli metin Y olmak üzere istatistiksel olarak iki bağımsız olasılık dağılımı olarak modelleyelim. $Y = f_K(X)$ şartı ile f kriptosistemi tüm x ve y değerleri için iki olasılık dağılımı

$$\Pr_{X|Y}[X = x \mid Y = y] = \Pr_{X}[X = x] \tag{1.10}$$

ifadesini sağlıyorsa mükemmel güvenlik özelliğine sahiptir.

Mükemmel güvenlik X açık metninin olasılık dağılımının, Y şifreli metni elde edildikten sonraki açık metin olasılık dağılımına eşit olmasıdır. Yani saldırganın şifreli metni elde etmesine rağmen açık metin hakkında önceden bildiğinden farklı birşey elde edememesidir.

Tanım 1.17. (Rastgele bir değişkenin entropisi): X sonlu küme $\mathcal{X}$ üzerinde tanımlı ayrık rastgele bir değişken olsun. Bu durumda X 'in entropisi aşağıdaki gibi tanımlanır:

$$H(X) = -\sum_{\substack{x \in \mathrm{X} \\ \Pr[X=x] \neq 0}} \Pr[X = x] \log_2 \Pr[X = x]. \tag{1.11}$$

Teorem 1.4. (Shannon): X ve Y açık metin ve şifreli metin olasılık dağılımlarına göre dağılan iki rastgele değişken olsun. Mükemmel güvenlik $H(X) = H(X|Y)$ ifadesi ile tanımlanır.

1.12.2 Sınırlandırılmış Saldırganlara Karşı Güvenlik

Mükemmel güvenlik çok güçlü bir güvenlik modeli olmakla birlikte gerçekleştirilmesi makul değildir. Bu nedenle modern kriptografide saldırganın gücünün sınırlı olduğu düşünülür.

1.12.3 Sınırlı Saklama Modeli

Maurer tarafından sunulan bu modelde saldırganın sonsuz hesaplama gücüne sahip olmasına rağmen saklama kapasitesinin sınırlı olduğu varsayılmıştır [51] [52].

1.12.4 Luby-Rackoff Modeli

Blok şifreleyiciler için ispatlanabilir güvenlik alanı ilk olarak Luby ve Rackoff tarafından sunulmuştur [53]. Luby ve Rackoff sözde rastsal permütasyon fonksiyonu içeren üç döngülü Feistel şifreleyicisinin sözde rastsal pseudorandom permütasyonla sonuçlandığını ispatlamıştır.

BÖLÜM 2

2 MATEMATİKSEL ALTYAPI VE GEREKLİLİKLER

Cisim toplama, çıkarma, çarpma ve bölme işlemlerinin gerçekleştirilebildiği bir alandır. Diğer bir deyişle, bir F kümesi aşağıdaki özellikleri sağlıyorsa cisimdir:

- F kümesi birim eleman 0 ile birlikte toplam işlemine göre bir abelian gruptur.
- F kümesinin sıfırdan farklı elemanları çarpma işlemine göre abelian grup oluşturur.
- Çarpmanın toplama işlemi üzerine dağılma özelliği mevcuttur ($a \bullet (b + c) = a \bullet b + a \bullet c$).

Cisim kendisini oluşturan kümenin sonlu veya sonsuz olmasına göre sonlu veya sonsuz cisim olarak isimlendirilir. Gerçel sayılar, rasyonel sayılar, karmaşık sayılar sonsuz cisimlere örnek olarak verilebilir.

2.1 Euclidean Alanı (Domain)

Euclid 300 yılında iki tamsayının en büyük ortak bölenini bulan basit bir yordam bulmuştur. Euclid algoritması geçen zaman içinde gelişmiş ve yeni sürüm tamsayıların, polinomların, hatta euclid alanından alınan eleman çiftlerinin en büyük ortak bölenini bulur hale gelmiştir.

Tanım 2.1. İntegral alanı (domain) toplama ve çıkarma işlemleri ile birlikte aşağıdaki özellikleri sağlayan bir D kümesidir:

- D kümesinin elemanları birim elemanla birlikte bir abelian grup oluşturur.
- Çarpma işleminin birim elemanı 1'dir, birleşme (associative) ve değişme özelliği (commutative) vardır.
- Götürme kuralı çalışır. Eğer $ab = ac$ ve $a \neq 0$ ise $b = c$ 'dir.
- Çarpmanın toplama işlemi üzerine dağılma özelliği mevcuttur. Eğer $a, b, c \in D$ ise $a \bullet (b + c) = a \bullet b + a \bullet c$ 'dir.

Euclidean alanı bir integral alan olmakla birlikte, büyüklükle ilgili ek bir özelliği daha vardır. $g(a)$ ile gösterilen a'nın büyüklüğü, $a \neq 0$ şartı için aşağıdaki özelliği sağlayan artı (positive) bir tamsayıdır: Euclidean alana verilebilecek örnekler aşağıda sıralanmıştır:

- $g(n) = |n|$ olan tamsayılar.
- $g(f(x)) = derece(f)$ eşitliğini sağlayan polinomlar.
- Gaussian tamsayıları: $\{a + b\sqrt{-1} : a, b \text{ tamsayılar } \}, \mathrm{g}(a + bi) = a^2 + b^2$

Tanım 2.2. Verilen bir $a \in D$ integral alanında $a \neq 0$ olan bir eleman diğer bir b elemanını, $b = c \bullet a$ koşulunu sağlayan bir c elemanı varsa böler ve $a|b$ şeklinde gösterilir.

Tanım 2.3. Eğer $a|b_i, i = 1,2,\ldots,n$ ise, a elemanı b_i'nin ortak böleni olarak isimlendirilir. Eğer d $\{b_1, b_2, \ldots, b_n\}$' nin ortak böleni ve $\{b_1, b_2, \ldots, b_n\}$'nin diğer tüm ortak bölenleri d'yi bölüyorsa, d $\{b_1, b_2, \ldots, b_n\}$'nin en büyük ortak böleni olarak isimlendirilir ve $d = \gcd(b_1, b_2, \ldots, b_n)$ şeklinde gösterilir.

Teorem 2.1. Eğer $B = (b_1, b_2, \ldots, b_n)$ Euclidean alanı D'nin herhangi bir alt kümesi ise, B, b_k'nin $\sum \lambda_k b_k$ doğrusal kombinasyonu olarak ifade edilebilecek olan (d) en büyük ortak bölenine sahiptir.

İspat: $S=\left\{\sum_{k=1}^{n}\mu_k b_k : \mu_k \in D\right\}$ ve d, S kümesinin $g(d)$ değerini mümkün olan en küçük değer yapacak olan sıfırdan farklı bir elemanı olsun. d b_k değerlerinin doğrusal kombinasyonu olarak ifade edilebilir ve b_k değerinin en büyük ortak bölenidir.

Öncelikle $d|b_i, i=1,2,\ldots,n$ eşitliğini ispatlayalım. $d\neq 0$ olduğuna göre, $r_i=0$ ve $g(r_i)<g(d)$ koşulları dahilinde $b_i=q_i d+r_i$ şeklinde ifade edilebilir. Buradan elde edilen $r_i=b_i-q_i d$ değeri S kümesinin bir elemanıdır. $r_i=0$ için $b_i=q_i d$ olur ve d b_i'nin ortak böleni olur.

e b_i'nin diğer bir ortak böleni ise $b_i=q_i' e, i=1,2,\ldots,n$ olur. d b_i'nin doğrusal kombinasyonu olduğuna göre, $d=\sum\lambda_i b_i=e\sum\lambda_i q_i'$ olur. Böylelikle d'nin $\{b_1,b_2,\ldots,b_n\}$'nin en büyük ortak böleni olduğunu ispatlamış olduk.

Teorem 2.2. s,t,r**'**nin tüm elemanları için $\gcd(s,t)=\gcd(s,t-rs)$ 'dir.

İspat: Eğer d hem t hem de s değerlerini bölerse $t-rs$ değerini de böler. Yani s ve t'nin ortak bölenleri, s ve $t-rs$ değerlerinin de ortak bölenidir.

D Euclidean alan olsun.

$a,b\in D,\ a>0,\ b>0$ ve $g(a)\geq g(b)$ için,

$d=\gcd(a,b)$ değerini bulmak istiyoruz.

$r_{-1}=a$, $r_0=b$, $r_{-1}>0$

$r_{i-2}=q_i r_{i-1}+r_i$, $\qquad g(r_i)>g(r_{i-1})$

r_i değeri, r_{i-2} değerinin r_{i-1} değerine bölünmesinden kalan değerdir.

r_i değeri sıfır olana kadar yukarıdaki tekrarlamalı ifade çalışır.

Eğer $r_{n+1}=0$ ise $r_n=\gcd(a,b)$ değeridir.

Algoritma 2.1. Euclid Algoritması

Örnek 2.1. $\gcd(84,54)$ değerini Euclid algoritması ile hesaplayalım.

$84 = 1.54 + 30 \qquad q_1 = 1, \qquad r_1 = 30$

$54 = 1.30 + 24 \qquad q_2 = 1, \qquad r_2 = 24$

$30 = 1.24 + 6 \qquad q_3 = 1, \qquad r_3 = 6$

$24 = 4.6 + 0 \qquad q_4 = 4, \qquad r_4 = 0$

$r_4 = 0$ olduğuna göre, $\gcd(84,54) = r_3 = 6$ olur.

D Euclidean domain.

$r_i, q_i, s_i, t_i \in D$

$s_{-1} = 1, \qquad s_0 = 0$
$t_{-1} = 0, \qquad t_{-1} = 1$

$i = -1,0,1,\ldots,n+1$

$t_i = t_{i-2} - q_i t_{i-1}$

$s_i = s_{i-2} - q_i s_{i-1}$

$r_i = as_i + bt_i$

Algoritma $r_{n+1} = 0$ olana kadar devam eder. Bu noktada

$r_n = \gcd(a,b)$ değeridir.

Algoritma 2.2. Genişletilmiş Euclid Algoritması

Tanım 2.4. D bir Euclidean alanı olsun. Birim $u \in D$ 1 değerinin herhangi bir bölenidir. u ancak ve ancak $uv = 1$ eşitliğini sağlayan bir $v \in D$ değeri varsa birimdir.

Tanım 2.5. Bir u birim değeri için $a = ub$ ise iki eleman $a, b \in D$ ilişkili (associative) olarak isimlendirilir.

Tanım 2.6. Euclidean alandaki iki a ve b değeri eğer en büyük ortak bölenleri $(\gcd = 1)$ 1 ise a ve b değerleri aralarında asaldır denir.

Önerme 2.1. Eğer a ve b değerleri aralarında asal ise $as+bt=1$ eşitliğini veren s ve t değerleri mevcuttur.

Önerme 2.2. Eğer $p \in D$ asal sayı ve p a değerini bölmüyorsa, p ve a aralarında asaldır.

Önerme 2.3. Eğer p asal sayı ve $p|ab$ ise $p|a$ veya $p|b$ veya hem a hem de b'yi böler.

İspat: Eğer p a'yı bölmüyorsa, $ps+at=1$ eşitliğini sağlayan s ve t değerleri mevcuttur. Bu ifadeyi b değeri ile çarptığımızda, $b=pbs+abt$ ifadesi elde edilir. $p|ab$ 'yi böldüğüne göre $p|b$'yi böler.

Önerme 2.4. Euclidean alanında eğer a değeri b değerini bölüyorsa $g(a)<g(b)$ 'dir.

Teorem 2.3. $b \in D$ birim elemandan farklı olsun. $b=p_1 p_2 \ldots p_r$ şeklinde asal sayıların çarpımı olarak yazılabilir. $b=q_1 q_2 \ldots q_s$ asal sayıların çarpımı olarak diğer bir şekilde yazılabilir. Burada $r=s$ ve uygun bir sıralamadan sonra $i=1,2,\ldots,s$ için p_i, q_i asosiyatiftir.

Önerme 2.5. (Gauss): Bir denklik bağıntısı a modül m'ye göre b'ye eşitse $m|(a-b)$'yi böler.

$$a \equiv b \pmod{m} \quad \text{eğer} \quad m|(a-b) \tag{2.1}$$

Teorem 2.4. Eğer p asal sayı ise, $D \bmod p$ bir cisimdir.

2.2 Sonlu Cisim Teorisi

p asal sayı, n pozitif tamsayı olma şartı ile $m=p^n$ ise m. dereceden bir sonlu cisim vardır demektir. p^n elemanlı bir sonlu cisim Galois alanı olarak tanımlanır ve $GF(p^n)$ ile gösterilir. p asal sayısı için $GF(p)$ Z_p'ye eşittir. $GF(p^n)$ sonlu cismi, $n>1$ için $GF(p)$' den $GF(p)$ uzayındaki n. dereceden indirgenemez polinom kullanılarak elde edilir [54].

Tanım 2.7. Cisim toplama ve çarpma işlemini göre aşağıdaki özellikleri sağlayan bir Z [3] kümesidir [54]:

i. Toplamada kapalılık özelliği;

$$a,b \in Z_m \rightarrow a+b \in Z_m$$

ii. Çarpmada kapalılık özelliği;

$$a,b \in Z_m \rightarrow a.b \in Z_m$$

iii. Toplamada değişme özelliği;

$$a,b \in Z_m \rightarrow a+b = b+a$$

iv. Çarpmada değişme özelliği;

$$a,b \in Z_m \rightarrow a.b = b.a$$

v. Toplamada geçişme özelliği;

$$a,b,c \in Z_m \rightarrow (a+b)+c = a+(b+c)$$

vi. Çarpmada geçişme özelliği;

$$a,b,c \in Z_m \rightarrow (a.b).c = a.(b.c)$$

vii. Çarpmada dağılma özelliği;

$$a,b,c \in Z_m \rightarrow a.(b+c) = a.b + a.c$$

Toplama işleminin birim elemanı 0 ve çarpma işlemininin birim elemanı 1 ile yapılan aşağıdaki işlemlerin sonucu Z_m kümesinde olmalıdır.

viii. $a+0=a, \forall a \in Z_m$

ix. $a.1=a$ ve $a.0=0, \forall a \in Z_m$

x. $a \in Z_m$ için a ' nın toplamaya göre tersi $(m-a)$'dır.

xi. $a \in Z_m$ için a 'nın çarpmaya göre tersi a^{-1} 'dir ve $a^{-1}a=1$ olmalıdır.

[3] Z bir cisimdir.

Tanım 2.8. *i, ii, v, vi* ve *vii*. özellikleri sağlayan Z_m kümesi toplama veya çarpma işlemine göre gruptur denir. Bu özelliklerle birlikte *iii* ve *iv*. özellikleri de sağlıyorsa abelyan gruptur denir.

Tanım 2.9. *i*'den *ix*'e kadar tüm özellikleri sağlayan Z_m kümesi halka olarak simlendirilir. Tam sayılar ve reel sayılar halkaya örnek olarak gösterilebilir.

Örnek 2.2. $m = 4$ için Z_4 bir cisim midir inceleyelelim. Z_4 0, 1, 2 ve 3 elemanlarından oluşur. Toplama ve çarpma işlemine göre elde edilen tablolar Tablo 2.1 'de gösterilmiştir.

Tablo 2.1. Z_4 Cisminde Toplama ve Çarpma İşlemleri

+	**0**	**1**	**2**	**3**
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

.	**0**	**1**	**2**	**3**
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

Z_4 kümesi *i, ii, v, vi* ve *vii*. özellikleri sağladığı için gruptur. Bunara ek olarak *iii* ve *iv*. özelliği de sağlayan Z_4 kümesi abelyan grup olarak isimlendirilir. *i*'den *ix*'e kadar tüm özellikleri sağladığı için halkadır. Fakat tüm bu özellikleri sağlamasına rağmen çarpma işlemine göre 2 elemanın tersi olmadığı için cisim değildir.

2.2.1 Sonlu Cisimde Polinomlar

$GF(p)$ uzayında *n.* dereceden bir polinom şu şekilde ifade edilir [54]:

$a_i \in GF(p), a_n \neq 0$ için $P(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 = \sum_{i=0}^{n} a_i x^i$. (2.2)

Tanım 2.10. Z_m bir cisim olmak üzere küme $Z_m(x) = \left\{ \sum_{i=0}^{n} a_i x^i : a_i \in Z_m, n \geq 0 \right\}$, Z_m üzerine bir polinom halka olarak isimlendirilir. $Z_m(x)$'in bir elemanı Z_m üzerine polinom olarak isimlendirilir. Artı dereceli bir polinom $f(x) = \sum_{i=0}^{n} a_i x^i$ için $derece(g(x)) < derece(f(x))$, $derece(h(x)) < derece(f(x))$ ve $f(x) = g(x)h(x)$ şartlarını

sağlayacak şekilde iki polinom varsa $f(x)$ polinomu Z_m üzerine indirgenebilir, aksi takdirde indirgenemez polinom olarak tanımlanabilir.

Teorem 2.5. $f(x)$ fonksiyonu Z_m cisminde derecesi birden büyük bir polinom olsun. $\frac{Z_m(x)}{f(x)}$ polinomu, $f(x)$ polinomu indirgenemez ise cisimdir.

Örnek 2.3. $\frac{Z_2(x)}{x^2+x+1}$ halkasının çarpma ve toplama işlemlerini kullanarak sonlu cisim olduğunu belirleyelim.

Tablo 2.2. $\frac{Z_2(x)}{x^2+x+1}$ Halkasında Toplama Çarpma İşlemleri

+	0	1	x	1+x
0	0	1	x	1+x
1	1	0	1+x	x
x	x	1+x	0	1
1+x	1+x	1	1	0

.	0	1	x	1+x
0	0	0	0	0
1	0	1	x	1+x
x	0	x	1+x	1
1+x	0	1+x	1	x

x^2+x+1 polinomu indirgenemez polinom olduğu için Teorem 2.5 gereği olarak $\frac{Z_2(x)}{x^2+x+1}$ polinom halkası sonlu bir cisimdir. Tablo 2.2'deki toplama ve çarpma işlemlerinin sonuçları bu gerçeği doğrulamaktadır.

2.2.2 Sonlu Cisimde İşlemler

2.2.2.1 Toplama

Polinomsal gösterimde, aynı cisim içerisinde bulunan iki elemanın toplanması ya da çıkarılması işlemi, standart polinomların toplama ve çıkarma işlemi gibidir. Sonlu cisim aritmetiğinde elemanlar $\{0,1\}$ katsayılarına sahip polinomlar olarak temsil edildiğinden toplama işlemi katsayılarının basitçe modül 2 aritmetiğine göre toplamıdır.

Örnek 2.4. $a=(01110111)$ ve $b=(10110101)$ olsun. O zaman $a+b=11000010$ olacaktır. Polinomsal olarak göstermek gerekirse $a=x^6+x^5+x^4+x^2+x+1$ ve

$b = x^7 + x^5 + x^4 + x^2 + 1$ olarak ifade edilir. Buradan $a + b = x^7 + x^6 + x$ olarak bulunacaktır.

2.2.2.2 Çarpma

Sonlu cisim aritmetiğinde çarpma polinomların birbirleri ile aritmetik çarpımı şeklindedir. Fakat çarpma sonucunda doğal olarak sonlu cismin derecesinden daha yüksek dereceli terimler oluşabilir. Böyle bir durumda yüksek dereceli elemanlar sonlu cismin derecesinden küçük olacak şekilde cismi oluşturan indirgenemez polinom aracılığı ile indirgenir. Bu işlem indirgenemez polinoma göre indirgeme ya da mod alma işlemidir.

Örnek 2.5. $a = 1101$, $b = 0101$ ve indirgenemez polinom $x^4 + x + 1$ seçilsin. Bu değerlere göre

$$\begin{aligned} a.b &= (x^3 + x^2 + 1).(x^2 + 1) \\ &= (x^5 + x^4 + x^2 + x^3 + x^2 + 1) \\ &= x^5 + x^4 + x^3 + 1 \\ &= x^2 + x + x + 1 + x^3 + 1 \\ &= x^3 + x^2 \end{aligned}$$

$$\begin{aligned} x^1 &= x \\ &\vdots \\ x^4 &= x + 1 \\ x^5 &= x^2 + x \\ &\vdots \\ x^{15} &= 1 \end{aligned}$$

şeklinde olacaktır.

2.2.2.3 Bölme

Sonlu cisim aritmetiğinde bölme normal aritmetikteki bölme işlemi gibi gerçekleşir. Fakat gerçekleşen işlemlerdeki aritmetik sonlu cisim aritmetiğidir.

Örnek 2.6. $Z_{13}[x]$ sonlu cisminde $a(x) = x^8 + x^6 + 10x^4 + 10x^3 + 8x^2 + 2x + 8$ ve $b(x) = 3x^6 + 5x^4 + 9x^2 + 4x + 8$ olsun. (2.3) ifadesini sağlayan $q(x)$ ve $r(x)$ polinomlarını bulmaya çalışalım.

$$a(x) = q(x)b(x) + r(x); \qquad \deg(r) < \deg(b) \tag{2.3}$$

Bu polinomları bulmak için polinomların bölünmesi işlemi kullanılır.

```
1 0 1 0 10 10 8 2 8   | 3 0 5 0 9 4 8
1 0 6 0  3  10 7      |_______________
    8 0  7  0 1  2 8  | 9 0 7
    8 0  9  0 11 2 4  |
       11  0  3 0 4
```

Buradan $q(x)=9x^2+7$, $r(x)=11x^4+3x^2+4$ olarak bulunur.

2.2.3 Ters Alma

n bit iki polinomun çarpımının kalanı, seçilen indirgenemez polinoma göre 1 ise o zaman iki polinom, birbirinin seçilen indirgenemez polinoma göre tersidir. İndirgenemez bir polinoma göre ters alma işlemi için iki yöntem önerilebilir. Bu yöntemlerden ilki $GF(2^n)$ için tablo oluşturmaktır. Eğer n değeri küçük bir değer ise bu yöntem etkili olabilir.

Teorem 2.6. Eleman sayısı q, bir asal sayının üssüdür: p asal sayı ise , $q=p^m$'dir.

Teorem 2.7. Eğer t α'nın derecesi ise, $t|(q-1)$'i böler (q eleman sayısı).

Önerme 2.6. $p(x)$ F alanında m. dereceden katsayıları olan bir polinom ise, $p(x)=0$ ifadesinin F alanında m farklı çözümü vardır.

Önerme 2.7. Eğer $derece(\alpha)=t$ ise, $derece(\alpha^i)=\dfrac{t}{\gcd(i,t)}$ olur.

Teorem 2.8. Eğer n artı bir tamsayı ise, $\sum_{d|n}\phi(d)=n$ 'dir.

Önerme 2.8. $\gcd(m,n)=1$ koşuluyla $derece(\alpha)=m$, $derece(\beta)=n$ ise, $derece(\alpha\beta)=m+n$ olur.

2.3 S-Kutuları

Yer değiştirme permütasyon ağlarını temel alan gizli anahtarlı kriptosistemlerde, kriptosistemin gücü doğrudan kullanılan S-kutusunun gücüne bağlıdır [55]. Biham ve

Shamir zayıf bir S-kutusu kullanılması durumunda diferansiyel kriptanaliz ile DES şifreleme algoritmasının kırılabileceğini göstermişlerdir [18].

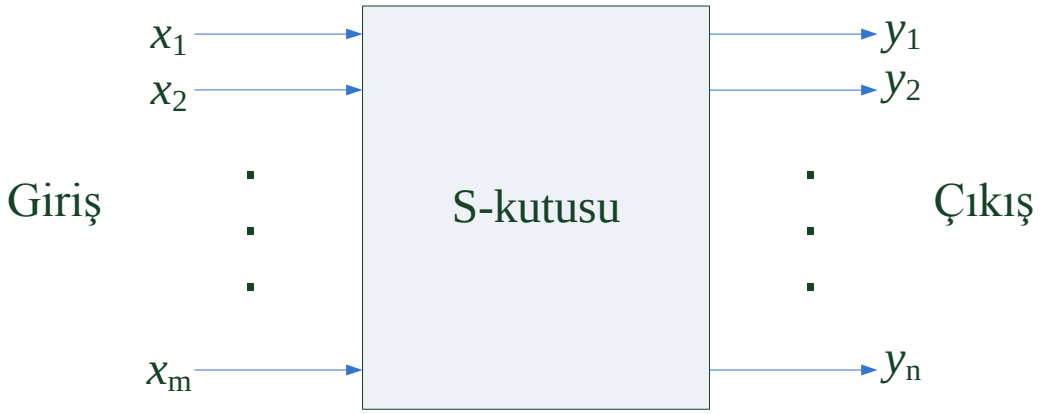
S-kutusu iki farklı bakış açısına sahiptir. Bunlar:

– *Durağan görünüm*: S-kutusunun giriş değerlerinin değişmediği durumu tanımlayan görünümdür.

– *Dinamik görünüm*: S-kutusunun giriş değerlerinin değiştiği durumu tanımlar.

$X=(x_1,\ldots,x_m)$ giriş değerleri ve $Y=(y_1,\ldots,y_n)$ çıkış değerleri ile S-kutusunun durağan görünümü Şekil 2.1'de görülmektedir.

S-kutusunun dinamik görünümü üzerinde düşünürken, Şekil 2.2'den yararlanmakta fayda vardır. Şekil 2.2'de, $X=(x_1,\ldots,x_m)$ vektörünün değerleri S-kutusunun güncel giriş değerleridir ve delta[4] S-kutusunun durumunu gösterir. Buradaki Δx_i ve Δy_i, giriş ve çıkış değerlerindeki göreli değişimlerdir. X vektörünün güncel durumu bilinmemekte olup, Δx_i ve Δy_i arasında herhangi bir ilişki olduğu varsayılır [55].



Şekil 2.1. $m\times n$ S-Kutusu Durağan Görünüm

Tanım 2.11. Olası değerleri $x_1,\ldots,x_n$ olan rastgele x değişkeni için, x değerindeki belirsizlik (uncertainty) $H(x)=\sum_{i=1}^{n}P(x_i)\log_2\left(\frac{1}{P(x_i)}\right)$ ifadesiyle tanımlanır. İki X ve Y

[4] delta: S-kutusunun dinamik görünümündeki değişim gösteren S-kutusudur.

rastgele değişkeni arasındaki karşılıklı bilgi $I(X:Y)=H(X)-H(X\mid Y)$ olarak tanımlanır.

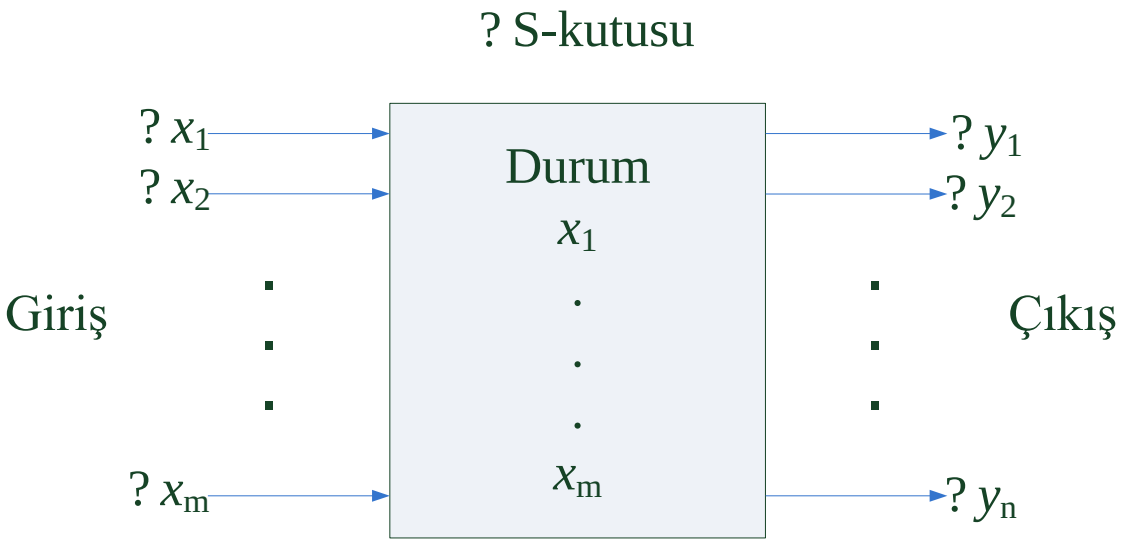
Bilgi teorisi temelinden bakıldığında amaç, S-kutusunun bilinen giriş ve çıkış değerlerinden bilinmeyen giriş ve çıkış değerleriyle ilgili edinilen bilgiyi en aza indirgemektir [55].

Teorem 2.9. S-kutusunun giriş ve çıkış değerleriyle ilgili kısmi bilgi bilinmeyen çıkış değerlerinin belirsizliğini azaltmaz. Matematiksel olarak ifade etmek gerekirse,

$\forall i,k,l,s,p \mid 1\le i\le n, 1\le k\le m-1, 1\le j_1,\ldots,j_k\le m, 1\le s\le n-1, 1\le (l_1,\ldots,l_s,p)\le n, l_p\neq i$ için

$$H(y_i \mid x_{j_1},\ldots,x_{j_k},y_{l_1},\ldots,y_{l_s}) = H(y_i) \tag{2.4}$$

olur.



Şekil 2.2. $m\times n$ S-Kutusu Dinamik Görünüm

Teorem 2.10. S-kutusunun giriş ve çıkış değerleriyle ilgili kısmi bilgi bilinmeyen giriş değerindeki belirsizliği azaltmaz. Matematiksel olarak ifade etmek gerekirse,

$\forall i,k,l,s,p \mid 1\le i\le m, 1\le k\le m-1, 1\le (j_1,\ldots,j_k,p)\le m, 1\le s\le n-1, 1\le l_1,\ldots,l_s\le n, j_p\neq i$ için

$$H(x_i \mid x_{j_1},\ldots,x_{j_k},y_{l_1},\ldots,y_{l_s}) = H(x_i) \tag{2.5}$$

olur.

Teorem 2.11. Verideki bilinmezlik S-kutusundan geçirildiğinde olası en küçük değer kadar azalır. Yani S-kutusunun giriş değerlerindeki bilinmezlik çıkış değerlerindeki bilinmezlik kadardır. Öyleki çıkış değerlerindeki bilinmezlik, çıkış bitleri sayısı için en fazla olacaktır.

S-kutusunun dinamik özellikleri, statik özellikleri ile aynı olmakla birlikte, giriş ve çıkış değerlerindeki değişimlerle ilgilidir.

Teorem 2.12. Giriş ve çıkış değerlerindeki değişimlerle ilgili kısmi bir bilgi, bilinmeyen çıkış değerlerindeki bilinmezliği azaltmaz. Matematiksel olarak ifade etmek gerekirse,

$$H\left(\Delta y_i \mid \Delta x_{j_1},\ldots,\Delta x_{j_k},\Delta y_{l_1},\ldots,\Delta y_{l_s}\right)=H\left(\Delta y_i\right) \tag{2.6}$$

olur.

Teorem 2.13. Giriş ve çıkış değerlerindeki değişimlerle ilgili kısmi bir bilgi, bilinmeyen giriş değerlerindeki bilinmezliği azaltmaz. Matematiksel olarak ifade etmek gerekirse,

$$H\left(\Delta x_i \mid \Delta x_{j_1},\ldots,\Delta x_{j_k},\Delta y_{l_1},\ldots,\Delta y_{l_s}\right)=H\left(\Delta x_i\right) \tag{2.7}$$

olur.

Teorem 2.14. Veri değerlerindeki değişimlerdeki bilinmezlik, S-kutusundan geçince mümkün olan en küçük değerde azalır. Yani S-kutusunun çıkış değerlerindeki değişimlerin bilinmezliği, $m>n$ olduğu için, çıkış değerlerindeki bilinmezlik çıkış bitleri sayısı için en fazla olacak şekilde, S-kutusunun giriş bitlerindeki bilinmezlik kadardır. Matematiksel olarak ifade etmek gerekirse,

$$H\left(\Delta Y\right)=\begin{Bmatrix} H\left(\Delta X\right) & if\ H\left(\Delta X\right)\le n \\ n & f\ H\left(\Delta X\right)>n \end{Bmatrix} \tag{2.8}$$

olur.

S-kutusu sabit tasarım kriterleri aşağıda verilmiştir [55]:

– *Giriş çıkış bağımsızlığı*: Giriş değerlerinin bilinmesi, çıkış değerlerinin bilinmezliğini değiştirmez.

- *Çıkış giriş bağımsızlığı*: Bazı çıkış değerlerinin bilinmesi, giriş değerlerinin bilinmezliğini değiştirmez.
- *Çıkış çıkış bağımsızlığı*: Çıkış bitleri ile ilgili kısmi bir bilgi, diğer bilinmeyen çıkış bitlerinin bilinmezliğini değiştirmez. Matematiksel olarak,

$$\forall y_j, a_k \left| 1 \le j,k \le n, \left(a_k, y_j\right) \in \{0,1\} \right. \text{ için,}$$

$$Prob\left(y_j \left| a_1 y_1, \ldots, a_n y_n \right.\right) = Prob\left(y_i\right) \tag{2.9}$$

şeklinde ifade edilir.

- *Doğrusal olmama*: S-kutusunun en önemli özelliğidir. S-kutusunun doğrusal eşitlikler halinde ifade edilmesini engelleyen özelliktir. Bu doğrusal eşitlikler, S-kutusunun bulunduğu kriptosistemleri kırmada kullanılır. Bu nedenle doğrusal olmama özelliği yüksek olan S-kutuları kullanılmalıdır.
- *Bilgi bütünlüğü*: Kam ve Davida [56]'daki çalışmalarında bilgi bütünlüğünü,

  "Her olası giriş değeri için her çıkış biti, giriş bitlerinin sadece uygun bir bölümüne değil, tüm olası giriş değerlerine bağımlıdır."

  olarak tanımlamıştır.
- *Tersi alınabilir*: Bu kriter $n \times n$ S-kutularının arzulanan bir özelliğidir. Bir S-kutusunun giriş ve çıkış değerleri arasında birebir haritalama mevcutsa tersi alınabilir. Eğer bir S-kutusunun tersi alınamıyorsa giriş değerlerinden daha az sayıda çıkış değerleri vardır. Böyle bir durumda çıkış değerlerinde giriş değerlerine göre daha az bilinmezlik olur.

2.3.1 S-Kutularının Özellikleri

S-kutularının doyurması gereken bazı kriptografik özellikler vardır. Bunlar sırasıyla doğrusal olmama, doğrusal saldırılar için önemli olan LAT (Linear Approximation Table-Doğrusal Yaklaşım Tablosu), diferansiyel saldırılar için önemli olan DDT (Difference Distribution Table-Fark Dağılım Tablosu-XOR Tablosu), bütünlük (completeness), çığ (avalanche), katı çığ (strict avalanche) gibi verilebilir [57].

2.3.1.1 S-Kutularının :Doğrusal Yaklaşım Tablosu (Linear Approximation Table-LAT)

Doğrusal yaklaşım tablosu (Lineer Approximation Table) (LAT) [58], doğrusal kriptanalize karşı S-kutularının güvenliğinin ölçülmesinde çok önemli bir değerlendirme kriteridir. Verilen bir S-kutusu $S: Z_2^n \to Z_2^n$'nin w. satır ve c. kolonun $LAT(w,c)$ değeri (2.10)' daki gibi tanımlanabilir [57].

(2.10)' daki ifadede P giriş bitlerini ve $S(P)$ S-kutusunun çıkış bitlerini, $\bullet$ nokta çarpımı göstermektedir. $n \times n$ boyutunda bir S-kutusu için doğrusal yaklaşım tablosu $2^n \times 2^n$ matrise denk düşer [57].

$$LAT(w,c) = \#\left\{P \in Z_2^n \middle| w \bullet P = c \bullet S(P)\right\} - 2^{n-1} \tag{2.10}$$

En büyük LAT girişi $\left(\max_{w,c} \left|LAT_s(w,c)\right|\right)$ doğrusal kriptanalizin karmaşıklığı en büyük girişe bağlı olduğu için önemlidir. Ayrıca en büyük LAT girişi bir S-kutusunun doğrusal olmama özelliğini bulmada kullanılabilir. (2.11) ifadesi bunu göstermektedir [57].

$$NLM_S = 2^{n-1} - \max_{w,c} \left|LAT_S(w,c)\right| \tag{2.11}$$

2.3.1.2 S-Kutularının XOR Tablosu (Fark Dağılım Tablosu)

Diferansiyel kriptanaliz saldırısı blok şifreleme algoritmasında kullanılan S-kutularının fark tablosundaki (XOR tablosunda) [59], [60] bazı özel girişlerin kullanılması esasına dayanır. $n \times n$ boyutunda bir S-kutusu için XOR tablosu $2^n \times 2^n$ matrise denk düşer. $S: Z_2^n \to Z_2^n$ şeklinde tanımlanan bir S-kutusu, $(\Delta a, \Delta b)$ XOR tablosuna giriş olarak indekslensin. P giriş vektörü, giriş farkı Δa ve çıkış farkı $\Delta b = S(P) \oplus S(P \oplus \Delta a)$ için, $\Delta a \in Z_2^n \neq (0,0,\ldots,0)$ ve $\Delta b \in Z_2^n$ olmak üzere $XOR_S(a,b)$ (2.12)'deki gibi ifade edilebilir [57].

$$XOR_s(a,b) = \#\left\{P \in Z_2^n \middle| S(P) \oplus S(P \oplus \Delta a) = \Delta b\right\}$$ [5] (2.12)

$$XOR_{\max}(S) = \max_{a,b} XOR_S(a,b) \quad (2.13)$$

(2.13) ifadesi bir S-kutusunun maksimum değerinin en geniş XOR tablosu girişi olduğunu göstermektedir [57].

2.3.1.3 Bütünlük (Completeness)

Kam ve Davida tarafından belirlenmiştir [56]. S-kutuları vektörel bir fonksiyondur ve bir fonksiyonun bütünlük özelliği taşıması için gerekli olan kurallar aşağıda verilmiştir [61]:

- $f : \{0,1\}^n \rightarrow \{0,1\}^n$ ifadesiyle tanımlanan bir fonksiyon olsun. $i, j \in \{1,2,\ldots,n\}$ için, $f(X)$ ve $f(X \oplus \Delta X_i)$ değerlerinin bir j değerinde en az bir tane X değeri için farklılaşıyorsa, bütünlük özelliği sağlanmış olur. Kısacası herhangi bir çıkış biti, giriş bitlerinin tümüne bağlıdır.
- S-kutusunun çığ (avalanche) vektörü (2.14) denklemindeki gibidir. [59], [62], [63].

$$\begin{aligned} \Delta Y^{\Delta X_i} &= f(X) \oplus f(X \oplus \Delta X_i) \\ &= \left[a^{\Delta X_i}{}_1 \; a_2^{\Delta X_i} \; \ldots\ldots a_n^{\Delta X_i}\right] \end{aligned} \quad (2.14)$$

- $\Delta Y^{\Delta X_i}$ çığ vektörü, giriş şeridinin sadece bir biti ($i.$ bit) değiştirilerek elde edilmiş fark şeridir. Çığ vektöründeki toplam değişme (2.15) ifadesiyle hesaplanır.

$$wt\left(a_j^{\Delta X_i}\right) = \sum_{\forall X} a_j^{\Delta X_i} \quad (2.15)$$

- (2.15) ifadesinin maksimum değeri 2^n'dir. Diğer bir deyişle $0 \le wt\left(a_j^{\Delta X_i}\right) \le 2^n$'dir. (2.16) 'daki ifadede ΔX_i vektörü görülmektedir.

[5] # sayı anlamına gelir.

$$\begin{aligned}\Delta X_1 &= [1,0,0,\ldots,0]\\ \Delta X_2 &= [0,1,0,\ldots,0]\\ &\vdots\\ \Delta X_n &= [0,0,0,\ldots,1]\end{aligned} \tag{2.16}$$

– Eğer $wt\left(a_j^{\Delta X_i}\right)=0$ ise yani çıkış bitleri giriş bitlerinden etkilenmiyorsa bütünlük yoktur denir. Öte yandan eğer $wt\left(a_j^{\Delta X_i}\right)=2^n$ ise giriş bitinin değili alındığında, çıkış bitinin doğrudan etkilendiği anlamına gelir ki bu da istenmeyen bir özelliktir. Bunun dışındaki tüm durumlar için S-kutusu bütünlük ölçütünü sağlayacaktır. Yani çığ vektöründeki toplam değişme (2.17)'de ifade edilmiştir.

$$0 < \frac{1}{2^n} wt\left(a_j^{\Delta X_i}\right) < 1 \tag{2.17}$$

2.3.1.4 Çığ (Avalanche) Kriteri

Çığ ölçütü (avalanche criterion - AVAL) Feistel [64] tarafından S-kutuları ve SPN tabanlı blok şifreler için tanımlanmıştır.

Bir $f:\{0,1\}^n \rightarrow \{0,1\}^n$ fonksiyonu için giriş bitinin bir biti değiştiğinde, çıkış bitlerinin yarısı değişecektir. Başka bir deyişle (2.18)' deki çığ vektöründeki toplam değişme, $i, j \in \{0,1,2,\ldots\}$ olmak üzere i giriş ve j çıkış bitleri için (2.19)' daki gibi ifade edilebilir ise AVAL [59] [62] [63] kriteri sağlanmış olur.

$$\frac{1}{2^n}\sum_{j=1}^{n} wt\left(a_j^{\Delta X_i}\right) = \frac{n}{2} \tag{2.18}$$

(2.18) ifadesi $k_{AVAL}(i)$ değerini elde etmek için tekrar düzenlenir ise (2.19) ifadesi elde edilir.

$$k_{AVAL}(i) = \frac{1}{n2^n}\sum_{j=1}^{n} wt\left(a_j^{\Delta X_i}\right) = \frac{1}{2} \tag{2.19}$$

(2.19) ifadesine göre $k_{AVAL}(i)$ parametresi [0,1] aralığında değerler almaktadır ve herhangi bir i değeri için $\frac{1}{2}$ değerinden farklı bir değer alırsa S-kutusu AVAL kriterini sağlamayacaktır [61].

2.3.1.5 Katı Çığ Kriteri (Strict Avalanche Criterion)

Webster ve Tavares [65] bütünlük ve çığ özelliklerini bileştirerek katı çığ özelliğini (Strict Avalanche Criterion - SAC) tanımlamışlardır. Buna göre $f:\{0,1\}^n \to \{0,1\}^n$ fonksiyonu için, $i, j \in \{0,1,2,\ldots,n\}$ olmak üzere eğer giriş biti i'yi değiştirmek çıkış biti j'nin kesinlikle $\frac{1}{2}$ olasılığında değişmesine neden oluyor ise, SAC özelliği sağlanmaktadır. Matematiksel olarak tüm i ve j değerleri için (2.14) ifadesi doğrulanır ise, S-kutusu katı çığ kriterini sağlıyor denir [59] [62] [63].

$$\frac{1}{2^n} wt\left(a_j^{\Delta X_i}\right) = \frac{1}{2} \tag{2.20}$$

(2.13) ifadesi (2.15) ifadesindeki gibi değiştirilerek $k_{SAC}(i,j)$ tipinde bir SAC parametresi tanımlanabilir.

$$k_{SAC}(i,j) = \frac{1}{2^n} wt\left(a_j^{\Delta X_i}\right) \tag{2.21}$$

Eğer ki, $k_{SAC}(i,j)$ parametresi [0,1] aralığında değerler alır ve herhangi bir (i,j) kombinasyonu için, $\frac{1}{2}$ değerinden farklı ise S-kutusu SAC kriterini sağlamaz. İfadelerden de görülebileceği gibi S-kutusu çığ ve bütünlük kriterlerinin ikisini de sağlıyor ise o zaman SAC ölçütünü de sağlar denilir [61].

2.4 Cebirsel Yapılar

2.4.1 Yarı Gruplar (Semi Groups)

Tanım 2.12. S , boş olmayan bir küme ve *, S üzerinde tanımlı bir işlem olsun. $(S,*)$ yapısı S üzerinde * işlemi birleştirme özelliğine sahip ise yarı gruptur. Eğer işlem hem birleştirme hem de değişme özelliğine sahip ise $(S,*)$ yapısı değişken yarı grup olarak isimlendirilir. Yarı gruplarda birleşme özelliği ile birlikte etkisiz elemanda mevcutsa monoid olarak isimlendirilir. Diğer bir deyişle monoid etkisiz elemana sahip $(S,*)$ yarı gruptur [61].

Tanım 2.13. Tüm elemanların tersinin olduğu monoid'e grup denir $(S,*)$. Grup aşağıda verilen üç şartı sağlar [61]:

i. * işlemi, S üzerinde birleşme özelliğine sahiptir.

ii. Etkisiz eleman mevcuttur.

iii. S grubunun tüm elemanlarının tersi vardır.

Tanım 2.14. Herhangi bir $(G,*)$ grubunun sahip olduğu eleman sayısı ya da derecesi G kümesinin kardinalitesidir ve $|G|$ şeklinde gösterilir.

Teorem 2.15. $(G,*)$ bir grup ise sol ve sağ sadeleşme kuralı uygulanabilir. Yani $a,x,y \in G$ ise,

- $ax = ay$ ifadesinin anlamı $x = y$ (sol sadeleştirme),
- $xa = ya$ ifadesinin anlamı $x = y$ (sağ sadeleştirme)

şeklinde ifade edilebilir.

Tanım 2.15. $(G,*)$ bir grup ve $a,b \in G$ ise,

- $ax = b$ denkleminin $x = a^{-1}b$ şeklinde tek bir çözümü vardır.
- $ya = b$ denkleminin $y = ba^{-1}$ şeklinde tek bir çözümü vardır.

Tanım 2.16. Eğer $(G,*)$ sonlu bir grupsa, bu grubun cayley tablosunda G'nin her elemanı, her bir satır ve sütunda sadece bir kez yer alır.

Tanım 2.17. En az bir üretece sahip gruplara periyodik (cyclic) denir.

Tanım 2.18. $(G,*)$ grubu, n bir tamsayı olmak üzere $\forall g \in G$ için, $a \in G$ elemanı mevcut ise halkadır. $(G,*)$ grubu a tarafından üretilmiştir ve a, $(G,*)$ grubunun üretecidir.

2.4.2 Permütasyon Gruplar

Tanım 2.19. S boş olmayan bir küme olsun. S kümesinin bir permütasyonu, S 'den S 'ye bir tam eşleme (bijection)'dir.

Bir tam eşleme tanımlamak için kullanılan yol genellikle S kümesinin tüm eleman eşleşmelerinin etkilerini göstermektir [66].

Örnek 2.7. $S = \{1,2,3,4\}$ ise aşağıdaki gibi p_1 tam eşlemesi tanımlanabilir.

$$p_1(1) = 2,\ p_1(2) = 4,\ p_1(3) = 3,\ p_1(4) = 1,$$

$p_1 = \begin{vmatrix} 1\,2\,3\,4 \\ 2\,4\,3\,1 \end{vmatrix}$ şeklinde gösterilebilir.

Örnek 2.8. $A = \{1,2,3\}$ kümesini düşünelim olası tüm permütasyonlar S_3 kümesi $P_1P_2 \ldots P_6$ olsun.

$$p_1 = \begin{vmatrix} 1\,2\,3 \\ 1\,2\,3 \end{vmatrix} \quad p_2 = \begin{vmatrix} 1\,2\,3 \\ 2\,3\,1 \end{vmatrix} \quad p_3 = \begin{vmatrix} 1\,2\,3 \\ 3\,1\,2 \end{vmatrix} \quad p_4 = \begin{vmatrix} 1\,2\,3 \\ 1\,3\,2 \end{vmatrix} \quad p_5 = \begin{vmatrix} 1\,2\,3 \\ 3\,2\,1 \end{vmatrix}$$

$$p_6 = \begin{vmatrix} 1\,2\,3 \\ 2\,1\,3 \end{vmatrix}$$

Bu küme üzerinde birleşme $P_iP_j\left(P_iP_j \in S_3\right)$ olmak üzere bileşke olarak tanımlansın.

$p_{35} = \begin{vmatrix} 1\,2\,3 \\ 3\,1\,2 \end{vmatrix} \begin{vmatrix} 1\,2\,3 \\ 3\,2\,1 \end{vmatrix} \Rightarrow \begin{vmatrix} 1\,2\,3 \\ 1\,3\,2 \end{vmatrix}$ olarak bulunur.

O zaman $(S_3,*)$ için cayley tablosu Tablo 2.3'teki gibi olmaktadır.

Tablo 2.3. Örnek Cayley Tablosu

*	p_1	P_2	p_3	p_4	p_5	p_6
p_1	p_1	p_2	p_3	p_4	p_5	p_6
p_2	p_2	p_3	p_1	p_5	p_6	p_4
p_3	p_3	p_1	p_2	p_6	p_4	p_5
p_4	p_4	p_6	p_5	p_1	p_3	p_2
p_5	p_5	p_4	p_6	p_2	p_1	p_3
p_6	p_6	p_5	p_4	p_3	p_2	p_1

BÖLÜM 3

3 BLOK ŞİFRELEYİCİLER VE MİMARİLER

3.1 İşlem Biçimleri

Blok şifreleyiciler mesajları veri blokları halinde işlerler. Genellikle şifrelenecek verinin uzunluğu, şifreleyicinin blok uzunluğundan (n) daha büyüktür. Bu nedenle, veri eşit uzunlukta mesaj blokları dizisine bölünür. Şifreleyici çeşitli amaçlarla farklı işlem biçimlerini destekleyecek şekilde çalışır. İşlem biçimleri blok şifreleyiciye rastgelelik ekleme, açık metinleri keyfi uzunluklara çıkarma, hata ilerletimi kontrolü, blok şifreleyicinin akış şifreleyicisine çevrilmesi gibi güvenlik açısından arzulanan özellikleri sağlar.

İşlem biçimleri şu ölçütlere göre incelenir [23]:

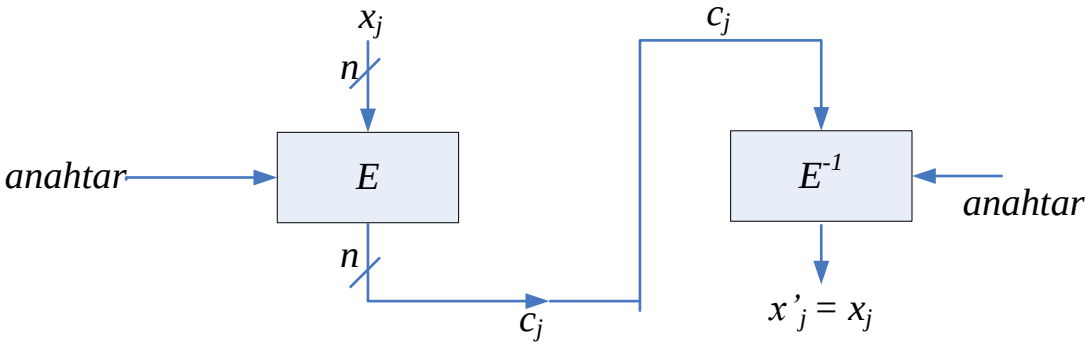
- *Hata genişlemesi*: Bir işlem biçimi tarafından işlenen bir mesajın iletiminde iki tip hata oluşabilir: Keyfi şekilde bitlerin eklenmesi ve silinmesi anlamına gelen *kayma (slip) hatası* ve bazı bitlerin döndüğü *bit-dönme (bit-flipping)* hatasıdır. Kullanılan işlem biçimine göre bu tarz hatalar, sadece hatanın oluştuğu blokları veya tüm veri bloklarını etkiler.
- *Açık metin artığı (redundancy)*: Açık metin olasılık dağılımının şifreli metnin tümüne mi, yoksa bir kısmına mı yayıldığını gösterir.
- *Rastgele erişim*: Bazı işlem biçimleri rastgele bir konumdaki veri bloklarını, diğer blokları okumadan veya güncellemeden, okumaya veya güncellemeye izin verir.

– *Paralel işleme*: Bazı işlem biçimlerinde bloklar paralel olarak işlenir. Bu işlem biçimlerinde her bir blok, diğer bloklara gerek olmadan hesaplanır. Bu işlem biçimleri yüksek başarımlı gerçeklemelerde kullanılır.

– *Rastgelelik eklemesi*: Bazı işlem biçimleri açık metin veya anahtardan farklı olarak başlangıç vektörü gibi ek rastgeleliklere ihtiyaç duyar.

3.1.1 Elektronik Kod Kitabı (Electronic Codebook – ECB)

Verinin aynı parçası, aynı anahtar ile üst üste iki kez şifrelendiğinde oluşan şifreli metin parçalarının da birbirine eşit olması durumuna Elektronik Kod Kitabı (Electronic Codebook – ECB) denir. Bu sisteme saldırıda bulunma amacı güden kişiler için, çok önemli bir bilgidir [3].



Şekil 3.1. Elektronik Kod Kitabı (Electronic Codebook – ECB)

ECB işlem biçimi aşağıda sıralanan özelliklere sahiptir [3]:

– Aynı açık metin içeren bloklar aynı anahtar ile şifrelendiğinde aynı şifreli metin elde edilir.

– *Zincir bağımlılıklar*: Bloklar diğer bloklardan bağımsız olarak şifrelenir.

– *Hata ilerletimi*: Bir veya daha fazla bitin tek bir şifreli metin bloğunda hatalı olması, sadece o bloğu etkiler.

Şifreli metin blokları birbirinden bağımsız olduğu için, ECB bloklarından birinin zararlı kod blokları ile değiştirilmesi diğer blokları etkilemez. ECB' nin veri örüntüsünü saklamaması, yani aynı anahtar ile şifrelenen aynı açık metnin aynı şifreli metni oluşturması nedeniyle, ECB birden fazla blok içeren mesajlar için önerilmez [3].

c_j : Şifreli metin

x_j : Açık metin

E_k : Şifreleme fonksiyonu, E_k^{-1} : Deşifreleme fonksiyonu

k : Anahtar, t : Blok sayısı

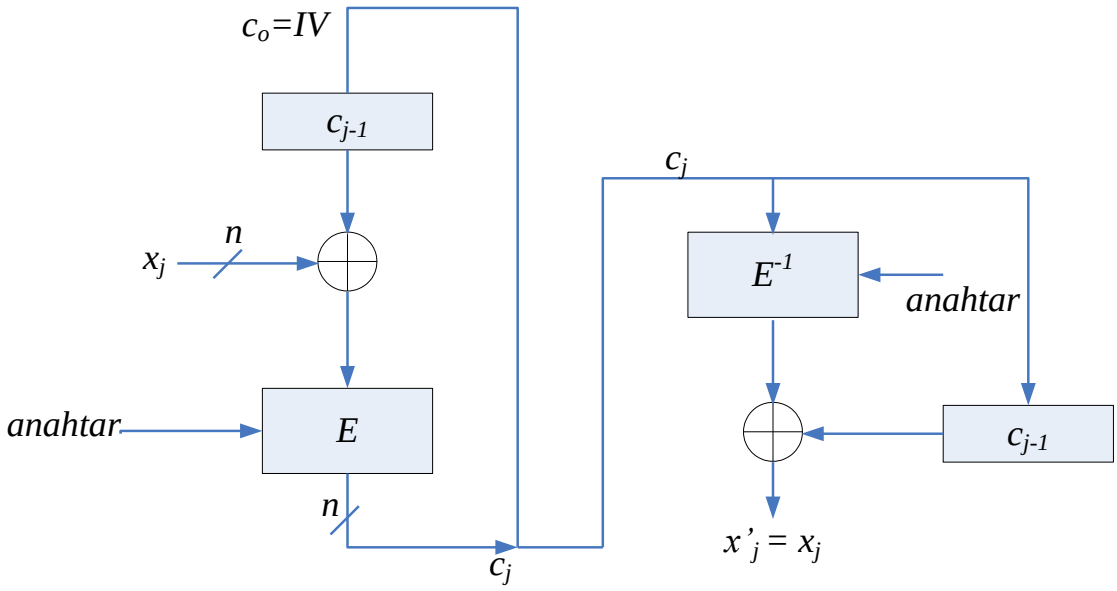
Şifreleme: $1 \le j \le t,\ c_j \leftarrow E_k(x_j)$

Deşifreleme: $1 \le j \le t,\ x_j \leftarrow E_k^{-1}(c_j)$

Algoritma 3.1. Elektronik Kod Kitabı (ECB) Şifreleme Deşifreleme

3.1.2 Şifreleyici Blok Zincirlemesi (Cipher Block Chaining – CBC)

Şifreleyici blok zincirlemesinde şifreli metin, açık metin bloklarının, daha önceki şifreli blok ile XOR işlemine tabi tutulduktan sonra şifrelenmesi ile elde edilir.



Şekil 3.2. Şifreleyici Blok Zincirlemesi (Cipher Block Chaining – CBC)

CBC işlem biçimi aşağıda sıralanan özelliklere sahiptir [3]:

– *Benzer açık metinler*: Benzer açık metin, benzer anahtar ve *IV* değeri ile şifrelendiğinde elde edilen şifreli metin değeri de benzerdir.

- *Zincir bağımlılıklar*: Şifreli metnin sıralaması değiştirilirse, deşifreleme süreci etkilenir.
- *Hata ilerletimi*: C_j bloğundaki bir hata C_j ve C_{j+1} bloklarını etkiler.
- *Hata kurtarımı*: CBC modunda C_j bloğu hatalı C_{j+1} bloğu hatasız olduğu durumda C_{j+2}, X_{j+2} değerine doğru olarak deşifrelenebilir.

CBC modunda *IV* değerinin gizli olmaması, tutarlığın korunması gerekliliğini ortaya çıkarmıştır. Çünkü saldırgan kurtarılan açık metin bloğu üzerinde, tahmin edilebilir zararlı bit değişiklikleri yapmış olabilir. Gizli bir *IV* değerinin kullanılması bu tehlikeyi engeller. Tutarlılığın gerektiği durumlarda ek yöntemlerin kullanılması gerekmektedir. Çünkü şifreleme algoritmasının ana görevi verinin güvenliğini sağlamaktır [3].

c_j : Şifreli metin

x_j : Açık metin

E_k : Şifreleme fonksiyonu, E_k^{-1} : Deşifreleme fonksiyonu

k : Anahtar, t : Blok sayısı

Şifreleme: $1 \le j \le t,\ c_j \leftarrow E_k\left(c_{j-1} \oplus x_j\right)$

Deşifreleme: $1 \le j \le t,\ x_j \leftarrow c_{j-1} \oplus E_k^{-1}\left(c_j\right)$

Algoritma 3.2. Şifreleyici Blok Zincirlemesi (CBC) Şifreleme Deşifreleme

3.1.3 Şifreleyici Geri Beslemesi (Cipher Feedback - CFB)

Bu yöntemde k. şifreli blok, $(k-1)$. şifreli metnin şifrelenip, k. açık metin bloğu ile XOR işlemine tabi tutulması sonucunda elde edilir.

CFB işlem biçimi aşağıda sıralanan özelliklere sahiptir [3]:

– *Benzer açık metinler*: Her bir CBC şifreleme için, *IV* değerinin değiştirilmesi halinde farklı çıkış değerleri elde edilir. Bazı uygulamalarda *IV* değerinin gizli olması gerekirken, CFB modunda gerekli değildir.

– *Zincir bağımlılıklar*: Zincirleme mekanizması CBC şifrelemedeki gibi, şifreli metin bloğu c_j, açık metin bloğu x_j ve öncül x_{j-1} bloğuna bağlı olmasına neden olur. Şifreli metin bloğunun doğru olarak deşifrelenmesi, $\left(\frac{n}{r}\right)$ öncül şifreli metin bloğunun doğru olmasına bağlıdır.

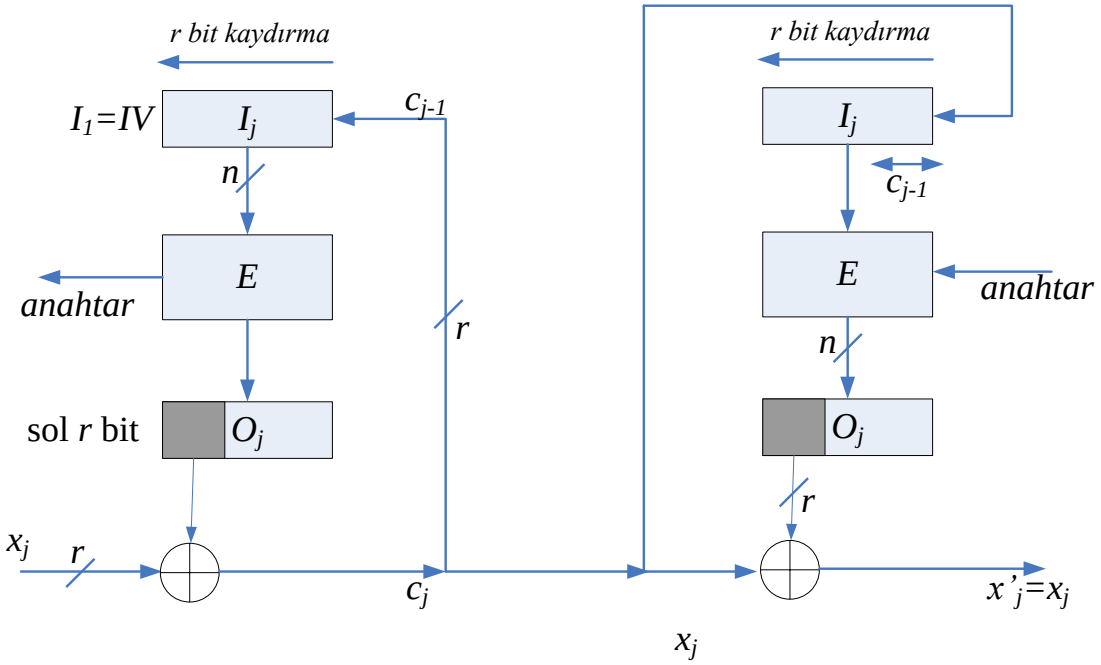
Giriş Değerleri:	k bit anahtar,
	n bit *IV* değeri,
	r bit açık metin blokları $x_1,\ldots,x_u (1 \le r \le n)$
Şifreleme:	$I_1 \leftarrow IV$ (I_j kaydırma saklacındaki giriş değeri)
	$1 \le j \le u$ için
	$O_j \leftarrow E_K(I_j)$
	$t_j \leftarrow O_j$'nin en son r biti
	$c_j \leftarrow x_j \oplus t_j$
	$I_{j+1} \leftarrow 2^r . I_j + c_j \bmod 2^n$
	c_j değeri kaydırma saklacının sağ taraftan sonuna kaydırılır.
Deşifreleme:	$1 \le j \le u$ için, şifreli metin bloğu c_j alınır alınmaz; $x_j \leftarrow c_j \oplus t_j \ldots O_j, t_j, I_j$ değerleri yukarıdaki gibi hesaplanır.

Algoritma 3.3. Şifreleyici Geri Beslemesi (Cipher Feedback - CFB) Şifreleme Deşifreleme

– *Hata ilerletimi*: Tek bir r bit şifreli metin bloğundaki, bir veya daha fazla bitin hatalı olması, bu bloğu takip eden $\left(\frac{n}{r}\right)$ bloğun deşifrelenmesini etkiler.

– *Hata kurtarımı*: CFB, CBC' deki gibi kendi kendine eşzamanlanabilir (synchronize); fakat hata kurtarımı için $\left(\frac{n}{r}\right)$ şifreli metin bloğuna gerek duyar.

– *Üretilen iş (Throughput)*: $r < n$ için, şifreli metnin r bitini oluşturan E fonksiyonunun herbir çalışmasında üretilen iş (throughput) $\left(\frac{n}{r}\right)$ oranında azalır [3].



Şekil 3.3. Şifreleyici Geri beslemesi (Cipher Feedback - CFB)

3.1.4 Çıkış Geribesleme Modu (Output Feedback - OFB)

Çıkış geribesleme modu hata ilerletiminden kaçınılan uygulamalarda kullanılır. CFB moduna benzemekte olan OFB, değişik uzunluktaki blokları şifreleyebilir. Farklı olarak, şifreli metin yerine, şifreleme fonksiyonu E geri besleme olarak sunulur.

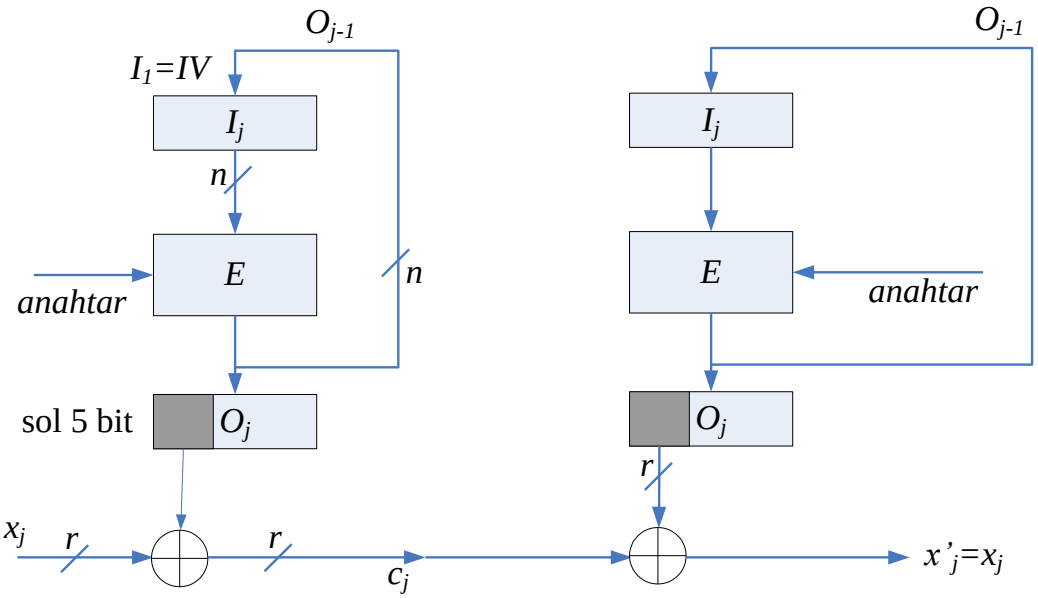
Giriş Değerleri:	k bit anahtar,
	n bit IV değeri,
	r bit açık metin blokları $x_1,\ldots,x_u (1 \le r \le n)$
Şifreleme:	$I_1 \leftarrow IV$
	$1 \le j \le u$ için açık metin bloğu x_j
	$O_j \leftarrow E_K(I_j)$
	$t_j \leftarrow O_j$ 'nin en son r biti
	$c_j \leftarrow x_j \oplus t_j$ (r bit şifreli metin bloğu c_j 'yi ilet.)
	$I_{j+1} \leftarrow O_j$ takip eden blok için blok şifreleme giriş değerini günceller.
	c_j değeri kaydırma saklacının sağ taraftan sonuna kaydırılır.
Deşifreleme:	$1 \le j \le u$ için, şifreli metin bloğu c_j alınır alınmaz; $x_j \leftarrow c_j \oplus t_j \ldots O_j, t_j, I_j$ değerleri yukarıdaki gibi hesaplanır.

Algoritma 3.4. Geribesleme Modu (Output Feedback - OFB) Şifreleme Deşifreleme

OFB işlem biçimi aşağıda sıralanan özelliklere sahiptir [3]:

- *Benzer açık metin*: Benzer açık metinlerin şifrelenmesinde *IV* değerinin değiştirilmesi, farklı bir çıkış değerinin oluşmasına neden olur.
- *Zincir bağımlılıklar*: Anahtar uzayı açık metinden bağımsızdır.
- *Hata ilerletimi*: Şifreli metin c_j üzerindeki bir veya daha fazla bitin hatalı olması, sadece o konumdaki bitin deşifrelenmesini etkiler.
- *Hata kurtarımı*: OFB modu şifreli metindeki bit hatalarını düzeltebilir; fakat şifreli metin bitlerinin yok olması durumunda kendi kendine eşzamanlayamaz.

– *Üretilen iş (Throughput)*: Üretilen iş, $r < n$ için, CFB modunun her çalışması sonucunda azalır. Şifreleme/deşifreleme süresince anahtarlar açık metin veya şifreli metne bağımlı olmadığına göre, önceden hesaplanabilir [3].



Şekil 3.4. Çıkış Geribesleme Modu (Output Feedback - OFB)

3.2 Feistel Blok Yapısı

Horst Feistel tarafından tasarlanan Feistel blok yapısı [64] tersi alınabilir çarpım şifresi kavramını temel almıştır. Feistel ağı, her bir aşamada güncel bloğun yarısını güncelleyen ve çift sayıda blok uzunluğu gerektiren bir blok şifreleme mimarisidir. Bu blok yapısında giriş bloğu iki parçaya bölünür ve döngüler boyunca işlenir; verinin sol parçasında, verinin sağ parçası ve alt anahtarın döngü fonksiyonunu temel alan yerine koyma (substitution) işlemi gerçekleştirilir. R döngülü bir Feistel ağında R adet $\left(k^1, k^2, \ldots, k^R\right)$ alt anahtar kullanılır. Alt anahtarların uzunlukları döngünün yapısına bağlı olarak değişir [29].

Tanım 3.1. r. döngünün N bit giriş değerlerinin sol ve sağ parçaları x_L^r ve x_R^r olarak tanımlansın. x_R^r sağ giriş verisi,

$$f_r : \{0,1\}^{N/2} \rightarrow \{0,1\}^{N/2} \tag{3.1}$$

fonksiyonuna, k^r alt anahtarını parametre olarak alır ve giriş verisi olur. f_r fonksiyonunun çıkış değeri x_L^r ile XOR'lanır ve sonraki döngünün giriş değerinin sağ tarafı $\left(x_R^{r+1}\right)$ elde edilir. Bu döngüde x_R^r değişmeden sonraki döngünün giriş değerinin sol parçası $\left(x_L^{r+1}\right)$ olur. Yarım blokların değişimi son döngü hariç tüm aşamalarda olur. Yukarıdaki gösterimde açık metin $p=\left(x_L^1,x_R^1\right)$, şifreli metin ise $c=\left(x_L^{r+1},x_R^{r+1}\right)$ olarak ifade edilir.

Feistel ağında yer değiştiren veri bloklarının permütasyonu XOR işlemi ile sağlanır. Shannon'un yerine koyma permütasyonu, ağ kavramını [10] gerçekler ve açık metin bir dizi döngüler boyunca şifrelenir. Feistel herbir aşamada giriş verisinin sadece yarısını güncellemektedir. Oysaki, SPN tüm giriş verisini günceller. Feistel ağının bir döngüsünün gösterimi Şekil 3.5' de, tüm ağın gösterimi ise Şekil 3.6' da görülebilir [29].

3.2.1 Feistel Ağının Özellikleri

- *Blok uzunluğu*: Büyük blok uzunluğu güvenliği artırır. Fakat, gerekli işlem sayısı arttığı için başarım düşer.
- *Anahtar uzunluğu*: Büyük anahtar uzunluğu güvenliği artırır, yoğun anahtar aramayı zorlaştırır; fakat şifreleme algoritmasını yavaşlatır.
- *Döngü sayısı*: Çoklu döngü sayısı güvenlik seviyesini artırır; fakat algoritmayı yavaşlatır.
- *Alt anahtar oluşturma algoritması*: Karmaşıklık derecesinin artması kriptanalizi zorlaştırır; fakat algoritmayı yavaşlatır.
- *Hızlı yazılım şifreleme/deşifreleme*: Algoritmanın çalışma zamanı hızlılığı önemlidir.

3.2.2 Feistel Ağının Gerçeklenmesi

Feistel ağının döngülerinin tasarımı için, değişik yaklaşımlar mevcut olmakla birlikte, ortak yaklaşım, Yer Değiştirme Permütasyon Ağ Şifresi (Substitution Permutation

Network - SPN) [67] S-kutularının düzenlenmesi ve doğrusal dönüşümler gibi temel özelliklerinin birleştirilmesidir. Feistel ağının döngü fonksiyonlarının tersinin alınması zorunluluğunun olmaması, tasarımda büyük bir esneklik sağlar.

Gerçekleme açısından bakıldığında, Feistel ağ yapısının önemli avantajlarından biri, şifreleme ve deşifreleme işlemlerinin temel olarak aynı olmasıdır. Yani şifreli metin, şifreleme algoritması kullanılarak deşifrelenir. Fakat farklı olarak döngü fonksiyonlarının ve karşılık gelen alt anahtarların sırası değişir. Eğer tüm döngülerde aynı döngü fonksiyonu kullanılmış ise, sadece alt anahtarların sırası değişir. Bu ters blokların oluşturulması ve saklanması gereksinimini ortadan kaldırır [29].

Tanım 3.2. Genel bir Feistel ağında x_L^r ve x_R^r eşit uzunluktadır ve böyle Feistel ağları dengeli olarak isimlendirilir. Schneier ve Kelsey tarafından sunulan Dengesiz Feistel Ağı (Unbalanced Feistel Network-UFN) 'nda [68] x_L^r ve x_R^r dengeli olarak adlandırılan Feistel ağlarından farklı olarak eşit uzunlukta değildirler. x_R^r giriş değerlerinin uzunlukları s ve t bit ve $(s+t=N)$ ise,

$$f_r : \{0,1\}^t \rightarrow \{0,1\}^s \tag{3.2}$$

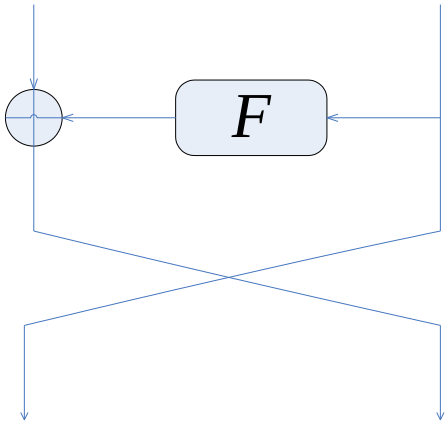
olarak gösterilir. Diğer döngünün giriş değerleri x_L^{r+1} ve x_R^{r+1},

$$x_L^{r+1} \parallel x_R^{r+1} = x_R^r \parallel \left(f_r\left(x_R^r\right) \oplus x_L^r\right) \tag{3.3}$$

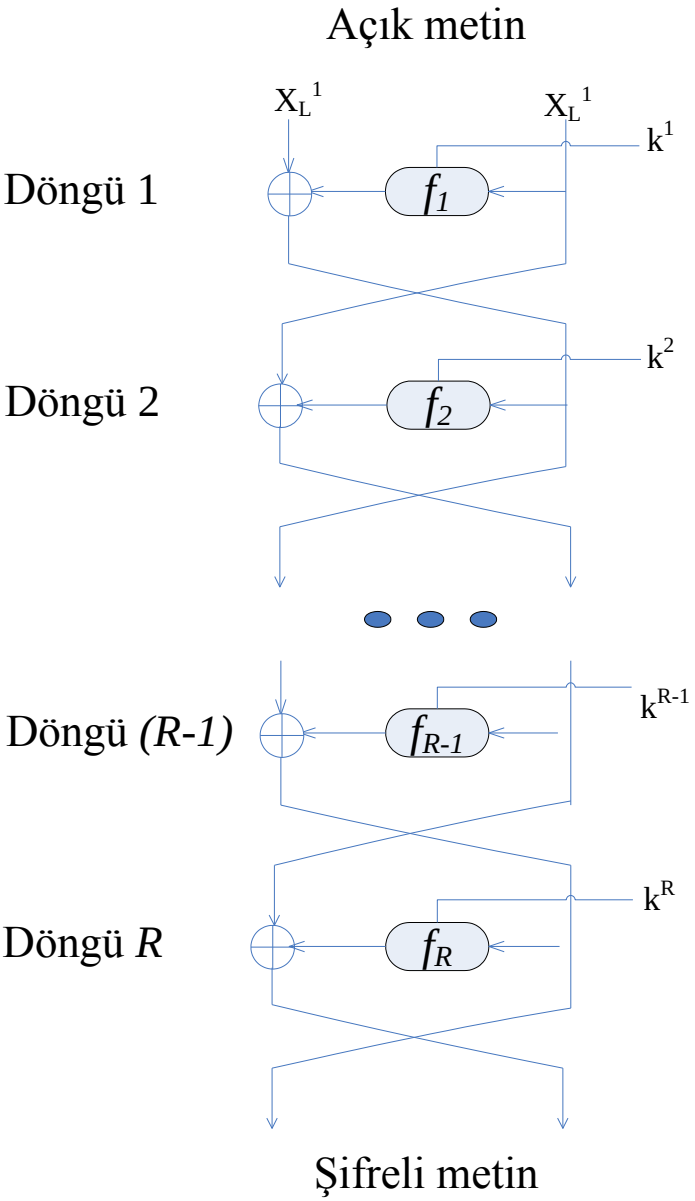
(|| birleştirme işlemi) olarak tanımlanır.

3.2.3 S-Kutuları ve Alt Anahtar

$n \times n$ S-kutusu, n bitten n bite tersi alınabilir bir dönüşümdür. Öyleki iki farklı giriş değeri, iki farklı çıkış değeri ile ilişkilendirilir. Alt anahtarlar, döngü anahtarları, k anahtarından bir anahtar planlama algoritması ile oluşturulur. k anahtar değeri bazen temel (master) anahtar olarak adlandırılır.



Şekil 3.5. Feistel Blok Yapısı (Bir Döngü)



Şekil 3.6. Feistel Blok Yapısı

3.3 Yer Değiştirme Permütasyon Ağ Şifresi (Substitution Permutation Netwok Cipher - SPN)

Yer değiştirme permütasyon şifresi (SPN), açık metin bloğunu giriş verisi olarak ve bu bloğu bir döngüdeki (round) temel işlemleri şifreleyicinin tasarımına göre çeşitli sayıda tekrar ederek işler.

Şifreleyicinin her bir döngüsü [67];

- Anahtar Karıştırma (XOR)
- Yer değiştirme (S kutularıyla karşılaştırma)
- Permütasyon (Doğrusal dönüşüm)

evrelerini içerir.

Bu işlemlerin ayrıntılı olarak anlatımında Heys'in [7] sunmuş olduğu, 16-bit açık metin bloğunu giriş verisi olarak kullanan 4 döngülü örnek bir SPN şifreleyicisi kullanılmıştır.

3.3.1 Anahtar Karıştırma (XOR)

Anahtar karıştırma, şifreleme algoritmasının bir döngüsündeki anahtar (alt anahtar) bitlerinin döngünün giriş değeri olan veri bloğunun bitleriyle XOR işlemine tabi tutulması ile gerçekleştirilebilir. Algoritmanın kriptanalizini zorlaştırmak üzere şifreli metin, son döngüye uygulanan bir alt anahtar ile bir yer değiştirme permütasyonuna tabi tutulur. Bu nedenledir ki, algoritmada dört döngü olmasına rağmen beş anahtar vardır.

Normal olarak bir şifrede bir döngü için alt anahtar, anahtar planlama olarak bilinen bir işlem yoluyla şifrenin ana anahtarından elde edilir. Anahtar planlamada, başlangıç değeri olarak kullanılan 32 bitlik anahtar değeri 5 ayrı alt anahtara bölünür.

Yer değiştirme permütasyonu ağ şifreleme algoritmasının sunumunda alt anahtarın tüm bitlerinin birbirinden bağımsız üretildiği ve ilişkisiz olduğu varsayılacaktır [2].

3.3.2 Yer Değiştirme

16 bitlik veri bloğu 4 bitlik alt bloklara ayırılır. Her alt blok 4×4'lük bir S-kutusuna giriş değeri oluşturmaktadır. 4×4'lük S-kutusu 16 4-bit değerin arama (look-up) tablosu ile tanımlanabilir. S-kutusunun en önemli özelliği doğrusal olmayan bir haritalama olmasıdır. Çıkış bitleri, giriş bitleri üzerinde doğrusal bir işlem olarak temsil edilemez. SPN şifreleme algoritmasında, tüm S-kutuları için aynı doğrusal olmayan haritalama kullanılacaktır [67].

DES şifreleme algoritmasında bir döngüdeki tüm S-kutuları farklıdır [5]. Aynı zamanda tüm döngüler aynı S-kutularının bir kümesini kullanır. Doğrusal ve diferansiyel kriptanaliz saldırıları S-kutularının tek bir haritasına veya farklı bir haritaya eşit olarak uygulanır.

SPN algoritmasının gösteriminde kullanılacak harita (bknz. Tablo 3.1), DES algoritmasının S-kutularından seçilmiştir (S-kutusunun ilk sırasıdır.). Tablodaki hexadecimal gösterimin en büyük biti, S-kutusunun soldaki en büyük bitini temsil eder.

Tablo 3.1. Hexadecimal Olarak S-kutusu Gösterimi

Giriş	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Çıkış	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7

3.3.3 Permütasyon

Bir döngünün permütasyon kısmı bitlerin yer değiştirmesi ya da bit konumlarının permütasyonu olarak tanımlanabilir. Tablo 3.1' in permütasyonu Tablo 3.2' de verilmiştir. Sayılar bloktaki bit konumunu temsil eder, 1 en soldaki bit, 16 en sağdaki bittir.

Tablo 3.2. Permütasyon

Giriş	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Çıkış	1	5	9	13	2	6	10	14	3	7	11	15	4	8	12	16

Sayılar, basitçe S-kutusu j'nin i. çıkışı S-kutusu i'nin j. girişine bağlanmıştır diye tasvir edilebilir [4].

3.3.4 Deşifreleme

Deşifreleme için, veri esasen ağ yoluyla geriye doğru geçirilir. Giriş çıkış, çıkışta giriş olmak koşulu ile deşifreleme ağında S-kutularında kullanılan haritalar şifreleme ağındaki haritaların tersidir. SPN ağının deşifrelemeye izin verebilmesi için tüm S-kutularının birebir örten (bijective) olması gerekir. Yani, aynı sayıdaki giriş çıkış bitleri birebir haritalamadır. Aynı yol ile, düzgün olarak bir ağın deşifrelenmesi için alt anahtarlar ters derecede uygulanır ve alt anahtarların bitleri permütasyona göre hareket etirilir. Şifreleme ağında son yer değiştirme katmanından sonra bir permütasyon varsa, deşifreleme ilk yer değiştirme katmanından önce bir permütasyon gerektirecektir.

Örnek 3.1. 32 bit ana anahtar, 5 alt anahtara bölünürken,

- 1-16 arasındaki bitler 1. anahtarı
- 5-20 arasındaki bitler 2. anahtarı
- 9-24 arasındaki bitler 3. anahtarı
- 13-28 arasındaki bitler 4. anahtarı
- 17-32 arasındaki bitler 5. anahtarı oluşturur.

0011 1010 1001 0100 1101 0110 0011 1111 1. anahtar

0011 **1010 1001 0100 1101** 0110 0011 1111 2. anahtar

0011 1010 **1001 0100 1101 0110** 0011 1111 3. anahtar

0011 1010 1001 **0100 1101 0110 0011** 1111 4. anahtar

0011 1010 1001 0100 **1101 0110 0011** 1111 5. anahtar

Kullanım açısından kolaylık sağlaması için SPN şifrelemede veri hexadecimal olarak girilir. Fakat şifreleme işlemi için, öncelikle 4 hexadecimal sayı 16 bit ikili veriye dönüştürülür.

Anahtarlama

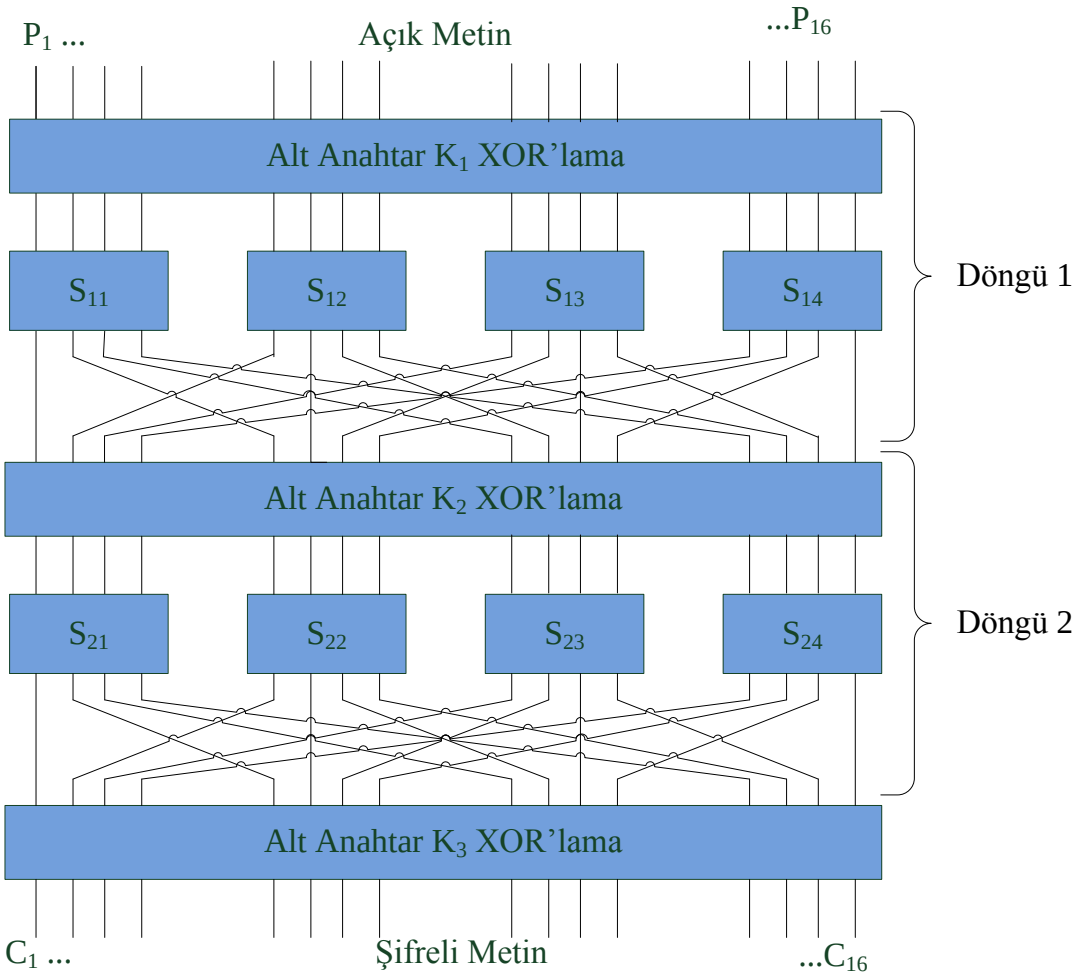
0010 0110 1011 0111 (26B7) Veri

XOR

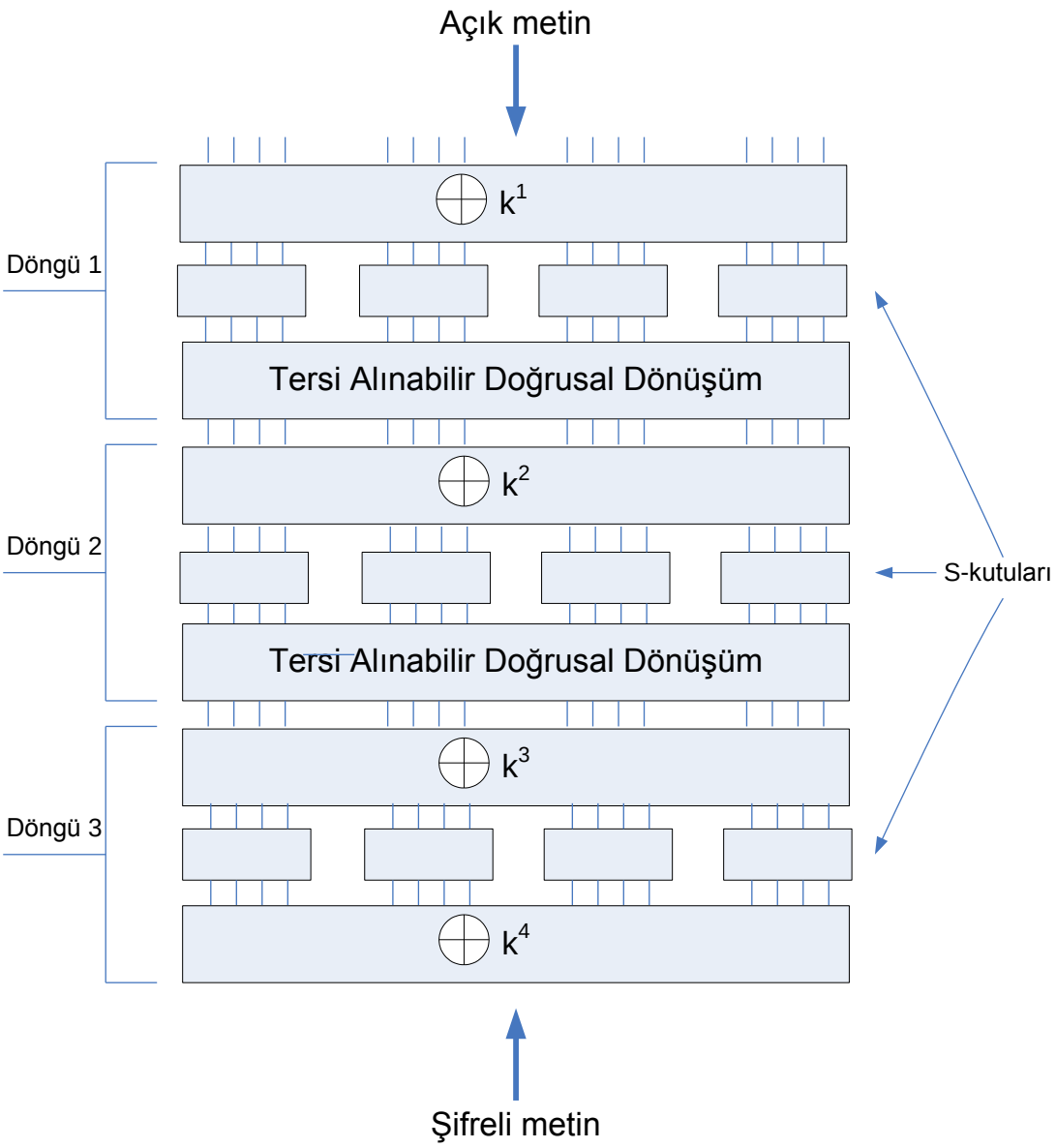
0011 1010 1001 0100 1.anahtar

⇩

0001 1100 0010 0011



Şekil 3.7. SPN Algoritması (N =16, M = n = 4, R = 2)



Şekil 3.8. SPN Blok Yapısı

Yer değiştirme

Bu aşamada sayıların hexadecimal değerlerine karşılık gelen değerler bulunur.

0001	1100	0010	0011
1	C	2	3
4	5	D	1
0100	0101	1101	0001

Karşılıkları bulunup yazıldıktan sonra yer değiştirme aşaması birinci döngü için sona erer ve bu 16 bit veri aynı döngünün üçüncü aşaması olan permütasyon aşamasına gider.

Permütasyon

0100 0101 1101 0001 bitlerinin permütasyon tablosuna göre yer değiştirmesi sonucunda elde edilen veri 0010 1110 0000 0111' dir. Elde edilen 16 bitlik veri ikinci döngünün giriş bloğu olarak alınır ve 2. anahtarla XOR işlemine tabi tutulur. Bu şekilde 4 döngü sonrasında işlem tamamlanmış olur. Giriş verisinde olduğu gibi çıkış ve ara veriler de hexadecimal tabana dönüştürülür.

3.4 Veri Şifreleme Standardı (Data Encryption Standard – DES)

DES [5] veriyi 64 bitlik bloklarla şifreleyen Feistel mimarisine sahip simetrik bir şifreleme algoritmasıdır. Algoritmada şifreleme ve deşifreleme için aynı anahtarlar (anahtar planlama algoritmasında küçük farklılıklar vardır.) kullanılırken, 64 bitlik açık metin, her bir blokta 64 bitlik şifreli metne dönüşür.

Anahtar uzunluğu 56 bit olan DES şifreleme algoritmasında, anahtar 64 bit ile gösterilmesine rağmen her bir sekizinci bit eşlik (parity) kontrolü için kullanılır ve gözardı edilir. Bu eşlik bitleri anahtar sekizlilerinin en önemsiz bitleridir [5].

DES şifreleme algoritması basit anlamda, şifrelemenin iki temel tekniği olan karıştırma ve yayılmanın kombinasyonundan oluşur. DES şifreleme algoritmasının temel bloğu, açık metin üzerinde bir anahtarı temel alan bir permütasyon tarafından takip edilen yerine koyma işlemini içerir ve döngü (round) olarak isimlendirilir. DES şifreleme algoritması 16 döngüye sahiptir ve tekniklerin aynı kombinasyonunu açık metne 16 kez uygular.

Algoritmanın 64 bitlik sayılar üzerinde sadece standart aritmetik ve mantıksal işlemleri kullanması, 1970' lerin donanım teknolojilerinde bu işlemlerin kolaylıkla gerçeklenmesini mümkün kılmıştır. Algoritmanın tekrarlama yapısı özel amaçlı yongalar (chip) üzerinde kullanımını rahatlaştırmaktadır [5].

Şifrelenecek blok bir başlangıç permütasyonundan (IP) geçirildikten sonra, karmaşık anahtar bağımlı bir hesaplamaya tabi tutulur. Son olarak başlangıç

permütasyonunun tersi olan bir permutasyondan $\left(IP^{-1}\right)$ geçirilir. Anahtar bağımlı fonksiyon, f şifreleme fonksiyonu ve KS anahtar eşleme fonksiyonu cinsinden ifade edilebilir [5].

L ve R bitten oluşan iki blok verilmesi durumunda LR, L bitlik bloğu takip eden R bitlik bloğun birleşimini gösterir.

3.4.1 Şifreleme

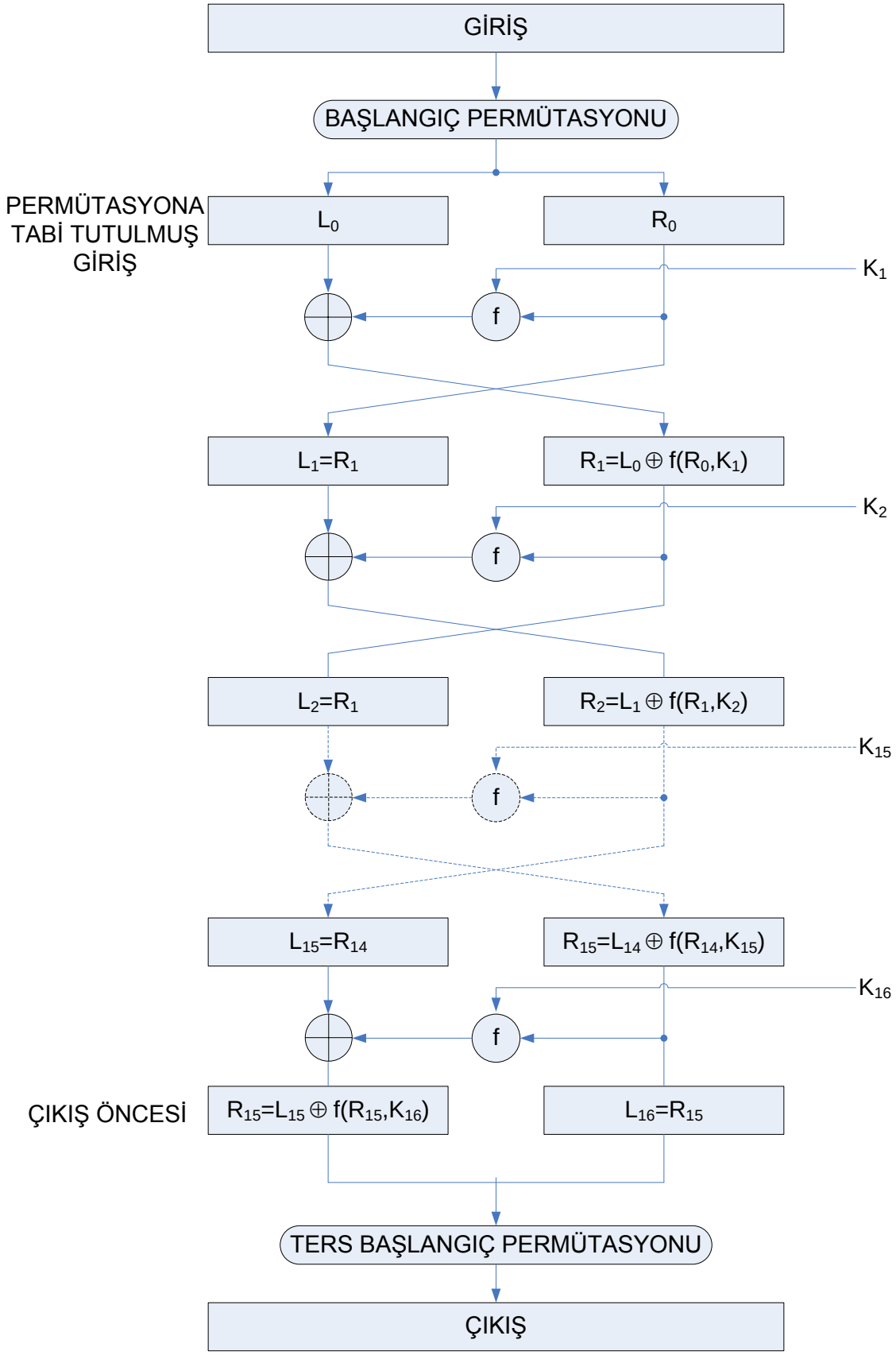
Şifreleme işleminin hesaplamaları Şekil 3.9'da görülmektedir. Şifrelenecek 64 bitlik giriş bloğu, Tablo 3.3'de değerleri görülmekte olan (IP) olarak adlandıran permütasyona tabi tutulur.

Permütasyona tabi tutulan giriş bloğunun 58. biti ilk bit, 50. biti ikinci bit, 7. biti son bit olarak seçilir. Permütasyona tabi tutulmuş giriş bloğu, aşağıda anlatılan karmaşık anahtar bağımlı hesaplamalara giriş değeri olarak verilir. Bu hesaplamaların çıkışı (öncül çıkış (preoutput) olarak adlandırılır.), Tablo 3.4'de gösterimi verilen başlangıç permutasyonunun tersi olan permutasyona tabi tutulur [5].

Permütasyona tabi tutulmuş giriş bloklarını, öncül çıkış bloklarını üretmek için giriş değerleri olarak kullanan hesaplamalar, 32 ve 48 bitlik iki blok üzerinde işlem yapan ve 32 bitlik bir çıkış bloğu oluşturan ve f şifreleme fonksiyonu cinsinden ifade edilen 16 döngülü hesaplamalardan oluşur [5].

Tablo 3.3. *IP* Başlangıç Permutasyonu (DES)

IP							
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



Şekil 3.9. DES Şifreleme Algoritması Şematik Gösterimi

Bir döngüye giriş değeri olarak verilen 64 bitlik blok, 32 bitlik R bloğu tarafından takip edilen 32 bitlik L bloğundan oluşur. Yukarıda bahsedilmiş olan gösterime göre giriş bloğu LR olarak temsil edilir [5].

K 64 bitlik anahtardan seçilmiş 48 bitlik blok olsun. $L'R'$ çıkış değerinin LR giriş değeri cinsinden ifadesi aşağıda gösterilmiştir:

$$L' = R$$

$$R' = L \oplus f(R,K) \tag{3.1}$$

Tablo 3.4. IP^{-1} Başlangıç Permutasyonunun Tersi (DES)

IP^{-1}

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

KS, 1 ile 16 arasında değerler alan bir tamsayı değeri n ve 64 bitlik *KEY (anahtar)* bloğunu giriş değeri olarak alan ve 48 bitlik K_n değeri oluşturan bir fonksiyon olsun. K_n, *KEY* değerinden permütasyona tabi tutularak seçilmiş bitlerdir.

$$K_n = KS(n, KEY) \tag{3.2}$$

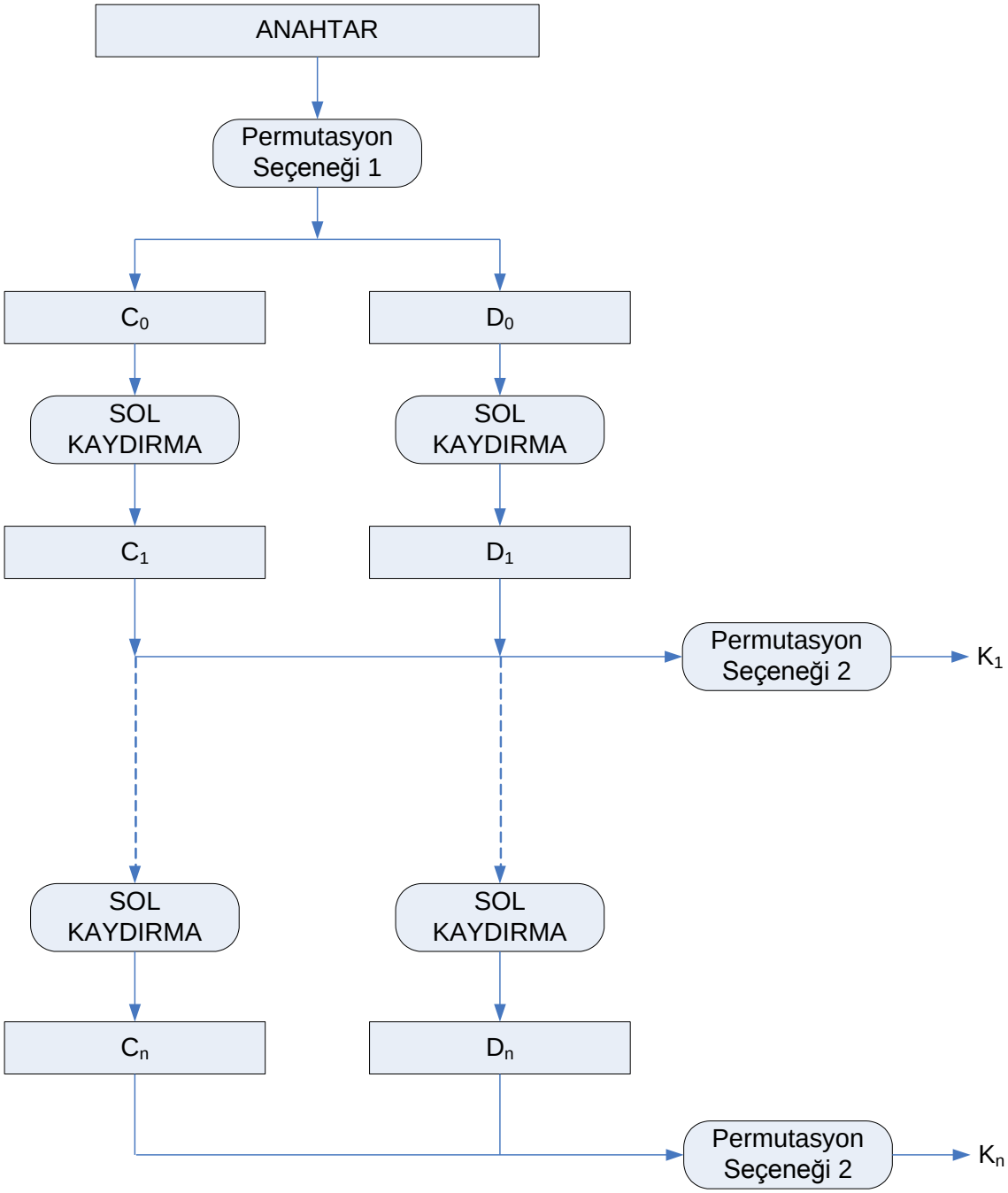
KS anahtar planlama olarak adlandırılır. Çünkü L' ve R'' nün *n.* döngüsünde kullanılan K anahtarı, (3.2) ifadesinden hesaplanan K_n değeridir.

L' ve R'' nün L_n ve R_n olduğu, göreceli olarak L_{n-1} ve R_{n-1}' inde L ve R olduğu, K' nında K_n olduğu düşünüldüğünde, yukarıdaki ifadeler aşağıdaki gibi ifade edilebilir (Bu ifadelerdeki n, 1 ile 16 arasında değerler alır.) (bknz.(3.3)):

$$L_n = R_n$$

$$R_n = L_{n-1} \oplus f(R_{n-1}, K_n) \tag{3.3}$$

Öncül çıkış bloğu $R_{16}L_{16}$ olarak ifade edilir.



Şekil 3.10. Anahtar Eşzamanlama Algoritması

3.4.2 Deşifreleme

IP başlangıç permutasyonunun tersi olan IP^{-1} permutasyonu öncül çıkış bloğuna uygulanır.

$$R = L'$$

$$L = R' \oplus f(L', K) \tag{3.4}$$

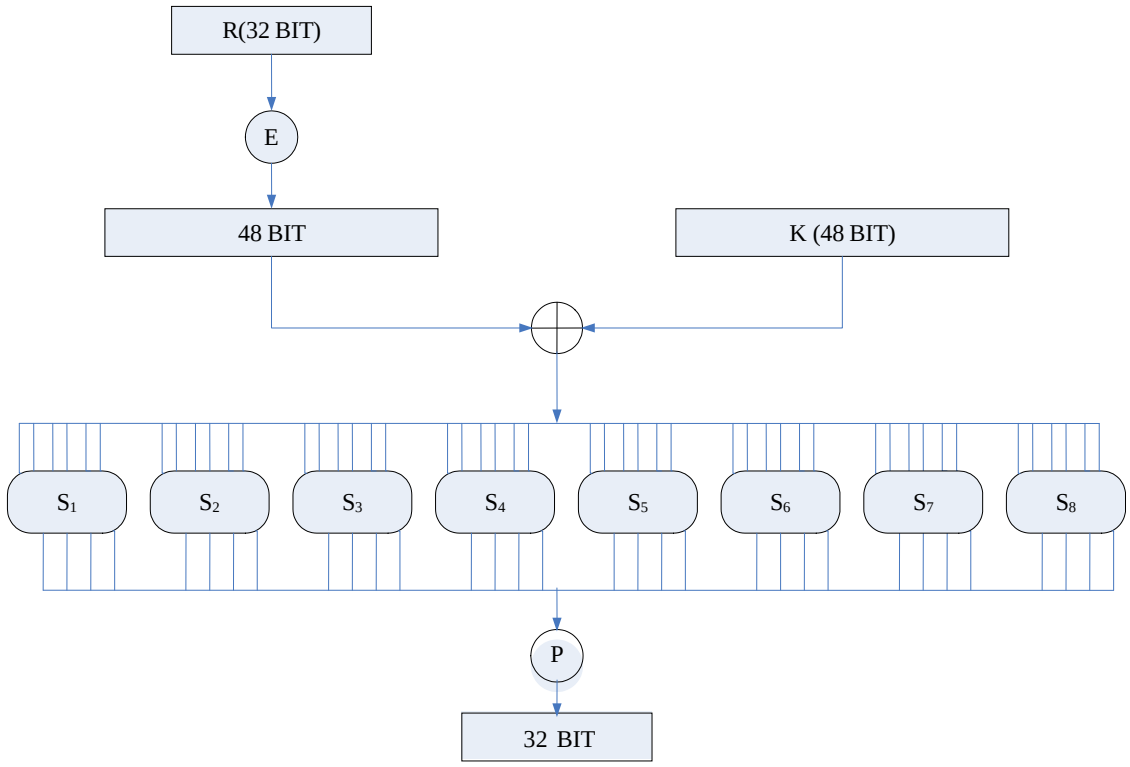
Deşifreleme algoritmasında, şifrelenmiş metin bloğuna şifrelemeyle aynı algoritma ve aynı anahtar bloğununun kullanılması, deşifrelemenin başarıyla gerçekleşmesini sağlar [5]. Deşifreleme işlemi şu şekilde ifade edilebilir:

$$R_{n-1} = L_n$$

$$L_{n-1} = R_n \oplus f(L_n, K_n) \tag{3.5}$$

3.4.3 Şifreleme Fonksiyonu

Şifreleme fonksiyonu f 'nin gösterimi Şekil 3.11'de görülmektedir.



Şekil 3.11. Şifreleme Fonksiyonunun Gösterimi

Tablo 3.5. E-Bit Seçme Tablosu (DES)

32	1	2	3	4	4
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

3.4.4 Yayılma Permutasyonu

E 32 bitlik bloğu giriş değeri olarak alıp, 48 bitlik çıkış bloğu oluşturan bir fonksiyon olsun. E fonksiyonunun, 6 bitlik 8 blok halinde gösterilen 48 bitlik çıkış bloğu, giriş bloğundaki bitler Tablo 3.5'deki seçme tablosuna göre seçilerek elde edilir [5].

3.4.5 S-Kutusu Yerine Koyma

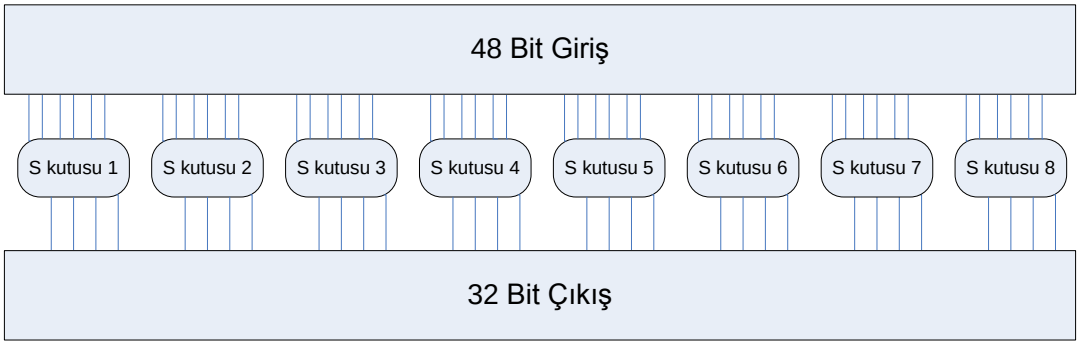
Yerine koyma işlemi 8 adet S-kutusu tarafından gerçekleştirilir. S-kutusu 6 bit giriş ve 4 bit çıkış değeri içerir. 48 bitlik bloklar 8 adet 6 bitlik alt bloğa bölünür. Her bir ayrı blok farklı bir S-kutusu ile işleme tabi tutulur. İlk blok S-kutusu 1 ile, ikinci blok S-kutusu 2 ile işleme tabi tutulur vs. Her bir S-kutusu 4 satır ve 16 kolon içerir. S-kutusu 4 bitlik sayılardan oluşur. S-kutusunun 6 bitlik giriş değeri, çıkış değeri için hangi satır ve kolona bakılması gerektiğini gösterir. Tablo 3.6' da S-kutusu değerleri görülmektedir.

S-kutusunun 6 bitlik giriş değeri $b_1\ b_2\ b_3\ b_4\ b_5$ ve b_6 olarak gösterilsin. b_1 ve b_6 bitleri 0 ve 3 aralığında değerler alan 2 bitlik bir sayı oluşturmak üzere birleştirilir. Bu değer S-kutusu üzerindeki satırı belirler. Ortadaki 4 bit $(b_2\, b_3\, b_4\, b_5)$ 0 ve 15 aralığında değerler alan 4 bitlik bir sayı oluşturur. Bu değerde S-kutusunun kolonunu belirler [14].

$$B_1B_2\ldots B_8 = K \oplus E(R)$$

$$P(S_1(B_1)S_2(B_2)\ldots S_8(B_8)) \tag{3.6}$$

Örneğin, (XOR fonksiyonun 31-36 bitleri) 110011 değeri için; baştaki ve sondaki bit (11) satırı oluşturur (3). Ortadaki 4 bit (1001) kolon değerini oluşturur (9). S-kutusu 6 için 3. satır 2. kolondaki değer 14' tür. 1110 değeri 110011 değerinin yerine koyulur.



Şekil 3.12. S kutuları Yerine Koyma İşlemi

3.4.6 DES Algoritmasının Adımları

Milli Standardlar Bürosu (National Bureau of Standards) tarafından yayınlanmış DES şifreleme algoritmasının aşamaları aşağıda sıralanmıştır [5]:

- Mesaj 64 bitlik bloklara bölünür.
- Bu bloklar bir başlangıç permütasyonundan geçirilir.
- 56 bitlik anahtarı kullanılarak, 16 adet 48 bitlik anahtar elde edilir.
    - 56 bitlik anahtar üzerinde permütasyon gerçekleştirerek, 2 adet 28 bitlik anahtar elde edilir.
    - Her iki blok, 1, 2, 9 ve 16. aşamalarda sola doğru bir, diğer aşamalarda 2 bit döndürme (rotate) işlemine tabi tutulur.
    - Her bir blok ayrı ayrı permütasyondan geçirilip, birinci anahtarın 9, 18, 22 ve 25., ikinci anahtarın 35, 38, 43 ve 54. bitleri elenir ve 48 bitlik anahtar elde edilir.

Tablo 3.6. S Kutuları (DES)

S kutusu 1:															
14,	**4,**	**13,**	**1,**	**2,**	**15,**	**11,**	**8,**	**3,**	**10,**	**6,**	**12,**	**5,**	**9,**	**0,**	**7,**
0,	**15,**	**7,**	**4,**	**14,**	**2,**	**13,**	**1,**	**10,**	**6,**	**12,**	**11,**	**9,**	**5,**	**3,**	**8,**
4,	**1,**	**14,**	**8,**	**13,**	**6,**	**2,**	**11,**	**15,**	**12,**	**9,**	**7,**	**3,**	**10,**	**5,**	**0,**
15,	**12,**	**8,**	**2,**	**4,**	**9,**	**1,**	**7,**	**5,**	**11,**	**3,**	**14,**	**10,**	**0,**	**6,**	**13,**
S kutusu 2:															
15,	**1,**	**8,**	**14,**	**6,**	**11,**	**3,**	**4,**	**9,**	**7,**	**2,**	**13,**	**12,**	**0,**	**5,**	**10,**
3,	**13,**	**4,**	**7,**	**15,**	**2,**	**8,**	**14,**	**12,**	**0,**	**1,**	**10,**	**6,**	**9,**	**11,**	**5,**
0,	**14,**	**7,**	**11,**	**10,**	**4,**	**13,**	**1,**	**5,**	**8,**	**12,**	**6,**	**9,**	**3,**	**2,**	**15,**
13,	**8,**	**10,**	**1,**	**3,**	**15,**	**4,**	**2,**	**11,**	**6,**	**7,**	**12,**	**0,**	**5,**	**14,**	**9,**
S kutusu 3:															
10,	**0,**	**9,**	**14,**	**6,**	**3,**	**15,**	**5,**	**1,**	**13,**	**12,**	**7,**	**11,**	**4,**	**2,**	**8,**
13,	**7,**	**0,**	**9,**	**3,**	**4,**	**6,**	**10,**	**2,**	**8,**	**5,**	**14,**	**12,**	**11,**	**15,**	**1,**
13,	**6,**	**4,**	**9,**	**8,**	**15,**	**3,**	**0,**	**11,**	**1,**	**2,**	**12,**	**5,**	**10,**	**14,**	**7,**
1,	**10,**	**13,**	**0,**	**6,**	**9,**	**8,**	**7,**	**4,**	**15,**	**14,**	**3,**	**11,**	**5,**	**2,**	**12,**
S kutusu 4:															
7,	**13,**	**14,**	**3,**	**0,**	**6,**	**9,**	**10,**	**1,**	**2,**	**8,**	**5,**	**11,**	**12,**	**4,**	**15,**
13,	**8,**	**11,**	**5,**	**6,**	**15,**	**0,**	**3,**	**4,**	**7,**	**2,**	**12,**	**1,**	**10,**	**14,**	**9,**
10,	**6,**	**9,**	**0,**	**12,**	**11,**	**7,**	**13,**	**15,**	**1,**	**3,**	**14,**	**5,**	**2,**	**8,**	**4,**
3,	**15,**	**0,**	**6,**	**10,**	**1,**	**13,**	**8,**	**9,**	**4,**	**5,**	**11,**	**12,**	**7,**	**2,**	**14,**
S kutusu 5:															
2,	**12,**	**4,**	**1,**	**7,**	**10,**	**11,**	**6,**	**8,**	**5,**	**3,**	**15,**	**13,**	**0,**	**14,**	**9,**
14,	**11,**	**2,**	**12,**	**4,**	**7,**	**13,**	**1,**	**5,**	**0,**	**15,**	**10,**	**3,**	**9,**	**8,**	**6,**
4,	**2,**	**1,**	**11,**	**10,**	**13,**	**7,**	**8,**	**15,**	**9,**	**12,**	**5,**	**6,**	**3,**	**0,**	**14,**
11,	**8,**	**12,**	**7,**	**1,**	**14,**	**2,**	**13,**	**6,**	**15,**	**0,**	**9,**	**10,**	**4,**	**5,**	**3,**
S kutusu 6:															
12,	**1,**	**10,**	**15,**	**9,**	**2,**	**6,**	**8,**	**0,**	**13,**	**3,**	**4,**	**14,**	**7,**	**5,**	**11,**
10,	**15,**	**4,**	**2,**	**7,**	**12,**	**9,**	**5,**	**6,**	**1,**	**13,**	**14,**	**0,**	**11,**	**3,**	**8,**
9,	**14,**	**15,**	**5,**	**2,**	**8,**	**12,**	**3,**	**7,**	**0,**	**4,**	**10,**	**1,**	**13,**	**11,**	**6,**
4,	**3,**	**2,**	**12,**	**9,**	**5,**	**15,**	**10,**	**11,**	**14,**	**1,**	**7,**	**6,**	**0,**	**8,**	**13,**
S kutusu 7:															
4,	**11,**	**2,**	**14,**	**15,**	**0,**	**8,**	**13,**	**3,**	**12,**	**9,**	**7,**	**5,**	**10,**	**6,**	**1,**
13,	**0,**	**11,**	**7,**	**4,**	**9,**	**1,**	**10,**	**14,**	**3,**	**5,**	**12,**	**2,**	**15,**	**8,**	**6,**
1,	**4,**	**11,**	**13,**	**12,**	**3,**	**7,**	**14,**	**10,**	**15,**	**6,**	**8,**	**0,**	**5,**	**9,**	**2,**
6,	**11,**	**13,**	**8,**	**1,**	**4,**	**10,**	**7,**	**9,**	**5,**	**0,**	**15,**	**14,**	**2,**	**3,**	**12,**
S kutusu 8:															
13,	**2,**	**8,**	**4,**	**6,**	**15,**	**11,**	**1,**	**10,**	**9,**	**3,**	**14,**	**5,**	**0,**	**12,**	**7,**
1,	**15,**	**13,**	**8,**	**10,**	**3,**	**7,**	**4,**	**12,**	**5,**	**6,**	**11,**	**0,**	**14,**	**9,**	**2,**
7,	**11,**	**4,**	**1,**	**9,**	**12,**	**14,**	**2,**	**0,**	**6,**	**10,**	**13,**	**15,**	**3,**	**5,**	**8,**
2,	**1,**	**14,**	**7,**	**4,**	**10,**	**8,**	**13,**	**15,**	**12,**	**9,**	**0,**	**3,**	**5,**	**6,**	**11**

```
public long getCipherFunction(long subBlock, long key)
{
    return (long)getCipherPermutation(
                    (long)getSBoxOutput(
                            (long)getExpandedSubBlock((ulong)subBlock) ^ key));
}

public long encryptBlock(ulong block,int roundNumber)
{
    ulong ipout = block;// getInitialPermutation((ulong)block);
    ulong left = (ipout >> 32);
    ulong right = (ipout);
    for (int round = 0; round < roundNumber; round++)
    {
            ulong temp = left;
            left = right;
            right = (ulong)(getCipherFunction((long)right,
                    key.getSubKey(round + 1)) ^ (long)temp);
    }
    long result = ((long)right << 32) | ((long)left & LowWordMask);
    return result;
}

public long decryptBlock(long block)
{
    ulong ipout = getInitialPermutation((ulong)block);
    ulong left  = (ipout >> 32);
    ulong right = (ipout);
    for (int round=15; round>-1; round--)
    {
    // apply cipher for that round and swap subblocks
            ulong temp = left;
            left = right;
            right = (ulong)(getCipherFunction((long)right,
            key.getSubKey(round+1)) ^ (long)temp);
    }
    long result = ((long)right << 32) | ((long)left & LowWordMask);
    return (long)getFinalPermutation(result);
}
```

Algoritma 3.5. DES Şifreleme Algoritması Örnek Kaynak Kodu

– DES şifresinin döngülerini gerçekleştirmek üzere, 16 adet 48 bitlik anahtarın her biri kullanılır.

– 64 bitlik giriş değeri 32 bitlik iki yarı bloğa bölünür.

– DES döngü giriş değerinin sağ tarafı, giriş çıkış değerinin sol tarafı olur.

- Döngü giriş değerinin sağ tarafına parçalayıcı işlevi, döngüye ait anahtarla uygulanır.
- Parçalayıcı işlevin sonucu, döngüye giren değerin sol tarafı ile döngüden çıkan değerin sağ tarafının XOR işleminden geçirilmesiyle elde edilir.

– Dördüncü döngü 16 kez tekrar edilir.

– Sonucun sol ve sağ yarı blokları yer değiştirilir.

– Son permütasyon gerçekleştirilir.

3.5 AES (Advanced Encryption Standard)

AES' teki Rijndael algoritması 128, 192 ve 256 bitlik şifre anahtarları (cipher key) kullanarak 128 bitlik veri bloklarını şifreleyebilen bir simetrik blok şifredir. Algoritma üç ayrı uzunlukta şifre anahtarı kullandığı için bu yöntemler "AES - 128" , "AES - 192" ve "AES - 256" olarak isimlendirilir [6].

Gösterim

– K : Şifre anahtarı

– Nb : Durumda bulunan 32 bit sözcük sayısı (4)

– Nk : Şifre anahtarındaki 32 bit sözcük sayısı (4-6-8)

– Nr : Nb, Nk ' nin döngü sayısı (10-12-14) 128 bit anahtar, 192 bit anahtar, 256 bit anahtar

AES 10, 12, 14 döngüden oluşabilen bir yer değiştirme doğrusal dönüşüm ağıdır. Tüm şifreleme işlemlerinin sekizli (byte) temelli olduğu Rijndael şifreleme algoritmasında şifrelenecek veri bloğu sekizli dizisine ayrılır.

AES algoritmasındaki bütün sekizli değerleri sırayla $\{b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0\}$ şeklinde gösterilir. Bu sekizliler sonlu alan elemanları olarak isimlendirilirler ve şu polinomsal gösterimi kullanırlar [6]:

$$b_7 x^7 + b_6 x^6 + b_5 x^5 + b_4 x^4 + b_3 x^3 + b_2 x^2 + b_1 x + b_0 = \sum_{i=0}^{7} b_i x^i . \qquad (3.7)$$

Örnek 3.2. {01100011} değeri sonlu alan elemanı olarak $x^6 + x^5 + x + 1$ şeklinde ifade edilir.

Sekizli dizlerinin AES algoritmasındaki gösterimi aşağıdaki gibidir:

$$a_0 a_1 a_2 \ldots a_{15} \tag{3.8}$$

Sekizlilerin ve sekizlilerin içindeki bitlerin sıralaması 128 bitlik giriş serisi (3.9) ifadesinden (3.10) ifadesindeki gibi türetilir.

$$input_0 input_1 input_2 \ldots input_{126} input_{127} \tag{3.9}$$

$$\begin{aligned} a_0 &= \{input_0, input_1, \ldots, input_7\}; \\ a_1 &= \{input_8, input_9, \ldots, input_{10}\}; \\ &\vdots \\ a_{15} &= \{input_{120}, input_{121}, \ldots, input_{127}\}. \end{aligned} \tag{3.10}$$

Örüntü daha uzun diziler için genişletilebilir (Örneğin 192 ve 256 bitlik anahtar uzunlukları için). Bu sayede ilgili ifade şu şekilde genelleştirilebilir:

$$a_n = \{input_{8n}, input_{8n+1}, \ldots, input_{8n+7}\}. \tag{3.11}$$

3.5.1 Durumlar

AES algoritmasının işlemleri durum olarak isimlendirilen iki boyutlu sekizli diziler üzerinde gerçekleşir. Durum, herbiri Nb (*blok uzunluğu / 32)* sekizli içeren 4 satırdan oluşur. s ile gösterilen durum dizisinde, her bir sekizli iki indise sahiptir (Satır numarası $r\,(0 \le r < 4)$ ve sütun numarası $c\,(0 \le c \le Nb)$). Durumdaki her bir dizinin gösterimi $s_{r,c}$ veya $s[r,c]$ şeklindedir. Bu standart için $Nb = 4$ yani $0 < c \le 4$' tür. Giriş yani $in_0, in_1, \ldots, in_{15}$ Şekil 3.13'teki gibi durum dizisine kopyalanmıştır. Şifre ve Ters-Şifre işlemleri durum dizisinde yürütülür, sonra son değeri $out_0, out_1, \ldots, out_{15}$ olarak gösterilen çıkışa kopyalanır [6].

Giriş Sekizlileri

in_0	in_4	in_8	in_{12}
in_1	in_5	in_9	in_{13}
in_2	in_6	in_{10}	in_{14}
in_3	in_7	in_{11}	$İn_{15}$

Durum Dizisi

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

Çıkış Sekizlileri

out_0	out_4	out_8	out_{12}
out_1	out_5	out_9	out_{13}
out_2	out_6	out_{10}	out_{14}
out_3	out_7	out_{11}	out_{15}

Şekil 3.13. Durum dizisi giriş ve çıkışlar

Şifreleyici veya ters şifreleyicinin başında giriş dizisi *in* durum dizisine aşağıdaki yönteme göre kopyalanır:

$$0 \le r < 4 \text{ ve } 0 \le c < Nb \text{ için, } s[r,c] = in[r + 4c]. \tag{3.12}$$

Şifreleyici veya ters şifreleyicinin sonunda ise, durum *out* çıkış dizisine aşağıdaki yönteme göre kopyalanır:

$$0 \le r < 4 \text{ ve } 0 \le c < Nb \text{ için, } out[r + 4c] = s[r,c] \tag{3.13}$$

Durum, kolon numarası c' nin indis olduğu tek boyutlu 32 bitlik bir sözcük dizisi $(w_0 \ldots w_3)$ olarak yorumlanabilir. Durum dört sözcük dizisi olarak aşağıdaki şekilde düşünülebilir [6]:

$$w_0 = s_{0,0}s_{1,0}s_{2,0}s_{3,0} \qquad w_0 = s_{0,2}s_{1,2}s_{2,2}s_{3,2} \tag{3.14}$$

$$w_0 = s_{0,1}s_{1,1}s_{2,1}s_{3,1} \qquad w_0 = s_{0,3}s_{1,3}s_{2,3}s_{3,3}$$

3.5.2 Matematiksel Önbilgiler

3.5.2.1 Toplama

Toplama işlemi XOR işlemi ile gerçekleştirilir ($1 \oplus 1 = 0$, $1 \oplus 0 = 1$, $0 \oplus 0 = 0$). Sonlu alan elemanlarında toplama işleminin alternatif bir gösterimi, sekizlilerdeki birbirine karşılık gelen bitlerin modül 2 toplamlarıdır. İki sekizlilik $\{a_0a_1a_2a_3a_4a_5a_6a_7\}$

dizisi ile $\{b_0 b_1 b_2 b_3 b_4 b_5 b_6 b_7\}$ dizisinin toplamı $c_i = a_i \oplus b_i$ şartı için $\{c_0 c_1 c_2 c_3 c_4 c_5 c_6 c_7\}$ dizisidir.

Toplama işlemi, polinomsal gösterim, ikili gösterim ve hexadecimal gösterim olmak üzere üç şekilde gösterilebilir. Aşağıda üç gösterim için örnek verilmiştir [6]:

i. Polinomsal gösterim: $(x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1)$
$= x^7 + x^6 + x^4 + x^2$

ii. İkili gösterim: {01010111} ⊕ {10000011} = {11010100}

iii. Hexadecimal gösterim: {57} ⊕ {83} = {d4}

3.5.2.2 Çarpma

$GF(2^8)$ uzayında $\bullet$ işareti ile gösterilen çarpma işlemi, polinomların çarpımının 8. dereceden indirgenemez bir polinoma göre modülüne karşılık gelir [6].

Tanım 3.3. Sadece kendisi ve bire bölünebilen polinomlara indirgenemez polinomlar denir. AES şifreleyicisinde kullanılan indirgenemez polinom

$$m(x) = x^8 + x^4 + x^3 + x + 1 \tag{3.15}$$

veya hexadecimal gösterimde {01} {1b}' dir.

Örnek 3.3. $\{57\} \bullet \{83\} = \{c1\}$' dir. Çünkü,

$$(x^6 + x^4 + x^2 + x + 1)(x^7 + x + 1) = x^{13} + x^{11} + x^9 + x^8 + x^7 + x^7 + x^5 + x^3 + x^2 + x + x^6 + x^4 + x^2 + x + 1$$

$$= x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$$

$$(x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1) \bmod (x^8 + x^4 + x^3 + x + 1)$$

$$= x^7 + x^6 + 1.$$

Yukarıda tanımlanan çarpma işlemi birleşmelidir ve {01} elemanı çarpımsal tanımlayıcı olarak tanımlanır.

Tanım 3.4. Sıfırdan farklı sekizinci dereceden küçük bir $b(x)$ ikili polinomunun çarpımsal tersi $b^{-1}(x)$ olarak tanımlanır.

$$b(x)a(x)+m(x)c(x)=1 \tag{3.16}$$

$$b^{-1}(x)=a(x) \bmod m(x)$$

$$a(x)\bullet(b(x)+c(x))=a(x)\bullet b(x)+a(x)\bullet c(x).$$

3.5.2.3 X ile Çarpma

$$b_7x^7+b_6x^6+b_5x^5+b_4x^4+b_3x^3+b_2x^2+b_1x+b_0 \tag{3.17}$$

polinomu ile *x* değerinin çarpımı sonucunda,

$$b_7x^8+b_6x^7+b_5x^6+b_4x^5+b_3x^4+b_2x^3+b_1x^3+b_0x \tag{3.18}$$

polinomu elde edilir.

$x\bullet b(x)$ işleminin sonucu, yukarıda ifade edilen sonucun $\bmod m(x)$ ile indirgenmesiyle elde edilir. $b_7=0$ ise sonuç zaten indirgenmiş biçimdedir. $b_7=1$ ise indirgeme işlemi indirgenecek polinomdan $m(x)$ polinomunun çıkarılması ile (yani XOR'lama ile) gerçekleştirilir. *x* ile çarpma işlemi (yani {00000010} veya {02}) sekizli seviyesinde sola kaydırma ve elde edilen değerin {1b} ile XOR işlemine tabi tutulmasıyla gerçekleştirilir. Sekizliler üzerindeki bu işlem *xtime()* ile gösterilir ve *x*'in yüksek dereceleriyle çarpma, *xtime()* işleminin tekrarlanmasıyla sağlanır. Elde edilen ara sonuçların toplanmasıyla, herhangi bir sabitle çarpma işlemi yapılmış olur [6].

Örnek 3.4.

$$\{57\}\bullet\{02\}=xtime(\{57\})=\{ae\}$$
$$\{57\}\bullet\{04\}=xtime(\{ae\})=\{47\}$$
$$\{57\}\bullet\{08\}=xtime(\{47\})=\{8e\}$$
$$\{57\}\bullet\{10\}=xtime(\{8e\})=\{07\}$$
$$\{57\}\bullet\{13\}=\{57\}\bullet(\{01\}\oplus\{02\}\oplus\{10\})$$

$$= \{57\} \oplus \{ae\} \oplus \{07\}$$
$$= \{fe\}$$

Bu işlem bit seviyesinde şu şekilde gösterilebilir:

$$\{57\} \bullet \{02\} = \{0101\ 0111\} \bullet \{0000\ 0010\}$$
$$\{x^6 + x^4 + x^2 + x + 1\} \bullet \{x\} = \{x^7 + x^5 + x^3 + x^2 + x\}$$
$$= \{1010\ 1110\}$$
$$= \{ae\}$$
$$\{57\} \bullet \{02\} = \{ae\}$$
$$\{57\} \bullet \{04\} = \{ae\} \bullet \{02\}$$
$$= (1010\ 1110) \bullet (0000\ 0010)$$
$$= \{x^7 + x^5 + x^3 + x^2 + x\} \bullet \{x\}$$
$$= \{x^8 + x^6 + x^4 + x^3 + x^2\}.$$

x ile çarpma işlemi bir bit sola kaydırma işleminden ibarettir. $b_7 = 1$ olduğundan $\{1b\}$ ile XOR işlemine tabi tutulur.

$$\{0101\ 1100\} \oplus \{0001\ 1011\} = \{0100\ 0111\} = \{47\}$$
$$\{57\} \cdot \{08\} = \{47\} \cdot \{02\}$$
$$= \{0100\ 0111\} \cdot \{0000\ 0010\}$$
$$= \{x^6 + x^2 + x + 1\} \bullet \{x\}$$
$$= \{x^7 + x^3 + x^2 + x\}.$$

$b_7 = 0$ olduğundan sonuç zaten indirgenmiş biçimdedir.

$$= \{1000\ 1110\} = \{8e\}$$
$$\{57\} \cdot \{10\} = \{8e\} \bullet \{02\} =$$
$$= \{1000\ 1110\} \bullet \{0000\ 0010\}$$

$\{8e\}$'de $b_7 = 1$ olduğundan x ile çarpmadan yani sola kaydırmadan elde edilecek sonuç $\{1b\}$ ile XOR' lanır.

$$= \{0001\ 1100\}$$

= {0001 1100} ⊕ {0001 1011}

= {0000 0111}

= {07}

3.5.3 GF(2^8) Cisminde Bulunan Katsayılara Sahip Polinomlar

Katsayıları sonlu alan elemanları olan dört terimli polinomlar,

$$a(x) = a_3x^3 + a_2x^2 + a_1x + a_0 \tag{3.19}$$

şeklinde tanımlanabildikleri gibi, bir sözcük olarak, yani $[a_0, a_1, a_2, a_3]$ biçiminde de gösterilebilirler. Dört terimli polinomların katsayıları sonlu elemanlar (yani bitler yerine sekizliler gibi) olup, çarpma işleminde farklı bir polinom indirgemesi kullanılır [6].

Toplama ve çarpma işlemlerinin sunuluşunda,

$$b(x) = b_3x^3 + b_2x^2 + b_1x + b_0 \tag{3.20}$$

dört terimli polinomu kullanılacaktır.

Tanım 3.5. Toplama, x'in benzer kuvvetlerinin sonlu cisim katsayılarının toplanmasıyla olur. Bu toplama, her bir sözcükteki uygun sekizlilerin arasındaki XOR işlemidir. Başka bir deyişle bütün sözcük değerlerinin XOR'lanmasıdır.

$$a(x) + b(x) = (a_3 \oplus b_3)x^3 + (a_2 + b_2)x^2 + (a_1 + b_1)x + (a_0 + b_0) \tag{3.21}$$

Tanım 3.6. Çarpma işlemi iki aşamada gerçekleştirilir:

i. Polinomsal çarpım $c(x) = a(x) \bullet b(x)$ cebirsel olarak genişletilir ve benzer üsler bir araya getirilir.

$$c(x) = c_6x^6 + c_5x^5 + c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c_0 \tag{3.22}$$

$$c_0 = a_0 \bullet b_0$$
$$c_4 = a_3 \bullet b_1 \oplus a_2 \bullet b_2 \oplus a_1 \bullet b_3$$
$$c_1 = a_1 \bullet b_0 \oplus a_0 \bullet b_1 \qquad c_5 = a_3 \bullet b_2 \oplus a_2 \bullet b_3$$
$$c_2 = a_2 \bullet b_0 \oplus a_1 \bullet b_1 \oplus a_0 \bullet b_2 \qquad c_6 = a_3 \bullet b_3$$
$$c_3 = a_3 \bullet b_0 \oplus a_2 \bullet b_1 \oplus a_1 \bullet b_2 \oplus a_0 \bullet b_3$$

ii. $c(x)$ polinomu dördüncü dereceden bir polinom moduna indirgenir. Sonuçta dörtten küçük bir dereceye indirgenmiş olur. AES şifreleyicisinde indirgeme işlemi x^4+1 polinomuyla gerçekleştirilir.

$$x^i \bmod (x^4+1) = x^{i \bmod 4}. \tag{3.23}$$

Tanım 3.7. $a(x)$ ve $b(x)$ polinomlarının modüler çarpımı 4. dereceden $d(x)$ polinomu ile gösterilirse $(d(x)=a(x)\otimes b(x))$,

$$d(x) = d_3x^3 + d_2x^2 + d_1x + d_0 \tag{3.24}$$

$$d_0 = (a_0 \bullet b_0) \oplus (a_3 \bullet b_1) \oplus (a_2 \bullet b_2) \oplus (a_1 \bullet b_3)$$

$$d_1 = (a_1 \bullet b_0) \oplus (a_0 \bullet b_1) \oplus (a_3 \bullet b_2) \oplus (a_2 \bullet b_3)$$

$$d_2 = (a_2 \bullet b_0) \oplus (a_1 \bullet b_1) \oplus (a_0 \bullet b_2) \oplus (a_3 \bullet b_3)$$

$$d_3 = (a_3 \bullet b_0) \oplus (a_2 \bullet b_1) \oplus (a_1 \bullet b_2) \oplus (a_0 \bullet b_3)$$

olur. Eğer $a(x)$ sabit bir polinom ise (3.24)'deki işlem matris biçiminde gösterilebilir (bknz.(3.25)).

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \tag{3.25}$$

3.5.3.1 AES Şifreleme Algoritması

AES algoritmasında girdi bloğu, çıktı bloğu ve durumun uzunluğu 128 bittir ve $Nb=4$'tür. Yani durumdaki 32-bit sözcük sayısını (sütun sayısını) gösterir. AES şifreleme algoritmasında şifre anahtarı (K)'nın uzunluğu Nk=4, 6 veya 8'dir. Yani şifre anahtarındaki 32-bit sözcük sayısını (sütun sayısını) gösterir [6].

AES şifreleme algoritmasının döngü sayısı (Nr) anahtarın boyutuna bağlıdır. $Nk=4$ iken $Nr=10$, $Nk=6$ iken $Nr=12$, $Nk=8$ iken $Nr=14$'tür. Anahtar–blok–döngü kombinasyonları Tablo 3.7'de verilmiştir.

Tablo 3.7. Anahtar - Blok - Döngü Kombinasyonları

	Anahtar Uzunluğu (Nk)	Blok Uzunluğu (Nb)	Döngü Sayısı (Nr)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

AES algoritması, hem şifre hem de ters-şifre için 4 farklı sekizli-yönelimli dönüşümden oluşan bir döngü fonksiyonu kullanır:

i. Yerine koyma (substitution) tablosu (S-box) kullanarak sekizlileri yerine koymak;

ii. Durum dizisinin satırlarını farklı uzaklıklarda kaydırmak;

iii. Durum dizisi sütunlarındaki verileri karıştırmak;

iv. Duruma döngü anahtarı (round key) eklemek.

3.5.3.2 Anahtar Planlama

AES şifreleme algoritması şifre anahtarı K'yı bir anahtar planlama algoritmasına tabi tutarak, alt döngülerinde kullanmak üzere bir anahtar çizelgesi (key schedule) oluşturur. $Nb(Nr+1)$ sözcük üreten anahtar planlama algoritması Nb sözcükten oluşan bir başlangıç kümesi gerektirir ve her bir Nr döngü anahtar verisinin Nb sözcüğüne ihtiyaç duyar. Sonuç olarak elde edilen anahtar çizelgesi 4 sekizlilik sözcüklerden oluşan doğrusal bir dizi içerir ve $0 \le i < Nb(Nr+1)$ şartı için $[w_i]$ olarak gösterilir.

Tanım 3.8. *SubWord()*, 4 sekizlilik sözcüğü giriş değeri olarak kullanan ve 4 sekizlinin her birini bir S-kutusundan geçirerek 4-sekizlilik çıkış sözcüğü oluşturan bir fonksiyondur.

```
public void KeyExpansion(byte[] key /*[4*Nk]*/, word[] w/*[Nb*(Nr+1)]*/, int Nk)
{
   word temp;
   i = 0;
   while (i < Nk)
   {
      w[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3]);
      i = i+1;
   }
   i = Nk;
   while (i < Nb * (Nr+1))
   {
      temp = w[i-1];
      if (i % Nk = 0)
      {
         temp = SubWord(RotWord(temp)) ^ Rcon[i/Nk];
      }
      else if (Nk > 6 and i % Nk = 4)
      {
         temp = SubWord(temp);
      }
      w[i] = w[i-Nk] ^ temp;
      i = i + 1;
   }
}
```

Algoritma 3.6. Anahtar Planlama Algoritması Örnek Kaynak Kodu

Tanım 3.9: *RotWord()* fonksiyonu $[a_0, a_1, a_2, a_3]$ sözcüğünü giriş değeri olarak alır ve dönüştürme permütasyonuna tabi tutarak $[a_{1,}, a_2, a_3, a_0]$ değerini üretir.

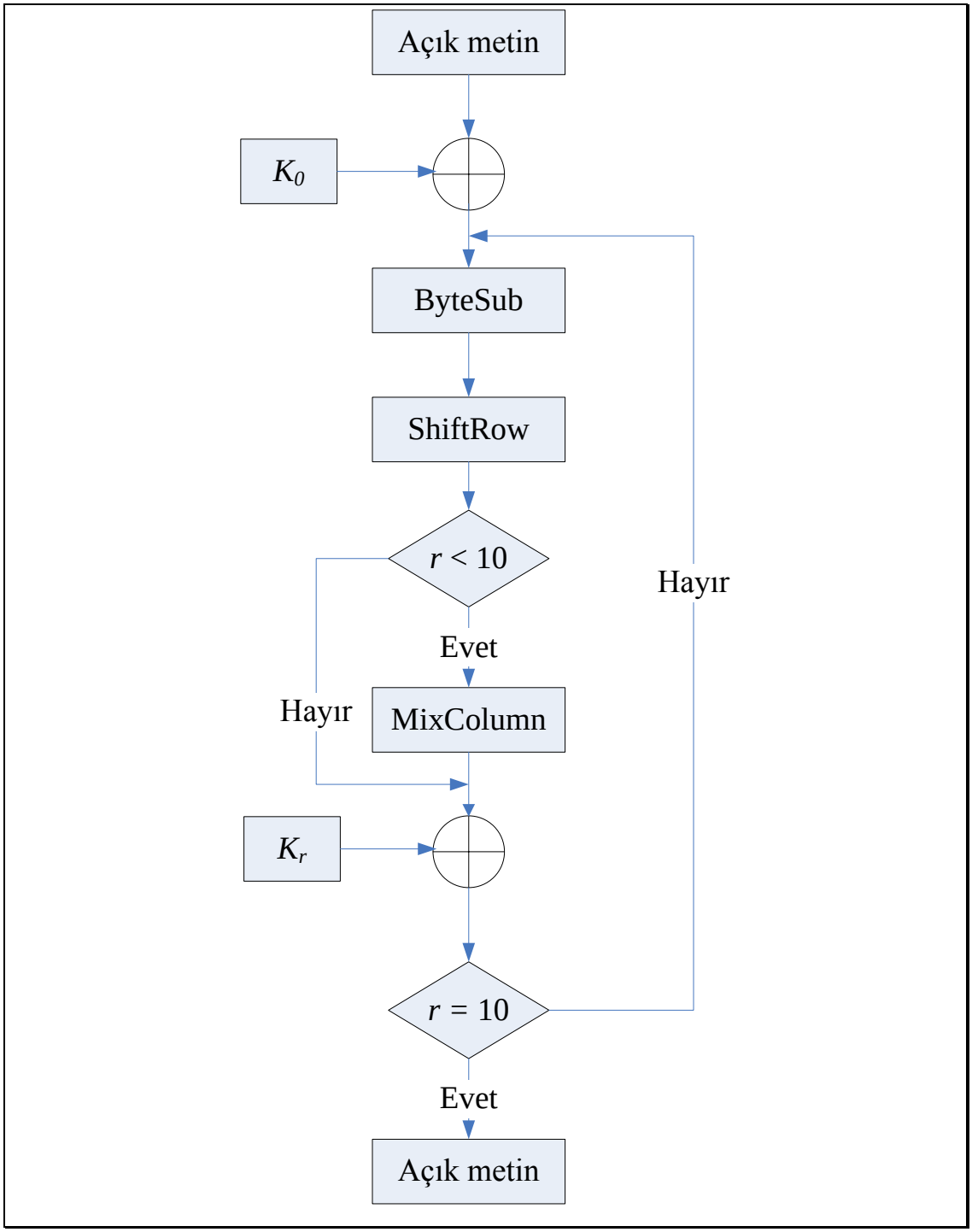
Döngü sabit sözcük dizisi $Rcon[i]$, $GF(2^8)$ sonlu cisiminde $[x^{i-1}, \{00\}, \{00\}, \{00\}]$ değerlerinden oluşur ve. x değeri $\{02\}$' dir.

Algoritma 3.6'dan görüldüğü gibi, genişletilmiş anahtarın ilk Nk sözcükleri şifre anahtarından oluşmaktadır. Takip eden herbir $w[i]$ sözcüğü, kendisinden bir önce gelen $w[i-1]$ sözcüğü ile $w[i-Nk]$' nin XOR'udur. Nk ' nın katları olan sözcükler için dönüşüm, $w[i-1]$ ve $Rcon[i]$' nin XOR' udur. Bu dönüşüm *RotWord()* ve *SubWord()* dönüşümlerini içerir. 256-bitlik şifre anahtarlarının $(Nk=8)$ anahtar genişletme (key expansion) algoritması 128 ve 192 bitlik şifre anahtarlarınınkinden biraz farklıdır. Eğer $Nk=8$ ise ve $(i-4)$, Nk 'nın katıysa, *SubWord()* fonksiyonu $w[i-1]$'e uygulanır [6].

3.5.3.3 Şifreleyici

Şifreleyici açık metin verisinin, giriş değeri olarak durum dizisine kopyalanması ile başlar. Başlangıç döngü anahtarının eklenmesinden sonra, durum dizisi son döngüsü $(Nr-1)$ döngüden biraz farklı olan bir döngü fonksiyonunun anahtar uzunluğuna bağlı olarak 10,12 veya 14 kez çalıştırılması ile dönüşüme tabi tutulur. Dönüşümlerden sonra elde edilen son durum dizisi çıkış değerini oluşturur.

```
function  AESK(M)
begin
        (K0,.....K10)  expand (K)    anahtar genişletilerek 10 tane
                                     alt anahtar üretilir.
        s       M  XOR  K0           ilk anahtar yani şifreleme anahtarı açık
                                     metinle XOR' lanır.
        for r = 1 to 10 do           (10) r döngü için ;
        s       S(s)                 durumdaki değerlerin S-kutusundaki karşılıkları
                                     bulunur.
        s       shift – rows (s)     durumdaki değerlerin satırları shift-rows'la
                                     kaydırılır.
        if r ≤ 9 then
          s     mix – cols (s)       mix-cols işlemi ilk 9 döngüde geçerli
                                     9. döngüde yok
        end if
          s     s  XOR Kr            son olarak 10. anahtar durum ile XOR lanır.
                end for
                return s
end
```



Algoritma 3.7. AES Yüksek Seviyeli Yapısı Akış Diyagramı

Döngü fonksiyonu anahtar genişletme algoritması kullanılarak elde edilen ve tek boyutlu dört sekizlilik sözcüklerden oluşan bir anahtar çizelgesinin kullanılması ile parametrik hale getirilmiştir.

Rijndael döngü fonksiyonu dört dönüşümden oluşur [69]:

i. *SubBytes dönüşümü*: 8×8'lik bir S-kutusu tüm sekizlilere uygulanır.

```
/// <summary>
/// Veriyi 16 sekizlilik bloklar halinde şifreler.
/// 16 sekizlilik şifreli metin bloğu döner.
/// </summary>
/// <param name="input">şifrelenecek 16 sekizlilik karakter bloğu</param>
/// <returns>16 sekizli uzunluğunda şifrelenmiş veri bloğu</returns>
private byte[] Cipher(byte[] input)  // 16-bit giriş verisini şifreler
{
 byte[] output = new byte[16];
 try
 {
        // durum = giriş
        this.State = new byte[4,Nb];  // always [4,4]
        for (int i = 0; i < (4 * Nb); ++i)
        {
                this.State[i % 4, i / 4] = input[i];
        }
        AddRoundKey(0);
        for (int round = 1; round <= (Nr - 1); ++round)  // main round loop
        {
                SubBytes();
                ShiftRows();
                MixColumns();
                AddRoundKey(round);
        }  // ana döngü fonksiyonu
        SubBytes();
        ShiftRows();
        AddRoundKey(Nr); // çıkış = durum
        for (int i = 0; i < (4 * Nb); ++i)
        {
                output[i] = this.State[i % 4, i / 4];
        }
 }
 catch (Exception excep)
 { AES_ShowError(excep, "Cipher"); }
  return output;
} // Cipher()}
```

Algoritma 3.8. AES Şifreleme Algoritması Örnek Kaynak Kodu

i. *ShiftRows dönüşümü*: Dizinin satırlarınn ötelendiği bir doğrusal karıştırma fonksiyonudur.

ii. *MixColumn dönüşümü*: Sütun karıştırma işleminin yapıldığı bir doğrusal karıştırma fonksiyonudur.

iii. AddRoundKey dönüşümü: Anahtar sekizlileri dizinin tüm sekizlileri ile XOR işlemine tabi tutulur.

Son döngüde dizinin sütun karıştırılması atlanır. AES' in yüksek – seviyeli yapısı Algoritma 3.7'de gösterilmiştir [70].

3.5.3.4 SubBytes() Dönüşümü

SubBytes() dönüşümü, durum dizisinin tüm sekizlileri üzerinde S-kutusu kullanarak bağımsız işlem yapan doğrusal olmayan bir sekizli yerine koyma işlemidir. Tersi alınabilen bir dönüşüm olan S-kutusu iki dönüşümün birleşiminden oluşur:

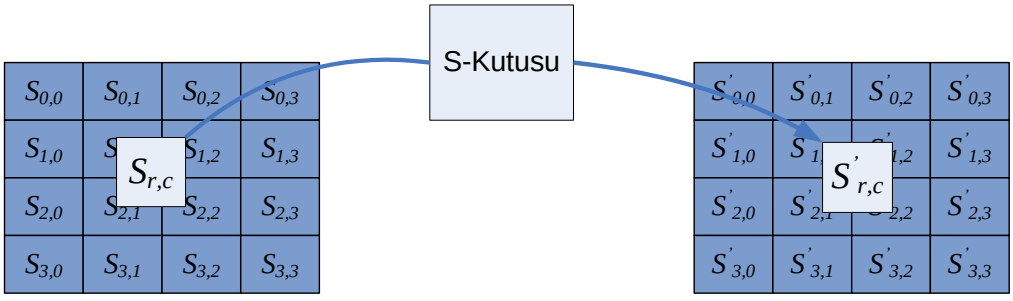
i. Önce S-kutusu giriş değerinin $GF(2^8)$ sonlu cisminde çarpmaya göre tersi alınır. $\{00\}$ elemanının tersi kendisidir.

ii. Sonra aşağıdaki affine dönüşüm uygulanır:

$$0 \le i < 8 \text{ için; } b_i' = b_i \oplus b_{(i+4) mod 8} \oplus b_{(i+5) mod 8} \oplus b_{(i+6) mod 8} \oplus b_{(i+7) mod 8} \oplus c_i \quad (3.26)$$

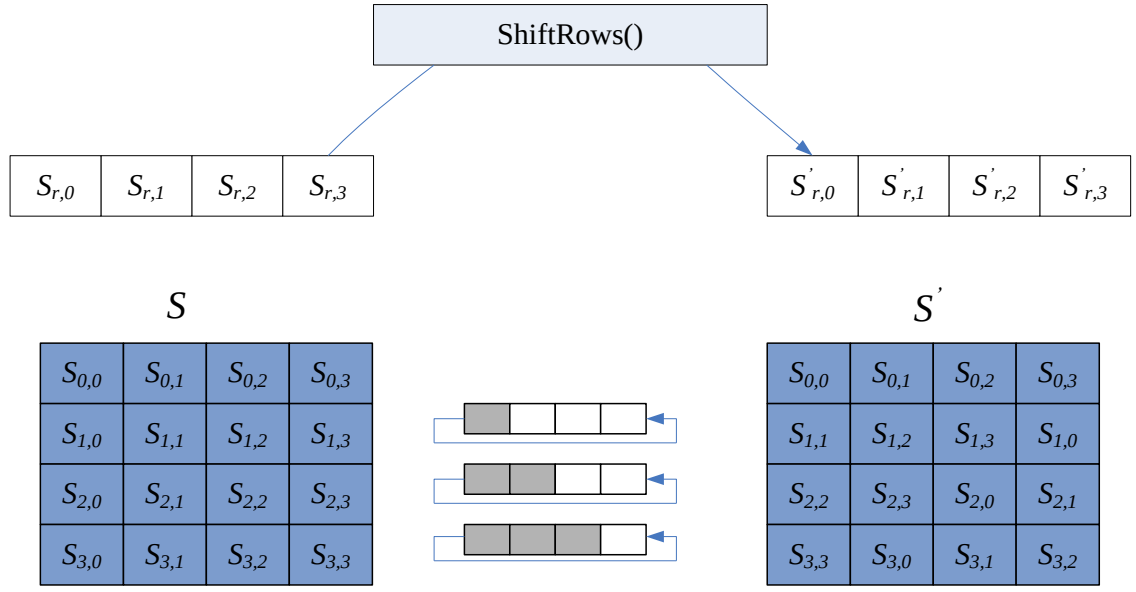
(3.26)'da gösterilen affine dönüşüm elemanı matris biçiminde aşağıdaki gibi ifade edilebilir [6]:

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad (3.27)$$



Şekil 3.14. SubBytes Dönüşümü

3.5.3.5 ShiftRows() Dönüşümü



Şekil 3.15. ShiftRows Dönüşümü

ShiftRows() dönüşümünde, durum dizisinin ilk satırı sabit olarak kalırken, son üç satırı farklı sayıda sekizliler üzerinde dairesel olarak kaydırılır. ShiftRows() dönüşümü matematiksel olarak şu şekilde ifade edilir [6]:

$$0 < r < 4 \text{ ve } 0 \le c < Nb \text{ için, } S'_{r,c} = S_{r,(c+shift(r,Nb))\bmod Nb} \tag{3.28}$$

$shift(r, Nb)$ değeri satır numarasına bağlıdır (r satır numarası).

$$shift(1,4) = 1 \; ; \; shift(2,4) = 2 \; ; \; shift(3,4) = 3 \tag{3.29}$$

3.5.3.6 MixColumns Dönüşümü

MixColumns() dönüşümü durum üzerinde sütun sütun işlem görür ve her sütunun 4 terimli polinom olduğunu düşünür. Kolonlar $GF(2^8)$ sonlu uzayında bir polinom olarak düşünülür ve (3.30) ifadesindeki $a(x)$ polinomu ile $x^4 + 1$ modülünde çarpılır.

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}. \tag{3.30}$$

İlgili ifade matris çarpımı olarak şu şekilde yazılabilir [6]:

$$s'(x) = a(x) * s(x)$$

$$0 \le c < Nb \text{ için } \begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 01 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \quad (3.31)$$

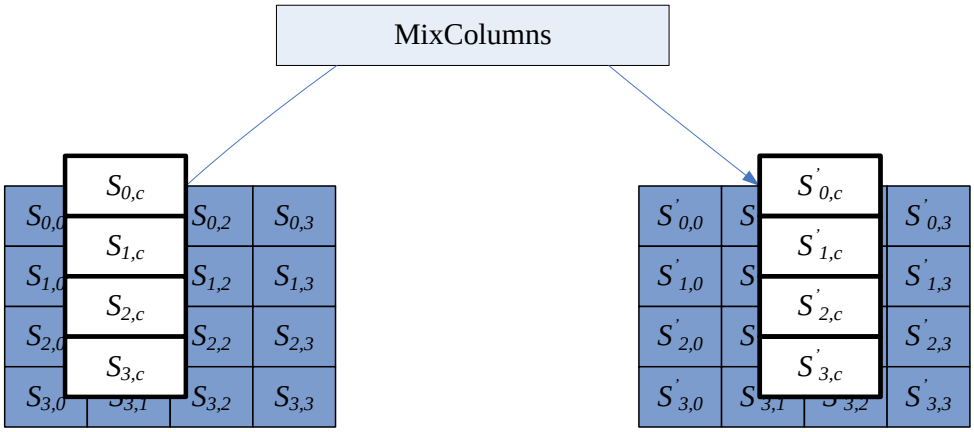
Çarpmanın sonucunda bir sütundaki 4 sekizli aşağıdaki gibi yer değiştirir:

$$s'_{0,c} = (\{02\} \bullet s_{0,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$$

$$s'_{1,c} = s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c}$$

$$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c})$$

$$s'_{3,c} = (\{03\} \bullet s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \bullet s_{3,c})$$



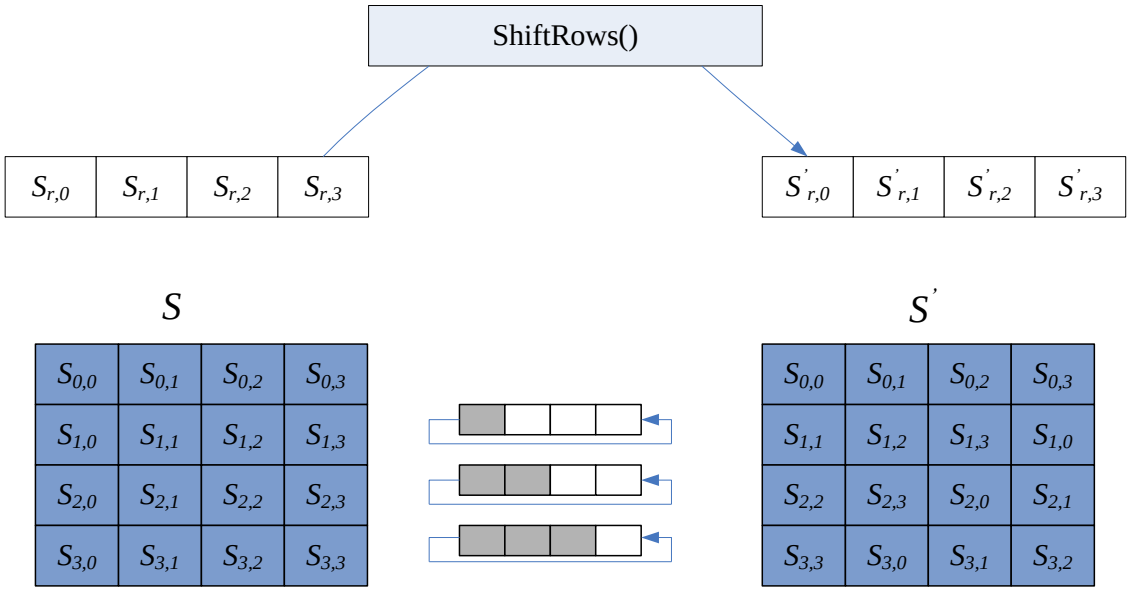
Şekil 3.16. MixColumns() Dönüşümü

3.5.3.7 AddRoundKey() Dönüşümü

AddRoundKey() dönüşümünde bir döngü anahtarı bit seviyesinde basit bir XOR işlemi ile duruma eklenir. Her bir döngü anahtarı, anahtar çizelgesinden Nb sözcük içerir. Bu Nb sözcük durumun kolonlarına şu şekilde eklenir:

$$0 \le c \le Nb \text{ için }, [s'_{0,c}, s'_{1,c}, s'_{2,c}, s'_{3,c}] = [s_{0,c}, s_{1,c}, s_{2,c}, s_{3,c}] \oplus [w_{round*Nb+c}] \quad (3.32)$$

w_i anahtar çizelgesi sözcükleri.



Şekil 3.17. AddRoundKey() Dönüşümü

3.5.3.8 Ters Şifre

Şifre dönüşümleri tersine çevrilebilir ve ters sırada ters şifreye uygulanır. Ters şifrede kullanılan dönüşümler olan InvShiftRows(), InvSubBytes(), InvMixColumns() ve AddRoundKey() dönüşümleri durum üzerinde işlem yaparlar. Ters şifre, Algoritma 3.9'da pseudo koduyla açıklanmaktadır.

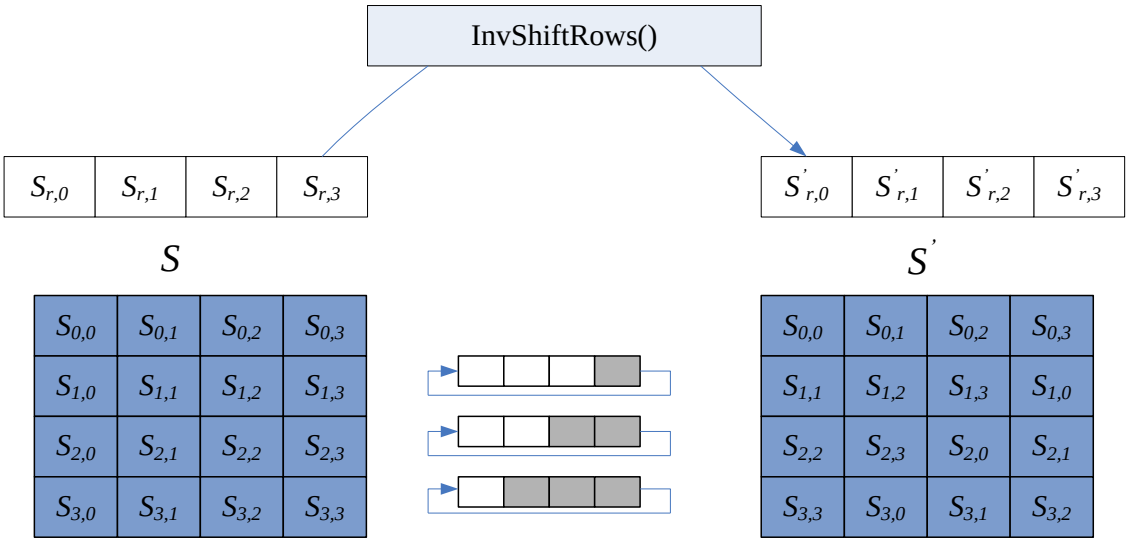
3.5.3.9 InvShiftRows() Dönüşümü

InvShiftRows() dönüşümü ShiftRows() dönüşümünün tersidir. Durumun son üç satırındaki sekizliler dairesel olarak farklı sayıda sekizli kadar kaydırılırlar. İlk satır $(r = 0)$ kaydırılmaz. Alttaki üç satır $Nb - shift(r, Nb)$ sekizli kadar dairesel olarak kaydırılır. Kaydırma değeri $shift(r, Nb)$ (3.33)'deki denklemde gösterildiği gibi satır sayısına bağlıdır. InvShiftRows() dönüşümü şu şekilde çalışır (bknz.Şekil 3.18):

$$0 < r < 4 \text{ ve } 0 \le c < Nb \text{ için } s'_{r,(c+shift(r,Nb))\bmod Nb} = s_{r,c} \qquad (3.33)$$

```
        /// <summary>
        /// 16 sekizlilik veri bloğunu deşifreler.
        /// </summary>
        /// <param name="input">16 sekizlilik deşifrelenecek blok</param>
        /// <returns>16 sekizlilik deşifrelenmiş blok</returns>
        private byte[] InvCipher(byte[] input)  // 16 bitlik veriyi deşifreler.
        {
                byte[] output = new byte[16];
                try
                {
                        // durum = giriş
                        this.State = new byte[4,Nb];
                        for (int i = 0; i < (4 * Nb); ++i)
                        {
                                this.State[i % 4, i / 4] = input[i];
                        }
                        AddRoundKey(Nr);
                        for (int round = Nr-1; round >= 1; --round)  // ana döngü
                        {
                                InvShiftRows();
                                InvSubBytes();
                                AddRoundKey(round);
                                InvMixColumns();
                        }
                        InvShiftRows();
                        InvSubBytes();
                        AddRoundKey(0);
                        // çıkış = durum
                        for (int i = 0; i < (4 * Nb); ++i)
                        {
                                output[i] = this.State[i % 4, i / 4];
                        }
                }
                catch (Exception excep)
                { AES_ShowError(excep, "InvCipher"); }
                        return output;
        }  // InvCipher()
```

Algoritma 3.9. Ters Şifre Örnek Kaynak Kodu



Şekil 3.18. InvShiftRows() Dönüşümü

3.5.3.10 InvSubBytes() Dönüşümü

InvSubBytes() dönüşümü, S-kutusunun durumun herbir sekizlisine uygulandığı sekizli yerine koyma dönüşümünün tersidir. Bu dönüşümde önce (3.26)'daki affine dönüşümün tersi uygulanır. Elde edilen değerin $GF(2^8)$ uzayında çarpımsal tersi alınarak istenilen değere ulaşılır.

InvSubBytes() dönüşümünde kullanılan ters S-kutusu Tablo 3.8' de verilmiştir [6].

3.5.3.11 InvMixColumns() Dönüşümü

InvMixColumns() dönüşümü MixColumns() dönüşümünün tersidir. InvMixColumns() durum üzerinde sütun sütun işlem yapar ve her sütunun dört terimli polinom olduğunu düşünür. Sütunlar $GF(2^8)$' deki polinomlar olarak düşünülür ve $a^{-1}(x)$ sabit polinomuyla x^4+1 modülünde çarpılır [6]:

$$a^{-1}(x)=\{0b\}x^3+\{0d\}x^2+\{09\}x+\{0e\}. \tag{3.34}$$

Bu ifade matris çarpımı olarak da yazılabilir:

Tablo 3.8. InvSubBytes() Dönüşümü[6]

		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	0	52	09	6	D	30	36	A	38	B	40	A	9E	81	F3	D	F
	1	7	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	D	E9	C
	2	54	7B	94	32	A	C2	23	3D	E	4C	95	0B	42	F	C	4E
	3	08	2E	A	66	28	D9	24	B2	76	5B	A	49	6D	8B	D	25
	4	72	F8	F6	64	86	68	98	16	D	A	5	A	5D	65	B	92
	5	6	70	48	50	F	E	B9	D	5	15	46	57	A7	8	9	84
x	6	90	D8	A	00	8C	B	D	0A	F7	E4	58	05	B8	B3	45	06
	7	D	2C	1E	8F	C	3F	0F	02	C	A	B	03	01	13	8	6
	8	3	91	11	41	4F	67	D	E	97	F2	C	C	F0	B4	E6	73
	9	96	A	74	22	E7	A	35	85	E	F9	37	E8	1C	75	D	6E
	A	47	F1	1	71	1	29	C5	89	6F	B7	62	0E	A	18	B	1
	B	F	56	3E	4	C6	D2	79	20	9	D	C0	FE	78	C	5	F4
	C	1F	D	A	33	88	07	C7	31	B	12	10	59	27	80	E	5F
	D	60	51	7F	A	19	B5	4	O	2	E5	7	9F	93	C9	9	E
	E	A	E0	3B	4	A	2A	F5	B0	C	E	B	3C	83	53	99	61
	F	17	2B	04	7	B	77	D	26	E	69	14	63	55	21	0	7

$$s'(x) = a^{-1}(x) \otimes s(x)$$

$$0 \le c < Nb \text{ için, } \begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \quad (3.35)$$

Bu çarpımın sonucunda bir sütundaki 4 sekizli aşağıdaki ifadede gösterildiği gibi yer değiştirir.

[6] AES S-kutusu Tablo 5.2'de gösterilmiştir.

$$s'_{0,c} = (\{0e\} \bullet s_{0,c}) \oplus (\{0b\} \bullet s_{1,c}) \oplus (\{0d\} \bullet s_{2,c}) \oplus (\{09\} \bullet s_{3,c})$$

$$s'_{1,c} = (\{09\} \bullet s_{0,c}) \oplus (\{0e\} \bullet s_{1,c}) \oplus (\{0b\} \bullet s_{2,c}) \oplus (\{0d\} \bullet s_{3,c})$$

$$s'_{2,c} = (\{0d\} \bullet s_{0,c}) \oplus (\{09\} \bullet s_{1,c}) \oplus (\{0e\} \bullet s_{2,c}) \oplus (\{0b\} \bullet s_{3,c})$$

$$s'_{3,c} = (\{0b\} \bullet s_{0,c}) \oplus (\{0d\} \bullet s_{1,c}) \oplus (\{09\} \bullet s_{2,c}) \oplus (\{0e\} \bullet s_{3,c})$$

3.5.3.12 Denk Ters Şifre (Equivalent Inverse Cipher)

Algoritma 3.10'da gösterilen ters şifredeki ardışık dönüşümler, denk ters şifreye göre farklı olmakla birlikte, şifreleme ve deşifreleme için anahtar çizelgesinin şekli aynı kalmaktadır.

AES'in birçok özelliği, denk ters şifreleyicinin normal şifreleyici ile tek farkı dönüşümlerin tersi olmak koşuluyla, aynı dönüşümler dizisini takip etmesine olanak tanır. Bu durum anahtar çizelgesinde yapılan bir değişiklikle sağlanmıştır [6].

AES şifreleyicisinin denk ters şifreleyicisinin kullanımına imkan veren iki özelliği şunlardır:

i. ShiftRows() dönüşümünden sonra SubBytes() dönüşümünün gelmesi ile, SubBytes() dönüşümünden sonra ShiftRows() dönüşümünün gelmesi birbirine denktir. Bu tersleri için de geçerlidir. Yani InvSubBytes() ve InvShiftRows() için.

ii. MixColumns() ve InvMixColumns() işlemleri sütun girdisiyle doğrusaldırlar. Yani bu şu anlama gelir:

InvMixColumns(state XOR Round Key) =
InvMixColumns(state) XOR InvMixColumns(Round Key).

Bu özellikler InvSubBytes() ve InvShiftRows() dönüşümlerinin sırasının yer değiştirmesine izin verir. AddRoundKey() ve InvMixColumns() dönüşümlerinin sırası da yer değiştirebilir. Çünkü deşifreleme anahtar çizelgesinin sütunları (sözcükleri) InvMixColumns() dönüşümünü kullanarak değiştirilir.

Denk ters şifreleyici InvSubBytes() ile InvShiftRows() dönüşümünün sırasının yer değiştirmesi ve AddRoundKey() ile InvMixColumns() dönüşümlerinin sırasının yer değiştirmesi olarak tanımlanabilir.

```
/// <summary>
/// 16 sekizlilik veri bloğunu deşifreler.
/// </summary>
/// <param name="input">16 sekizlilik deşifrelenecek blok</param>
/// <returns>16 sekizlilik deşifrelenmiş blok</returns>
private byte[] EqInvCipher(byte[] input)  // 16 bitlik veriyi deşifreler.
{
        byte[] output = new byte[16];
        try
        {
                // durum = giriş
                this.State = new byte[4,Nb];
                for (int i = 0; i < (4 * Nb); ++i)
                {
                        this.State[i % 4, i / 4] = input[i];
                }
                AddRoundKey(Nr);
                for (int round = Nr-1; round >= 1; --round)  // ana döngü
                {
                        InvSubBytes();
                        InvShiftRows();
                        InvMixColumns();
                        AddRoundKey(round);
                }
                InvSubBytes();
                InvShiftRows();
                AddRoundKey(0);
                // çıkış = durum
                for (int i = 0; i < (4 * Nb); ++i)
                {
                        output[i] = this.State[i % 4, i / 4];
                }
        }
        catch (Exception excep)
        {
                AES_ShowError(excep, "EqInvCipher"); }
                return output;
        }  // EqInvCipher()
```

Algoritma 3.10. Ters Şifre Dengi Örnek Kaynak Kodu

BÖLÜM 4

4 İNCELENEN KRİPTANALİZ YÖNTEMLERİ

4.1 Diferansiyel Kriptanaliz

Diferansiyel kriptanaliz [19] [71] [72] farklı sınıflarda kriptosistemleri kırmada kullanılan günümüzde bilinen en önemli saldırı yöntemlerinden biridir. İsrail'li araştırmacılar Eli Biham ve Adi Shamir tarafından 1990 yılında bulunmuştur. Bu kriptanaliz yöntemi açık metin ikilileri farkının, bunlara karşılık gelen şifreli metin ikilileri üzerindeki etkisini kullanarak analiz yapar. Bu farklar olası anahtarlara ihtimal atamak ve ihtimali en yüksek anahtarları belirlemek için kullanılır. Aynı farka sahip olan bir çok açık metin ikilisini ve karşı gelen şifreli metin ikililerini kullanır [19].

Diferansiyel kriptanaliz, kriptanalist açık metni seçebildiği ve şifreli metni de elde edebildiği durumlarda (seçilen açık metin kriptanalizi) verimlidir. Bilinen açık metin diferansiyel kriptanalizi de olasıdır; ancak çoğunlukla bilinen metin çiftlerinin boyutu çok uzun olmaktadır. Yöntem, farkları sabit olan açık metin ve şifreli metin çiftlerini arar ve kriptosistemin diferansiyel davranışını inceler.

İki şifreli metin arasındaki fark, bu şifreli metinlere karşılık gelen açık metinler arasındaki farkın bir fonksiyonu olarak gözlemlenir. Birkaç döngü boyunca incelenen karekteristik olarak isimlendirilen en yüksek olasılıktaki diferansiyeli sağlayan açık metin ve şifreli metin ikilileri incelenir.

Tanım 4.1. X_1 ve X_2 arasındaki fark, $X_1 \oplus X_2$ olarak tanımlanır (Bit bazında XOR). Bazı kriptosistemlerde fark hesabı XOR' dan farklı bir işlemle de tanımlanabilir.

for $(L_1, L_2) \leftarrow (0,0)$ to (F, F) — 4 bitlik L_1 ve L_2 değerlerinin oluşturduğu olası tüm

do $Count[L_1, L_2] \leftarrow 0$ — anahtar çiftlerinin sayaçları sıfırlanır.

for each (x,y,x*,y*) $\in \tau$

if $((y_{(1)} = (y_{(1)})^*)$ and $((y_{(3)} = (y_{(3)})^*)$ — Diferansiyel Karakteristik ile veri filtrelenir.

for $(L_1, L_2) \leftarrow (0,0)$ to (F, F) — Karakteristiği sağlayan veriler için tüm olası anahtar değerleri ile analiz edilir.

do then

$v^4_{(2)} \leftarrow L_1 \oplus y_{(2)}$

$v^4_{(4)} \leftarrow L_2 \oplus y_{(4)}$

$u^4_{(2)} \leftarrow \pi_S^{-1}(v^4_{(2)})$

$u^4_{(4)} \leftarrow \pi_S^{-1}(v^4_{(4)})$

do $(v^4_{(2)})^* \leftarrow L_1 \oplus (y_{(2)})^*$

$(v^4_{(4)})^* \leftarrow L_2 \oplus (y_{(4)})^*$

$(u^4_{(2)})^* \leftarrow \pi_S^{-1}((v^4_{(2)})^*)$

$(u^4_{(4)})^* \leftarrow \pi_S^{-1}((v^4_{(4)})^*)$

$(u^4_{(2)})' \leftarrow u^4_{(2)} \oplus (u^4_{(2)})^*$

$(u^4_{(4)})' \leftarrow u^4_{(4)} \oplus (u^4_{(4)})^*$

if $((u^4_{(2)})' = 0101)$ and $((u^4_{(4)})' = 0101)$ — Doğru değeri veren anahtar değerleri için sayaç değeri 1 artırılır

then $Count[L_1, L_2] \leftarrow Count[L_1, L_2] + 1$

$max \leftarrow -1$

for $(L_1, L_2) \leftarrow (0,0)$ to (F, F) — Olasılığı en yüksek olan anahtar değeri belirlenir.

do if $Count[L_1, L_2] > max$

then $max = Count[L_1, L_2]$

$max\,key = (L_1, L_2)$

$output(max\,key)$

Algoritma 4.1. Diferansiyel Kriptanaliz Algoritması[7]

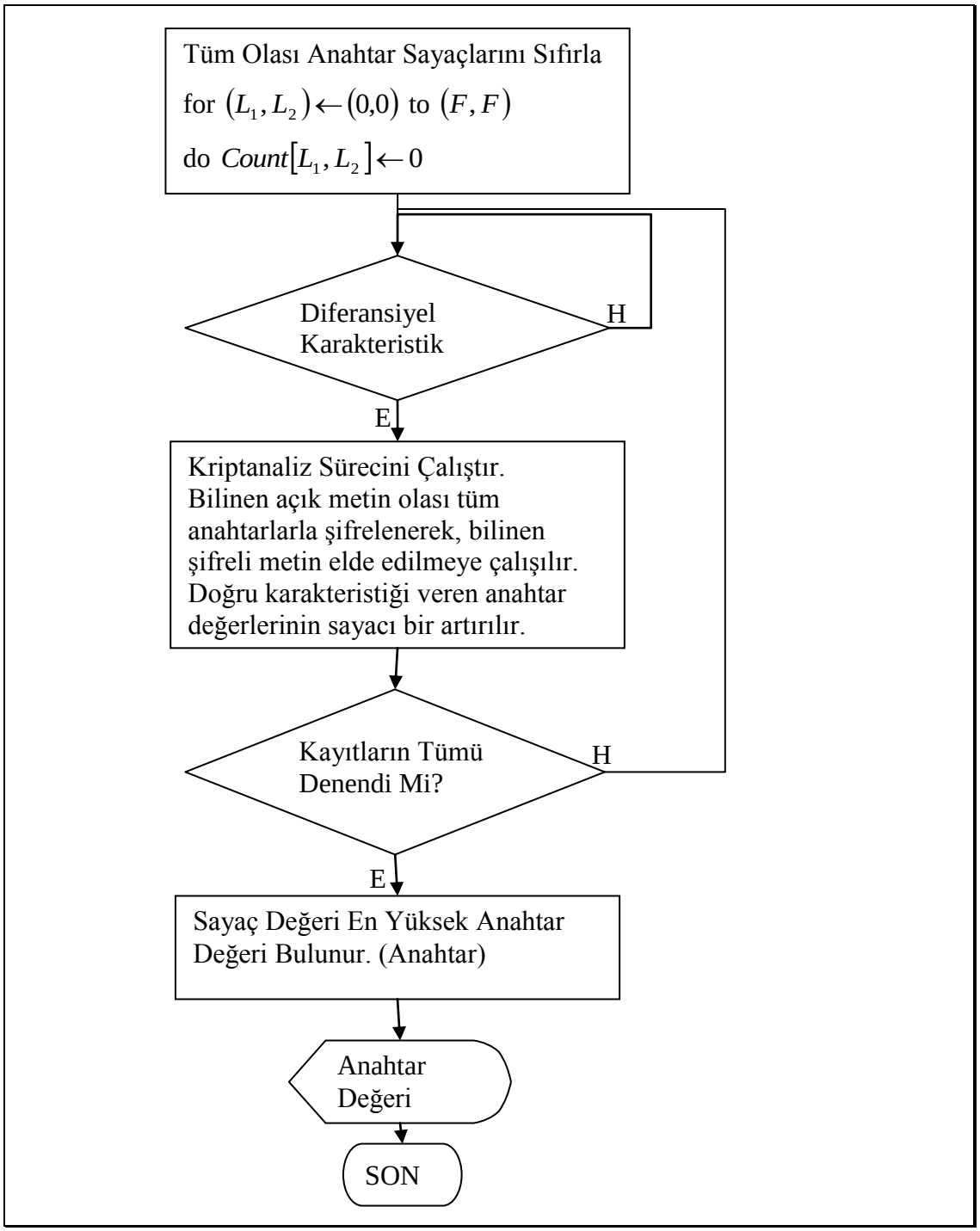
$\Delta X = X \otimes X^{-1}$ ifadesinde $\otimes$ bit dizi gruplar üzerinde, döngü (round) fonksiyonu içinde, anahtar ile metin girdisinin birleştirilmesini sağlayan bir grup işlemidir ve X^{-1} $\otimes$ operasyonuna göre X ' in tersidir.

Farkı tanımlamadaki asıl amaç, metin girdileri arasındaki farkın anahtar eklenmeden ve eklendikten sonra aynı olması, yani farkın anahtardan bağımsız hale getirilmesidir.

[7] Diferansiyel kriptanaliz algoritması Douglas Stinson'ın kriptografi kitabından alınmıştır [73].

$$\Delta X = (X_1 \otimes K) \otimes (X_2 \otimes K)^{-1} = X_1 \otimes K \otimes K^{-1} \otimes X_2^{-1} = X_1 \otimes X_2^{-1} \qquad (4.1)$$

Feistel yapısındaki blok şifre sistemlerinin bir çoğu için, bu farkı kullanarak şifre sisteminin bir döngüsü için, olası tüm metin girdi farkları ve bunlara karşılık gelen olası çıktı farklarının olasılıklarını içeren, fark dağılım tabloları oluşturmak mümkündür.



Algoritma 4.2. Diferansiyel Kriptanaliz Akış Diyagramı

4.1.1 S-Kutuları

S-kutuları bir blok şifreleme algoritmasının en önemli ana elemanıdır. Çünkü algoritmadaki tek doğrusal olmayan yapıdır ve dolayısıyla algoritmaya gücünü vermektedir [55]. Çıkış bitlerinin giriş bitleri cinsinden doğrusal bir fonksiyon ile ifade edilememesi, şifreli metinden açık metnin elde edilmesini önler.

S-kutuları için üç önemli nokta vardır. Bunların belirlenmesinde doğrusal kriptanaliz, diferansiyel kriptanaliz ve Davies [9] saldırıları etkili olmuştur. Bunlar;

i. *Katı çığ ölçütü (Strict Avalance Criteria-SAC)*: 1 bit giriş değişimi sonucunda her çıkış bitinin değişme olasılığı $\frac{1}{2}$ olur.

ii. *S-kutularının genişliği*: Kriptanaliz saldırıları düşünüldüğünde, büyük bir kutu küçüğüne oranla daha iyi olacaktır. Ayrıca diferansiyel saldırılardan korunmak için büyük sayıda çıkış bitleri, doğrusal saldırılardan korunmak için ise büyük sayıda giriş bitleri gereklidir.

iii. *S kutusu gereksinimleri*: Çıkışların dağılımları Davies saldırısına karşın kontrol edilmeli, çıkışlar girişe göre doğrusal olmamalı, S-kutusunun her sırasındaki değerler tek olmalıdır.

4.1.2 Örnek SPN S Kutusu

Heys'in SPN algoritması [7] [46] [73] [74] 4 giriş bitini, 4 çıkış bitine dönüştüren 4×4'lük doğrusal olmayan bir dönüşüm fonksiyonudur. Çıkış bitlerinin giriş bitleri cinsinden doğrusal bir fonksiyon ile ifade edilememesi, şifreli metinden açık metnin elde edilmesini önler. Heys'in 16 bitlik SPN algoritmasında veri blokları, 4 bitlik alt veri bloklarına bölünürek dönüşüme tabi tutulur [7].

4.1.3 Permütasyon

Bitlerin yer değiştirmesi veya bit konumlarının permütasyonu olarak tanımlanabilir. S-kutusu j'nin $i.$ çıkışı, S-kutusu i'nin $j.$ girişine bağlanır.

Tablo 4.1. S-Kutusu

X	Y	X	Y
0000	1110	1000	0011
0110	1011	0100	0010
0001	0100	1001	1010
1011	1100	1110	0000
1100	0101	0111	1000
0010	1101	0011	0001
1111	0111	1101	1001
0101	1111	1010	0110

Permütasyon ile sağlanan yayılma (diffusion), açık metnin artığını şifreli metin üzerine dağıtır. Böylece, açık metnin istatistiksel yapısı, şifreli metnin uzun aralıklı istatistiklerine dağıtılır. Her bir şifreli metin rakamının birçok açık metin rakamı tarafından etkilenmesi nedeniyle, artıkları bulmaya çalışan bir kriptanalist oldukça zaman harcayacaktır. Yayılmayı gerçeklemenin en basit yolu doğrusal dönüşüm[8] kullanmaktır [7].

Tablo 4.2. Permütasyon

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	5	9	13	2	6	10	14	3	7	11	15	4	8	12	16

4.1.4 Heys'in 4 Döngülü SPN Algoritması

4.1.4.1 Anahtar Planlama

4 Döngülü 16 bit SPN için

- 32 bitlik anahtar dizesi seçilir.
- Bu dizeden 16 bitlik 5 alt anahtar elde edilir.
- Dizenin ilk 16 biti, 1. döngünün anahtarı olarak seçilir.
- 2. döngünün anahtarı dize sola 4 bit kaydırıldıktan sonra, ilk 16 bit alınarak elde edilir.

[8] Permütasyon da denir.

– Bu algoritma devam ettirilerek tüm alt anahtarlar hesaplanır.

Örnek 4.1. 00001011000000011100101100001011 anahtar değeri olsun. Döngü anahtarları aşağıdaki gibi hesaplanır:

Anahtar 1: 0000101100000001

Anahtar 2: 1011000000011100

Anahtar 3: 0000000111001011

Anahtar 4: 0001110010110000

Anahtar 5: 1100101100001011

Örnek 4.2. Giriş değeri 1011101100110011 için örnek hesaplama aşağıdaki gibi gerçekleştirilir:

Döngü 1: 1011101100110011

Döngü 2: 1110111110010011

Döngü 3: 0010001000110011

Döngü 4: 0001000000110001

Sonuç: 1101101100111010

Döngü 1:

1011 1011 0011 0011 ⊕ 0000 1011 0000 0001 = 1011 1110 0101 0101

S-Kutusu Dönüşümü: 1110 0000 1111 1111

Permütasyon: 1011 1011 0011 0011

Döngü 2:

1011 1011 0011 0011 ⊕ 1011 0000 0001 1100 = 0000 1011 0010 1111

S-Kutusu Dönüşüm: 1110 1100 1101 0111

Permütasyon: 1110 1111 1001 0011

Döngü 3:

1110 1111 1001 0011 ⊕ 0000 0001 1100 1011 = 1110 1110 0101 1000

S-Kutusu Dönüşüm: 0000 0000 1111 0011

Permütasyon: 0010 0010 0011 0011

Döngü 4:

0010 0010 0011 0011 $\oplus$ 0001 1100 1011 0000 = 0011 1110 1000 0011

S-Kutusu Dönüşüm: 0001 0000 0011 0001

Döngü 5:

0001 0000 0011 0001 $\oplus$ 1100 1011 0000 1011 = 1101 1011 0011 1010

Sonuç: 1101 1011 0011 1010

4.1.5 Fark Dağılımları

X , Y ve verilen (X , $X \oplus \Delta X$) giriş çiftine karşılık gelen ΔY değerlerinin ikili değerleri Tablo 4.3'de görülmektedir. Tablonun son üç kolonu X değeri ve ΔX değeri için, ΔY değerlerini göstermektedir. ΔX = 1001 için ΔY = 0010 değerinin oluşma sayısı 16 olası değer arasından 8 (olasılık 8/16), ΔX = 1000 için ΔY = 1011 değerinin oluşma sayısı 16 olası değer arasından 4 (olasılık 4/16), ΔX = 0100 için ΔY = 1010 değerinin oluşma sayısı 16 olası değer arasından 0 (olasılık 0/16)' dır. Kusursuz bir S-kutusunda tüm fark çiftlerinin oluşma sayısı 1, yani verilen bir ΔX değeri için ΔY değerinin oluşma olasılığı 1/16 olur. Bu da kusursuz bir S-kutusunun matematiksel olarak mümkün olmadığını gösterir.

Fark dağılım tablosunda (bknz. Tablo 4.4) satırlar ΔX değerlerini, kolonlar ΔY değerlerini gösterirler (hexadecimal olarak). Tablonun herbir elemanı verilen bir ΔX giriş farkına karşılık gelen ΔY farkının oluşma sayısını gösterir. Özel durum (ΔX = 0, ΔY = 0) dışında, tablodaki en büyük değer ΔX = B ve ΔY = 2 farklarına karşılık gelen 8 değeridir. ΔX = B ve ΔY = 2 değer çiftinin oluşma olasılığı 8/16' dır.

4.1.6 Diferansiyel Karakteristiğin Belirlenmesi

Kriptanaliz süresince ΔP = [0000 1011 0000 0000] için birçok açık metin çifti şifrelenmiştir. $\frac{27}{1024}$ değerindeki bir yüksek olasılıkla gösterilmiş olan fark

karekteristiği elde edilmiştir. Böyle çiftler ΔP fark değeri için doğru çift olarak tanımlanır. Bu karekteristiğin oluşmadığı açık metin çiftleri, yanlış açık metin çifti olarak tanımlanır.

Analizimizde kullanacağımız diferansiyel karakteristik, S-kutusu fark çiftlerinin birbirinden bağımsız olması şartı ile aşağıda ayrıntılı olarak anlatılacaktır:

Şifreleyecinin giriş farkı, şifreleyecinin ilk döngüsünün giriş farkına eşittir.

$$\Delta P = \Delta U_1 = [0000\ 1011\ 0000\ 0000]$$

U_i i. döngünün giriş, V_i ise çıkış değerini gösterirken, ΔU_i ve ΔV_i ise bunlara karşılık gelen fark değerlerini gösterir.

Tablo 4.3. S-Kutusu Fark Çiftleri

X	Y	$\triangle Y$		
		$\triangle X$=1011	$\triangle X$=1000	$\triangle X$=0100
0000	1110	0010	1101	1100
0001	0100	0010	1110	1011
0010	1101	0111	0101	0110
0011	0001	0010	1011	1001
0100	0010	0101	0111	1100
0101	1111	1111	0110	1011
0110	1011	0010	1011	0110
0111	1000	1101	1111	1001
1000	0011	0010	1101	0110
1001	1010	0111	1110	0011
1010	0110	0010	0101	0110
1011	1100	0010	1011	1011
1100	0101	1101	0111	0110
1101	1001	0010	0110	0011
1110	0000	1111	1011	0110
1111	0111	0101	1111	1011

Tablo 4.4. Fark Dağılım Tablosu

		Çıkış Farkı															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Giriş Farkı	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	2	0	0	0	2	0	2	4	0	4	2	0	0
	2	0	0	0	2	0	6	2	2	0	2	0	0	0	0	2	0
	3	0	0	2	0	2	0	0	0	0	4	2	0	2	0	0	4
	4	0	0	0	2	0	0	6	0	0	2	0	4	2	0	0	0
	5	0	4	0	0	0	2	2	0	0	0	4	0	2	0	0	2
	6	0	0	0	4	0	4	0	0	0	0	0	0	2	2	2	2
	7	0	0	2	2	2	0	2	0	0	2	2	0	0	0	0	4
	8	0	0	0	0	0	0	2	2	0	0	0	4	0	4	2	2
	9	0	2	0	0	2	0	0	4	2	0	2	2	2	0	0	0
	A	0	2	2	0	0	0	0	0	6	0	0	2	0	0	4	0
	B	0	0	8	0	0	2	0	2	0	0	0	0	0	2	0	2
	C	0	2	0	0	2	2	2	0	0	0	0	2	0	6	0	0
	D	0	4	0	0	0	0	0	4	2	0	2	0	2	0	2	0
	E	0	0	2	4	2	0	0	0	6	0	0	0	0	0	2	0
	F	0	2	0	0	6	0	0	0	0	4	0	2	0	0	2	0

Tablo 4.4'te, B (1011) giriş farkının 8/16 olasılıkla 2 (0010) çıkış farkına karşılık geldiği görülmektedir. Sıfır değerli 4 bit grupların çıkış farkı tüm olası çiftler için sıfır değerini alır.

$$\Delta V_1 = [0000\ 0010\ 0000\ 0000] \qquad (p = 8/16)$$

ΔV_1 değerine, Tablo 4.2'deki permütasyonun uygulanması ile ikinci döngünün giriş değeri olan ΔU_2 değeri elde edilir.

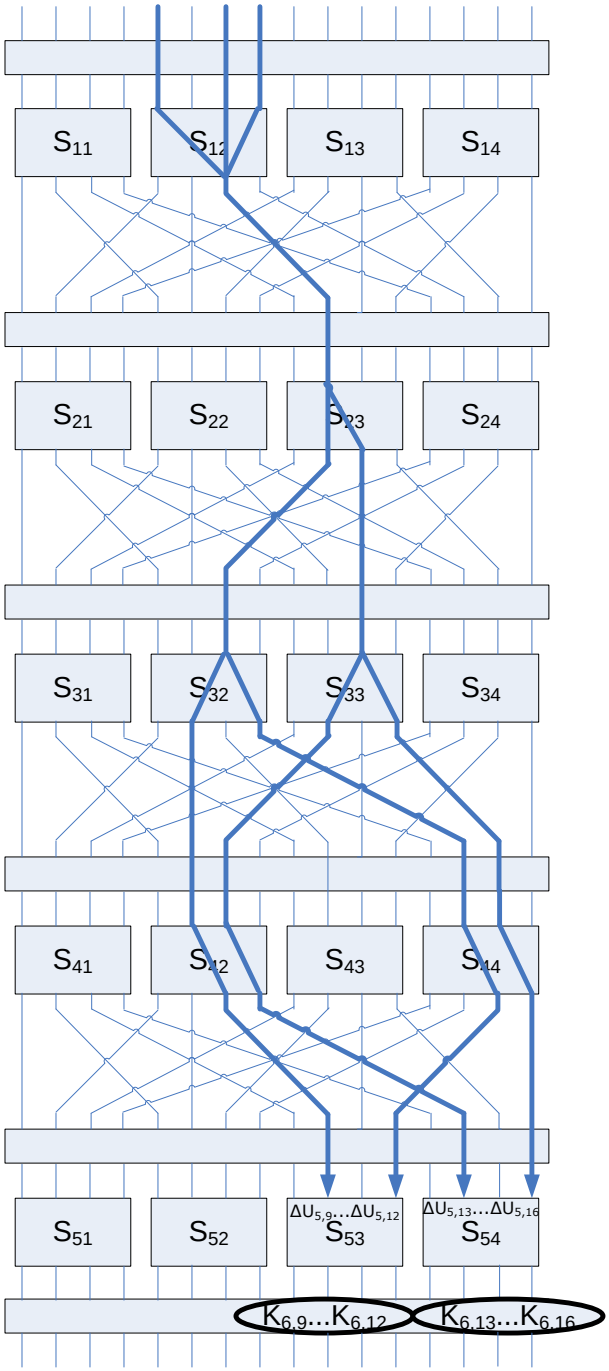
$$\Delta U_2 = [0000\ 0000\ 0100\ 0000]$$

4 (0100) giriş farkı değeri, 6/16 olasılık ile 6 (0110) çıkış farkı değerine karşılık gelir.

$$\Delta V_2 = [0000\ 0000\ 0110\ 0000] \qquad (p = 8/16 * 6/16 = 3/16)$$

Permütasyonun uygulanması ile ΔU_3 değeri elde edilir.

$$\Delta U_3 = [0000\ 0010\ 0010\ 0000]$$



Şekil 4.1. Örnek Diferansiyel Karakteristik

2 (0010) giriş farkı 6/16 olasılıkla 5 (0101) çıkış farkına karşılık gelir.

ΔV_3 = [0000 0101 0000 0101] (p = 8/16* 6/16*(6/16)^2 = 27/1024)

Permütasyonun uygulanması ile ΔU_4 değeri elde edilir.

ΔU_4 = [0000 0110 0100 0110]

6 (0110) giriş farkı 4/16 olasılıkla 3 (0011) çıkış farkına karşılık gelir.

ΔV_4 = 0000 0011 0000 0011 (p = 8/16* 6/16*(6/16)^2 *(4/16)^2= 27/16384

Permütasyonun uygulanması ile ΔU_5 değeri elde edilir.

$$\Delta U_5 = 0000\ 0000\ 0101\ 0101$$

Yukarıda ayrıntılı olarak anlatılan ve Şekil 4.1'de şekilsel olarak gösterimi yapılan 4 döngülük diferansiyel karakteristik kullanılarak gerçekleştirilen Beş Döngülü SPN Diferansiyel Kriptanaliz çalışması (5.1) bölümünde ayrıntılı olarak anlatılmıştır.

4.1.7 Saldırının Gerçeklenmesi İçin Gerekli Veri Sayısı

Karakteristikte yer alan sıfırdan farklı giriş farkı olan S-kutularına aktif S-kutuları denir. Genel olarak, aktif S-kutularının diferansiyel olasılıkları ne kadar büyük olursa, tüm şifreleyici için karakteristik özelliği de büyük olur. Aynı zamanda, az sayıda aktif S-kutusu büyük karakteristik olasılığı demektir.

Saldırıyı gerçekleştirmek için gerekli açık metin sayısını tam olarak belirlemek çok karmaşıktır.

$$N_D \approx \frac{c}{p_D} \tag{4.2}$$

N_D: Açık metin sayısı,

p_D: R döngülü şifreleyicide, $R-1$ döngü için diferansiyel karakteristik olasılığı

c: Küçük bir sabit değer

$$p_D = \prod_{i=1}^{\gamma} \beta_i \tag{4.3}$$

γ: Aktif S-kutusu sayısı,

β: Karakteristiğin $i.$ S-kutusunda fark çiftinin oluşma olasılığı

Her bir aktif S-kutusunda fark çiftlerinin oluşması bağımsızdır.

4.1.8 AES Diferansiyel Kriptanaliz

4.1.8.1 AES Diferansiyel Özellikleri

Beomsik Song ve Jennifer Seberry yaptıkları çalışmalarında, AES şifreleme algoritmasının temel fonksiyonlarının cebirsel özelliklerini incelemiş ve şifreleyicinin şu diferansiyel örüntülerini belirlemişlerdir [75]:

- Eğer iki açık metin sadece bir sekizlide farklılık gösteriyorsa, ikinci döngünün çıkış farkında aynı değere sahip olan 4 çift vardır.
- Eğer iki açık metin belli bir yerde 4 sekizli kadar farklılık gösteriyorsa, bu örüntü ikinci döngü çıkış farkında da gözükür.
- Herhangi n sekizli konumunda farklılık gösteren, diğer konumlarda aynı olan her bir $2^8 n$ açık metin için, bu açık metinlerden birini diğer açık metinlerden biri ile çiftleştirirsek, çıkış farklarının herhangi biri 3. döngüden sonra diğer çıkış farklarının XOR'una eşittir.
- Belli 4 sekizli konumunda farklılık gösteren, diğer konumlarda aynı olan her bir 2^{32} açık metin için, bu açık metinlerden birini diğer açık metinlerden biri ile çiftleştirirsek, çıkış farklarının herhangi biri 4. döngüden sonra diğer çıkış farklarının XOR'una eşittir.

4.1.8.2 MixColumn Özellikleri

MixColumn işlemi doğrusal bir işlemdir. Diferansiyel ve doğrusal kriptanalizde dallanma sayısı (branch number) önemlidir. MixColumn dönüşümü için dallanma sayısı 5'dir.

Teorem 4.1. X ve X', MixColumn dönüşümü için iki giriş değeri olsun. Bu iki giriş değerinin MixColumn dönüşümlerinin XOR'u, iki giriş değerinin XOR'unun MixColumn dönüşümüne eşittir..

$$MixColumn(X) \oplus MixColumn(X') = MixColumn(\Delta X = X \oplus X') \tag{4.4}$$

Teorem 4.2. MixColumn dönüşümünün giriş değerlerinin herhangi dört sekizli değeri için, bir sekizlinin değeri α ve diğer 3 sekizli değerinden farklı (bu 3 sekizli değeri aynı değerde ve β) ise, α değeri çıkış değerinin 2 sekizlisinde yer alır. Matematiksel olarak I, I', I'', I''' MixColumn dönüşümünün giriş değeri olması şartı ile aşağıdaki gibi gösterilir:

$$I = (\alpha, \beta, \beta, \beta) \qquad MixColumn(I) = (\gamma, \alpha, \alpha, \alpha)$$

$$I' = (\beta, \alpha, \beta, \beta) \qquad MixColumn(I') = (\delta, \gamma, \alpha, \alpha)$$

$$I'' = (\beta, \beta, \alpha, \beta) \qquad MixColumn(I'') = (\alpha, \delta, \gamma, \alpha)$$

$$I''' = (\beta, \beta, \beta, \alpha) \qquad MixColumn(I''') = (\alpha, \alpha, \delta, \gamma)$$

$$\gamma \oplus \delta = \alpha \oplus \beta$$

4.1.8.3 SubBytes Dönüşümünün Diferansiyel Karakteristiği (S-Kutusu)

S-kutusu doğrusal olmayan bir dönüşümdür. S-kutusunun herhangi bir ΔX giriş farkı için oluşan olası çıkış farkı ΔY sayısı her durumda 127' dir. Yani tüm 128 giriş çifti için, belli bir değer çıkış farkında iki kez gözükürken diğerleri bir kez gözükmektedir.

2 Döngülük AES Şifreleyicisi üzerinde gerçekleştirdiğimiz diferansiyel kriptanaliz uygulamasının ayrıntısı (5.2) bölümünde verilmiştir.

4.2 Kesilmiş (Truncated) Diferansiyel

Geleneksel diferansiyel saldırıda, diferansiyel belirli bir sayıda döngüden sonra şifreli metnin n bitini tahmin edebilmek için kullanılan bir araçtır. Her zaman n bitlik değerin tamamının tahmin edilmesi gerekli değildir. Bazı durumlarda 1 bitlik değer bile yeterli olabilmektedir. n bit değerin sadece bir kısmını tahmin eden diferansiyel, kesilmiş (truncated) diferansiyel olarak isimlendirilir.

4.2.1 Kesilmiş (Truncated) Diferansiyel Saldırı

$2n$ bitlik bir Feistel şifreleyicisi üzerindeki geleneksel diferansiyel saldırıda [8], [18] diferansiyel belirli bir sayıda döngüden sonra şifreli metnin n bitini tahmin edebilmek için kullanılan bir araçtır.

Tanım 4.2. i döngü kadar şifrelemeden sonra a giriş farkı, b çıkış farkını oluşturuyorsa (a,b) i döngülük diferansiyel olarak isimlendirilir.

Her zaman n bitlik değerin tamamının tahmin edilmesi gerekli değildir. Bazı durumlarda 1 bitlik değer bile yeterli olabilmektedir. n bit değerin sadece bir kısmını tahmin eden diferansiyel kesilmiş (truncated) diferansiyel olarak isimlendirilir.

Tanım 4.3. (a,b) i döngülük diferansiyel olsun. a' a'nın alt dizisi, b' b'nin alt dizisi olsun. (a',b') i döngülük kesilmiş (truncated) diferansiyel olarak tanımlanır.

Kesilmiş diferansiyeller Knudsen tarafından bulunmuştur [36]. Sekizli temelli işlemler içermesi nedeniyle SAFER [76] şifreleyicisine uygulanmıştır [77].

Örnek 4.3. n bit blok üzerinde m bitlik kısıtlar tanımlanır. $(m<n):(A,-A,B,2B)$, A ve B herhangi bir değer.

Örnek 4.4. Veri bloğunun bir kısmı sabitlenir ve geri kalan kısmı rastgele olarak değişebilir. (0, $*$, 3, $*$, 255, $*$, $*$), $*$ herhangi bir değer alabilir.

4.2.2 Kesilmiş Diferansiyel Özellikleri

- Sözcük temelli yapıya sahip şifreleyicilere karşı güçlü bir araçtır.
- İmkansız diferansiyeller ve boomerang saldırısı [20] [21] gibi diferansiyel tekniklerin uzantılarında faydalıdır.
- Kesilmiş diferansiyeller sık sık verileri yapılara paketleyen bir teknik ile birleştirilir. Bu bazen kesilmiş diferansiyellerin 2^{-n}'den düşük olasılıkta elde edilmesini sağlar.

Teorem 4.3. (Kesilmiş (Truncated) Diferansiyeller).

- $f(x,k): GF(2^n) \times GF(2^n) \to GF(2^n)$ $2n$ blok uzunluğuna sahip 5 döngülük Feistel şifreleyicisinin döngü fonksiyonu olsun. Her bir döngü anahtarı n bit uzunluğunda olsun.
- $\alpha(\neq 0)$ f fonksiyonunun iki giriş değerinin farkı olsun.
- Çıkış farkının (W) sadece bir kısmının oluşması şartı olsun.

Kesilmiş diferansiyeller kullanan diferansiyel saldırı $2L$ açık metin gerektirir. $L \times 2^{2n}$ çalışma zamanı karmaşıklığı gerektirir (L en küçük tamsayı), $(W)^L < 2^{-2n}$. L en fazla $2n+1$ olur.

İspat:

Aşağıda anlatılan saldırıyı düşünelim.

- α f fonksiyonunun iki giriş değerinin farkı olsun.
  - Çıkış farkının sadece bir kısmının oluşması şartı ile.
- Başlangıç değeri tüm elemanlar için 0 olan T tablosu hesaplanır.
  - For $i = 0,\ldots,2^{n-1}, T[f(i) \oplus f(i \oplus \alpha)] = 1$
  - Rastgele olarak P_1 açık metni seçilir ve $P_2 = P_1 \oplus (\alpha \| 0)$[9]ifadesi hesaplanır.
- P_1 ve P_2 açık metinlerine karşılık gelen C_1 ve C_2 şifreli metinleri hesaplanır.
- RK_5 döngü anahtarının herbir k_5 anahtarı için;
  - C_1 ve C_2 şifreli metinleri, k_5 döngü anahtarı ile deşifrelenerek D_1 ve D_2 şifreli metinleri elde edilir.
  - RK_4 döngü anahtarının herbir k_4 anahtarı için;
    - $i = 1,2$ için $t_i = f(D_i^R \oplus k_4)$ ifadesi hesaplanır.

[9] || tekrarlamayı gösterir.

- $T\left[t_1 \oplus t_2 \oplus D_1^R \oplus D_2^R\right] > 0$ ise, k_4 ve k_5 değerleri gösterilir.

– α giriş farkı için T tablosu hesaplanır. $T[\beta] > 0$ için, β çıkış farkı olasıdır.

– Birinci döngünün giriş değerleri eşittir. İkinci döngünün giriş değerleri farkı α ' dır.

– Dördüncü döngünün çıkış farkının W parçasının tüm olası değerleri şifreli metnin sağ parçası ve T tablosundan hesaplanabilir.

– L açık metin çifti denendiğinde doğru anahtar W^L olasılığı ile L kez önerilir.

– $W^L < 2^{-2n}$ ise doğru anahtar tekil olarak belirlenebilir.

$$W \le \frac{1}{2},\ \min_L : \left(\frac{1}{2}\right)^L < 2^{-2n} = 2n+1.$$

Tanım 4.4. (Sinyal Gürültü Oranı). Sinyal Gürültü Oranı, doğru anahtarın sayılma sayısının rastgele bir anahtarın sayılma sayısına oranıdır. Sinyal gürültü oranı aşağıdaki ifade ile hesaplanır:

$$S/N = \frac{|K| \times p}{\gamma \times \lambda} \tag{4.5}$$

– p : Saldırıda kullanılan diferansiyelin olasılığı

– $|K|$: Olası anahtarların sayısı

– γ : Her bir açık metin çifti için önerilen anahtarların sayısı.

– λ : Gözden çıkarılmayan çiftlerin tüm çiftlere oranıdır.

– $S/N \le 1$ ise diferansiyel saldırı başarısız olur.

4.2.3 DES Kesilmiş Diferansiyelleri

DES şifreleyicisinde olasılık değeri 1 olan kesilmiş diferansiyeller vardır. F fonksiyonunun iki giriş değerinin S-kutusu giriş değeri eşit ise, bu S-kutularının çıkış değeri de, diğer S-kutularının giriş değerlerinden bağımsız olarak eşittir.

Bir S-kutusunun çıkışı, P permütasyonu nedeniyle izleyen döngüdeki S-kutularının en fazla altısının giriş değerini etkiler. Bu etkileşim Tablo 4.5' deki etkileşim matrisinden görülebilir. Bu sayede DES için, 1 olasılıklı 4 döngülük bir kesilmiş diferansiyel elde edilebilir. Bu 4 döngülük diferansiyel şifreli metin farkının 8 biti hakkında bilgi verir.

4.2.3.1 4 Döngülük Kesilmiş (Truncated) DES Diferansiyel

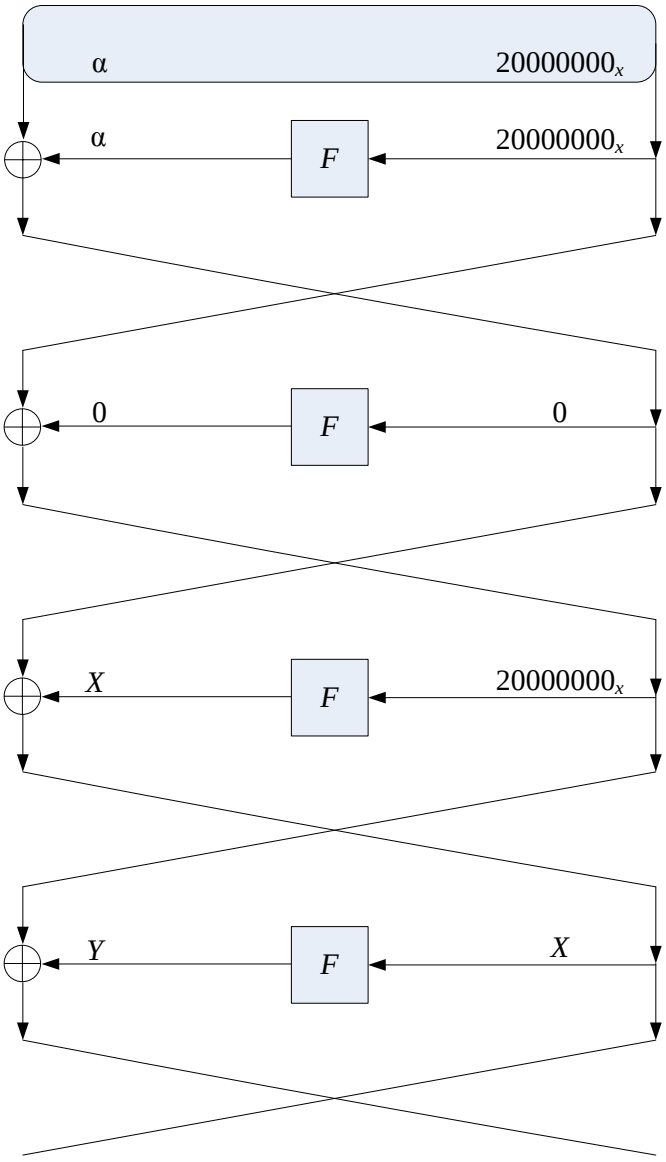
DES üzerinde 6 döngüde 46 açık metin kullanarak, yaklaşık 3500 şifreleme işlemi ile gizli anahtarı bulan bir saldırı gerçekleştirilebilir.

Tablo 4.5. S Kutusu Etkileşimleri

S-kutusu Çıkış Değeri	Etkilemediği S-kutuları
1	**1,7**
2	**2,6**
3	**3,1**
4	**4,2**
5	**5,8**
6	**6,4**
7	**7,5**
8	**8,3**

DES şifreleme algoritmasına Şekil 4.2' deki diferansiyel ve 20000000_x, değerinin 40000000_x değeri ile değiştiği benzer diferansiyeli kullanan, diferansiyel seçilen açık metin saldırısı düzenleyelim. Saldırının ayrıntıları aşağıda verilmiştir:

- Birinci döngünün çıkış değerlerinin farkı α olsun.
- Üçüncü döngünün giriş değerleri sadece iki bitte farklılaşır ve bu farklılık sadece birinci S-kutusunu etkiler.
- X giriş farkının dördüncü döngü girişleri, birinci ve yedinci S-kutularının girişlerinde eşit değerdedir. Bu nedenle, Y çıkış farkının 8 bitinin değeri 0' dır.



Şekil 4.2. 4 Döngülük DES Diferansiyel

- Saldırgan bu bilgiden faydalanarak, anahtar değerinin 64 olası değerinin tümünü S-kutusu 1 ve S-kutusu 7 için dener.
- Her bir şifreli metin çifti için ortalama 4 anahtar değeri önerilir.
- Denenen şifreli metin çifti sayısı artırılarak olasılığı en yüksek olan anahtar, önerilen 4 anahtar değeri arasından belirlenir.

Şekil 4.2'deki 4 döngülük diferansiyel karakteristiği kullanarak gerçekleştirilen saldırının ayrıntıları (5.3) bölümünde verilmiştir.

4.3 İmkansız Diferansiyel Saldırı

İmkansız diferansiyel kriptanaliz, blok şifreleyiciler üzerindeki diferansiyel kriptanalizin bir çeşididir. Klasik diferansiyel kriptanaliz döngüler boyunca ilerletilen, beklenen olasılığın üzerindeki farkları izlerken, imkansız diferansiyel kriptanaliz algoritmanın ara aşamalarında oluşan imkansız farklara odaklanır.

Anahtarın gerçek değerlerini elde etmek için imkansız diferansiyelleri kullanan imkansız diferansiyel kriptanaliz, blok şifreleyicilerin döngüleri boyunca imkansız olayların bulunmasına bağlıdır. Bu sayede, tüm olası gizli anahtarlar tahmin edilebilir ve belirlenen imkansız olayı oluşturan tüm anahtarlar yanlış anahtarlar olarak işaretlenir. Çünkü, doğru anahtarlar böyle bir imkansız olayı hiçbir zaman oluşturmazlar [78].

İmkansız olayları oluşturmanın en verimli yöntemi Biham tarafından bulunan ortada ıskalama (miss-in-the-middle) tekniğidir [34]. Bu teknik sürekli gerçekleşen iki olay üzerine odaklanır. Çelişkiye neden olan bu iki olay birleştirilerek, imkansız olay oluşturulur.

Tanım 4.5. (İmkansız Diferansiyel). İmkansız diferansiyel olasılık değeri 0 olan veya var olmayan diferansiyel demektir. $\{\alpha, \beta\}$ çifti AES şifreleyici algoritmasında imkansız diferansiyel olsun. α, çıkış XOR örüntüsünde belirli sekizli konumlarında β pasif sekizlilerini hiçbir zaman oluşturmayan, giriş XOR örüntüsündeki aktif sekizli indisidir.

AES için XOR örüntüsü bir AES veri bloğu çiftinin pasif ve aktif sekizli konumlarını tanımlayan 4×4 'lük bir bloktur (Mini-AES için 2×2).

Gösterim

P : 16-bitlik açık metin

C : 16-bitlik şifreli metin

Tanım 4.6. (Nibble). 16-bitlik veri bloğu 4 adet 4 bitlik alt bloktan oluşur. Bu alt blokların her biri bir nibble'dır. P ve P' sadece tek bir nibble'da farklı olan açık metin çiftleridir.

Tanım 4.7. (Pasif Nibble). P ve P' açık metin çiftlerinin nibble'larının eşit olduğu nibble'lar, pasif nibble'lar olarak isimlendirilir.

Tanım 4.8. (Aktif Nibble). P ve P' açık metin çiftlerinin nibble'larının farklı olduğu nibble'lar aktif nibble'lar olarak isimlendirilir.

Örnek 4.5. P = 0100 0011 1110 1001 P' = 1110 0011 1110 1001

4.3.1 Mini-AES

Mini-AES [79] AES'in temel kavramlarının anlaşılması için eğitimsel amaçlı olarak geliştirilmiştir. İmkansız diferansiyel kriptanaliz, Mini-AES ile aynı şekilde AES üzerinde de uygulanabilir.

4.3.1.1 Matematik Altyapısı

Mini-AES şifreleyicisinin dört elemanından ikisi sonlu cisim aritmetiğini kullanan NibbleSub ve MixColumn elemanlarıdır. Mini-AES şifreleyicisi $GF(2^4)$ sonlu cismini kullanır. İşlemde kapalılık özelliğinin gereği olarak, sonlu cisim aritmetiğinde gerçekleştirilen işlemlerin sonucunda elde edilen değerler yine seçilen sonlu cismin içerisinde olmak zorundadır. $GF(2^4)$ sonlu cisminde her bir nibble $\{0,1\}$ katsayılarını kullanan ve en yüksek dereceli terimi, sonlu cisim derecesinden bir eksik olan polinomlar olarak ifade edilirler. Örneğin, $P=(p_0, p_1, p_2, p_3)$ bir nibble ise, $P = a_3x^3 + a_2x^2 + a_1x + a_0$ olarak ifade edilir.

Örnek 4.6. $P = 1101$ bir nibble olsun. $P = x^3 + x^2 + 1$ olarak ifade edilir.

Mini-AES şifreleyicisinde toplama işlemi $GF(2^4)$ sonlu cisminde aynı derecedeki terimlerin $\{0,1\}$ değerlerinden oluşan katsayıları mod 2 aritmetiğinde toplanarak gerçekleştirilir. $GF(2^4)$ sonlu cisminde toplama işlemi XOR işlemine denk düşer.

Örnek 4.7. $P = 1101$, $P' = 0110$ olmak üzere iki nibble verilsin. $P = x^3 + x + 1$, $P' = x^2 + x$ olarak ifade edilir. $P + P' = (x^3 + x + 1) + (x^2 + x) = x^3 + x^2 + 1 = (1101)$.

$GF(2^4)$ sonlu cisminde çarpma işleminde, önce çarpılacak değerlerin polinomsal gösterimi elde edilir. Elde edilen iki polinom, polinom çarpımı ile mod 2

aritmetiğinde çarpılarak işlem gerçekleştirilir. Polinom çarpma işlemi sonucunda sonlu cismin derecesinden büyük derecede terimler oluşabilir. Bu terimler bir indirgenemez polinom aracılığı ile indirgenerek, işlemin sonucunun sonlu cisim içeresinde olması sağlanır.

Bölüm (1.6)' da belirtildiği gibi, çarpanlarına ayrılamayan indirgenemez polinom denir. Mini-AES şifreleyicisinde $x^4 + x + 1$ indirgenemez polinomu kullanılmıştır [79]. İndirgeme işlemi, x^4 yerine $x + 1$ yazılarak yapılır.

Örnek 4.8. $P = 1101$, $P' = 1011$ olmak üzere iki nibble verilsin. $P = x^3 + x^2 + 1$, $P' = x^3 + x + 1$ olarak ifade edilir. Bu iki nibble ile gerçekleştirilen çarpma işlemi aşağıda gösterilmiştir:

$$\begin{aligned} P \bullet P' = (x^3 + x^2 + 1)(x^3 + x + 1) &= x^6 + x^4 + x^3 + x^5 + x^3 + x^2 + x^3 + x + 1 \\ &= x^6 + x^5 + x^3 + x^2 + x + 1 \\ &= x^2(x+1) + x(x+1) + x^3 + x^2 + x + 1 \\ &= x^3 + x + x^2 + x + x^3 + x^2 + x + 1 \\ &= x + 1 \\ &= (0011). \end{aligned}$$

4.3.1.2 Mini-AES Yapısı

AES ile aynı temel yapıya sahiptir. 16 bitlik giriş değeri, 16 bitlik döngü anahtarları ile işleme tabi tutularak 16 bitlik çıkış değeri elde edilir. Şifreleme işleminde işlenen 16 bitlik bloklar 2×2 'lik matrisler olarak ifade edilir.

P_0	P_2
P_1	P_3

$$P = P_0 P_1 P_2 P_3$$

Mini-AES şifreleme algoritması 2 döngüden oluşur ve 4 temel işlem içerir. Bu dört temel işlem şunlardır [79]:

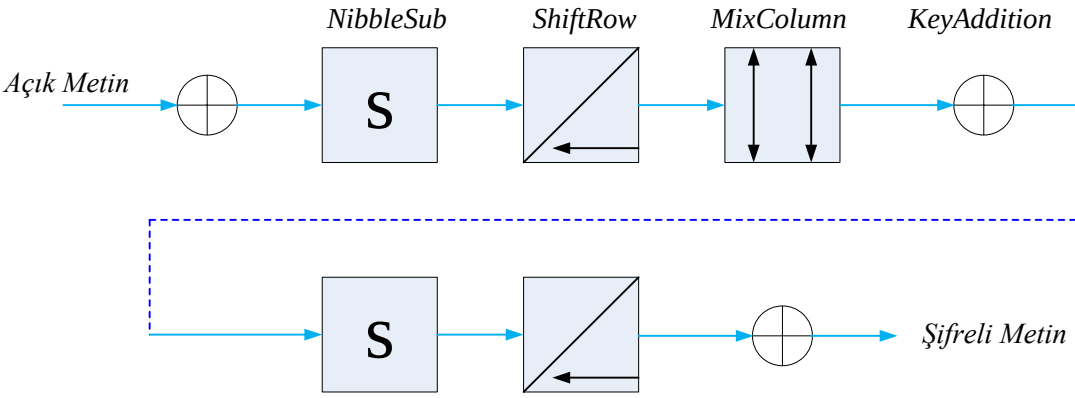
i. NibbleSub

ii. ShiftRow

iii. MixColumn

iv. RoundKeyAddition

Birinci döngüden önce RoundKeyAddition işlemi gerçekleştirilir. Son döngüde ise MixColumn dönüşümü gerçekleştirilmez.

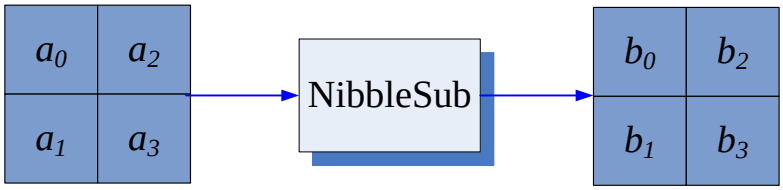


Şekil 4.3. Mini-AES Şifreleme Algoritması Şekilsel Gösterimi

4.3.1.3 NibbleSub, γ

NibbleSub, Mini-AES algoritmasındaki doğrusal olmayan yapıdır [79]. NibbleSub her bir giriş nibble'ını 4×4' lük bir yerine koyma tablosuna göre bir çıkış nibble'ına çevirir. 16 bitlik $A\ (a_0,a_1,a_2,a_3)$ giriş değeri, 16 bitlik $B\ (b_0,b_1,b_2,b_3)$ çıkış değerine, dört bit dört bit işlenerek Tablo 4.6' daki şemaya göre dönüşür.

Örnek 4.9. $a_0=1111\rightarrow b_0=0111.$



Şekil 4.4. NibbleSub Dönüşümünün Şekilsel Gösterimi

NibbleSub dönüşümü, P ve P' açık metinleri arasındaki aktif ve pasif nibble sayısını değiştirmez. Dolayısıyla XOR örüntüsünü de değiştirmez.

Tablo 4.6. Mini-AES S Kutusu

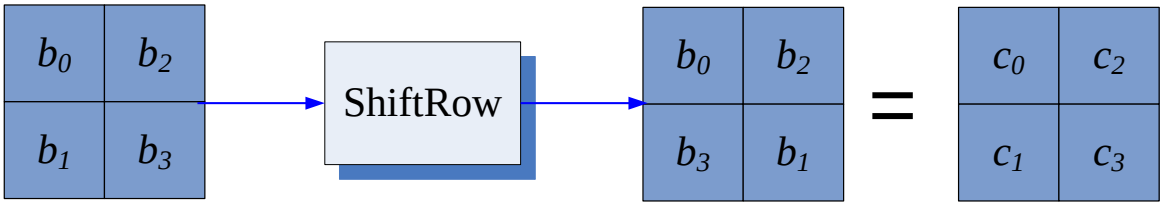
Giriş	Çıkış	Giriş	Çıkış
0000	1110	1000	0011
0110	1011	0100	0010
0001	0100	1001	1010
1011	1100	1110	0000
1100	0101	0111	1000
0010	1101	0011	0001
1111	0111	1101	1001
0101	1111	1010	0110

4.3.1.4 ShiftRow, π

ShiftRow dönüşümü [79] aktif nibble'ların aynı satır üzerinde diğer nibble konumlarına kaydırılmasını sağlar. ShiftRow giriş bloğunun her bir satırını farklı nibble sayılarında sola döndürme işlemine tabi tutar. İlk satır üzerinde herhangi bir değişiklik olmazken, ikinci satır sola bir nibble miktarında döndürülür.

ShiftRow dönüşümü ikinci satırdaki nibble'ların yerlerini değiştirdiği için, aktif nibble sayısını değiştirmez. İkinci ShiftRow dönüşümünden sonra, aktif sekizliler farklı kolonlara yerleşir.

$B=(b_0,b_1,b_2,b_3)$ giriş değeri, $C=(c_0,c_1,c_2,c_3)$ çıkış değeri olsun. ShiftRow dönüşümü sonrası, $C=(c_0,c_1,c_2,c_3)=B=(b_0,b_3,b_2,b_1)$ olur.



Şekil 4.5. ShiftRow Dönüşümünün Şekilsel Gösterimi

4.3.1.5 MixColumn, θ

MixColumn [79] aktif nibble'ların yayılmasını sağlaması nedeniyle imkansız diferansiyellerin davranışı üzerinde büyük etkiye sahiptir. MixColumn giriş bloğunun her bir kolonunu alır ve sabit bir matris ile çarpım işlemine tabi tutarak çıkış kolonunu

elde eder. MixColumn dönüşümü $C=(c_0,c_1,c_2,c_3)$ giriş değerini $D=(d_0,d_1,d_2,d_3)$ çıkış değerine dönüştürür.

$$\begin{bmatrix} d_0 \\ d_1 \end{bmatrix} = \begin{bmatrix} 3 & 2 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} \quad \begin{bmatrix} d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} 3 & 2 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} c_2 \\ c_3 \end{bmatrix}$$

Şekil 4.6. MixColumn Dönüşümünün Şekilsel Gösterimi

MixColumn aktif nibble'ın tüm 4 nibble üzerine dağılmasını sağlar. Dolayısıyla XOR örüntüsünü değiştirir.

Örnek 4.10.

$$\begin{bmatrix} d_0 \\ d_1 \end{bmatrix} = \begin{bmatrix} 0011 & 0010 \\ 0010 & 0011 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \end{bmatrix}$$

$$d_0 = (0011 \bullet c_0) \oplus (0010 \bullet c_1)$$

$$d_1 = (0010 \bullet c_0) \oplus (0011 \bullet c_1)$$

$$\begin{bmatrix} d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} 0011 & 0010 \\ 0010 & 0011 \end{bmatrix} \begin{bmatrix} c_2 \\ c_3 \end{bmatrix}$$

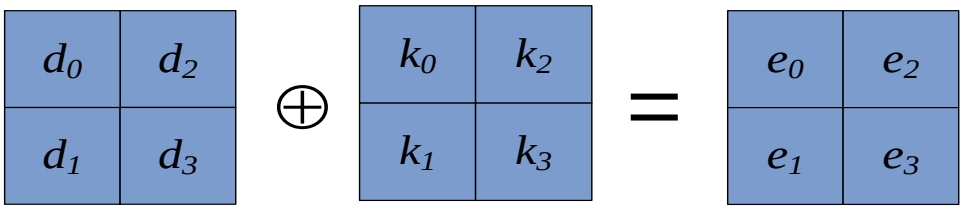
$$d_2 = (0011 \bullet c_2) \oplus (0010 \bullet c_3)$$

$$d_3 = (0010 \bullet c_2) \oplus (0011 \bullet c_3)$$

4.3.1.6 KeyAddition, σK_i

Giriş değerinin bitleri döngü anahtarının bitleri ile XOR işlemine tabi tutularak çıkış değeri elde edilir.

$$E(e_0,e_1,e_2,e_3) = D(d_0,d_1,d_2,d_3) \oplus K_i(k_0,k_1,k_2,k_3) \tag{4.6}$$



Şekil 4.7. KeyAddition Dönüşümünün Şekilsel Gösterimi

KeyAddition doğrusal bir işlem olduğu için aktif sekizli sayısı üzerinde bir etkisi yoktur [79]. Dolayısıyla XOR örüntüsü üzerinde de.

4.3.1.7 Anahtar Planlama:

Aşağıdaki algoritmadan da görüleceği gibi K_0 ile gösterilen şifreleme sürecinin başındaki alt anahtar değeri, anahtar planlama algoritmasının giriş değeri olan K değeridir. 16 bitlik K anahtarı dört bitlik gruplara bölünür ve dört bit dört bit işlemlere tabi tutularak sonraki döngünün alt anahtarları hesaplanır. Algoritma 4.3'deki Mini-AES anahtar planlama algoritmasında *rcon* değeri, $rcon(i) = 2^i$ şeklinde ifade edilir.

4.3.1.8 Mini-AES Şifreleme

$$Mini - AES = \sigma K_2 \circ \pi \circ \gamma \circ \sigma K_1 \circ \theta \circ \pi \circ \gamma \circ \sigma K_0 \tag{4.7}$$

Yukarıda iki döngülük Mini-AES şifreleme algoritmasının gösterimi yapılmıştır. $\circ$ sembolü fonksiyonların birleştirilmesini ifade etmektedir ve çalışma sırası sağdan sola doğrudur. Birinci döngüden önce açık metin K_0 döngü anahtarı, RoundKeyAddition(σK_0) işlemine tabi tutulur. Elde edilen sonuç doğrusal olmayan yerine koyma işlemi olan NibbleSub(γ) işlemine tabi tutulur. Elde edilen sonuç karıştırmayı sağlayan ShiftRow(π) dönüşümüne tabi tutulur. ShiftRow dönüşümünün sonucu yayılmayı sağlayan doğrusal MixColumn(θ) dönüşümüne tabi tutulur. Elde edilen sonuç K_1 döngü anahtarı ile KeyAddition (σK_1) işlemine tabi tutulur ve birinci döngünün çıkış değeri elde edilir.

İkinci döngüde MixColumn dönüşümü yoktur. İkinci döngüde sırayla NibbleSub, ShiftRow ve KeyAddition işlemleri gerçekleştirilerek şifreli metin elde edilir [79].

Giriş Değeri: $K=(k_0,k_1,k_2,k_3)$

Döngü 0: $K_0=(w_0,w_1,w_2,w_3)$

$w_0=k_0, w_1=k_1, w_2=k_2, w_3=k_3$

Döngü 1: $K_1=(w_4,w_5,w_6,w_7)$

$w_4=w_0 \oplus NibbleSub(w_3)\oplus rcon(1)$

$w_5=w_1 \oplus w_4$

$w_6=w_2 \oplus w_5$

$w_7=w_3 \oplus w_6$

Döngü 2: $K_2=(w_8,w_9,w_{10},w_{11})$

$w_8=w_4 \oplus NibbleSub(w_7)\oplus rcon(2)$

$w_9=w_5 \oplus w_8$

$w_{10}=w_6 \oplus w_9$

$w_{11}=w_7 \oplus w_{10}$

Algoritma 4.3. Mini-AES Anahtar Planlama Algoritması

Örnek 4.11. $K=1100\ 0011\ 1111\ 0000$ değeri için döngü anahtarları aşağıdaki gibi hesaplanır:

$$K_0=(w_0,w_1,w_2,w_3)=(k_0,k_1,k_2,k_3)=K=1100\,0011\,1111\,0000$$

$$w_0=1100,\ w_1=0011,\ w_2=1111,\ w_3=0000$$

$$K_1=(w_4,w_5,w_6,w_7)$$

$$w_4 = w_0 \oplus NibbleSub(w_3) \oplus 0001$$
$$= 1100 \oplus NibbleSub(0000) \oplus 0001$$
$$= 1100 \oplus 1110 \oplus 0001$$
$$= 0011$$

$$w_5 = w_1 \oplus w_4 = 0011 \oplus 0011 = 0000$$

$$w_6 = w_2 \oplus w_5 = 1111 \oplus 0000 = 1111$$

$$w_7 = w_3 \oplus w_6 = 0000 \oplus 1111 = 1111$$

$$K_2 = (w_8, w_9, w_{10}, w_{11})$$

$$w_8 = w_4 \oplus NibbleSub(w_7) \oplus 0010$$
$$= 0011 \oplus NibbleSub(1111) \oplus 0010$$
$$= 0011 \oplus 0111 \oplus 0010$$
$$= 0110$$

$$w_9 = w_5 \oplus w_8 = 0000 \oplus 0110 = 0110$$

$$w_{10} = w_6 \oplus w_9 = 1111 \oplus 0110 = 1001$$

$$w_{11} = w_7 \oplus w_{10} = 1111 \oplus 1001 = 0110$$

$P = 1001\ 1100\ 0110\ 0011$ açık metin değeri için Mini-AES şifreleme işlemi aşağıda gösterilecektir:

$$A = P \oplus K_0 = 1001\ 1100\ 0110\ 0011 \oplus 1100\ 0011\ 1111\ 0000$$
$$= 0101\ 1111\ 1001\ 0011$$

Döngü 1:

NibbleSub

$$B = NibbleSub(0101),\ NibbleSub(1111),\ NibbleSub(1001),\ NibbleSub(0011)$$
$$= 1111\ 0111\ 1010\ 0001$$

ShiftRow

$$C = ShiftRow(1111\ 0111\ 1010\ 0001) = 1111\ 0001\ 1010\ 0111$$

MixColumn

$$C = \begin{bmatrix} 1111 & 1010 \\ 0001 & 0111 \end{bmatrix}$$

$$\begin{bmatrix} d_0 \\ d_1 \end{bmatrix} = \begin{bmatrix} 0011 & 0010 \\ 0010 & 0011 \end{bmatrix} \begin{bmatrix} 1111 \\ 0001 \end{bmatrix}$$

$(0011 \bullet 1111) \oplus (0010 \bullet 0001) = 0010 \oplus 0010 = 0000$

$(0010 \bullet 1111) \oplus (0011 \bullet 0001) = 1101 \oplus 0011 = 1110$

$$\begin{bmatrix} d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} 0011 & 0010 \\ 0010 & 0011 \end{bmatrix} \begin{bmatrix} 1010 \\ 0111 \end{bmatrix}$$

$(0011 \bullet 1010) \oplus (0010 \bullet 0111) = 1101 \oplus 1110 = 0011$

$(0010 \bullet 1010) \oplus (0011 \bullet 0111) = 0111 \oplus 1001 = 1110$

$$D = \begin{bmatrix} 0000 & 0011 \\ 1110 & 1110 \end{bmatrix} \text{ veya } \quad D = 0000\ 1110\ 0011\ 1110$$

KeyAddition

$$E = D \oplus K_1 = 0000\ 1110\ 0011\ 1110 \ \oplus \ 0011\ 0000\ 1111\ 1111$$
$$= 0011\ 1110\ 1100\ 0001$$

Döngü 2:

NibbleSub

$$F = NibbleSub(0011),\ NibbleSub(1110),$$
$$NibbleSub(1100),\ NibbleSub(0001)$$
$$= 0001\ 0000\ 0101\ 0100$$

ShiftRow

$$G = ShiftRow(0001\ 0000\ 0101\ 0100) = 0001\ 0100\ 0101\ 0000$$

Son döngüde MixColumn yok.

KeyAddition

$$H = G \oplus K_2 = 0001\ 0100\ 0101\ 0000 \oplus 0110\ 0110\ 1001\ 0110$$
$$= 0111\ 0010\ 1100\ 0110$$

$H =$ 0111 0010 1100 0110 (son şifreli metin)

4.3.1.9 Mini-AES Deşifreleme

Şifreli metinden açık metni elde etmek için şifreleme sürecinin tersi bir süreç işletilir.

$$\text{Mini-AES Deşifreleme} = \left(\sigma K_2 \circ \pi \circ \gamma \circ \sigma K_1 \circ \theta \circ \pi \circ \gamma \circ \sigma K_0\right)^{-1} \quad (4.8)$$

$$= \left(\sigma K_2 \circ \pi \circ \gamma \circ \sigma K_1 \circ \theta \circ \pi \circ \gamma \circ \sigma K_0\right)^{-1}$$

$$= \sigma K_0^{-1} \circ \gamma^{-1} \circ \pi^{-1} \circ \theta^{-1} \circ \sigma K_1^{-1} \circ \gamma^{-1} \circ \pi^{-1} \circ \sigma K_2$$

$$= \sigma K_0 \circ \gamma^{-1} \circ \pi \circ \theta \circ \sigma K_1 \circ \gamma^{-1} \circ \pi \circ \sigma K_2$$

σK_i^{-1} ifadesi XOR işleminin tersi, kendisi olduğu için σK_i ifadesine eşittir. MixColumn işleminde özel olarak seçilen sabit matris, MixColumn işleminin tersinin yine kendisi olmasını sağlamaktadır $\left(\theta^{-1} = \theta\right)$. ShiftRow işleminde ise, birinci satır sabit kalmakta, ikinci satır ise bir sütün sola döndürülmektedir. Satırların iki nibble'dan oluşması nedeniyle ikinci satır bir nibble daha döndürüldüğünde ilk değer elde edilir. Bu nedenle ShiftRow dönüşümünün tersi yine kendisidir $\left(\pi^{-1} = \pi\right)$.

4.3.1.10 Mini-AES ve AES Karşılaştırması

- Mini-AES veriyi 16 bit bloklara bölerek şifrelerken, AES 128 bitlik bir blok şifreleyicidir.
- AES 128, 192 ve 256 bitlik anahtar uzunlukları kullanırken, Mini-AES 16 bitlik anahtar uzunlukları kullanır.

- AES algoritmasının 128 biti 4×4 sekizlilik matris olarak ifade edilirken, Mini-AES'de 2×2 nibble'lık olarak ifade edilir.
- AES 10 döngüden oluşur ve Mini-AES ile aynı yapıya sahiptir.
- Son döngüde MixColumn dönüşümü yoktur ve ilk döngüden önce ek bir KeyAddition vardır. Bu özellikler sayesinde şifreleme ve deşifreleme aynı yapıya sahip olur.
- AES şifreleme algoritması ByteSub, ShiftRow, MixColumn ve KeyAddition elemanlarına sahiptir.
- ByteSub, NibbleSub ile aynı fakat biri sekizli üzerinde işlem yaparken diğeri nibble'lar üzerinde işlem yapar.
- AES ShiftRow dönüşümünde birinci satır aynı kalır, ikinci satır bir sekizli, üçüncü satır iki sekizli, dördüncü satır üç sekizli sola döndürülür.
- MixColumn AES'te verinin tüm kolonlarını 4×4'lük bir matris ile çarpım işlemine tabi tutar (Mini-AES 2×2).
- KeyAddition Mini-AES ile aynıdır.

5 Döngülük Mini-AES Şifreleyicisine uygulanan imkansız diferansiyel saldırının ayrıntıları (5.4) bölümünde verilmiştir.

4.3.2 AES İmkansız Diferansiyel Saldırı

2000 yılında Biham ve Keller AES şifreleyicisinde 5 döngüye kadar imkansız diferansiyel saldırıyı sunmuştur [80]. Bu saldırı 2001 yılında Cheon tarafından AES'in 6 döngüsüne uygulanarak iyileştirilmiştir [81]. Raphael C.-W. Phan 2003 yılında AES şifreleyicisi üzerinde 7. döngüye kadar geliştirmiştir. Bu saldırı AES şifreleyiciler üzerinde bilinen en iyi imkansız diferansiyel saldırıdır. Bu saldırı AES anahtar planlamasındaki zayıflığı kullanarak gerçekleşir. Algoritma 3.6'da kaynak kodu gösterilen anahtar planlama algoritması Algoritma 4.4'de pseudo kod olarak verilmiştir.

$j \in \{0,\ldots,4\times(R+1)-1\}$

W, $4\times(R+1)$ elemanlı bir dizidir. W dizisinin her bir 4 sözcüğü birleştirilerek bir döngü anahtarı elde edilir.

N: AES N adet 32 bitlik sözcükler içerir. AES-128 için $N=4$, AES-192 için $N=6$, AES-256 için $N=8$' dir.

$k \in \{1,2,\ldots\}, rcon[k]$ döngü sabitidir.

$f, g : \{0,1\}^{32} \rightarrow \{0,1\}^{32}$ doğrusal olmayan permütasyon.

if $(j \bmod N)=0$ then

$$W[j]=W[j-N]\oplus f(W[j-1])\oplus rcon[jdivN]$$

else if $((N>6)\ and\ (j \bmod N)=4)$ then

$$W[j]=W[j-N]\oplus g(W[j-1])$$

else

$$W[j]=W[j-N]\oplus W[j-1]$$

Algoritma 4.4. AES Anahtar Planlama Algoritması

4.3.2.1 4 Döngülük İmkansız Diferansiyel

İlk sekizli dışındaki sekizli değerleri eşit olan iki açık metin çifti 4 döngü sonra (1, 8, 11, 14), (2, 5, 12, 15), (3, 6, 9, 16), (4, 7, 10, 13) sekizli konumlarının herhangi birinde eşit olamaz [82] [83].

4.3.2.2 7 Döngülük AES-192'ye Saldırı

- Eğer 7. döngü anahtarını (RK_7) bilirsek veya tahmin edebilirsek, anahtarı oluşturan 4 anahtar sözcüğü $W[28], W[29], W[30]$ ve $W[31]$ değerlerini elde edebiliriz [84].
- Anahtar planlama algoritmasından aşağıdaki eşitlikler elde edilir:

$$W[28] = W[22] \oplus W[27] \tag{4.9}$$

$$W[29] = W[23] \oplus W[28]$$

$$W[30] = W[24] \oplus W[29] \oplus rcon[5]$$

$$W[31] = W[25] \oplus W[30]$$

- 2. eşitlikten $W[29]$ ve $W[28]$ değerleri bilindiği için, $W[23]$ değeri elde edilir.
- 3. eşitlikten $W[30]$, $W[29]$ ve $rcon[5]$ değerleri bilindiği için, $W[24]$ değeri elde edilir.
- 4. eşitlikten $W[30]$ ve $W[31]$ değerleri bilindiği için, $W[25]$ değeri elde edilebilir.
- Böylelikle RK_5 döngü anahtarının son kolonu $(W[23])$, RK_6 döngü anahtarının ilk ve ikinci kolonuna ($W[24]$ ve $W[25]$) sahip oluruz.

Saldırının Aşamaları

i. 2^{32} adet açık metin seçiniz. İlgili açık metinler (1, 6, 11, 16). sekizliler dışında aynı değerlere sahip olsunlar. Bu küme bir yapı olarak nitelendirilir. $(P, P*)$ açık metin çifti sayısı $2^{32} \times 2^{32} \times \left(\frac{1}{2}\right) \approx 2^{63}$ 'tür.

ii. 2^{60} yapı seçiniz. Böylelikle $2^{60} \times 2^{32} = 2^{92}$ açık metin, $2^{60} \times 2^{63} = 2^{123}$ açık metin çiftine sahip oluruz. Şifreli metinleri 3, 4, 6, 7, 9, 10, 13 ve 16 nolu sekizli konumları eşit olan (yani farkı 0 olan) açık metin çiftleri seçilir. Bu ölçüte uygun çift sayısı $2^{123} \times 2^{-64} = 2^{59}$'dur.

iii. $RK_7 \in \{0,1\}^{128}$ tüm değerleri için $W[28]$, $W[29]$, $W[30]$, $W[31]$ sözcükleri tahmin edilir. Anahtar planlama algoritması kullanılarak RK_6 döngü anahtarının ilk ve ikinci kolonu olan $W[24]$ ve $W[25]$ sözcükleri elde edilir.

iv. Tüm şifreli metin çiftleri $(C, C*)$ için,

$$C_6 = SB^{-1} \circ SR^{-1}(C \oplus RK_7) \tag{4.10}$$

$$C_6* = SB^{-1} \circ SR^{-1}(C* \oplus RK_7) \tag{4.11}$$

değerleri hesaplanır.

v. RK_6 anahtarının hesaplanmış olan birinci ve ikinci kolonu ile,

$$C_5 = SB^{-1} \circ SR^{-1} \circ MC^{-1}(C_6 \oplus RK_6) \tag{4.12}$$

$$C_5* = SB^{-1} \circ SR^{-1} \circ MC^{-1}(C_6* \oplus RK_6) \tag{4.13}$$

ilk iki kolon için hesaplanır. $MC^{-1}(C_5 \oplus C_5*)$ farkı imkansız sekizli konumlarında 0 olan çiftler seçilir. Bu olayın oluşma olasılığı $q = 2^{-32} \times 4 = 2^{-30}$'dur. Kalan çift sayısı $2^{59} \times 2^{-30} = 2^{29}$ olur.

vi. 2^{29} $(P, P*)$ açık metin çiftinin herbiri için, RK_0 anahtarının (1, 6, 11, 16) sekizli konumlarına denk düşen 4 sekizli için 2^{32} adet değerin tümü tahmin edilir.

a. $MC \circ SR \circ SB(P \oplus RK_0)$ ve $MC \circ SR \circ SB(P* \oplus RK_0)$ ifadeleri hesaplanır.

b. MC dönüşümünden sonra bir sekizli dışında 0 farkına sahip çiftler elde etme olasılığı, $p = 2^{-24} \times 4 = 2^{-22}$ olarak elde edilir.

c. Böyle bir fark imkansızdır ve böyle bir farkı üreten RK_0 yanlış anahtar değeridir.

vii. Kalan 2^{29} çiftin analizinden sonra, $2^{32}(1-2^{-22})^{2^{29}} \approx 2^{32e^{-27}} \approx 2^{-152}$ RK_0 anahtarının 4 sekizlisinin yanlış değerleri kalır.

```
public imkansizDiferansiyelSaldiri(string _anahtar,int _donguSayisi)
{
        anahtar = _anahtar;
        donguSayisi = _donguSayisi;
        aes = new Phd.Cryptanalysis.Cipher.AES(anahtar, donguSayisi);
        int j =0;
        for (int i = 0; i < 16; i++)
        {
           if (i == 0 || i == 5 || i == 10 || i == 15)
           {
              text = text + "{" + j.ToString() + "}";
```

```
            }
            else
            {
                text = text + "".PadLeft(32,'0');
            }
        }
        for (int i = 0; i < n; i++)
        {
            string x = BitConversion.ConvertToBaseTwoNotation(i, 32);
            string p1 =  string.Format(text, x.Substring(0, 32), x.Substring(32, 32),
                                        x.Substring(64, 32), x.Substring(96, 32));
            pText.Add(p1);
            cText.Add(p1, aes.Calculate(p1));
        }
        foreach (string s1 in pText)
        {
            foreach (string s2 in pText)
            {
                if (s1 != s2 && kriterKontrol(cText[s1].ToString(),
cText[s2].ToString()))
                {
                    Cift pCift = new Cift();
                    pCift.P1 = s1;
                    pCift.P2 = s2;
                    pTextCift.Add(pCift);
                }
            }
        }

        Anahtar a = new Anahtar(128, donguSayisi);
        for (long li = 0; li <= long.MaxValue; li++)
        {
            string k71 = BitConversion.ConvertToBaseTwoNotation(li,64);
            for (long lj = 0; lj <= long.MaxValue; lj++)
            {
                string k72 = BitConversion.ConvertToBaseTwoNotation(lj, 64);
                string k7 =k71 + k72;
                a.anahtarEldeEt(k7, 7);
                string k6 = a.W[23] + a.W[24];
                foreach (Cift pCift in pTextCift)
                {
                    string C61 =   aes.InvSubBytes(
                                   aes.InvShiftRows(BitIslem.XOR(pCift.P1,  k7)));
                    string C62 = aes.InvSubBytes(
                                   aes.InvShiftRows(BitIslem.XOR(pCift.P1,  k7)));
                    string C51 =   aes.InvSubBytes(aes.InvShiftRows(
                              aes.InvMixColumns(BitIslem.XOR(C61, k6), 2)));
                   string C52 =    aes.InvSubBytes(aes.InvShiftRows(
```

```
                    aes.InvMixColumns(BitIslem.XOR(C62, k6), 2)));
                byte[,] MC5 = aes.InvMixColumns(
                                        BitIslem.XOR(C51, C52),2);
                if (    MC5[0, 0] == 0 && MC5[1, 3] == 0 ||
                        MC5[0, 1] == 0 && MC5[1, 0] ==0 ||
                        MC5[0, 2] == 0 && MC5[1, 1] == 0 ||
                        MC5[0, 3] == 0 && MC5[1, 2] ==0 )
                {
                   pTextCift.Remove(pCift);
                 }
             }
          }
       }
       ArrayList kDogruAnahtar = new ArrayList();
       for (int i = 0; i < n; i++)
       {
          string x = BitConversion.ConvertToBaseTwoNotation(i, 32);
          string k = string.Format(text, x.Substring(0, 32), x.Substring(32, 32),
                                   x.Substring(64, 32), x.Substring(96, 32));
          int count = 0;
          foreach (Cift pCift in pTextCift)
          {
             byte[,] C11 =  aes.MixColumns(aes.ShiftRows(
                                   aes.SubBytes(BitIslem.XOR(pCift.P1, k))));
             byte[,] C12 = aes.MixColumns(aes.ShiftRows(
                                   aes.SubBytes(BitIslem.XOR(pCift.P2, k))));
             count =     (C11[0, 0] == C12[0, 0] ? 1 : 0) +
                         (C11[1, 1] == C12[1, 1] ? 1 : 0) +
                         (C11[2, 2] == C12[2, 2] ? 1 : 0) +
                         (C11[3, 3] == C12[3, 3] ? 1 : 0);
             if (count > 1)
             {
                break;
             }
          }
          if (count < 2)
          {
             kDogruAnahtar.Add(i);
          }
       }
}
```

Algoritma 4.5. AES İmkansız Diferansiyel Saldırı Örnek Kaynak Kodu

4.3.2.3 Sonuçlar

- Aşama *iv*, $2\times2^{128}\times2^{59} = 2^{188}$ bir döngülük şifreleme gerektirir.
- Aşama *v*, $2\times2^{128}\times2^{59} = 2^{188}$ bir döngülük şifreleme gerektirir.
- Aşama *vi*, $2^{128}\times2\times2^{32}\left\{1+\left(1-2^{-22}\right)+\left(1-2^{-22}\right)^2+\cdots+\left(1-2^{-22}\right)^{2^{29}}\right\}\approx 2^{183}$ bir döngülük şifreleme gerektirir.
- Sonuç olarak, $2^{188}+2^{188}+2^{183}\approx 2^{189}$ bir döngülük şifreleme veya $\frac{2^{189}}{7}\approx 2^{186}$ 7 döngülük AES-192 şifreleme gerektirir.
- $\left(RK_0, RK_6, RK_7\right)$ silinen anahtar değerlerinin saklanması için $2^{128}\times2^{32} = 2^{160}$ bit = $\frac{2^{160}}{128}\approx 2^{153}$ n-bit sözcük bellek gereklidir.

4.4 İnterpolasyon Saldırıları

Blok şifreleyicilerdeki interpolasyon saldırılarının karmaşıklığı, polinomsal ifadedeki terim sayısına ve elde edilen polinomun derecesine bağlıdır. Bazı durumlarda, S-kutusu veya döngü fonksiyonu cebirsel olarak ifade edilebilmektedir. Böyle durumlarda Lagrange interpolasyonu kullanılarak, döngü fonksiyonu ve S-kutusunun cebirsel ifadesi elde edilebilmektedir.

Bu çalışmada Lagrange interpolasyonu kullanılarak $GF(2^4)$ sonlu cisminde çeşitli S-kutularının cebirsel ifadeleri yazılan uygulama aracılığıyla bulunmuş ve güçleri karşılaştırılmıştır. Yine AES şifreleyecisinin S-kutusuna Lagrange interpolasyonu uygulanmış ve $GF(2^8)$ sonlu cisminde AES S-kutusunun cebirsel ifadesi elde edilmiştir. Bu işlemlerin gerçeklenmesinde kullanılmak üzere sonlu cisimde ters alma işlemini gereçekleştiren Euclid algoritması gerçeklenmiştir.

4.4.1 Sonlu Cisim Teorisi

p asal sayı, n pozitif tamsayı olma şartı ile $m. = p^n$ ise $m.$ dereceden bir sonlu cisim vardır demektir. p^n elemanlı bir sonlu cisim Galois alanı olarak tanımlanır ve $GF\left(p^n\right)$ ile gösterilir. p asal sayısı için $GF(p)$, Z_p'ye eşittir. $GF\left(p^n\right)$, $n>1$ için $GF(p)$' den $GF(p)$ uzayındaki $n.$ dereceden indirgenemez polinom kullanılarak elde edilir [54] (bknz. Bölüm 2.2).

4.4.2 Sonlu Cisimde Ters Alma

n bit iki polinomun çarpımının kalanı, seçilen indirgenemez polinoma göre 1 ise o zaman iki polinom birbirinin seçilen indirgenemez polinoma göre tersidir. İndirgenemez bir polinoma göre ters alma işlemi için iki yöntem önerilebilir. Bu yöntemlerden ilki $GF\left(2^n\right)$ için tablo oluşturmaktır. Eğer n değeri küçük bir değer ise bu yöntem etkili olabilir.

Örnek 4.12. $GF\left(2^4\right)$ için indirgenemez polinom olarak $x^4+x^3+x^2+x+1$ polinomu seçilsin. Bu cismin karakteristiği 2, eleman sayısı 16 ve bu cisimdeki üreteç eleman $\beta=(0011)=\alpha+1$'dir[10]. β üreteç elemanının üslerini düşünelim.

$$\beta^0=(0001),\ \beta^1=(0011),\ \beta^2=0101),\ \beta^3=(1111)$$
$$\beta^4=(1110),\ \beta^5=(1101),\ \beta^6=(1000),\ \beta^7=(0111)$$
$$\beta^8=(1001),\ \beta^9=(0100),\ \beta^{10}=(1100),\ \beta^{11}=(1011)$$
$$\beta^{12}=(0010),\ \beta^{13}=(0110),\ \beta^{14}=(1010),\ \beta^{15}=(0001)$$

Dolayısıyla $a\in GF\left(2^n\right)$ ve $a=\beta^i$ olmak üzere a değişkenin çarpmaya göre tersi $a^{-1}=\beta^{(-i)\bmod\left(2^n-1\right)}$ şeklinde verilebilir. Bunu göz önüne alarak elemanların tersi ve polinomsal yazılışları aşağıdaki gibidir:

β^0	(0001)	1	Tersi	β^{15}	(0001)	1
β^1	(0011)	$x+1$	Tersi	β^{14}	(1010)	x^3+x
β^2	(0101)	x^2+1	Tersi	β^{13}	(0110)	x^2+x

[10] α ile tüm cisim elemanları üretilememektedir.

β^3	(1111)	x^3+x^2+x+1	Tersi	β^{12}	(0010)	x
β^4	(1110)	x^3+x^2+x	Tersi	β^{11}	(1011)	x^3+x+1
β^5	(1101)	x^3+x^2+1	Tersi	β^{10}	(1100)	x^3+x^2
β^6	(1000)	x^3	Tersi	β^9	(0100)	x^2
β^7	(0111)	x^2+x+1	Tersi	β^8	(1001)	x^3+1
β^8	(1001)	x^3+1	Tersi	β^7	(0111)	x^2+x+1
β^9	(0100)	x^2+1	Tersi	β^6	(1000)	x^3
β^{10}	(1100)	x^3+x^2	Tersi	β^5	(1101)	x^3+x^2+1
β^{11}	(1011)	x^3+x+1	Tersi	β^4	(1110)	x^3+x^2+x
β^{12}	(0010)	x^2	Tersi	β^3	(1111)	x^3+x^2+x+1
β^{13}	(0110)	x^2+x	Tersi	β^2	(0101)	x^2+1
β^{14}	(1010)	x^3+x	Tersi	β^1	(0011)	$x+1$
β^{15}	(0001)	1	Tersi	β^0	(0001)	1

Örnek 4.13. İndirgenemez polinomunun $p(x)=x^4+x+1$ olması ve α değerinin bir kök olması koşulu ile aşağıdaki tablo elde edilir.

1	$\alpha^8=\alpha^4+\alpha^2+\alpha=\alpha^2+1$
α	$\alpha^9=\alpha^3+\alpha$
α^2	$\alpha^{10}=\alpha^4+\alpha^2=\alpha^2+\alpha+1$
α^3	$\alpha^{11}=\alpha^3+\alpha^2+\alpha$
$\alpha^4=\alpha+1$	$\alpha^{12}=\alpha^4+\alpha^3+\alpha^2=\alpha^3+\alpha^2+\alpha+1$
$\alpha^5=\alpha^2+\alpha$	$\alpha^{13}=\alpha^4+\alpha^3+\alpha^2+\alpha=\alpha^3+\alpha^2+1$
$\alpha^6=\alpha^3+\alpha^2$	$\alpha^{14}=\alpha^4+\alpha^3+\alpha=\alpha^3+1$
$\alpha^7=\alpha^4+\alpha^3=\alpha^3+\alpha+1$	$\alpha^{15}=\alpha^4+\alpha=1$

$GF(2^4)$ uzayında bir elemanın tersi aşağıdaki gibi bulunur.

$0\rightarrow 0$

$1\rightarrow 1$

$2\ (0010)=\alpha\rightarrow tersi\ \alpha^{14}=1001=9$

$3\ (0011)=\alpha+1\rightarrow tersi\ \alpha^{11}=1110=E$

$4\ (0100)=\alpha^2\rightarrow tersi\ \alpha^{13}=1101=D$

$5\ (0101)=\alpha^2+1=\alpha^8\rightarrow tersi\ \alpha^7=1011=B$

$6\ (0110)=\alpha^2+\alpha=\alpha^5\rightarrow tersi\ \alpha^{10}=0111=7$

$7\ (0111)=\alpha^2+\alpha+1=\alpha^{10}\rightarrow tersi\ \alpha^5=0110=6$

$8\ (1000) = \alpha^3 \rightarrow tersi\ \alpha^{12} = 1111 = F$

$9\ (1001) = \alpha^3 + 1 = \alpha^{14} \rightarrow tersi\ \alpha = 0010 = 2$

$\mathrm{A}\ (1010) = \alpha^3 + \alpha = \alpha^9 \rightarrow tersi\ \alpha^6 = 1100 = \mathrm{C}$

$\mathrm{B}\ (1011) = \alpha^3 + \alpha + 1 = \alpha^7 \rightarrow tersi\ \alpha^8 = 0101 = 5$

$\mathrm{C}\ (1100) = \alpha^3 + \alpha^2 = \alpha^6 \rightarrow tersi\ \alpha^9 = 1010 = \mathrm{A}$

$\mathrm{D}\ (1101) = \alpha^3 + \alpha^2 + 1 = \alpha^{13} \rightarrow tersi\ \alpha^2 = 0100 = 4$

$\mathrm{E}\ (1110) = \alpha^3 + \alpha^2 + \alpha = \alpha^{11} \rightarrow tersi\ \alpha^4 = 0011 = 3$

$\mathrm{F}\ (1111) = \alpha^3 + \alpha^2 + \alpha + 1 = \alpha^{12} \rightarrow tersi\ \alpha^3 = 1000 = 8$

Örnek 4.13'de, n (örnek için 4) küçük olduğu için tablo kolay bir şekilde elde edilmiştir. Örneğin $n = 8$ için $GF\left(2^8\right)$ sonlu cisminde 0 elemanı ile birlikte 256 adet eleman mevcuttur. Bu durumda, sonlu cisimde ters alma işlemi ikili Euclidean algoritması kullanılarak gerçeklenebilir. Sonlu cisimde ters alma işlemi için ikili Euclidean algoritması Algoritma 4.6' da gösterilmiştir [85]:

Giriş Değeri : $a \in GF\left(2^m\right), a \neq 0$

Çıkış Değeri: $a^{-1} \bmod f$

$u = a, v = f, g_1 = 1, g_2 = 0$

x değeri, *u* değerini tam olarak böldüğü sürece aşağıdaki işlemler gerçeklenir:

$u = \dfrac{u}{x}$

Eğer x, g_1 değerini tam olarak bölerse $g_1 = \dfrac{g_1}{x}$

aksi takdirde $g_1 = \dfrac{(g_1 + f)}{x}$ ifadesi hesaplanır.

Eğer $u = 1$ ise g_1 değeri döndürülür.

Eğer $derece(u) < derece(v)$ ise $u \leftrightarrow v$, $g_1 \leftrightarrow g_2$ yer değiştirmeleri yapılır.

$u = u + v, \quad g_1 = g_1 + g_2$

Algoritma 4.6. Ters Alma İşlemi İçin İkili Euclidean Algoritması

```
private int gcd(int x, int y)
{
        int[] remainder = new int[100];
        int[] auxiliary = new int[100];
        int[] quotient = new int[100];
        remainder[1] = y;
        remainder[2] = x;
        auxiliary[1] = 0;
        auxiliary[2] = 1;
        int i = 2;
        int inverse =0;
        while (remainder[i] > 1)
        {
                i = i + 1;
                Sonuc bolum = fieldBol(remainder[i - 2], remainder[i - 1]);
                remainder[i] = bolum.Kalan ;
                quotient[i] = bolum.Bolum;
                auxiliary[i] = fieldCarp(quotient[i] , auxiliary[i - 1]) ^ auxiliary[i - 2];
          inverse = auxiliary[i];
        }
        return inverse;
}
```

Algoritma 4.7. Ters Alma İşlemi İçin Euclidean Algoritması Örnek Kaynak Kodu

Algoritma 4.6' da gösterilen ikili Euclidean algoritması (1110) değerinin tersini Örnek 4.13'de gösterildiği gibi, $P_1(x)=x+1$ ya da (0011) şeklinde bulacaktır. Bu sonuç, derece($P_1(x)$)<4 ve derece($P_2(x)$)<3 olmak üzere

$$P_1(x)(x^3+x^2+x)+P_2(x)(x^4+x+1)=1$$

ifadesinde $P_1(x)$, x^3+x^2+x polinomunun çarpmaya göre tersi olacak şekilde gösterilebilir. Yukarıdaki ifadede $P_1(x)$ ve $P_2(x)\in Z_2[x]$'tir.

4.4.3 İnterpolasyon Saldırısı

Blok şifreleyiciler üzerinde yeni bir saldırı türü olan interpolasyon saldırısı Jakobsen ve Knudsen tarafından sunulmuştur [47]. S-kutuları gibi basit cebirsel fonksiyonlara saldırmada faydalıdır. İnterpolasyon saldırısı [47] [86] lagrange interpolasyon formülasyonunu [86] temel alır.

Tanım 4.9. R bir cisim ve $x_1,\ldots,x_n,\ y_1,\ldots,y_n,\ x_i$ değerlerinin tek olması koşulu ile $2n$ eleman olsun. Bu durumda, R cisminde en fazla $(n-1).$ dereceden bir polinom olan $s(x)$ matematiksel olarak şu şekilde ifade edilir:

$$s(x)=\sum_{i=1}^{n} y_i \prod_{1\le j\le n,\, j\ne i} \frac{x-x_j}{x_i-x_j}. \tag{4.14}$$

Nyberg doğrusal kriptanaliz ve diferansiyel kriptanalize karşı güçlü olan ve ters haritalamayı temel alan ve matematiksel ifadesi aşağıda bulunan bir S-kutusu sunmuştur [87]:

$$s(x)=x^{-1}, x\in GF(2^n),\ f(0)=0. \tag{4.15}$$

[87]'deki çalışmada da ifade edildiği gibi, bu S-kutusunun zayıf tarafı cebirsel yapısının basit olmasıdır. Bu zaaf şifreleyicinin interpolasyon saldırılarına maruz kalmasına neden olmaktadır. Bu zaafın giderilmesi amacı ile Shark, Square ve AES şifreleyicilerinde ters haritalamadan sonra doğrusal dönüşüm kullanılmıştır [87]. Camellia [88] şifreleyicisinde hem S-kutusu girişinden önce, hem de ters haritalamadan sonra doğrusal dönüşüm kullanılmıştır. Örnek 4.14'te $GF(2^4)$ sonlu cisminde $s(x)=x^{-1}$ ifadesini lagrange interpolasyonu ile elde edeceğiz.

Örnek 4.14. Tablo 4.7'de verilen S-kutusu giriş ve çıkış değerleri kullanılarak lagrange interpolasyonu ile $GF(2^4)$ sonlu cisminde, sonlu cisim aritmetiği ile S-kutusunun cebirsel ifadesi bulunacaktır. Lagrange interpolasyonu uygulanacak S-kutusu x^4+x+1 indirgenemez polinomu kullanılarak elde edilmiştir.

$$s(x)_0=0.\frac{x+1}{0+1}\frac{x+2}{0+2}.\frac{x+3}{0+3}.\frac{x+4}{0+4}.\frac{x+5}{0+5}.\frac{x+6}{0+6}.\frac{x+7}{0+7}.\frac{x+8}{0+8}.\frac{x+9}{0+9}.\frac{x+a}{0+a}.\frac{x+b}{0+b}.\frac{x+c}{0+c}.\frac{x+d}{0+d}.\frac{x+e}{0+e}.\frac{x+f}{0+f}$$

$$s(x)_1=1.\frac{x}{1+0}.\frac{x+2}{1+2}.\frac{x+3}{1+3}.\frac{x+4}{1+4}.\frac{x+5}{1+5}.\frac{x+6}{1+6}.\frac{x+7}{1+7}.\frac{x+8}{1+8}.\frac{x+9}{1+9}.\frac{x+a}{1+a}.\frac{x+b}{1+b}.\frac{x+c}{1+c}.\frac{x+d}{1+d}.\frac{x+e}{1+e}.\frac{x+f}{1+f}$$

$$s(x)_2=9.\frac{x}{2+0}.\frac{x+1}{1+2}.\frac{x+3}{2+3}.\frac{x+4}{2+4}.\frac{x+5}{2+5}.\frac{x+6}{2+6}.\frac{x+7}{2+7}.\frac{x+8}{2+8}.\frac{x+9}{2+9}.\frac{x+a}{2+a}.\frac{x+b}{2+b}.\frac{x+c}{2+c}.\frac{x+d}{2+d}.\frac{x+e}{2+e}.\frac{x+f}{2+f}$$

$$s(x)_3=e.\frac{x}{3+0}.\frac{x+1}{3+1}.\frac{x+2}{3+2}.\frac{x+4}{3+4}.\frac{x+5}{3+5}.\frac{x+6}{3+6}.\frac{x+7}{3+7}.\frac{x+8}{3+8}.\frac{x+9}{3+9}.\frac{x+a}{3+a}.\frac{x+b}{3+b}.\frac{x+c}{3+c}.\frac{x+d}{3+d}.\frac{x+e}{3+e}.\frac{x+f}{3+f}$$

$$s(x)_4=d.\frac{x}{4+0}.\frac{x+1}{4+1}.\frac{x+2}{4+2}.\frac{x+3}{4+3}.\frac{x+5}{4+5}.\frac{x+6}{4+6}.\frac{x+7}{4+7}.\frac{x+8}{4+8}.\frac{x+9}{4+9}.\frac{x+a}{4+a}.\frac{x+b}{4+b}.\frac{x+c}{4+c}.\frac{x+d}{4+d}.\frac{x+e}{4+e}.\frac{x+f}{4+f}$$

$$s(x)_5=b.\frac{x}{5+0}.\frac{x+1}{5+1}.\frac{x+2}{5+2}.\frac{x+3}{5+3}.\frac{x+4}{5+4}.\frac{x+6}{5+6}.\frac{x+7}{4+7}.\frac{x+8}{4+8}.\frac{x+9}{4+9}.\frac{x+a}{5+a}.\frac{x+b}{5+b}.\frac{x+c}{5+c}.\frac{x+d}{5+d}.\frac{x+e}{5+e}.\frac{x+f}{5+f}$$

$$s(x)_6=7.\frac{x}{6+0}.\frac{x+1}{6+1}.\frac{x+2}{6+2}.\frac{x+3}{6+3}.\frac{x+4}{6+4}.\frac{x+5}{6+5}.\frac{x+7}{6+7}.\frac{x+8}{6+8}.\frac{x+9}{6+9}.\frac{x+a}{6+a}.\frac{x+b}{6+b}.\frac{x+c}{6+c}.\frac{x+d}{6+d}.\frac{x+e}{6+e}.\frac{x+f}{6+f}$$

$$s(x)_7 = 6.\frac{x}{7+0}.\frac{x+1}{7+1}.\frac{x+2}{7+2}.\frac{x+3}{7+3}.\frac{x+4}{7+4}.\frac{x+5}{7+5}.\frac{x+6}{7+6}.\frac{x+8}{7+8}.\frac{x+9}{7+9}.\frac{x+a}{7+a}.\frac{x+b}{7+b}.\frac{x+c}{7+c}.\frac{x+d}{7+d}.\frac{x+e}{7+e}.\frac{x+f}{7+f}$$

$$s(x)_8 = f.\frac{x}{8+0}.\frac{x+1}{8+1}.\frac{x+2}{8+2}.\frac{x+3}{8+3}.\frac{x+4}{8+4}.\frac{x+5}{8+5}.\frac{x+6}{8+6}.\frac{x+7}{8+7}.\frac{x+9}{8+9}.\frac{x+a}{8+a}.\frac{x+b}{8+b}.\frac{x+c}{8+c}.\frac{x+d}{8+d}.\frac{x+e}{8+e}.\frac{x+f}{8+f}$$

$$s(x)_9 = 2.\frac{x}{9+0}.\frac{x+1}{9+1}.\frac{x+2}{9+2}.\frac{x+3}{9+3}.\frac{x+4}{9+4}.\frac{x+5}{9+5}.\frac{x+6}{9+6}.\frac{x+7}{9+7}.\frac{x+8}{9+8}.\frac{x+a}{9+a}.\frac{x+b}{9+b}.\frac{x+c}{9+c}.\frac{x+d}{9+d}.\frac{x+e}{9+e}.\frac{x+f}{9+f}$$

$$s(x)_a = c.\frac{x}{a+0}.\frac{x+1}{a+1}.\frac{x+2}{a+2}.\frac{x+3}{a+3}.\frac{x+4}{a+4}.\frac{x+5}{a+5}.\frac{x+6}{a+6}.\frac{x+7}{a+7}.\frac{x+8}{a+8}.\frac{x+9}{a+9}.\frac{x+b}{a+b}.\frac{x+c}{a+c}.\frac{x+d}{a+d}.\frac{x+e}{a+e}.\frac{x+f}{a+f}$$

$$s(x)_b = 5.\frac{x}{b+0}.\frac{x+1}{b+1}.\frac{x+2}{b+2}.\frac{x+3}{b+3}.\frac{x+4}{b+4}.\frac{x+5}{b+5}.\frac{x+6}{b+6}.\frac{x+7}{b+7}.\frac{x+8}{b+8}.\frac{x+9}{b+9}.\frac{x+a}{b+a}.\frac{x+c}{b+c}.\frac{x+d}{b+d}.\frac{x+e}{b+e}.\frac{x+f}{b+f}$$

$$s(x)_c = a.\frac{x}{c+0}.\frac{x+1}{c+1}.\frac{x+2}{c+2}.\frac{x+3}{c+3}.\frac{x+4}{c+4}.\frac{x+5}{c+5}.\frac{x+6}{c+6}.\frac{x+7}{c+7}.\frac{x+8}{c+8}.\frac{x+9}{c+9}.\frac{x+a}{c+a}.\frac{x+b}{c+b}.\frac{x+d}{c+d}.\frac{x+e}{c+e}.\frac{x+f}{c+f}$$

$$s(x)_d = 4.\frac{x}{d+0}.\frac{x+1}{d+1}.\frac{x+2}{d+2}.\frac{x+3}{d+3}.\frac{x+4}{d+4}.\frac{x+5}{d+5}.\frac{x+6}{d+6}.\frac{x+7}{d+7}.\frac{x+8}{d+8}.\frac{x+9}{d+9}.\frac{x+a}{d+a}.\frac{x+b}{d+b}.\frac{x+c}{d+c}.\frac{x+e}{d+e}.\frac{x+f}{d+f}$$

$$s(x)_e = 3.\frac{x}{e+0}.\frac{x+1}{e+1}.\frac{x+2}{e+2}.\frac{x+3}{e+3}.\frac{x+4}{e+4}.\frac{x+5}{e+5}.\frac{x+6}{e+6}.\frac{x+7}{e+7}.\frac{x+8}{e+8}.\frac{x+9}{e+9}.\frac{x+a}{e+a}.\frac{x+b}{e+b}.\frac{x+c}{e+c}.\frac{x+d}{e+d}.\frac{x+f}{e+f}$$

$$s(x)_f = 8.\frac{x}{f+0}.\frac{x+1}{f+1}.\frac{x+2}{f+2}.\frac{x+3}{f+3}.\frac{x+4}{f+4}.\frac{x+5}{f+5}.\frac{x+6}{f+6}.\frac{x+7}{f+7}.\frac{x+8}{f+8}.\frac{x+9}{f+9}.\frac{x+a}{f+a}.\frac{x+b}{f+b}.\frac{x+c}{f+c}.\frac{x+d}{f+d}.\frac{x+e}{f+e}$$

$$s(x)_0 = 0$$

$$s(x)_1 = x^{15} + x^{14} + x^{13} + x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x$$

$$s(x)_2 = 9x^{15} + x^{14} + 2x^{13} + 4x^{12} + 8x^{11} + 3x^{10} + 6x^9 + cx^8 + bx^7 + 5x^6 + ax^5 + 7x^4 + ex^3 + fx^2 + dx$$

$$s(x)_3 = ex^{15} + x^{14} + 3x^{13} + 5x^{12} + fx^{11} + 2x^{10} + 6x^9 + ax^8 + dx^7 + 4x^6 + cx^5 + 7x^4 + 9x^3 + 8x^2 + bx$$

$$s(x)_4 = dx^{15} + x^{14} + 4x^{13} + 3x^{12} + cx^{11} + 5x^{10} + 7x^9 + fx^8 + 9x^7 + 2x^6 + 8x^5 + 6x^4 + bx^3 + ax^2 + ex$$

$$s(x)_5 = bx^{15} + x^{14} + 5x^{13} + 2x^{12} + ax^{11} + 4x^{10} + 7x^9 + 8x^8 + ex^7 + 3x^6 + fx^5 + 6x^4 + dx^3 + cx^2 + 9x$$

$$s(x)_6 = 7x^{15} + x^{14} + 6x^{13} + 7x^{12} + x^{11} + 6x^{10} + 7x^9 + x^8 + 6x^7 + 7x^6 + x^5 + 6x^4 + 7x^3 + x^2 + 6x$$

$$s(x)_7 = 6x^{15} + x^{14} + 7x^{13} + 6x^{12} + x^{11} + 7x^{10} + 6x^9 + x^8 + 7x^7 + 6x^6 + x^5 + 7x^4 + 6x^3 + x^2 + 7x$$

$$s(x)_8 = fx^{15} + x^{14} + 8x^{13} + cx^{12} + ax^{11} + fx^{10} + x^9 + 8x^8 + cx^7 + ax^6 + fx^5 + x^4 + 8x^3 + cx^2 + ax$$

$$s(x)_9 = 2x^{15} + x^{14} + 9x^{13} + dx^{12} + fx^{11} + ex^{10} + 7x^9 + ax^8 + 5x^7 + bx^6 + cx^5 + 6x^4 + 3x^3 + 8x^2 + 4x$$

$$s(x)_a = cx^{15} + x^{14} + ax^{13} + 8x^{12} + fx^{11} + cx^{10} + x^9 + ax^8 + 8x^7 + fx^6 + cx^5 + x^4 + ax^3 + 8x^2 + fx$$

$$s(x)_b = 5x^{15} + x^{14} + bx^{13} + 9x^{12} + cx^{11} + dx^{10} + 6x^9 + fx^8 + 3x^7 + ex^6 + 8x^5 + 7x^4 + 4x^3 + ax^2 + 2x$$

$$s(x)_c = ax^{15} + x^{14} + cx^{13} + fx^{12} + 8x^{11} + ax^{10} + x^9 + cx^8 + fx^7 + 8x^6 + ax^5 + x^4 + cx^3 + fx^2 + 8x$$

$$s(x)_d = 4x^{15} + x^{14} + dx^{13} + ex^{12} + ax^{11} + bx^{10} + 6x^9 + 8x^8 + 2x^7 + 9x^6 + fx^5 + 7x^4 + 5x^3 + cx^2 + 3x$$

$$s(x)_e = 3x^{15} + x^{14} + ex^{13} + bx^{12} + 8x^{11} + 9x^{10} + 7x^9 + cx^8 + 4x^7 + dx^6 + ax^5 + 6x^4 + 2x^3 + fx^2 + 5x$$

$$s(x)_f = 8x^{15} + x^{14} + fx^{13} + ax^{12} + cx^{11} + 8x^{10} + x^9 + fx^8 + ax^7 + cx^6 + 8x^5 + x^4 + fx^3 + ax^2 + cx$$

$$k_{15} = 1 \oplus 9 \oplus e \oplus d \oplus b \oplus 7 \oplus 6 \oplus f \oplus 2 \oplus c \oplus 5 \oplus a \oplus 4 \oplus 3 \oplus 8$$

$$0001 \oplus 1001 \oplus 1110 \oplus 1101 \oplus 1011 \oplus 0111 \oplus 0110 \oplus 1111 \oplus$$
$$0010 \oplus 1100 \oplus 0101 \oplus 1010 \oplus 0100 \oplus 0011 \oplus 1000 = 0000$$

$$k_{14} = 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 = 1$$

$k_{13} = 1 \oplus 2 \oplus 3 \oplus 4 \oplus 5 \oplus 6 \oplus 7 \oplus 8 \oplus 9 \oplus a \oplus b \oplus c \oplus d \oplus e \oplus f = 0$
$k_{12} = 1 \oplus 4 \oplus 5 \oplus 3 \oplus 2 \oplus 7 \oplus 6 \oplus c \oplus d \oplus 8 \oplus 9 \oplus f \oplus e \oplus b \oplus a = 0$
$k_{11} = 1 \oplus 8 \oplus f \oplus c \oplus a \oplus 1 \oplus 1 \oplus a \oplus f \oplus f \oplus c \oplus 8 \oplus a \oplus 8 \oplus c = 0$
$k_{10} = 1 \oplus 3 \oplus 2 \oplus 5 \oplus 4 \oplus 6 \oplus 7 \oplus f \oplus e \oplus c \oplus d \oplus a \oplus b \oplus 9 \oplus 8 = 0$
$k_9 = 1 \oplus 6 \oplus 6 \oplus 7 \oplus 7 \oplus 7 \oplus 6 \oplus 1 \oplus 7 \oplus 1 \oplus 6 \oplus 1 \oplus 6 \oplus 7 \oplus 1 = 0$
$k_8 = 1 \oplus c \oplus a \oplus f \oplus 8 \oplus 1 \oplus 8 \oplus 8 \oplus a \oplus a \oplus f \oplus c \oplus 8 \oplus c \oplus f = 0$
$k_7 = 1 \oplus b \oplus d \oplus 9 \oplus e \oplus 6 \oplus 7 \oplus c \oplus 5 \oplus 8 \oplus 3 \oplus f \oplus 2 \oplus 4 \oplus a = 0$
$k_6 = 1 \oplus 5 \oplus 4 \oplus 2 \oplus 3 \oplus 7 \oplus 6 \oplus a \oplus b \oplus f \oplus e \oplus 8 \oplus 9 \oplus d \oplus c = 0$
$k_5 = 1 \oplus a \oplus c \oplus 8 \oplus f \oplus 1 \oplus 1 \oplus f \oplus c \oplus c \oplus 8 \oplus a \oplus f \oplus a \oplus 8 = 0$
$k_4 = 1 \oplus 7 \oplus 7 \oplus 6 \oplus 6 \oplus 6 \oplus 7 \oplus 1 \oplus 6 \oplus 1 \oplus 7 \oplus 1 \oplus 7 \oplus 6 \oplus 1 = 0$
$k_3 = 1 \oplus e \oplus 9 \oplus b \oplus d \oplus 7 \oplus 6 \oplus 8 \oplus 3 \oplus a \oplus 4 \oplus c \oplus 5 \oplus 2 \oplus f = 0$
$k_2 = 1 \oplus f \oplus 8 \oplus a \oplus c \oplus 1 \oplus 1 \oplus c \oplus 8 \oplus 8 \oplus a \oplus f \oplus c \oplus f \oplus a = 0$
$k_1 = 1 \oplus b \oplus d \oplus e \oplus 9 \oplus 6 \oplus 7 \oplus a \oplus 4 \oplus f \oplus 2 \oplus 8 \oplus 3 \oplus 5 \oplus c = 0$
$k_0 = 0$

Tablo 4.7. *GF(2^4)* Sonlu Cisminde $s(x) = x^{-1}$ İfadesinin Noktaları

GF(2^4)'te 4-bit değer	İndirgenemez polinom $x^4 + x + 1$ 'e göre ters alma	
	İkili	Hex.
0001	0001	1
0010	1001	9
0011	1110	E
0100	1101	D
0101	1011	B
0110	0111	7
0111	0110	6
1000	1111	F
1001	0010	2
1010	1100	C
1011	0101	5
1100	1010	A
1101	0100	4
1110	0011	3
1111	1000	8

$$s(x) = k_{15}x^{15} + k_{14}x^{14} + k_{13}x^{13} + k_{12}x^{12} + k_{11}x^{11} + k_{10}x^{10} + k_9x^9 + k_8x^8 + k_7x^7 + k_6x^6 + k_5x^5 + k_4x^4 + k_3x^3 + k_2x^2 + k_1x + k_0$$

k^{14} dışında tüm değerler sıfır olduğu için, $s(x) = x^{14}$ olur. Böylelikle lagrange interpolasyonu ile S-kutusunun ifadesini $GF(2^4)$ sonlu cisminde elde etmiş oluruz.

Lagrange Interpolasyonu ile cebirsel ifadenin elde edilmesi konusunda yapılan çalışmanın ayrıntıları (5.5) bölümünde verilmiştir.

BÖLÜM 5

5 YAPILAN ÇALIŞMALAR

5.1 Beş Döngülü SPN Diferansiyel Kriptanaliz

Dört döngülü SPN üzerinde diferansiyel kriptanaliz işleminin gerçekleştirilmesine, (X_1, Y_1) ve (X_2, Y_2) olmak üzere iki açık ve şifreli metin çifti seçerek başlanır. Açık metin ve şifreli metin çifti değerleri aşağıda gösterilmektedir.

X_1 = 0011 0110 0101 0001 , Y_1 = 1110 1011 0100 0100

X_2 = 0011 1101 0101 0001 , Y_2 = 1110 1011 0101 1000

Açık metin şifreli metin çiftleri ilk olarak uygulanan karakteristiğe göre filtrelenir. Son döngünün karakteristiğinde birinci ve üçüncü dörtlüler (0000) olduğu için,

$$Y_1(1)\ (1110) = Y_2(1)\ (1110) \tag{5.1}$$

$$Y_1(3)\ (1011) = Y_2(3)\ (1011) \tag{5.2}$$

eşitliğini sağlayan çiftler doğru çiftlerdir.

Anahtarın kaç bitinin elde edileceğini, kullanılan karakteristik belirler. Bu örnekte kullanılan diferansiyel karakteristik 16 bitlik alt anahtarın 8 bitlik kısmını elde etmeyi sağlar.

Beş döngülü bir SPN şifreleyicisinin diferansiyel kriptanalizi için 4 döngülük bir diferansiyel karakteristik yeterli olacaktır. Diferansiyel kriptanaliz saldırısı, son döngü alt anahtarından bitler elde etmeyi amaçlar.

Şifreli metin, son döngünün S-kutusu çıkışını hesaplamak için tüm olası anahtarlarla XOR işlemine tabi tutulur ve ΔV aşağıda gösterildiği gibi elde edilir.

$$V_1(3) = L_1 \oplus Y_1(3) \tag{5.3}$$

$$L_1 = 1011$$

$$V_1(3) = L_1 \oplus Y_1(3)$$

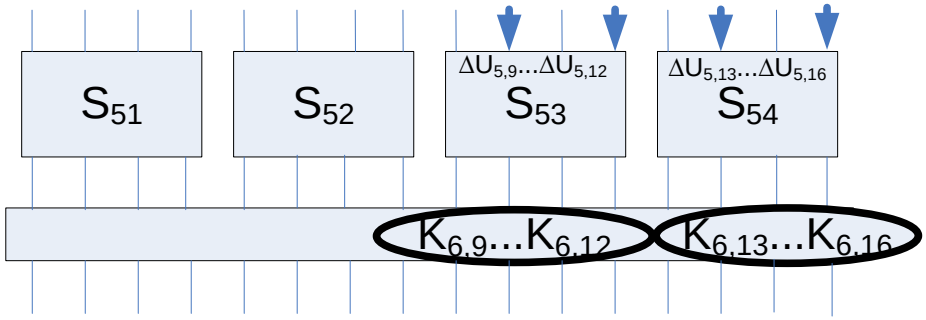
$$V_1(3) = 1011 \oplus 0100$$

$$V_1(3) = 1111$$

$$V_2(3) = L_1 \oplus Y_2(2) \tag{5.4}$$

$$V_2(3) = 1011 \oplus 1010$$

$$V_2(3) = 0001$$



Şekil 5.1. 4 Döngülü SPN Son Döngü

S-kutusunun çıkış değeri ΔV ' den giriş değeri ΔU ' yu elde etmek için, ΔV değerinin S-kutusu fonksiyonuna göre tersi alınır. Böylelikle bilinen Y değerini üreten X değeri aşağıda gösterildiği gibi bulunur:

$$U_1(3) = S^{-1}(V_1(3)) \tag{5.5}$$

$$U_1(3) = S^{-1}(1111)$$

$$U_1(3) = 0101$$

$$U_2(3) = S^{-1}(V_2(3)) \tag{5.6}$$

$$U_2(3) = S^{-1}(1110)$$

$$U_2(3) = 0000$$

Elde edilen U_1 ve U_2 değerleri XOR' lanarak ΔU değeri elde edilir.

$$U_{12}(3) = U_1(3) \oplus U_2(3) \quad (\Delta U \text{ değeri}) \tag{5.7}$$

$$U_{12}(3) = 0101 \oplus 0000$$

$$U_{12}(3) = 0101$$

Aynı işlem diferansiyel karakteristiğin sıfırdan farklı diğer 4 bitlik bloğu için tekrarlanır.

$$L_2 = 1000$$

$$V_1(4) = L_2 \oplus Y_1(4)$$

$$V_1(4) = 1000 \oplus 0100$$

$$V_1(4) = 1100$$

$$U_1(4) = S^{-1}(V_1(4))$$

$$U_1(4) = S^{-1}(1100)$$

$$U_1(4) = 1011$$

$$V_2(4) = L_2 \oplus Y_2(4)$$

$$V_2(4) = 1000 \oplus 1000$$

$$V_2(4) = 0000$$

$$U_2(4) = S^{-1}(V_2(4))$$

$$U_2(4) = S^{-1}(0000)$$

$$U_2(4) = 1110$$

$$U_{12}(4) = U_1(4) \oplus U_2(4)$$

$$U_{12}(4) = 1011 \oplus 1110$$

$$U_{12}(4) = 0101$$

0101 = 0101 0101 = 0101

Anahtar = 10111000, Gerçeklenme Sayı = 1

5.1.1 Anahtar Bulmaya Çalışma

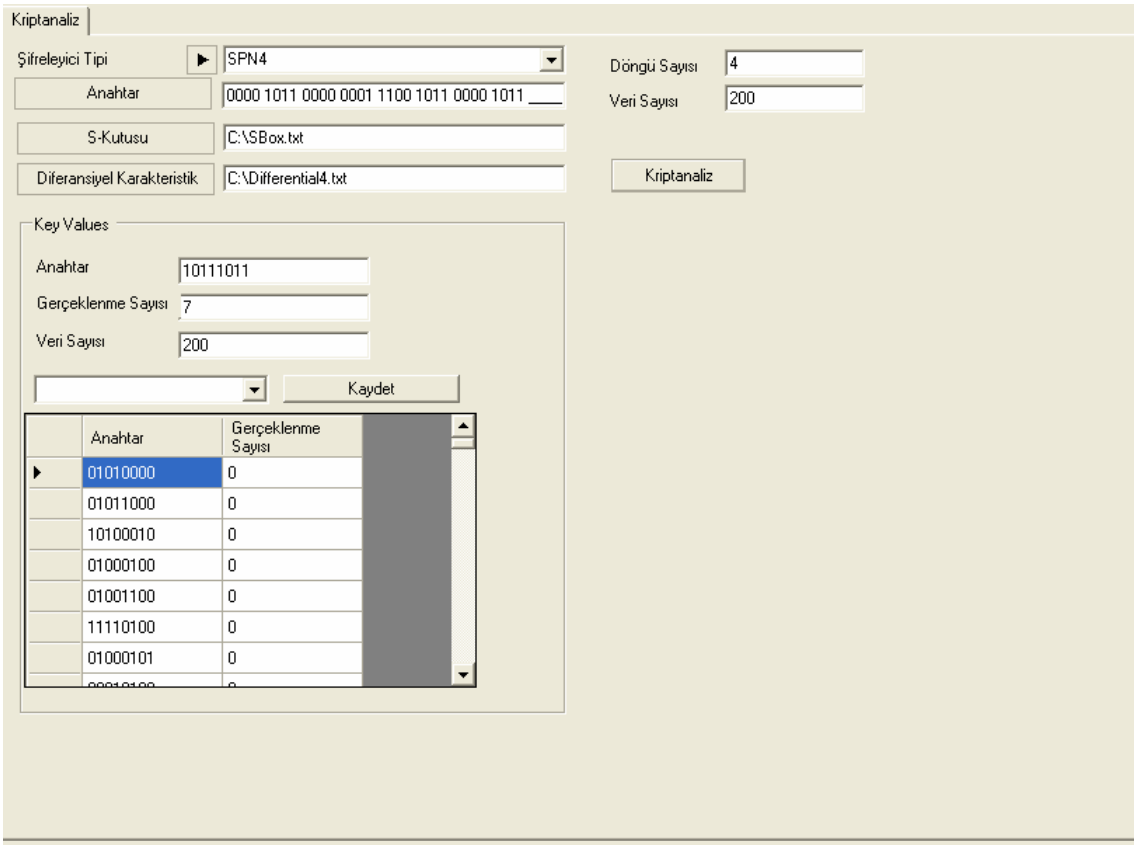
n döngülü bir şifreleyici için, $n-1$ döngü fark karekteristiği yeterli büyüklükte bir olasılık ile elde edilebilirse, son alt anahtardan bitler elde ederek şifreleyiciye saldırmak mümkün olabilir. İncelenen örnek için K_6 alt anahtarından bitler elde etmek mümkün olabilecektir. Kriptanaliz süreci şifreleyicinin son döngüsünü kısmi olarak deşifrelemeyi ve son döngünün girişlerini, doğru çift oluşup oluşmadığını belirlemek için incelemeyi içerir. Son döngünün deşifrelenmesi şifreli metnin hedef kısmi alt anahtar bitleri ile XOR' lanması ve hedef alt anahtar bitlerinin olası tüm değerleri denenerek verilerin S-kutuları boyunca geriye ilerletilmesini içerir. Bu süreç içerisinde tüm alt anahtarlar denenir ve hangisinin son aşamaya daha sık doğru giriş verdiği belirlenir. En sık doğru giriş veren büyük ihtimalle alt anahtardır.

Örnek şifreleyicimiz üzerindeki saldırı düşünüldüğünde, diferansiyel karakteristik son döngüdeki S_{53} ve S_{54} S-kutularının girişlerini etkiler. Her bir şifreli metin çiftleri için, $[K_{6,9} \ldots K_{6,12}, K_{6,13} \ldots K_{6,16}]$ anahtar bitleri için tüm 256 değer denenir. Her bir kısmi alt anahtar değeri için, son döngüye kısmi deşifreleme tarafından belirlenen giriş farkı ΔU_4'e eşitse sayaç bir artırılır. $[\Delta U_{5,9} \ldots \Delta U_{5,12}, \Delta U_{5,13} \ldots \Delta U_{5,16}]$ değerleri veriler kısmi alt anahtarlar ve S-kutuları boyunca hareket ettirelerek belirlenir. Tüm kısmi alt anahtar değerleri için, sayaç değeri doğru çiftlerle tutarlı farkların oluşma sayısını gösterir (kısmi alt anahtarın doğru olması varsayımı ile). Doğru çiftin yüksek olasılıklı oluşmasını araştırdığımız için, sayaç değeri en büyük olan doğru değer olarak seçilir.

Tüm şifreli metin çiftleri için kısmi deşifrelemeyi çalıştırmak gerekli değildir. Doğru çiftler için diferansiyel karakteristik oluştuğunda, son döngünün giriş farkı sadece 2 S-kutusunu etkilediği için, S_{51} ve S_{52} S-kutularına karşılık gelen şifreli metin

bit farkları sıfır olmalıdır. Birçok yanlış çift şifreli metin farklarının uygun alt blokları sıfır olmayan şifreli metin çiftleri elenerek filtrelenir. Bu durumlarda, şifreli metin çifti doğru çifte karşılık gelmediğinden, $[\Delta U_{5,1} \ldots \Delta U_{5,4}, \Delta U_{5,5} \ldots \Delta U_{5,8}]$ değerlerini incelemeye gerek yoktur.

5.1.2 Diferansiyel Kriptanaliz Uygulaması



Şekil 5.2. Diferansiyel Kriptanaliz Uygulaması

- Kriptanalizi yapılacak şifreleyici tipi seçilir.
- Bitleri elde edilmeye çalışılacak anahtar dizesi girilir.
- S-Kutusu değerlerinin bulunduğu dosya seçilir.

```
public void DiferansiyelKriptanaliz(int n)
{
    foreach (Difference d in DifferencePair)
    {
        BitArray deger = new BitArray(new int[] { 0, 1, 1, 0 });
```

```
        if (  BitConversion.ConvertToBaseTenNotation(Y(d.Y1, 1)) ==
              BitConversion.ConvertToBaseTenNotation(Y(d.Y2, 1)) &&
              BitConversion.ConvertToBaseTenNotation(Y(d.Y1, 3)) ==
              BitConversion.ConvertToBaseTenNotation(Y(d.Y2, 3)))
        {     for (int L1 = 0; L1 < 16; L1++)
              {
                for (int L2 = 0; L2 < 16; L2++)
                {
                  BitArray v42 = ((BitArray)hBitConversion[L1]).Xor(Y(d.Y1, 2));
                  BitArray v44 = ((BitArray)hBitConversion[L2]).Xor(Y(d.Y1, 4));
                  BitArray u42 = S(v42)
                  BitArray u44 = S(v44);
                  BitArray v42y = ((BitArray)hBitConversion[L1]).Xor(Y(d.Y2, 2));
                  BitArray v44y = ((BitArray)hBitConversion[L2]).Xor(Y(d.Y2, 4));
                  BitArray u42y = S(v42y);
                  BitArray u44y = S(v44y);
                  BitArray u42y1 = u42.Xor(u42y);
                  BitArray u44y1 = u44.Xor(u44y);
                  if   (BitConversion.ConvertToBaseTenNotation(
                              u42y1.Xor(deger).Or(u44y1.Xor(deger))) == 0)
                  {
                      string key = keyL1L2(L1, L2);
                      hCount[key] = ((int)hCount[key]) + 1;
                  }
                }
              }
         }
    }
    int max = 0;    string maxKey = "";
    for (int L1 = 0; L1 < 16; L1++) {
              for (int L2 = 0; L2 < 16; L2++) {
                       string key = keyL1L2(L1, L2);
                      if ((int)hCount[key] > max)
                      {
                              max = (int)hCount[key];
                               maxKey = key;
                      }
              }
    }
    Console.WriteLine(maxKey);
}
```

Algoritma 5.1. Diferansiyel Kriptanaliz Uygulaması Örnek Kaynak Kodu

– Diferansiyel karakteristiğin yer aldığı dosya seçilir.

– SPN algoritmasının döngü sayısı seçilir.

– Örnek veri sayısı girilir.

- Kriptanaliz düğmesine basıldığında kriptanaliz süreci başlar.
- Elde edilen anahtar dizesi ve gerçeklenme değeri ekranda gösterilir.

5.1.3 SPN Diferansiyel Kriptanaliz Deneysel Sonuçların Yorumlanması

Yapılan analizlerden 4 Döngülü SPN için elde edilen sonuçlar aşağıda yorumlanmıştır:

a. Kullanılan diferansiyel karakteristiğin olasılığı 27/1024 = 0,02637' dir.

b. 200 açık metin ve şifreli metin çifti için gerçeklenme sayısı (hit count) 7' dir.

c. Anahtarın elde edilme olasılığı yüksek bir olasılık olan 7/200 = 0,035' dir.

d. Örnek veri sayısı artırıldığında anahtarın elde edilmesinde beklenen olasılığa daha yakın olasılık değerleri ile karşılaşılmaktadır. Örneğin, 500 veri için gerçeklenme sayısı 11 değeri, beklenen olasılığa daha yakın olan 11/500 = 0,022 olasılık değerini doğurur. 1000 veri için, 21 olan gerçeklenme sayısı 21/1000=0,021 olasılık değerini oluşturur. Buradan da görüldüğü gibi veri sayısı belirli bir değerin üzerine çıktığında anahtarın elde edilme olasılığı azalmaktadır. 300 veri için, 7 gerçeklenme sayısı daha yakın bir olasılık olan 7/300 = 0,023 olasılık değerini oluşturur.

5 Döngülü SPN için;

a. Diferansiyel karakteristiğin olasılığı $(27/1024)*(4/16)^2$ = 27/16384 = 0.00165'dir.

   ΔU_4 = 0000 0110 0000 0110

   $\Delta X = 6 \rightarrow \Delta Y = 3$ değerine 4/16 olasılık değeri ile karşılık gelir.

   $\Delta X = 6 \rightarrow \Delta Y = 3$ olasılık 4/16

   ΔV_4 = 0000 0011 0000 0011

   ΔU_5 = 0000 0000 0101 0101

b. Toplam Olasılık = $(27/1024)*(4/16)^2$ = 27/16384

c. Örnek Veri Sayısı = 2000, Gerçeklenme Sayısı = 4

d. İzlenen Olasılık = 4/2000 = 0,002

e. Beklenen Olasılık = 27/16384 = 0,00165

6 Döngülü SPN için;

a. 4 bitlik anahtarı veren diferansiyel karakteristik olasılığı $\frac{27}{16384} \times \frac{4}{16} \times \frac{4}{16}$ ’dır.

ΔU_5 = 0000 0000 0101 0101

$\Delta X = 5 \rightarrow \Delta Y = 1$ olasılık: 4/16

$\Delta X = 5 \rightarrow \Delta Y = 1$ olasılık: 4/16

ΔV_5 = 0000 0000 0001 0001

ΔU_6 = 0000 0000 0000 0011 (Anahtarın 4 bitini verir.)

b. Toplam Olasılık = $\frac{27}{16384} \times \frac{4}{16} \times \frac{4}{16}$.

c. 6 döngülük SPN şifreleme algoritmasına yapılan diferansiyel kriptanaliz saldırısı, anahtarı elde etmede başarısız olmuştur.

d. Olasılığı $\frac{27}{16384} \times \frac{4}{16} \times \frac{4}{16}$ olan başka bir diferansiyel deneyelim.

ΔU_5 = 0000 0000 0101 0101

$\Delta X = 5 \rightarrow \Delta Y = 10$ olasılık: 4/16

$\Delta X = 5 \rightarrow \Delta Y = 10$ olasılık: 4/16

ΔV_5 = 0000 0000 1010 1010

ΔU_6 = 0011 0000 0011 0000 (Anahtarın 4 bitini verir.)

e. Toplam Olasılık = $\frac{27}{16384} \times \frac{4}{16} \times \frac{4}{16}$.

f. Yine doğru anahtar elde edilemedi.

Yaptığımız analizlerden gördük ki; diferansiyel kriptanaliz SPN şifreleyicisinin anahtarlarını geniş anahtar aramadan daha az zaman karmaşıklığında elde edebilmektedir. Dolayısıyla, blok şifreleyicilerin gücünün ölçülmesinde önemli bir tekniktir.

Bu çalışmada diferansiyel kriptanalizi dört, beş ve altı döngülük SPN şifreleyisi üzerinde denedik. Dört döngülük ve beş döngülük SPN şifreleyicisinde hedef anahtarın bir kısmını elde ederken, altı döngülük SPN şifreleyicisinde hedef anahtarı elde edemedik.

Diferansiyel kriptanaliz n döngü SPN için $n-1$ döngü diferansiyel karakteristik gerektirmektedir. Dört döngülü ve beş döngülük SPN'e saldırırken kullanmış olduğumuz diferansiyel karakteristik hedef anahtarın 8 bitini sağlamaktadır. Altı döngülü SPN için kullanılan diferansiyel karakteristik ise anahtarın 4 bitini vermektedir.

Yapılan analizler sonucunda altı döngülük SPN şifreleyicinin diferansiyel kriptanalize direnç gösterdiği görülmüştür. Diferansiyel kriptanalizin, kesilmiş diferansiyel kriptanaliz, ilişkili anahtar diferansiyel kriptanaliz gibi gelişmiş yöntemleri kullanılarak SPN'nin daha çok sayıda döngüsüne saldırılabilir.

5.2 AES Diferansiyel Kriptanaliz Uygulaması

a. 613300 çiftte 152 kez diferansiyel karakteristiğe uygun veri elde edildi. 152/ 613300 ≈ 1/4035 dir.

b. Diferansiyel karakteristik olasılığı = 1/4035.

c. 128 bitlik AES şifreleyicisi

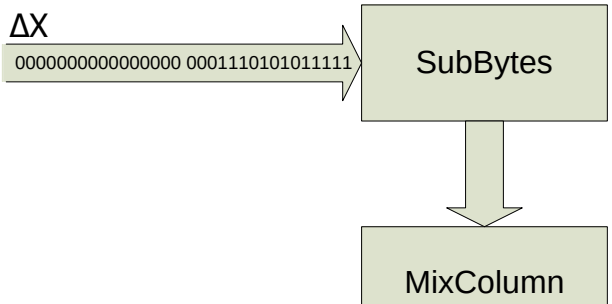
d. m : Örnek Veri Sayısı, n : Alt Blok Uzunlugu

e. Karmaşıklık = $m \times 2^n$ 'dir.

f. $m = 4096$, $n = 32$ için, Karmaşıklık = $4096 \times 2^{32} = 2^{12} \times 2^{32} = 2^{44}$.

g. Teorik olarak geniş anahtar aramadan daha hızlı olmasına rağmen pratikte bu hesaplamanın sonucunu göremedik.

,



Şekil 5.3. AES 1 Döngülük Diferansiyel Karakteristik

Bu çalışmada, AES şifreleyicisinin iki döngüsüne 1 döngülük diferansiyel karakteristik kullanılarak diferansiyel kriptanaliz ile saldırdık. Yoğun anahtar aramadan (exhaustive search) daha iyi olasılıkta bir diferansiyel karakteristik ile saldırmamıza rağmen uygulamada başarıya ulaşamadık.

5.3 DES Kesilmiş (Truncated) Diferansiyel Saldırı

5.3.1 Açık Metinlerin Belirlenmesi

Gösterim

$K_{i,j}$: i. döngüde S-kutusu j'nin 6 sekizlilik anahtarı,

P : DES döngü fonksiyonundaki 32 bitlik permütasyon,

a_i : 4 bitlik sayı,

A_i : 64 bitlik verinin 32 bitlik sol parçası,

r_k : Rastgele olarak seçilmiş 4 bitlik bir sayı,

P_R : Rastgele seçilmiş 32 bitlik dize olsun.

b_j : 4 bitlik sayı.

B_j : 64 bitlik verinin 32 bitlik sol parçası.

| : 4 bit birleştirme, || : 32 bit birleştirme olsun.

Kesilmiş diferansiyel saldırıların belirlenmesinde izlenen yöntem aşağıda ayrıntılı olarak anlatılacaktır:

– 4 açık metin seçiniz.

for $i = 0 \ldots 3$

$a_i = i$

$A_i = P(a_i \mid r_0 \mid r_1 \mid \ldots \mid r_5 \mid r_6)$

$P_i = A_i \parallel P_R$

– 4 açık metin daha seçiniz.

for $j = 0 \ldots 3$

$B_i = P(b_i \mid r_0 \mid r_1 \mid \ldots \mid r_5 \mid r_6)$

$P_{1,j} = B_i \parallel P_R \oplus \Phi_{1,1}$

$\Phi_{1,1} = 20000000_x \;\; b_0 = 0_x \;\; b_1 = 4_x \;\; b_2 = 8_x \;\; b_3 = c_x$

5.3.2 Açık Metin Çiftleri Oluşturma

– P_i ve $P_{1,j}$ ile her bir 4 açık metin birleştirilerek, $P(h_x \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid 0) \parallel \Phi_{1,1}$ $h_x = (0 \ldots f_x) = a_0 \oplus b_0$ *(h_x: 4 bitlik sayı)* farkına sahip açık metin çiftleri elde edilir.

– Elde edilen bu 8 çiftten biri seçilen karakteristiğe göre doğru çift olur.

– Daha fazla doğru çift elde etmek için 4 açık metin daha seçelim.

for $j = 0 \ldots 3$

$P_{2,j} = B_j \parallel P_R \oplus \Phi_{1,2}$

$\Phi_{1,2} = 4000000_x$

for $j = 0 \ldots 3$

$$P_{3,j} = A_j \,\|\, P_R \oplus \Phi_{1,1} \oplus \Phi_{1,2}$$

– $P_{2,j}$ ile $P_{3,j}$ birleştirilerek, $P(h_x \,|\, 0 \,|\, 0 \,|\, 0 \,|\, 0 \,|\, 0) \| \Phi_{1,1}$ farkını sağlayan bir doğru çift elde edilir.

– $P_{1,j}$ ile $P_{2,j}$ ve P_i ile $P_{3,j}$ birleştirilerek $h_x = (0 \ldots f_x)$ için $P(h_x \,|\, 0 \,|\, 0 \,|\, 0 \,|\, 0 \,|\, 0) \| \Phi_{1,2}$ farkına sahip açık metin çiftleri elde edilir.

– $P_{1,j}$ ve $P_{2,j}$ ile, P_i ve $P_{3,j}$ birleştirilerek doğru 2 çift daha elde edilir.

5.3.3 Diferansiyel Saldırı

– Birinci döngüde S-kutusu 1'de $K_{1,1}$ anahtarının her bir $k_{1,1}$ değeri için;

  - $k_{1,*}$, $k_{1,1}$ ve rastgele seçilmiş 42 bitin birleştirilmesinden elde edilmiş 48 bitlik anahtar değeri olsun.
  - $c_0 = F(k_{1,*}, P_R)$, $c_1 = F(k_{1,*}, P_R \oplus \Phi_{1,1})$ ifadeleri hesaplanır.
  - y (4 bitlik bir değer)' nin herhangi bir değeri için, $c_0 \oplus c_1 = P(y \,|\, 0 \,|\, 0 \,|\, \ldots \,|\, 0)$ ifadesi hesaplanır.
  - $c_0 \oplus c_1 = A_i \oplus B_j$ ifadesini sağlayan P_i ve $P_{1,j}$ açık metinleri bulunur.
  - P_i ve $P_{1,j}$ karakteristiğe göre doğru çifttir.
  - $c_2 = F(k_{1,*}, P_R \oplus \Phi_{1,2}), c_3 = F(k_{1,*}, P_R \oplus \Phi_{1,1} \oplus \Phi_{1,2})$ ifadeleri hesaplanır.
  - $c_2 \oplus c_3 = A_i \oplus B_j$ ifadesini sağlayan $P_{2,j}$ ve $P_{3,j}$ açık metinleri bulunur. $P_{2,j}$ ve $P_{3,j}$ karakteristiğe göre doğru çifttir.
  - Yukarıdaki işlem aşamaları uygulanarak, P_i ve $P_{2,j}$, $P_{1,j}$ ve $P_{3,j}$ doğru çiftleri diğer karakteristik için bulunur.

```
private void AcikMetinBelirle(      string PR,string r0,string r1,string r2,
                                    string r3,string r4, string r5,string r6 )
{
string[] A = new string[4];
string[] B = new string[4];
string[] C = new string[4];
string[] D = new string[4];
int n = (int)Math.Pow(2, 16);
DES des = new DES();
for (int i = 0; i < 4; i++)
{
   string a = BitConversion.ConvertToBaseTwoNotation(i, 4);
   A[i] = des.P(a + r0 + r1 + r2 + r3 + r4 + r5 + r6) + PR;
}
string[] b = new string[] { "0000", "0100", "1000", "1100" };
string PRQ = XOR(PR, "0010" + "".PadLeft(28, '0')); //20000000x
for (int i = 0; i < 4; i++)
{
   B[i] = des.P(b[i] + r0 + r1 + r2 + r3 + r4 + r5 + r6) + PRQ;
}
string PRQ2 = XOR(PR, "0100" + "".PadLeft(28, '0'));//40000000x
for (int i = 0; i < 4; i++)
{
   C[i] = des.P(b[i] + r0 + r1 + r2 + r3 + r4 + r5 + r6) + PRQ2;
}
string PRQ3 = XOR(PRQ2, "0010" + "".PadLeft(28, '0'));
for (int i = 0; i < 4; i++)
{
   string a = BitConversion.ConvertToBaseTwoNotation(i, 4);
   D[i] = des.P(a + r0 + r1 + r2 + r3 + r4 + r5 + r6) + PRQ3;
}
dogruCiftleriBelirle(A, B);
dogruCiftleriBelirle(C, D);
dogruCiftleriBelirle(A, C);
dogruCiftleriBelirle(B, D);
DataTable dtRightPair = dogruCifler();
Console.WriteLine(findKey(dtRightPair));
}
```

Algoritma 5.2. Kesilmiş Diferansiyel Kriptanaliz Uygulaması Örnek Kaynak Kodu

- Elde edilen bu 4 doğru çift diferansiyel saldırıda kullanılır.
- Öncelikle son döngüdeki S-kutusu 1'e saldırılır.
- $K_{6,1}$'in $k_{6,1}$ anahtar değeri 4 çift tarafından da sağlanırsa, son döngüdeki S-kutusu 7 için diferansiyel saldırı gerçekleştirilir.

– $K_{6,7}$'nin $k_{6,7}$ anahtar değeri 4 çift tarafından da sağlanırsa, $k_{6,1}$ ve $k_{6,7}$, $K_{6,7}$'nin anahtar değerleri olarak alınırken $k_{1,1}$'de $K_{1,1}$'in anahtar değeri olarak kabul edilir.

5.3.3.1 DES Kesilmiş Diferansiyel Saldırı Sonuçları

– Yukarıda anlatılan saldırı anahtarın 18 bitini yüksek bir olasılıkla elde eder.

– Doğru çiftlerin bulunması aşamasında F fonksiyonunun tamamının hesaplanmasına gerek yoktur. Sadece S-kutularının yer aldığı hesaplamaların yapılması yeterlidir.

– $K_{1,1}$'in her bir değeri için 4 S-kutusu hesaplaması yapılmıştır.

– $K_{6,1}$ için 4 doğru çiftteki, 8 şifreli metnin her biri için bir hesaplama olmak üzere, 8 S-kutusu hesaplaması yapılmıştır.

– $K_{6,7}$ için arama, $K_{6,1}$ dört çiftin tümü tarafından da önerildiği durumda yapılmıştır.

– Toplam karmaşıklık 215 S-kutusu hesaplamasıdır.

– Yaklaşık olarak 500 6 döngülük DES hesaplamasına eşittir.

– Saldırıda kullanılan kesilmiş diferansiyelin olasılığı birdir.

– Farklı karakteristikleri sağlayan açık metinlerle gerçekleştirilecek benzer saldırılarla, anahtarın daha fazla bitini elde etmek mümkün olabilecektir.

5.4 Mini-AES İmkansız Diferansiyel Saldırı

5.4.1 İmkansız Olayın Belirlenmesi

NibbleSub

– P ve P' sadece en sol nibble'ları aktif olan iki açık metin bloğu olsun.

P = 0100 0011 1110 1001 , P' = 1110 0011 1110 1001

– NibbleSub dönüşümünden sonra,

P = 1110 1111 0110 0111 , P' = 0100 1111 0110 0111

olur.

– İki çıkış değeri sadece aynı nibble'larda farklılaşır. Yani ne aktif nibble sayısı, ne de yeri değişir.

ShiftRow

– NibbleSub dönüşümünün çıktısı olan P ve P' değerlerine ShiftRow dönüşümü uygulanır.

P = 1110 1111 0110 0111 , P' = 0100 1111 0110 0111

– ShiftRow dönüşümünden sonra,

P = 1110 0111 0110 1111 , P' = 0100 0111 0110 1111

olur.

– ShiftRow dönüşümü aktif nibble'ların sayısını değiştirmez. Sadece iki nibble yer değiştirir.

MixColumn

– ShiftRow dönüşümünün çıktısı olan P ve P' değerlerine MixColumn dönüşümü uygulanır.

P = 1110 0111 0110 1111 , P' = 0100 0111 0110 1111

– MixColumn dönüşümünden sonra,

P = 1111 0110 0111 1110 , P' = 0010 0001 0111 1110

olur.

– Giriş değerlerindeki bir aktif nibble, MixColumn dönüşümü ile aynı kolonda iki aktif nibble'a dönüşür.

RoundKeyAddition

– MixColumn dönüşümünün çıktısı olan P ve P' değerlerine MixColumn dönüşümü uygulanır.

P = 1111 0110 0111 1110 , P' = 0010 0001 0111 1110

– RoundKeyAddition dönüşümünden sonra,

$P = P \oplus K_i = P_0 P_1 P_2 P_3$, $P' = P' \oplus K_i = P_0' P_1' P_2' P_3'$

olur.

– Aktif nibble sayısı üzerinde herhangi bir etkisi olmaz.

5.4.2 4 Döngülük İmkansız Diferansiyelin Belirlenmesi

– P ve P' sadece en sol nibble'ları aktif olan iki açık metin bloğu olsun. Aktif nibble değerleri birbirinden farklı 4 bitlik gruplardır. P ve P' açık metin çifti için aktif grup, en sol dört bitlik grup olup kırmızı renkte gösterilmiştir.

P = 0101 1111 0110 1100

P' = 0100 1111 0110 1100

– Döngü 0 ifadesi, Mini-AES şifreleme algoritmasındaki birinci döngüden önceki anahtar ekleme işlemi için kullanılmıştır. Mini-AES şifreleyicide anahtar ekleme işlemi, açık metin ile döngü anahtarının XOR işlemine tabi tutulmasından ibaret olup aşağıda gösterilmiştir:

- K_0 = 0101 1010 1100 001
- RoundKeyAddition:

$$P = P \oplus K_0 = 0101\ 1111\ 0110\ 1100 \oplus 0101\ 1010\ 1100\ 001$$

$$= 0000\ 0101\ 1010\ 1111$$

$$P' = P' \oplus K_0 = 0100\ 1111\ 0110\ 1100 \oplus 0101\ 1010\ 1100\ 001$$

$$= 0001\ 0101\ 1010\ 1111$$

- Yukarıdaki değerlerden de görüldüğü üzere XOR işleminin yapısı gereği aktif nibble sayısı değişmemiştir.

– Döngü 1: Mini-AES şifreleyicisinde birinci döngü sırasıyla doğrusal olmayan NibbleSub, yer değiştirme işlemi yapan, doğrusal bir işlem olan ShiftRow, karıştırma işlemi yapan MixColumn ve yine doğrusal anahtar ekleme işlemi olan RoundKeyAdditon işlemlerinden oluşur.

- NibbleSub

P = 0000 0101 1010 1111, P' = 0001 0101 1010 1111

P = *NibbleSub*(0000) *NibbleSub*(0101)

NibbleSub(1010) *NibbleSub*(1111)

= 1110 1111 0110 0111

P' = *NibbleSub*(1110) *NibbleSub*(1111)

NibbleSub(0110) *NibbleSub*(0111)

= 0100 1111 0110 0111

- Elde edilen değerlerden de görüldüğü gibi, NibbleSub dönüşümü ne aktif nibble sayısını, ne de yerini değiştirmiştir.
- ShiftRow

$$P = (p_0, p_1, p_2, p_3) \quad ShiftRow = (p_0, p_3, p_2, p_1)$$

$$P = (p_0 = 1110, p_1 = 1111, p_2 = 0110, p_3 = 0111)$$

$$ShiftRow = (p_0 = 1110, p_3 = 0111, p_2 = 0110, p_1 = 1111)$$

$$P = 1110\ 0111\ 0110\ 1111$$

$$P' = (p_0 = 0100, p_1 = 1111, p_2 = 0110, p_3 = 0111)$$

$$ShiftRow = (p_0 = 0100, p_3 = 0111, p_2 = 0110, p_1 = 1111)$$

$$P' = 0100\ 0111\ 0110\ 1111$$

- ShiftRow dönüşümünün aktif nibble'lar üzerinde bir etkisi olmamıştır. Gerçekleştirilen ShiftRow dönüşümü sonucunda, iki pasif nibble yer değiştirmiştir. Aktif nibble'ın birinci satırda olması ve ShiftRow dönüşümü sonucunda, matrissel gösterimde birinci satırda olan elemanların yerlerini koruması nedeniyle, ne aktif nibble sayısı ne de yeri değişmemiştir.
- MixColumn

$$p = \begin{bmatrix} 1110 & 0110 \\ 0111 & 1111 \end{bmatrix}$$

1111	0111
0110	1110

$$\begin{bmatrix} p_0 \\ p_1 \end{bmatrix} = \begin{bmatrix} 0011 & 0010 \\ 0010 & 0011 \end{bmatrix} \begin{bmatrix} 1110 \\ 0111 \end{bmatrix}$$

$$\begin{aligned} p_0 &= (0011) \bullet (1110) \oplus (0010) \bullet (0111) \\ &= (x+1)(x^3 + x^2 + x) + x(x^2 + x + 1) \\ &= x^4 + x^3 + x^2 + x^3 + x^2 + x + x^3 + x^2 + x \\ &= (x+1) + x^3 + x^2 \\ &= x^3 + x^2 + x + 1 \\ &= (1111) \end{aligned}$$

$$p_1 = (0010) \bullet (1110) \oplus (0011) \bullet (0111) = 0110$$

$$p_2 = (0011) \bullet (0110) \oplus (0010) \bullet (1111) = 0111$$

$$p_3 = (0010) \bullet (0110) \oplus (0011) \bullet (1111) = 1110$$

$$P = 1111\ 0110\ 0111\ 1110$$

$p_0 = (0010) \bullet (0100) \oplus (0011) \bullet (0111) = 0010$

$p_1 = (0011) \bullet (0100) \oplus (0010) \bullet (0111) = 0001$

$p_2 = (0010) \bullet (0110) \oplus (0011) \bullet (1111) = 0111$

$p_3 = (0011) \bullet (0110) \oplus (0010) \bullet (1111) = 1110$

0010	0111
0001	1110

P' = 0010 0001 0111 1110

- Elde edilen değerlerden de görüldüğü üzere, MixColumn dönüşümü aktif nibble sayısını aynı kolonda ikiye çıkarmıştır. MixColumn işleminin yapısındaki sütun karıştırma işlemi, aktif nibble sayısının artmasına neden olmuştur. Birinci sütündaki aktif nibble'ın aynı sütunun diğer satırlarının hesabına katılması sonucu, aktif nibble'ın diğer sütun ve satırlara yayılması sağlanmıştır.
- RoundKeyAddition: K_1 = 1100 0011 0101 1010

$P = P \oplus K_1$ = 1111 0110 0111 1110 ⊕ 1100 0011 0101 1010

= 0011 0101 0010 0100

$P' = P' \oplus K_1$ = 0010 0001 0111 1110 ⊕ 1100 0011 0101 1010

= 1110 0010 0010 0100

0011	0010
0101	0100

1110	0010
0010	0100

- Hala aynı kolonda iki aktif nibble var.

– Döngü 2:

- NibbleSub

P = 0011 0101 0010 0100, P' = 1110 0010 0010 0100

P = *NibbleSub*(0011) *NibbleSub*(0101)

NibbleSub(0010) *NibbleSub*(0100)

$=$ 0001 1111 1101 0010

P' $=$ *NibbleSub*(1110) *NibbleSub*(0010)

NibbleSub(0010) *NibbleSub*(0100)

$=$ 0000 1101 1101 0010

- Aktif nibble sayısı iki olarak aynı konumda kalmıştır.
- ShiftRow dönüşümü sonucunda birinci satır sabit kalırken, ikinci satır bir sütun döndürülür.

P = 0001 0010 1101 1111, P' = 0000 0010 1101 1101

0001	1101
0010	1111

0000	1101
0010	1101

- Aktif nibble sayısı aynı fakat yerleri değişmiştir.
- Bu işlemin sonucu olarak herbir kolonda bir aktif nibble oluşmuştur.
- MixColumn

$$p_0 = (0010) \bullet (0001) \oplus (0011) \bullet (0010) = 0100$$

$$p_1 = (0011) \bullet (0001) \oplus (0010) \bullet (0010) = 0010$$

$$p_2 = (0010) \bullet (1101) \oplus (0011) \bullet (1111) = 1001$$

$$p_3 = (0011) \bullet (1101) \oplus (0010) \bullet (1111) = 1011$$

P = 0111 0100 1001 1011

$$p_0 = (0010) \bullet (0000) \oplus (0011) \bullet (0010) = 0100$$

$$p_1 = (0011) \bullet (0000) \oplus (0010) \bullet (0010) = 0110$$

$$p_2 = (0010) \bullet (1101) \oplus (0011) \bullet (1101) = 1101$$

$$p_3 = (0011) \bullet (1101) \oplus (0010) \bullet (1101) = 1101$$

P' = 0100 0110 1101 1101

- Elde edilen değerlerden de görüldüğü üzere, MixColumn dönüşümünün sonunda tüm nibble'lar aktif duruma gelmiştir.
- RoundKeyAddition: K_2 = 1111 0010 1011 1100

  P = 1000 0110 0010 0111

  P' = 1011 0100 0110 0001
- RounKeyAddition işleminin sonucunda da tüm nibble'lar aktifliğini korur.

İlk nibble dışında eşit olan iki açık metin üzerinde çalıştık. Birinci döngüden sonra ilk kolonda iki aktif nibble'ı olan iki çıkış değeri elde ettik. İkinci döngü sonunda tüm nibble'ları aktif olan iki çıkış değerine sahip olduk. Şimdi de şifreleyicinin diğer ucuna, dördüncü, yani 4 döngülük Mini-AES şifreleyicisinin son döngüsünün çıkış değerlerine odaklanalım.

Son İki Döngünün Tersten Taranması

– Döngü 4

- C ve C' her bir satır ve kolonunda tek bir nibble'ları eşit olan iki şifreli metin olsun.

  C = 0100 0011 1001 0101

  C' = 1110 0011 1001 1110

0100	1001
0011	0101

1110	1001
0011	1110

- Şifreleme yaparken kullanılanan anahtar ekleme işleminin elenmesi, Ters KeyAddition işlemi ile sağlanır. KeyAddition işlemi bir XOR işlemidir. XOR işleminin tersi de XOR işlemidir. Dolayısıyla, Ters KeyAddition işlemi de XOR işlemidir.

  K_4 = 0010 1011 1100 0111

  $C = C \oplus K_4$ = 0100 0011 1001 0101 $\oplus$ 0010 1011 1100 0111

= 0110 1000 0101 0010

$C' = C' \oplus K_4$ = 1110 0011 1001 1110 $\oplus$ 0010 1011 1100 0111

= 1100 1000 0101 1001

- Hala iki aktif ve pasif nibble'a, hem de aynı komumlarda sahibiz.
- Mini-AES şifreleme algoritmasının son döngüsünde, MixColumn dönüşümü olmadığı için Ters ShiftRow dönüşümü ile devam edelim.
- Ters ShiftRow, ShiftRow dönüşümü ile aynıdır.

0110	0101
0010	1000

1100	0101
1001	1000

C = 0110 0010 0101 1000

C' = 1100 1001 0101 1000

- Ters ShiftRow dönüşümü sonrası aktif ve pasif nibble sayısı aynıdır. Dönüşüm sonucunda iki nibble'ın yer değiştirmesi aktif nibble'ların sadece ilk kolonda yerleşmesine neden olmuştur.
- Ters NibbleSub dönüşümü, NibbleSub dönüşümünün tersi olup, bilinen y değerini, hangi x değerinin oluşturduğunu gösterir.

C = *TNSub*(0110) *TNSub*(0010) *TNSub*(0101) *TNSub*(1000)

C = 1010 0100 1100 0111

C' =*TNSub*(1100) *TNSub*(1001) *TNSub*(0101) *TNSub*(1000)

C' =1011 1101 1100 0111

- Elde edilen değerlerden de görüldüğü üzere aktif ve pasif nibble sayısı ve konumu aynı kalmıştır.

– Döngü 3

- Ters KeyAddition: K_3 = 1011 1100 0111 1101

$C = C \oplus K_4$ = 1010 0100 1100 0111 $\oplus$ 1011 1100 0111 1101

= 0001 1000 1011 1010

$C' = C' \oplus K_4$ = 1011 1101 1100 0111 $\oplus$ 1011 1100 0111 1101

= 0000 0001 1011 1010

- Elde edilen değerlerden de görüldüğü üzere aktif ve pasif nibble sayısı ve konumu aynı kalmıştır.
- Ters MixColumn

$$c_0 = (0010) \bullet (0001) \oplus (0011) \bullet (1000) = 0000$$

$$c_1 = (0011) \bullet (0001) \oplus (0010) \bullet (1000) = 1001$$

$$c_2 = (0010) \bullet (1011) \oplus (0011) \bullet (1010) = 1001$$

$$c_3 = (0011) \bullet (1011) \oplus (0010) \bullet (1010) = 1000$$

C = 0000 1001 1001 1000

$$c_0 = (0010) \bullet (0000) \oplus (0011) \bullet (0001) = 0010$$

$$c_1 = (0011) \bullet (0000) \oplus (0010) \bullet (0001) = 0011$$

$$c_2 = (0010) \bullet (1011) \oplus (0011) \bullet (1010) = 1001$$

$$c_3 = (0011) \bullet (1011) \oplus (0010) \bullet (1010) = 1000$$

C' = 0010 0011 1001 1000

- Elde edilen değerlerden de görüldüğü üzere, aktif ve pasif nibble sayısı ve konumu aynı kalmıştır.
- Ters ShiftRow

C = 0000 1000 1001 1001

C' = 0010 1000 1001 0011

0000	1001
1000	1001

0010	1001
1000	0011

- Aktif nibble sayısı aynı, fakat yerleri değişmiştir.
- Bu işlemin sonucu olarak, her bir kolonda bir aktif nibble oluşmuştur.
- Ters NibbleSub

C = *TNSub*(0000) *TNSub*(1000) *TNSub*(1001) *TNSub*(1001)

C = 1110 0111 1101 1101

C' = *TNSub*(0010) *TNSub(*1000*)* *TNSub*(1001) *TNSub*(0011)

C' = 0100 0111 1101 1000

- Elde edilen değerlerden de görüldüğü üzere, aktif ve pasif nibble sayısı ve pozisyonu aynı kalmıştır.

İkinci Döngüden Sonra

P = 1000 0110 0010 0111 P' = 1011 0100 0110 0001

C = 1110 0111 1101 1101 C' = 0100 0111 1101 1000

- İlk iki döngü boyunca yapmış olduğumuz analizlerden elde etmiş olduğumuz sonuçlar ile, son iki döngü boyunca yaptığımız analizlerden elde ettiğimiz sonuçlar çelişti.
- İlk iki döngünün analizi neticesinde, ikinci döngünün sonucunda tüm nibble'ların aktif olduğu çıkarımına varılmışken, son iki döngünün analizi sadece iki nibble'ın aktif olduğu çıkarımını öne sürmüştür.
- Buradaki çelişkiden faydalanılarak imkansız diferansiyel oluşturulabilir.
- Sadece bir nibble'ları farklı olan iki P ve P' açık metni, 4 döngülük Mini-AES algoritması ile şifrelenirse, hiçbir zaman sadece her bir kolon ve satırda bir nibble'ları farklı olan C ve C' şifreli metinleri elde edilemez.
- Bu dört döngülük imkansız diferansiyelin kullanımı ile Mini-AES şifreleyicisine imkansız diferansiyel saldırı gerçekleştirmek mümkün olacaktır.
- İmkansız diferansiyel ortadaki döngülere yerleştirilerek, dış döngülerdeki döngü anahtarları tahmin edilir ve bu anahtarlar kullanılarak imkansız diferansiyelin oluşup oluşmadığı doğrulanır.
- Eğer doğrulama başarısızlıkla sonuçlanırsa, tahmin edilen döngü anahtarları yanlıştır ve olası anahtarlar listesinden silinir.
- Bu yetenek imkansız diferansiyel kriptanalizin arkasındaki büyük güçtür.

5.4.3 5 Döngülük Mini-AES'e Saldırı

- 2^{11} açık metin elde edilir, P.
- 2^{11} açık metin elde edilir, P'.
- P' açık metinleri, P açık metinlerinden birinci ve dördüncü nibble'larda farklıdır.
- Her bir P ve P' açık metinleri için C ve C' şifreli metinleri elde edilir.
- Her bir kolonda sadece bir nibble'ları farklı olan C ve C' çiftleri seçilir.
- Kullanılabilir C/C' çiftleri olasılığı: $(2^{-4} \text{ x } 2^{-4}) + (2^{-4} \text{ x } 2^{-4}) = 2^{-7}$

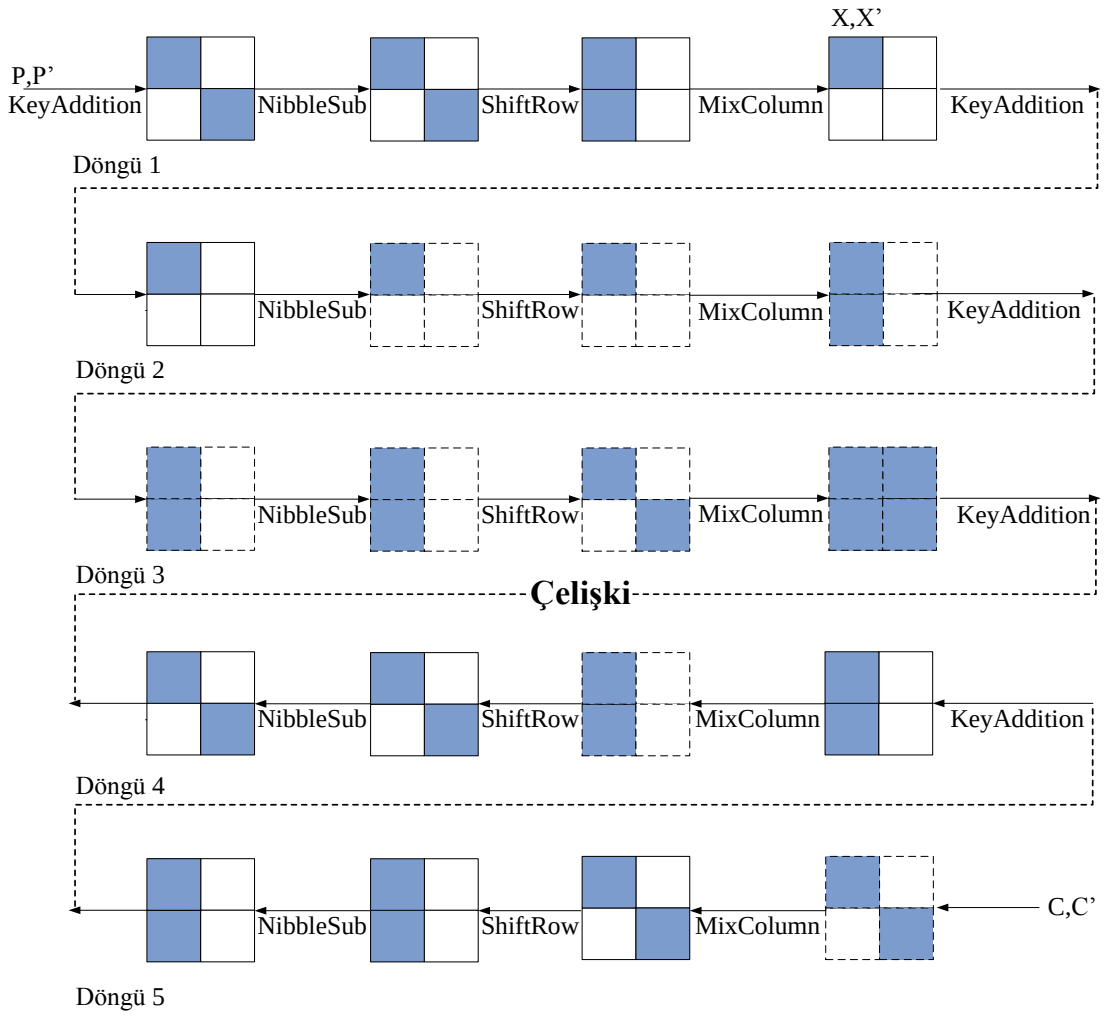
  $(2^{-7} \text{ x } 2^{11} = 2^4$ çift)
- Kullanılabilir C/C' çiftleri sayısı: $2^{11} \text{ x } 2^{-7} = 2^4$
- 2^4 çiftin herbiri için,
  - x ve x' değerleri, tüm olası K (2^8 değer) değerleri için hesaplanır.
  - $x = P$ değeri birinci döngüde MixColumn dönüşümü boyunca şifrelenir.

$$\left(x = \theta \circ \pi \circ \gamma \circ \sigma K_0(P)\right) \tag{5.8}$$

  - $x' = P'$ değeri birinci döngüde MixColumn dönüşümü boyunca şifrelenir.

$$\left(x' = \theta \circ \pi \circ \gamma \circ \sigma K_0(P')\right) \tag{5.9}$$

  - İlk kolonda sadece tek bir nibble'da farklı olan x ve x' çiftleri seçilir.
- Olasılık = $2^{-4} \times 2 = 2^{-3}$
- Rastgele seçilen bir anahtarın reddedilememe olasılığı: $\left(1 - 2^{-3}\right)^{2^6}$.
- Yanlış Anahtarlar: $2^8\left(1 - 2^{-3}\right)^{2^6} \approx 0$.
- Kalan sadece doğru K_0 değeridir.



Şekil 5.4. İmkansız Diferansiyeldeki Çelişki

```
private void imkansizDiferansiyel(      int ornekVeriSayisi,
                                        int bitSayisi,
                                        int donguSayisi)
{
    int n1 = (int)Math.Pow(2, 16);
    int n  = (int)Math.Pow(2, bitSayisi);
    for (int kA = 0; kA < n1; kA++)
    {
      string K0 = BitConversion.ConvertToBaseTwoNotation(kA, 16);
      cipher = new Cipher( CipherTypes.MiniAES,
                                  K0,
                                  m_FileSBox.FileReader,
                                  donguSayisi);
      DataTable dtAcikSifreliMetin = sifreliMetinAcikMetinCifti();
```

```
            MiniAES mini = new MiniAES( K0,
                                        _FileSBox.FileReader,
                                        donguSayisi));
            double maxCount = 0
            for (int i = 0; i < 16; i++)
            {
                string s = BitConversion.ConvertToBaseTwoNotation(i, 4);
                string k0 = s.Substring(0, 4) + "000000000000
                string[,] kOMatris = dizeMatrisDonusumu(k0, 2);
                double count = 0;
                DataRow drKey = dtKey.NewRow();
                foreach (DataRow dr in dtAcikSifreliMetin.Rows)
                {
                  string[,] Y1=mini.DonguFonksiyonu( dr["X1"].ToString(),
                                                     kOMatris);
                  string[,] Y2=mini.DonguFonksiyonu( dr["X2"].ToString(),
                                                     kOMatris);
                  int belirtec1 = (XOR(Y1[0, 0], Y2[0, 0]) == "0000" ? 0
                                  : 1) + (XOR(Y1[1, 0], Y2[1, 0]) ==
                                  "0000" ? 0 : 1);
                  int belirtec2 = (XOR(Y1[0, 1], Y2[0, 1]) == "0000" ? 0
                                  : 1) + (XOR(Y1[1, 1], Y2[1, 1]) ==
                                  "0000" ? 0 : 1);
                  if (belirtec1 == 2 && belirtec2 == 0)
                  {
                        drKey["X1"] = dr["X1"];
                        drKey["X2"] = dr["X2"];
                        count++;
                  }
            }
            if (maxCount < count) maxCount = count;
            drKey["Key"] = k0;
            drKey["Count"] = count;
            drKey["Probability"] = count /
                                   (double)dtAcikSifreliMetin.Rows.Count;
            if (Convert.ToDouble(drKey["Probability"]) > (1 / 8))
            {
                drKey["RightKey"] = "Evet";
            }
            else
            {
                drKey["RightKey"] = "Hayır";
            }
            dtKey.Rows.Add(drKey);
          }
}
```

Algoritma 5.3. Mini-AES İmkansız Diferansiyel Saldırı Örnek Kaynak Kodu

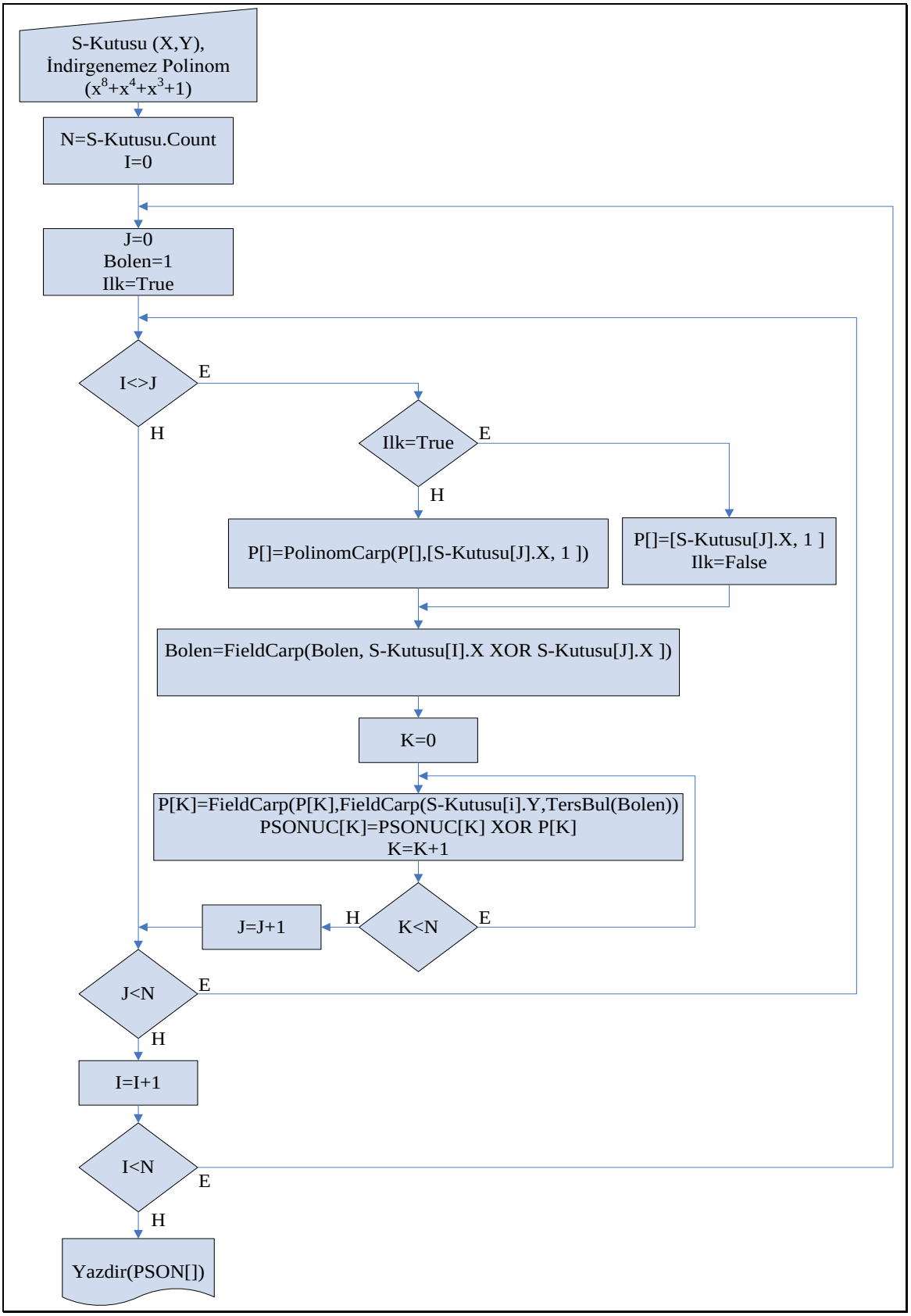
5.4.4 Mini-AES İmkansız Diferansiyel Saldırı Sonuçları

Beş Döngülülük Mini-AES şifreleyicisine imkansız diferansiyel saldırı düzenlemek için dört döngülük imkansız diferansiyel gereklidir. İmkansız diferansiyel saldırı bir şifreleyici ile giriş farkının oluşturduğu çıkış farkının, doğru anahtarla oluşturması imkansız olan farkları, oluşturan anahtarların elenmesi yöntemiyle, olası anahtarlar arasından doğru anahtarı bulmayı amaçlar.

Bu çalışmada dört döngülük imkansız diferansiyeli belirlemek amacıyla, ilk nibble dışında eşit olan iki açık metin ile çalıştık. İkinci döngü sonunda, tüm nibble'ları aktif olan iki çıkış değerine sahip olduk. Daha sonra bu analizde kullanmış olduğumuz iki açık metne karşılık gelen şifreli metin çiftlerini, şifreleyicinin döngüleri boyunca ters yönde iki döngü ilerlettik (tüm nibble'lar aktif) ve gördük ki; iki analizden elde ettiğimiz sonuçlar birbiriyle çelişti. Bu çelişkiden faydalanarak oluşturduğumuz imkansız diferansiyel ile, beş döngülük Mini-AES şifreleyicisine imkansız diferansiyel saldırı düzenledik. Bu saldırı sonucunda 1 olasılıkla doğru anahtar belirlenebilmektedir.

5.5 Lagrange Interpolasyonu İle Cebirsel İfadenin Elde Edilmesi

Algoritma 5.4'de lagrange interpolasyonunun akış diyagramı görülmektedir. Lagrange interpolasyonu algoritması parametrik olarak herhangi bir sonlu cisimde herhangi bir indrigenemez polinom kullanarak, herhangi bir S-kutusunun cebirsel ifadesini bulabilmektedir. Lagrange interpolasyonu algoritmasında iç içe iki döngüde tüm noktalar $(i \neq j)$ şartı için, S-kutusunun tüm giriş değerleri $(x_i + x_j)$ ifadesi şeklinde yazılarak sonlu cisim aritmetiğinde birbirleri ile polinom çarpımı işlemine tabi tutulur. İç içe döngü içinde $(x_i + x_j)$ değerleri sonlu cisim aritmetiğinde birbirleriyle çarpma işlemine tabi tutularak lagrange interpolasyonu ifadesinin paydası elde edilir. Sonlu cisimde payın paydaya bölünmesi paydanın sonlu cisimde tersi alınarak pay ile çarpılması ile gerçeklenir. Elde edilen değer y_i değeri ile çarpıldıktan sonra bir önceki aşamada elde edilen değer ile toplanır (XOR işlemine tabi tutulur.). Tüm bu işlemler tüm S-kutusu giriş ve çıkış değerleri için tekrarlanır. Çarpma ve ters alma işleminde sonlu cismin derecesinden yüksek derecede terimler elde edildiğinde, algoritmaya giriş değeri olarak verilen indirgenemez polinomla indirgeme işlemi yapılır.



Algoritma 5.4. Lagrange Interpolasyonu Akış Diyagramı

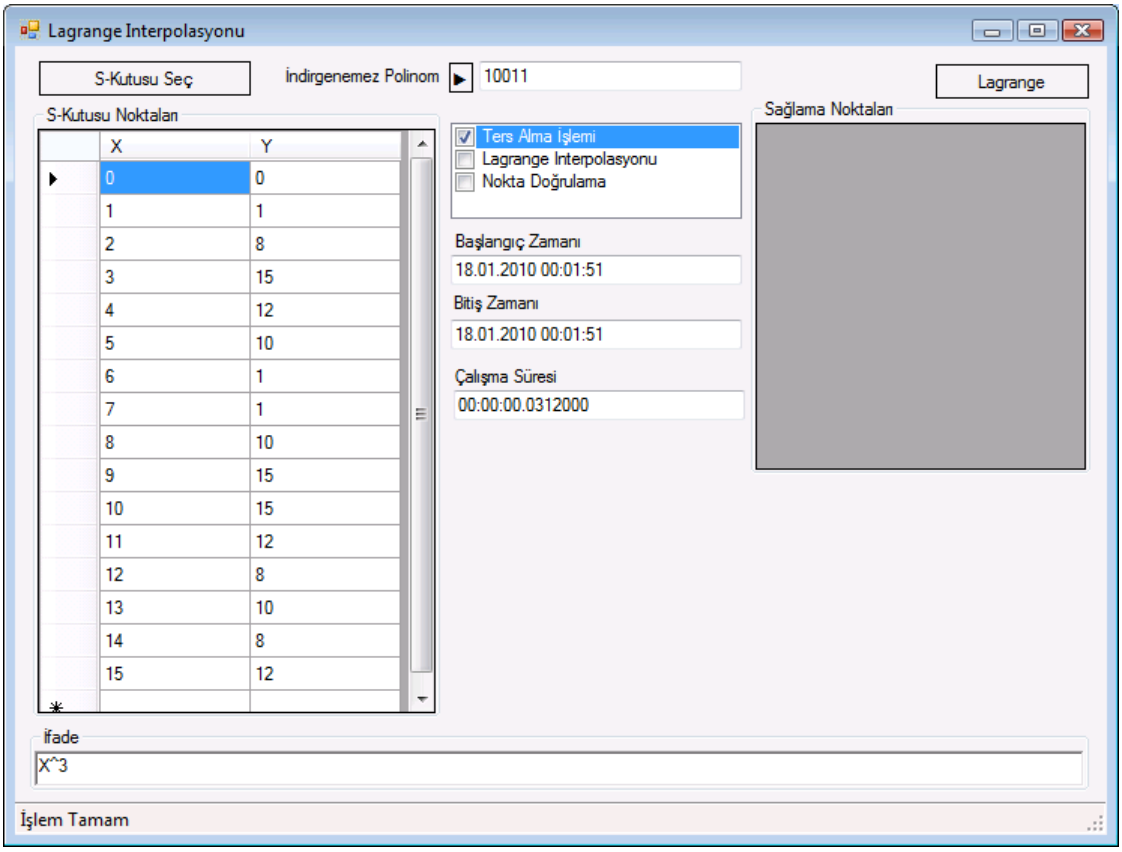
```
private void Lagrange(List<PointF> points)
{
   int[] p = new int[points.Count];
   int[] p5 = new int[points.Count * 2];
   bool ilk = true;
   for (int i = 0; i < points.Count; i++)
   {
      int bolen = 1;
      ilk = true;
      for (int k = 0; k < points.Count; k++)
      {
         if (i != k)
         {
            if (ilk)
            {
               p = new int[] { (int)points[k].X, 1 };
               ilk = false;
            }
            else
            {
                p = Polinom.Carpim(p, new int[] { (int)points[k].X, 1 });
            }
            bolen = fieldCarp(bolen, (int)((int)points[i].X ^ (int)points[k].X));
         }
        for (int j = 0; j < p.Length; j++)
        {
           p[j] = fieldCarp(p[j], fieldCarp((int)points[i].Y, tersBul(bolen)));
            p5[j] ^= p[j];
        }
   }
   yazdir(p5);
}
```

Algoritma 5.5. Lagrange Interpolasyon Algoritması Örnek Kaynak Kodu

5.5.1 Lagrange Interpolasyonu Analizleri

Yapılan analizler sonucunda $GF(2^4)$'de kullanılan indirgenemez polinom, elde edilen $s(x)$ polinomunun terim sayısında belirleyici olmaktadır.

Tablo 5.1'de gösterilen haritalama için iki farklı indirgenemez polinom kullanılarak, lagrange interpolasyonu sonucunda şu ifadeler elde edilmiştir:



Şekil 5.5. x^4+x+1 İndirgenemez Polinomu İle İnterpolasyon

Tablo 5.1. Örnek S-Kutusu

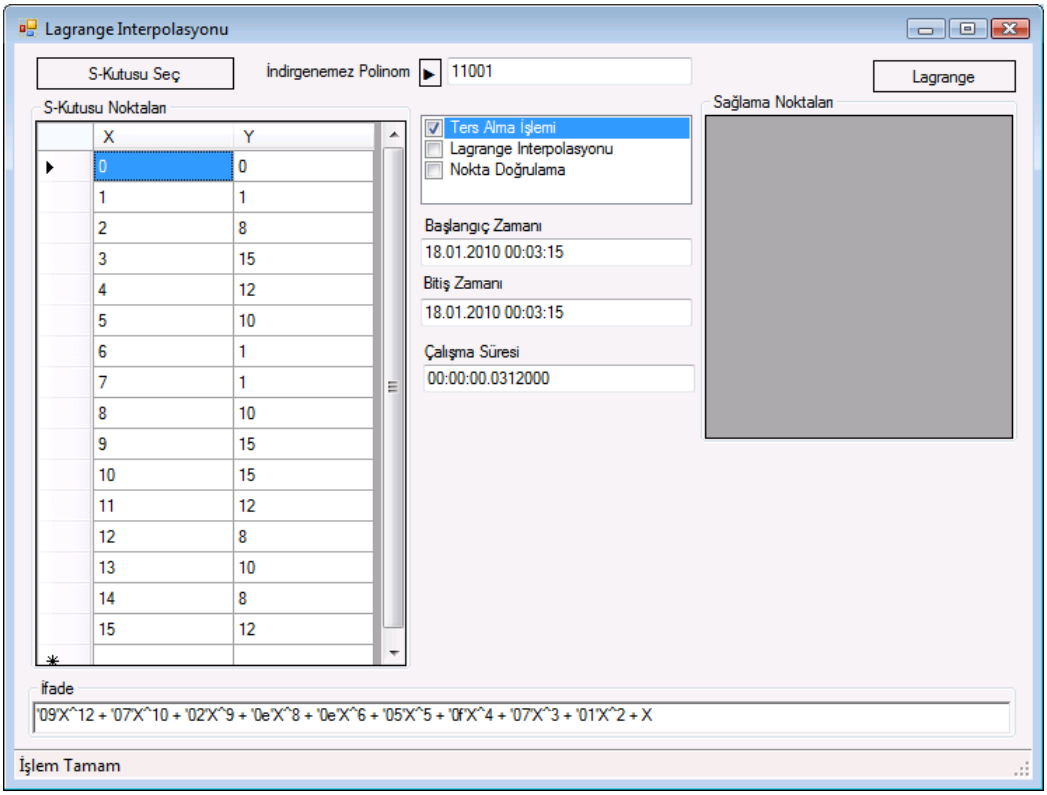
1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$s(x)$	0	1	8	15	12	10	1	1	10	15	15	12	8	10	8	2

– x^4+x+1 indirgenemez polinomu kullanılarak $s(x)=x^3$ fadesi elde edilmiştir (bknz.Şekil 5.5).

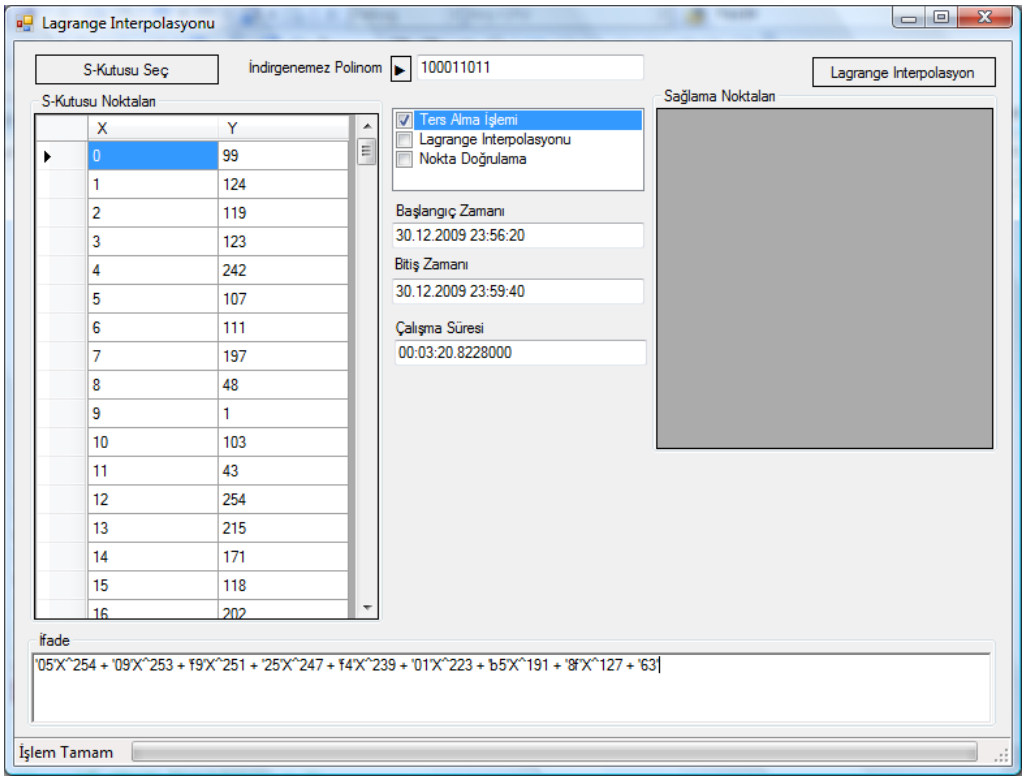
– x^4+x^3+1 indirgenemez polinomu kullanıldığında elde edilen ifadenin terim sayısının arttığı aşağıdaki ifadeden görülmektedir (bknz. Şekil 5.6).

$$s(x)='09'x^{12}+'07'x^{10}+'02'x^9+'0e'x^8+'0e'x^6 +'05'x^5+'0f'x^4+'07'x^3+x^2+x \tag{5.10}$$

AES S-kutusuna (bknz. Tablo 5.2) $GF(2^8)$ sonlu cisiminde $x^8+x^4+x^3+x+1$ indirgenemez polinomu kullanılarak lagrange interpolasyonu uygulandığında [87] [89] ve [90]'deki çalışmada olduğu gibi,



Şekil 5.6. $x^4 + x^3 + 1$ İndirgenemez Polinomu İle İnterpolasyon



Şekil 5.7. AES S-kutusu İnterpolasyon

$$s(x) = '05'x^{254} + '09'x^{253} + 'f9'x^{251} + '25'x^{247} + 'f4'x^{239} + '01'x^{223} + 'b5'x^{191} + '8f'x^{127} + '63' \quad (5.11)$$

ifadesi elde edilmiştir (bknz. Şekil 5.7).

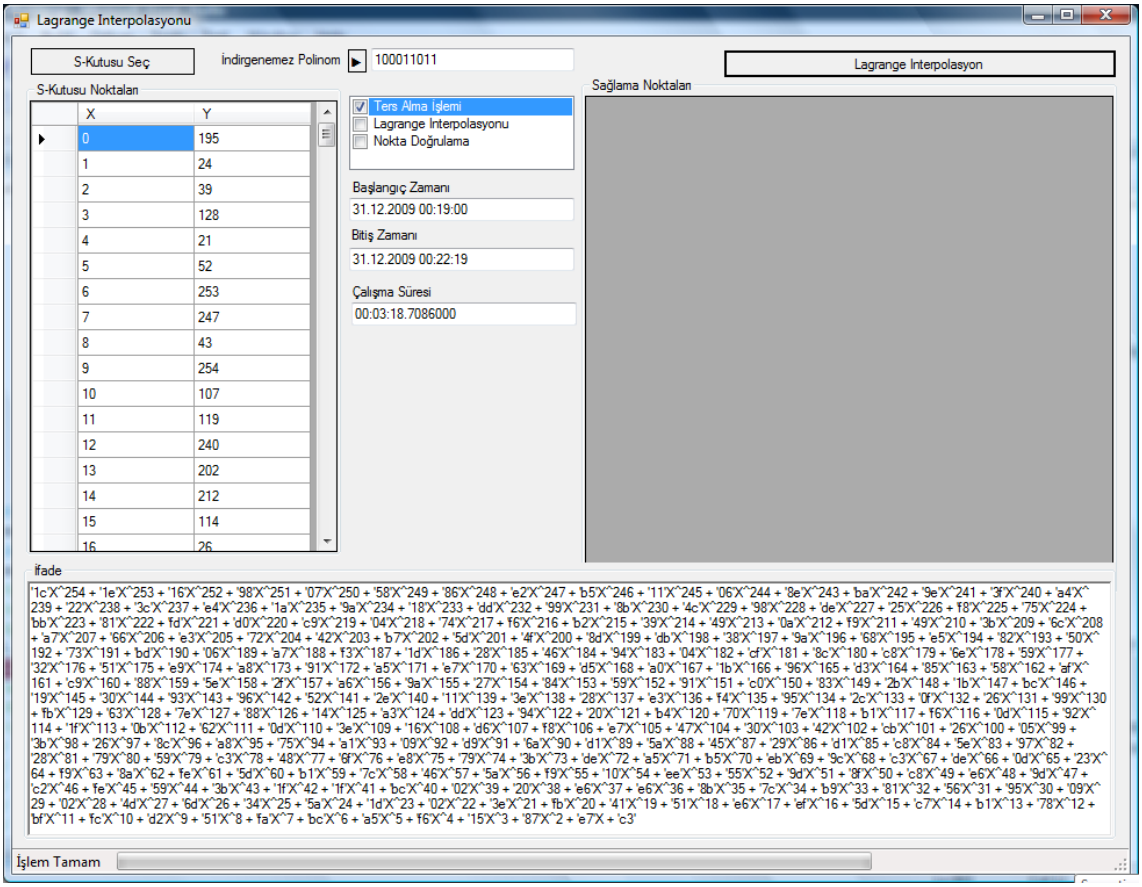
Tablo 5.2. $x \to x^{-1}$ Üst Haritalamalı AES S-kutusu

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

AES S-kutusu $x \to x^{-1}$ üs haritalamasını takip eden bir doğrusal dönüşümden oluşur. AES S-kutusuna uygulanan lagrange interpolasyonunda da görüldüğü gibi, AES S-kutusu biri sabit değer olmak üzere 9 terimden oluşmaktadır. Oysa $GF(2^8)$ sonlu cisminde 256 terim vardır. Terim sayısının az olması AES S-kutusunun cebirsel saldırılara karşı zayıf kalmasına neden olabilir. AES S-kutusundaki oluşabilecek bu zayıflık doğrusal dönüşümün yeri ve sayısı ile orantılı olarak giderilebilir. Doğrusal dönüşüm üs haritamaladan önce, sonra veya hem önce hem de sonra olmak üzere üç farklı şekilde uygulanabilir. Bu çalışmada [57]'deki çalışmada doğrusal dönüşüm üs haritalamadan hem önce hem de sonra kullanılarak tasarlanan S-kutularının lagrange interpolasyonu ile cebirsel ifadesi elde edilecektir.

$x \to x^{127}$ ve $x \to x^{254}$ üs haritalamalarında önce ve sonra olmak üzere (5.12) ve (5.13) ifadelerindeki $LA_1(x)$ ve $LA_2(x)$ doğrusal dönüşümleri uygulanarak Tablo 5.3 ve Tablo 5.4' deki S-kutuları elde edilmiştir [91]. (bknz.(5.14)) ($LA_1(x)$ ve $LA_2(x)$ doğrusal dönüşümleri Sakallı-Aslan-Buluş vd. [92]'deki çalışmasından alınmıştır.).

Elde edilen bu S-kutuları $GF(2^8)$ sonlu cisminde S-kutusu giriş ve çıkış değerleri, noktalar olarak kullanılarak lagrange interpolasyonuna tabi tutulmuşlardır.



Şekil 5.8. $x \to x^{254}$ Üs Haritalamalı AES S-kutusu Interpolasyonu

$$LA_1(x) = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{5.12}$$

$$LA_2(x)=\begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \end{bmatrix}=\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix}+\begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{5.13}$$

$$s_1(x)=L_{,A2}\left((L_{A1}(x))^{127}\right) \tag{5.14}$$

Tablo 5.3. $x \to x^{254}$ Üs Haritalama Uygulanarak Elde Edilen S-Kutusu

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	C3	18	27	80	15	34	FD	F7	2B	FE	6B	77	F0	CA	D4	72
1	1A	1B	E3	D6	CF	6A	D1	B1	21	10	9D	40	85	D0	F9	9F
2	66	48	C1	57	8A	E8	78	B4	E9	CE	D9	98	68	8C	99	BB
3	0A	49	95	AC	08	6C	C8	4E	14	DE	2A	4F	17	CD	A7	19
4	89	E6	B0	0F	28	1E	E1	94	74	BD	1C	2E	F6	3E	61	9E
5	13	97	64	3D	0B	EE	60	88	F4	7A	8D	6D	24	32	C2	79
6	C9	59	9C	AF	AB	01	63	C5	E5	D8	36	26	05	C7	07	75
7	AA	4D	50	7F	F3	B6	51	F5	BE	4C	20	ED	5A	83	52	84
8	E7	A9	AE	56	91	62	3A	06	C4	73	44	0C	22	DC	B8	5E
9	BA	C6	8B	DD	86	B9	B5	03	41	16	42	A1	69	11	87	55
A	53	5B	58	CB	29	B3	2C	6E	45	A8	33	EF	92	8F	DA	FF
B	B7	CC	31	A5	EB	E2	23	96	AD	C0	47	82	F2	7B	67	D7
C	A3	38	D2	BC	3C	02	FB	43	3B	2F	A0	09	FC	00	39	4A
D	7C	6F	76	30	A4	A2	7D	FA	12	B2	9A	04	3F	93	F1	71
E	81	90	DB	46	5D	7E	EC	5F	D3	E4	5C	E0	D5	37	EA	65
F	F8	8E	DF	9B	54	2D	0D	BF	35	1D	0E	70	A6	25	1F	4B

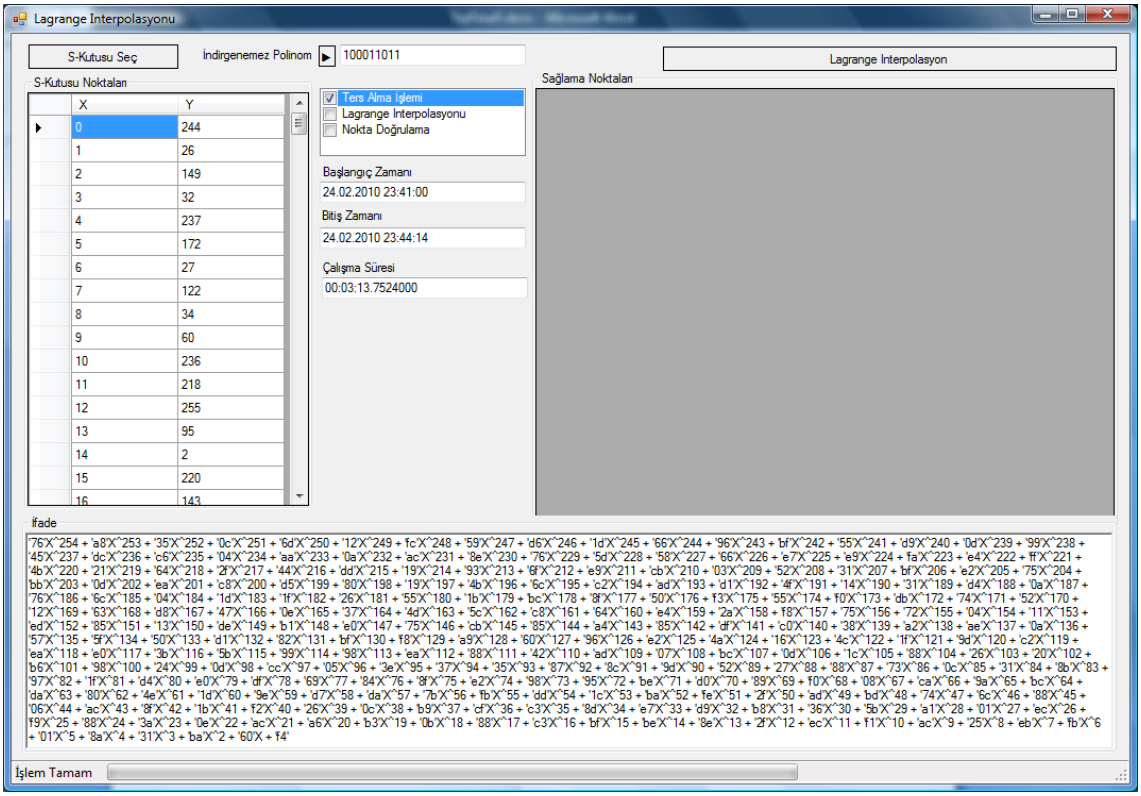
Tablo 5.3'de gösterilen $x \to x^{254}$ üs haritalamalı AES S-kutusuna lagrange interpolasyonu uygulandığında elde edilen ifadede, $GF(2^8)$ sonlu cismindeki $x^{255}=1$ dışındaki terimlerin tümü yer alır. Lagrange interpolasyonu sonucunda elde edilen cebirsel ifade sonraki sayfada gösterilmiştir.

$$s(x)='1c'x^{254}+'1e'x^{253}+'16'x^{252}+'98'x^{251}+'07'x^{250}+'58'x^{249}+'86'x^{248}+'e2'x^{247}+'b5'x^{246}+$$
$$'11'x^{245}+'06'x^{244}+'8e'x^{243}+'ba'x^{242}+'9e'x^{241}+63x^{240}+'a4'x^{239}+'22'x^{238}+'3c'x^{237}+'e4'x^{236}+$$
$$'1a'x^{235}+'9a'x^{234}+'18'x^{233}+'dd'x^{232}+'99'x^{231}+'8b'x^{230}+'4c'x^{229}+'98'x^{228}+'de'x^{227}+'25'x^{226}+$$
$$'f8'x^{225}+'75'x^{224}+'bb'x^{223}+'81'x^{222}+'fd'x^{221}+'d0'x^{220}+'c9'x^{219}+'04'x^{218}+'74'x^{217}+'f6'x^{216}+$$
$$'b2'x^{215}+'39'x^{214}+'49'x^{213}+'0a'x^{212}+'f9'x^{211}+'49'x^{210}+'3b'x^{209}+'6c'x^{208}+'a7'x^{207}+'66'x^{206}+$$

$$'e3'x^{205}+'72'x^{204}+'42'x^{203}+'b7'x^{202}+'5d'x^{201}+'4f'x^{200}+'8d'x^{199}+'db'x^{198}+'38'x^{197}+'9a'x^{196}+$$
$$'68'x^{195}+'e5'x^{194}+'82'x^{193}+'50'x^{192}+'73'x^{191}+'bd'x^{190}+'06'x^{189}+'a7'x^{188}+'f3'x^{187}+'1d'x^{186}+$$
$$'28'x^{185}+'46'x^{184}+'94'x^{183}+'04'x^{182}+'cf'x^{181}+'8c'x^{180}+'c8'x^{179}+'6e'x^{178}+'59'x^{177}+'32'x^{176}$$
$$'51'x^{175}+'e9'x^{174}+'a8'x^{173}+'91'x^{172}+'a5'x^{171}+'e7'x^{170}+'63'x^{169}+'d5'x^{168}+'a0'x^{167}+'1b'x^{166}+$$
$$'96'x^{165}+'d3'x^{164}+'85'x^{163}+'58'x^{162}+'af'x^{161}+'c9'x^{160}+'88'x^{159}+'5e'x^{158}+'2f'x^{157}+'a6'x^{156}+$$
$$'9a'x^{155}+'27'x^{154}+'84'x^{153}+'59'x^{152}+'91'x^{151}+'c0'x^{150}+'83'x^{149}+'2b'x^{148}+'1b'x^{147}+'bc'x^{146}+$$
$$'19'x^{145}+'30'x^{144}+'93'x^{143}+'96'x^{142}+'52'x^{141}+'2e'x^{140}+'11'x^{139}+'3e'x^{138}+'28'x^{137}+'e3'x^{136}+$$
$$'f4'x^{135}+'95'x^{134}+'2c'x^{133}+'0f'x^{132}+'26'x^{131}+'99'x^{130}+'fb'x^{129}+'63'x^{128}+'7e'x^{127}+'88'x^{126}+$$
$$'14'x^{125}+'a3'x^{124}+'dd'x^{123}+'94'x^{122}+'20'x^{121}+'b4'x^{120}+'70'x^{119}+'7e'x^{118}+'b1'x^{117}+'f6'x^{116}+$$
$$'0d'x^{115}+'92'x^{114}+'1f'x^{113}+'0b'x^{112}+'62'x^{111}+'0d'x^{110}+'3e'x^{109}+'16'x^{108}+'d6'x^{107}+'f8'x^{106}+$$
$$'e7'x^{105}+'47'x^{104}+'30'x^{103}+'42'x^{102}+'cb'x^{101}+'26'x^{100}+'05'x^{99}+'3b'x^{98}+'26'x^{97}+'8c'x^{96}+$$
$$'a8'x^{95}+'75'x^{94}+'a1'x^{93}+'09'x^{92}+'d9'x^{91}+'6a'x^{90}+'d1'x^{89}+'5a'x^{88}+'45'x^{87}+'29'x^{86}+$$
$$'d1'x^{85}+'c8'x^{84}+'5e'x^{83}+'97'x^{82}+'28'x^{81}+'79'x^{80}+'59'x^{79}+'c3'x^{78}+'48'x^{77}+'6f'x^{76}+$$
$$'e8'x^{75}+'79'x^{74}+'3b'x^{73}+'de'x^{72}+'a5'x^{71}+'b5'x^{70}+'eb'x^{69}+'9c'x^{68}+'c3'x^{67}+'de'x^{66}+$$
$$130d'x^{65}+'23'x^{64}+'f9'x^{63}+'8a'x^{62}+'fe'x^{61}+'5d'x^{60}+'b1'x^{59}+'7c'x^{58}+'46'x^{57}+'5a'x^{56}+$$
$$'f9'x^{55}+'10'x^{54}+'ee'x^{53}+'55'x^{52}+'9d'x^{51}+'8f'x^{50}+'c8'x^{49}+'e6'x^{48}+'9d'x^{47}+'c2'x^{46}+$$
$$'fe'x^{45}+'59'x^{44}+'3b'x^{43}+'1f'x^{42}+'1f'x^{41}+'bc'x^{40}+'02'x^{39}+'20'x^{38}+'e6'x^{37}+'e6'x^{36}+$$
$$'8b'x^{35}+'7c'x^{34}+'b9'x^{33}+'81'x^{32}+'56'x^{31}+'95'x^{30}+'09'x^{29}+'02'x^{28}+'4d'x^{27}+'6d'x^{26}+$$
$$'34'x^{25}+'5a'x^{24}+'1d'x^{23}+'02'x^{22}+'3e'x^{21}+'fb'x^{20}+'41'x^{19}+'51'x^{18}+'e6'x^{17}+'ef'x^{16}+$$
$$'5d'x^{15}+'c7'x^{14}+'b1'x^{13}+'78'x^{12}+'bf'x^{11}+'fc'x^{10}+'d2'x^{9}+'51'x^{8}+'fa'x^{7}+'bc'x^{6}+$$
$$'a5'x^{5}+'f6'x^{4}+'15'x^{3}+'87'x^{2}+'e7'x+'c3'.$$

Tablo 5.4. $x \to x^{127}$ Üs Haritalama Uygulanarak Elde Edilen S-Kutusu

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	F4	1A	95	20	ED	AC	1B	7A	22	3C	EC	DA	FF	5F	02	DC
1	8F	7E	43	5E	B8	FB	5D	92	4C	49	D6	6D	8D	E3	0A	BE
2	8E	C0	A1	30	76	42	AD	F8	E8	C7	9E	F7	FD	61	05	8A
3	15	F6	07	78	B4	75	3E	34	7C	41	47	DF	9B	82	16	6F
4	93	B9	96	4B	1F	23	86	3F	FE	E6	AF	98	D9	9C	89	2B
5	E9	C9	B0	6B	77	0F	59	CE	A6	EB	B5	F1	71	03	E2	25
6	F3	17	F9	60	88	9F	CD	B6	53	D3	A0	AE	56	09	E4	5B
7	A9	2F	E0	CB	58	5C	3B	E7	EE	D0	19	F5	51	A7	85	0B
8	70	A5	C6	B2	12	BA	31	0E	BB	C5	1E	CA	F0	28	63	99
9	D4	E1	AB	68	8C	01	94	72	9A	8B	B7	35	1D	CC	DD	04
A	64	9D	14	AA	4D	DE	40	3D	3A	39	D5	29	B3	37	4A	06
B	6E	65	00	62	91	FC	EA	2C	7F	4F	10	97	13	21	2A	50
C	52	C8	CF	24	32	6A	4E	84	33	DB	A3	C4	73	38	46	EF
D	0C	BC	90	A8	45	81	D2	44	79	B1	6C	83	5A	08	D1	C2
E	FA	55	7B	2D	54	F2	87	7D	80	2E	D8	48	66	E5	1C	27
F	C1	36	74	C3	18	BD	26	57	0D	67	69	A4	A2	11	D7	BF



Şekil 5.9. $x \to x^{127}$ Üst Haritalamalı AES S-kutusu Interpolasyonu

$x \to x^{127}$ üs haritalamalı AES S-kutusu lagrange interpolasyonu uygulandığında elde edilen ifade aşağıda gösterilmektedir:

$$s(x) = {}'76'x^{254} + {}'a8'x^{253} + {}'35'x^{252} + {}'0c'x^{251} + {}'6d'x^{250} + {}'12'x^{249} + {}'fc'x^{248} + {}'59'x^{247} + {}'d6'x^{246} +$$
$$`1d'x^{245} + {}'66'x^{244} + {}'96'x^{243} + {}'bf'x^{242} + {}'55'x^{241} + {}'d9'x^{240} + {}'0d'x^{239} + {}'99'x^{238} + {}'45'x^{237} + {}'dc'x^{236} +$$
$$`c6'x^{235} + {}'04'x^{234} + {}'aa'x^{233} + {}'0a'x^{232} + {}'ac'x^{231} + {}'8e'x^{230} + {}'76'x^{229} + {}'5d'x^{228} + {}'58'x^{227} + {}'66'x^{226} +$$
$$`e7'x^{225} + {}'e9'x^{224} + {}'fa'x^{223} + {}'e4'x^{222} + {}'ff'x^{221} + {}'4b'x^{220} + {}'21'x^{219} + {}'64'x^{218} + {}'2f'x^{217} + {}'44'x^{216} +$$
$$`dd'x^{215} + {}'19'x^{214} + {}'93'x^{213} + {}'6f'x^{212} + {}'e9'x^{211} + {}'cb'x^{210} + {}'03'x^{209} + {}'52'x^{208} + {}'31'x^{207} + {}'bf'x^{206} +$$
$$`e2'x^{205} + {}'75'x^{204} + {}'bb'x^{203} + {}'0d'x^{202} + {}'ea'x^{201} + {}'c8'x^{200} + {}'d5'x^{199} + {}'80'x^{198} + {}'19'x^{197} + {}'4b'x^{196} +$$
$$`6c'x^{195} + {}'c2'x^{194} + {}'ad'x^{193} + {}'d1'x^{192} + {}'4f'x^{191} + {}'14'x^{190} + {}'31'x^{189} + {}'d4'x^{188} + {}'0a'x^{187} +$$
$$`6c'x^{185} + {}'04'x^{184} + {}'1d'x^{183} + {}'1f'x^{182} + {}'26'x^{181} + {}'55'x^{180} + {}'1b'x^{179} + {}'bc'x^{178} + {}'8f'x^{177} + {}'50'x^{176} +$$
$$`f3'x^{175} + {}'55'x^{174} + {}'f0'x^{173} + {}'db'x^{172} + {}'74'x^{171} + {}'52'x^{170} + {}'12'x^{169} + {}'63'x^{168} + {}'d8'x^{167} + {}'47'x^{166} +$$
$$`0e'x^{165} + {}'37'x^{164} + {}'4d'x^{163} + {}'5c'x^{162} + {}'c8'x^{161} + {}'64'x^{160} + {}'e4'x^{159} + {}'2a'x^{158} + {}'f8'x^{157} + {}'75'x^{156} +$$
$$`72'x^{155} + {}'04'x^{154} + {}'11'x^{153} + {}'ed'x^{152} + {}'85'x^{151} + {}'13'x^{150} + {}'de'x^{149} + {}'b1'x^{148} + {}'e0'x^{147} + {}'75'x^{146} +$$
$$`cb'x^{145} + {}'85'x^{144} + {}'a4'x^{143} + {}'85'x^{142} + {}'df'x^{141} + {}'c0'x^{140} + {}'38'x^{139} + {}'a2'x^{138} + {}'ae'x^{137} + {}'0a'x^{136} +$$
$$`57'x^{135} + {}'5f'x^{134} + {}'50'x^{133} + {}'d1'x^{132} + {}'82'x^{131} + {}'bf'x^{130} + {}'f8'x^{129} + {}'a9'x^{128} + {}'60'x^{127} + {}'96'x^{126}$$

$$`e2'x^{125} +' 4a'x^{124} +'16'x^{123} +' 4c'x^{122} +'1f'x^{121} +' 9d'x^{120} +' c2'x^{119} +' ea'x^{118} +' e0'x^{117} +' 3b'x^{116} +$$
$$`5b'x^{115} +' 99'x^{114} +' 98'x^{113} +' ea'x^{112} +' 88'x^{111} +' 42'x^{110} +' ad'x^{109} +' 07'x^{108} +' bc'x^{107} +' 0d'x^{106} +$$
$$`1c'x^{105} +' 88'x^{104} +' 26'x^{103} +' 20'x^{102} +' b6'x^{101} +' 98'x^{100} +' 24'x^{99} +' 0d'x^{98} +' cc'x^{97} +' 05'x^{96} +$$
$$`3e'x^{95} +' 37'x^{94} +' 35'x^{93} +' 87'x^{92} +' 8c'x^{91} +' 9d'x^{90} +' 52'x^{89} +' 27'x^{88} +' 88'x^{87} +' 73'x^{86} +$$
$$`0c'x^{85} +' 31'x^{84} +' 8b'x^{83} +' 97'x^{82} +'1f'x^{81} +' d4'x^{80} +' e0'x^{79} +' df'x^{78} +' 69'x^{77} +' 84'x^{76} +$$
$$`8f'x^{75} +' e2'x^{74} +' 98'x^{73} +' 95'x^{72} +' be'x^{71} +' d0'x^{70} +' 89'x^{69} +' f0'x^{68} +' 08'x^{67} +' ca'x^{66} +$$
$$`9a'x^{65} +' bc'x^{64} +' da'x^{63} +' 80'x^{62} +' 4e'x^{61} +'1d'x^{60} +' 9e'x^{59} +' d7'x^{58} +' da'x^{57} +' 7b'x^{56} +$$
$$`fb'x^{55} +' dd'x^{54} +'1c'x^{53} +' ba'x^{52} +' fe'x^{51} +' 2f'x^{50} +' ad'x^{49} +' bd'x^{48} +' 74'x^{47} +' 6c'x^{46} +$$
$$`88'x^{45} +' 06'x^{44} +' ac'x^{43} +'8f'x^{42} +'1b'x^{41} +' f2'x^{40} +' 26'x^{39} +' 0c'x^{38} +' b9'x^{37} +' cf'x^{36} +$$
$$`c3'x^{35} +' 8d'x^{34} +' e7'x^{33} +' d9'x^{32} +' b8'x^{31} +' 36'x^{30} +' 5b'x^{29} +' a1'x^{28} +' 01'x^{27} +' ec'x^{26} +$$
$$`f9'x^{25} +' 88'x^{24} +' 3a'x^{23} +' 0e'x^{22} +' ac'x^{21} +' a6'x^{20} +' b3'x^{19} +' 0b'x^{18} +' 88'x^{17} +' c3'x^{16} +$$
$$`bf'x^{15} +' be'x^{14} +' 8e'x^{13} +' 2f'x^{12} +' ec'x^{11} +' f1'x^{10} +' ac'x^{9} +' 25'x^{8} +' eb'x^{7} +' fb'x^{6} +$$
$$`01'x^{5} +' 8a'x^{4} +' 31'x^{3} +' ba'x^{2} +' 60'x +' f4'.$$

$x \to x^7$ üs haritalamalı AES S-kutusu lagrange interpolasyonu uygulandığında elde edilen cebirsel ifade aşağıda gösterilmektedir:

$$s(x) =' 5f'x^{224} +' 25'x^{208} +' e2'x^{200} +' f0'x^{196} +' 6e'x^{194} +' d6'x^{193} +' 45'x^{192} +' 01'x^{176} +'1c'x^{168} +$$
$$`a9'x^{164} +' ba'x^{162} +' 0b'x^{161} +' 70'x^{160} +' 04'x^{152} +' 56'x^{148} +' 7c'x^{146} +' c7'x^{145} +' 85'x^{144} +$$
$$`fd'x^{140} +' 9f'x^{138} +' f0'x^{137} +' eb'x^{136} +'13'x^{134} +' a7'x^{133} +' 4d'x^{132} +' ed'x^{131} +' 41'x^{130} +$$
$$`44'x^{129} +' ff'x^{128} +' 66'x^{112} +' 31'x^{104} +' b7'x^{100} +' d8'x^{98} +' 84'x^{97} +' 98'x^{96} +' cc'x^{88} +$$
$$37'x^{84} +' e6'x^{82} +' f3'x^{81} +' a6'x^{80} +' 8c'x^{76} +' d5'x^{74} +' 6f'x^{73} +' 6a'x^{72} +' f8'x^{70} +' c4'x^{69} +$$
$$`8f'x^{68} +' 36'x^{67} +' a3'x^{66} +' 52'x^{65} +' bf'x^{64} +' f7'x^{56} +' 58'x^{52} +' 30'x^{50} +' fe'x^{49} +' ca'x^{48} +$$
$$`d5'x^{44} +' f6'x^{42} +'13'x^{41} +' dc'x^{40} +' 04'x^{38} +' ea'x^{37} +' e2'x^{36} +' 65'x^{35} +' 24'x^{34} +' 67'x^{33} +$$
$$`21'x^{32} +' e0'x^{28} +' ff'x^{26} +' 8b'x^{25} +' c8'x^{24} +' 53'x^{22} +' 4b'x^{21} +' e9'x^{20} +' 0c'x^{19} +' be'x^{18} +$$
$$`81'x^{17} +' df'x^{16} +' fc'x^{14} +' d9'x^{13} +' d8'x^{12} +' 09'x^{11} +' 6a'x^{10} +' 42'x^{9} +' f4'x^{8} +'14'x^{7} +$$
$$`1e'x^{6} +' 8e'x^{5} +' c2'x^{4} +' ba'x^{3} +' 2d'x^{2} +' b8'x +' 68'.$$

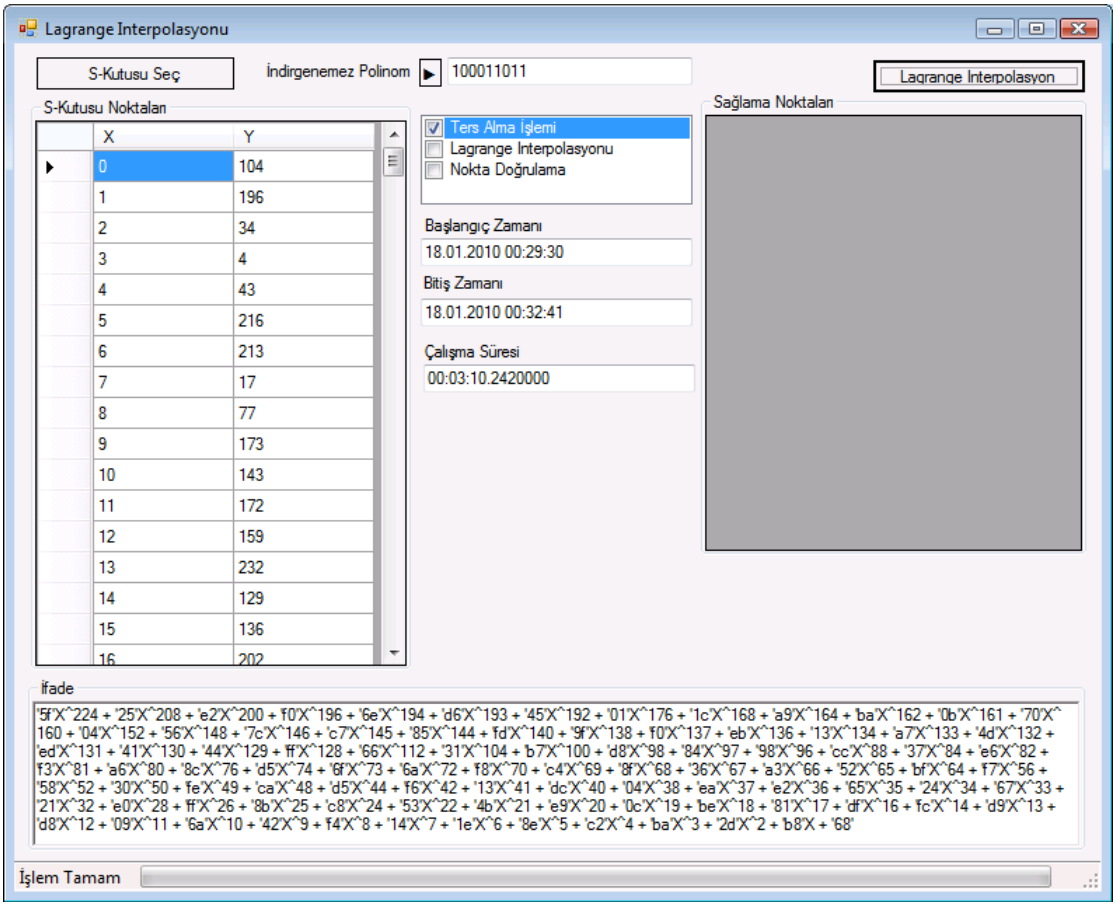
$x \to x^7$ üs haritalaması kullanılarak tasarlanan Tablo 5.5'de giriş ve çıkış değerleri verilen S-kutusunun lagrange interpolasyonu sonucunda elde edilen cebirsel ifade 93 terimden oluşmakta olup yukarıda görülmektedir. Şekil 5.10'da lagrange interpolasyonu uygulamasının ürettiği sonuçları gösteren ekran görüntüsü yer almaktadır.

Herhangi bir üst haritalamayı kullanararak tasarlanan S-kutularındaki terim sayısı kullanılan üs haritalamanın hamming ağırlığına göre belirlenebilir. Kullanılan üst

haritalamanın hamming ağırlığına eşit veya küçük sayıdaki hamming ağırlığına sahip terimler, S-kutusunun cebirsel ifadesinde yer alır.

Tablo 5.5. $x \rightarrow x^7$ Üs Haritalama Uygulanarak Elde Edilen S-Kutusu

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	68	C4	22	04	2B	D8	D5	11	4D	AD	8F	AC	9F	E8	81	88
1	CA	4E	7B	6A	FD	7A	5F	F0	C3	09	59	4F	86	87	67	07
2	8B	EC	7C	3E	5D	CC	C7	CE	49	1F	E3	D9	17	7F	B8	01
3	12	96	ED	53	E5	CB	D0	79	63	D4	62	A6	FF	2A	5C	47
4	28	69	1B	39	08	61	FB	4C	41	3C	26	71	EA	77	25	2F
5	21	31	14	78	4A	2E	18	BD	03	2D	C1	DA	00	92	CD	D7
6	65	30	3A	A3	E6	32	A0	05	89	9D	57	C6	98	C5	37	52
7	54	9B	FE	E2	C9	DB	DD	A1	6C	E0	E4	F2	80	19	DE	60
8	58	D1	2C	7E	38	A8	EB	1D	B3	F3	45	97	23	B2	1A	A4
9	15	10	13	D2	3D	7D	3B	02	8D	43	AA	E9	B6	F5	F9	8A
A	16	74	94	82	48	9A	B4	AF	BB	6D	6E	85	0C	A2	CF	E1
B	B9	9C	BA	F4	FA	33	20	3F	F8	6B	0F	BE	B1	06	F7	DF
C	73	5E	29	36	35	76	BC	70	42	D6	46	A9	EE	DC	51	A5
D	8E	56	5B	AE	BF	55	1E	64	AB	C8	83	84	93	0A	A7	E7
E	8C	95	F1	75	C2	1C	B5	4B	99	44	6F	66	24	F6	B0	0B
F	90	B7	91	34	FC	40	50	D3	0E	EF	27	0D	72	C0	9E	5A



Şekil 5.10. $x \rightarrow x^7$ Üs Haritalamalı AES S-kutusu Interpolasyonu

5.5.2 Başarım Analizleri

Bu bölümde Lagrange interpolasyonu algoritmasının karmaşıklığı hesaplanmakta, çalışma zamanı ölçülmekte ve algoritma üzerinde iyileştirmeler yapılmaktadır. Bu iyileştirmeler sonunda ortaya çıkan farklı Lagrange interpolasyonu algoritmaları için, karmaşıklık ve çalışma zamanı ölçütleri bazında kıyaslamalar yapılmıştır.

Kıyaslamalarda kullanılan yöntemlerde Lagrange Normal olarak isimlendirilen Lagrange interpolasyonu algoritması (5.4) bölümünde sunulan Lagrange interpolasyonu algoritmasıdır. Lagrange interpolasyonu algoritmasını incelediğimizde payda kısmı 1'den n'e kadar $(x+x_j)$ ifadelerinin $(i \neq j)$ şartı ile çarpımından oluşmaktadır. $GF(2^n)$' deki tüm elemanlar için bu çarpımı yapmak yerine, önce 1'den n'e kadar tüm $(x+x_j)$ ifadelerini çarpıp bir polinom elde edelim. Daha sonra bu polinomu her nokta $(i=j)$ için, $(x+x_j)$ ifadesini bölerek payı hesaplayalım. Böylelikle $(n-1)^2$ çarpma yerine $(n-1)$ çarpma ve $(n-1)$ bölme işlemi yapılarak Lagrange interpolasyonu gerçeklenebilir. Bu yöntemi Lagrange Paydalı olarak isimlendirdik. Lagrange interpolasyonu algoritmasını incelediğimizde paydayı oluşturan $(i+j)$ ifadelerinin $(i \neq j)$ şartı ile sonlu cisimde çarpımlarının sonucunun $GF(2^n)$' de 1 olmasından yola çıkarak payda çarpımlarının hesaplanması ve payın paydaya bölünmesi işlemini Lagrange interpolasyonu algoritmasından çıkarabiliriz. Bu yöntemi ise Lagrange Paydasız olarak isimlendirdik.

Tablo 5.6. Lagrange İnterpolasyon Yöntemleri Çalışma Zamanı Kontrolü

Yöntem	Zaman		
	Dakika	Saniye	Milisaniye
Lagrange	02	55	838
Lagrange	00	06	552
Lagrange	00	04	56

Tüm bu yöntemler için uygulamalar geliştirilmiş olup, doğru cebirsel ifadeyi oluşturduğu görülmüştür. Bu yöntemlerin çalışma zamanları ölçülerek yapmış olduğumuz kıyaslama Tablo 5.6'da görülmektedir.

Bu üç farklı yöntemin zaman karmaşıklığı *n* cinsinden hesaplanmış olup aşağıda verilmiştir:

- Lagrange Normal: $= (2*(n-1)+n)*n = 3n^2 - 2n$ çarpma
- Lagrange Paydalı : $= n + n*(2n-1)$ çarpma $+ n$ bölme
  $= 2n^2$ çarpma $+ n$ bölme
- Lagrange Paydasız: $= n + n*n$ çarpma $+ n$ bölme
  $= n^2 + n$ çarpma $+ n$ bölme

5.5.3 S-Kutusu Analiz Sonuçları

$x \to x^{254}$ ve $x \to x^{127}$ üs haritalama ve haritalamaların önüne ve arkasına doğrusal dönüşüm konulması ile elde edilen S-kutularının cebirsel ifadelerindeki terim sayısının 9'dan 255'e arttığını gördük.

$x \to x^7$ üs haritalaması kullanılarak tasarlanan S-kutusunun cebirsel ifadesi 93 terimden oluşmaktadır. Bunun nedeni 7 değerinin hamming ağırlığının 3 olmasından dolayı bu üs fonksiyonunun önüne ve arkasına konulan dönüşümler terim sayısını 93'e çıkarabilmektedir.

Klasik Lagrange interpolasyon algoritmasındaki polinom çarpımı sayısında yapılan iyileştirmelerle çalışma zamanında iyileşmeler sağlanmıştır. Lagrange Paydalı olarak isimlendirilen yöntemde, önce *n* eleman birbiriyle bir defaya mahsus çarpılmış olup $(i = j)$ için $(x + x_j)$ ifadesine bölünmüştür. Yapılan çarpma işlemi sayısı azalmasına rağmen $(n-1)$ bölme işlemi eklenmiştir. Buna rağmen çalışma zamanında Tablo 5.6'dan da görüldüğü gibi büyük bir iyileşme sağlanmıştır. Yukarıda ifade edildiği gibi $GF(2^n)$ sonlu cisminde Lagrange interpolasyonu algoritmasında paydadaki tüm terimlerin çarpımının sonucu 1 olacağından, paydanın çıkarılması sonucu yaklaşık olarak 2,5 saniyelik bir iyileşme daha sağlanmıştır.

Tüm bunlara ek olarak yapılan analizlerde sonlu cisimde çarpma işlemlerinin hesaplama maliyeti yüksek olduğu için sonlu cisimde örneğin, $GF(2^8)$' de olası tüm çarpımları önceden başka bir programla hesaplayıp bir tabloda sakladık. Daha sonra

herhangi bir (x, y) çifti için sonlu cisimde çarpma işlemi gerçekleştirmek yerine sonucu, bir referans (lookup) tablosundan getirdik. Böylelikle işlemin gerçekleşme zamanında yarı yarıya kazanç sağladık.

5.6 Cebirsel İfadenin Elde Edilmesi İçin Geliştirilen Yeni Yöntem

Bu bölümde AES S-kutusuna benzer S-kutularının cebirsel ifadesini Lagrange interpolasyonu yönteminden, hesaplama gücü ve zaman karmaşıklığı açısından daha iyi bir maliyetle elde eden yeni bir yöntem geliştirilmiştir.

AES S-kutusunun cebirsel ifadesindeki terim sayısının azlığı, AES şifreleyicisinde interpolasyon saldırılarına karşı bir zaaf oluşturmaktadır. Bu zaaf çeşitli şifreleme doğrusal dönüşümlerin yeri ve sayısı ile giderilmeye çalışılmıştır.

Square [45] ve Shark [93] şifreleyicilerinde, AES şifreleyicisinde olduğu gibi S-kutusu $GF(2^n)$'de ters haritalamayı takip eden bir doğrusal dönüşümden oluşmaktadır. Öte yandan Camelia [88] şifreleyicisinde doğrusal dönüşüm hem üs haritalamadan önce, hem de sonra uygulanmıştır.

AES S-kutusuna benzer S-kutularının tasarımında doğrusal dönüşümün kullanıldığı yerler irdelendiğinde, karşımıza aşağıda listelenmiş olan üç durum çıkmaktadır:

i. İkili doğrusal dönüşümü ters haritalama işleminden sonra kullanmak (durum 1, örnek AES),

ii. İkili doğrusal dönüşümü ters haritalama işleminden önce kullanmak (durum 2),

iii. İkili doğrusal dönüşümleri ters haritalama işleminden hem önce, hem de sonra kullanmak (durum 3, örnek Camellia).

Bu bahsedilen durumlardan durum 2 ve durum 3 ile tasarlanacak S-kutuları, cebirsel ifadesindeki terim sayısı açısından durum 1 ile tasarlanacak S-kutusuna göre önemli bir iyileştirme sunmaktadır [90]. Buna ek olarak uygulanacak doğrusal dönüşümün yeri, diğer kriptografik özellikleri değiştirmemektedir. Bu çalışmada belirtilen durumlara göre tasarlanacak S-kutuları için sonlu cisim teorisinden de

faydalanılarak hızlı, durum 2 ve durum 3 ile tasarlanacak S-kutularının neden terim sayısında iyileştirme yaptığını açıklayıcı bir yöntem geliştirilmiştir. Buna ek olarak sunulan yöntem Lagrange interpolasyonuna alternatif ve daha hızlı bir yöntemdir.

5.6.1 Lagrange İnterpolasyonu İle Doğrusal Dönüşümün Cebirsel İfadesinin Elde Edilmesi

Öncelikle bir uygulama aracılığı ile $GF(2^8)$'de (5.13) ifadesinde görüldüğü gibi matris biçiminde verilen doğrusal dönüşümün (x, y) veri çiftleri elde edilmiştir.

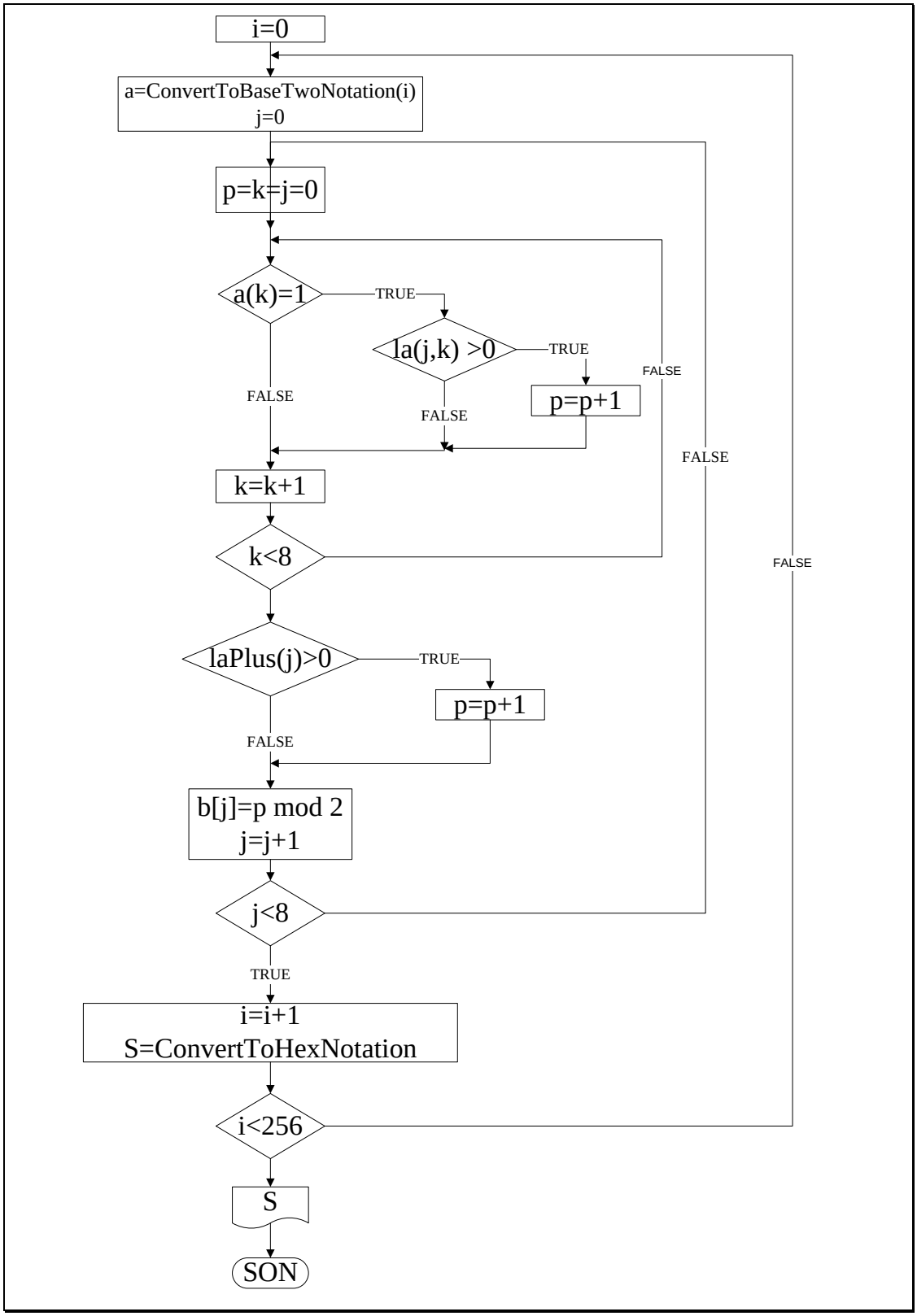
AES S-kutusu biçiminde olan bu veri çiftleri, AES'in indirgenemez polinomu kullanılarak Lagrange interpolasyonuna tabi tutulmuştur ve doğrusal dönüşümün cebirsel ifadesi elde edilmiştir.

Doğrusal dönüşümün cebirsel ifadesini elde etmek üzere geliştirilen uygulamaya, (5.13) ifadesindeki LA_2 doğrusal dönüşümü giriş değeri olarak verildiğinde aşağıda verilen cebirsel ifade elde edilmiştir.

$$\begin{aligned} LA_2(x) = {} & '33' + '52'x + '77'x^2 + '13'x^4 + 'e0'x^8 + \\ & 'fe'x^{16} + '9e'x^{32} + '96'x^{64} + '27'x^{128} \end{aligned} \tag{5.15}$$

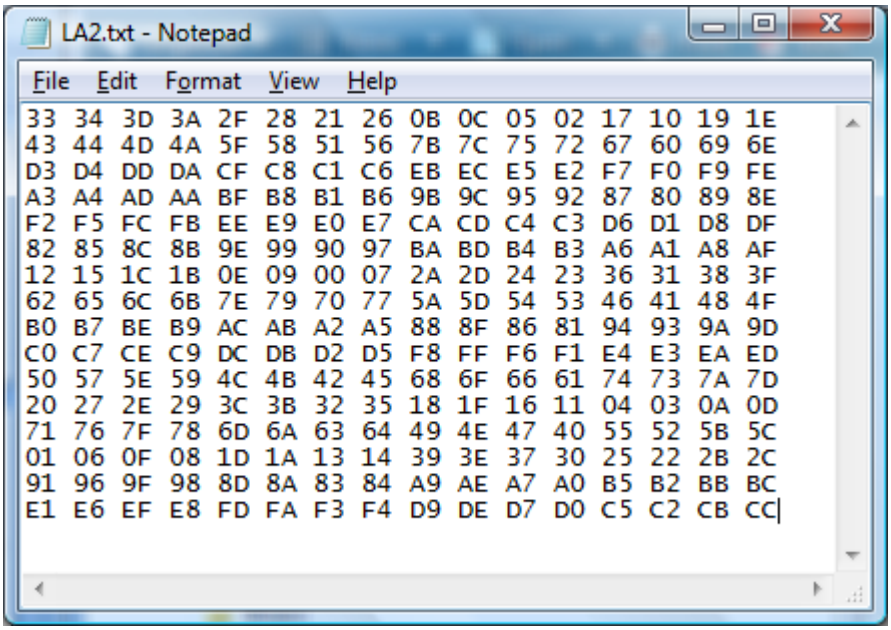
Algoritma 5.6' da akış diyagramı verilen veri çifti elde etme algoritması $GF(2^8)$'de tüm x değerleri için, y çıkış değerlerini hesaplar. $x_0, x_1, \ldots, x_7$ ifadesi, 1 ile 256 arasında değerler alan, x değişkeninin ikilik tabanda gösterimidir.

Akış diyagramındaki ConvertToBaseTwoNotation fonksiyonu ile ikilik tabana çevirme işlemi gerçeklenmiştir. Tüm 256 değer için, elde edilen değerler $x_0, x_1, \ldots, x_7$ ifadesinde yerine konulduktan sonra $GF(2^8)$'de matris çarpımı yapılıp, 8×1'lik matris ile toplanarak ikilik tabanda 8 bitlik bir değer elde edilir (Bu işlemlerde yer alan toplama işlemleri ikilik tabanda olduğundan XOR işlemidir.). Çarpma işleminde birinci matrisin satırları, ikinci matrisin sütunları ile tek tek çarpılır ve aynı satır elemanları için çarpımlardan elde edilen değerler toplanarak, sonuç matrisinin ilgili satır ve sütun değerleri bulunur. Bu algoritmada a(k) ve la(j,k) değerlerinin ikisi de 1 ise, işlem sonucu 1 olacağı için bir p değişkeni bu şart için 1 artırılırak satır sütün çarpımı gerçekleştirilir.



Algoritma 5.6. Doğrusal Dönüşüm Veri Çifti Elde Algoritması

laPlus(j) ile gösterilen son 8×1'lik matris değeri 0'dan büyükse (yani 1 ise) p değişkeni 1 daha artırılır. Yapılan işlemler $GF(2^8)$'de olduğu için, elde edilen değer modül 2 işlemine tabi tutulduktan sonra elde edilen değer, verilen x değeri için elde edilen y çıkış değerinin ikilik gösterimde j. bitidir. Tüm satırlar için, bu işlemi gerçekledikten sonra elde edilen ikilik tabandaki 8 bitlik değer ConvertToHexNotation fonksiyonu ile hexadecimal gösterime çevrilir.



Şekil 5.11. $x \to x^{254}$ Doğrusal Dönüşümün S-Kutusu Biçiminde Veri Tablosu

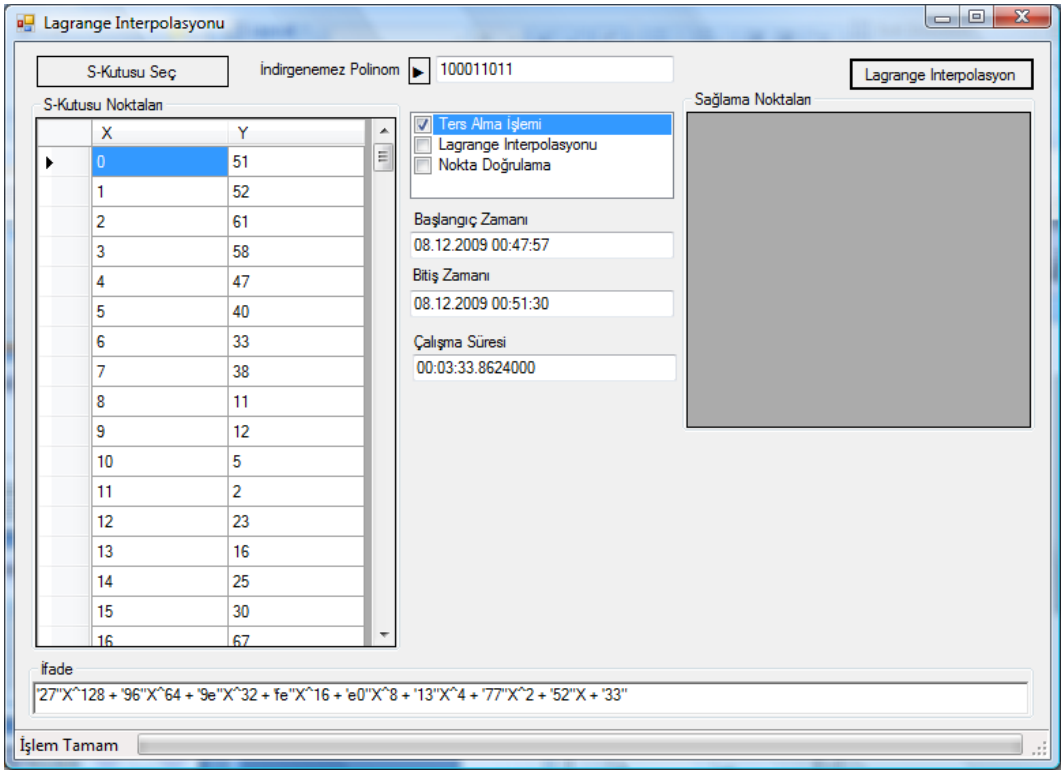
(5.13) ifadesindeki doğrusal dönüşüm için bu algoritma çalıştırıldığında elde edilen veri çiftlerinin S-kutusu biçiminde gösterimi Şekil 5.11'de görülmektedir.

Şekil 5.11'de gösterilen veri tablosu için AES şifreleme algoritmasının indirgenemez polinomu olan $x^8 + x^4 + x^3 + x + 1$ indirgenemez polinomu kullanılarak Lagrange interpolasyonu uygulandığında, (5.15) ifadesinde gösterilen cebirsel ifade elde edilmiştir. Şekil 5.12' de bu uygulamanın ekran görüntüsü görülmektedir.

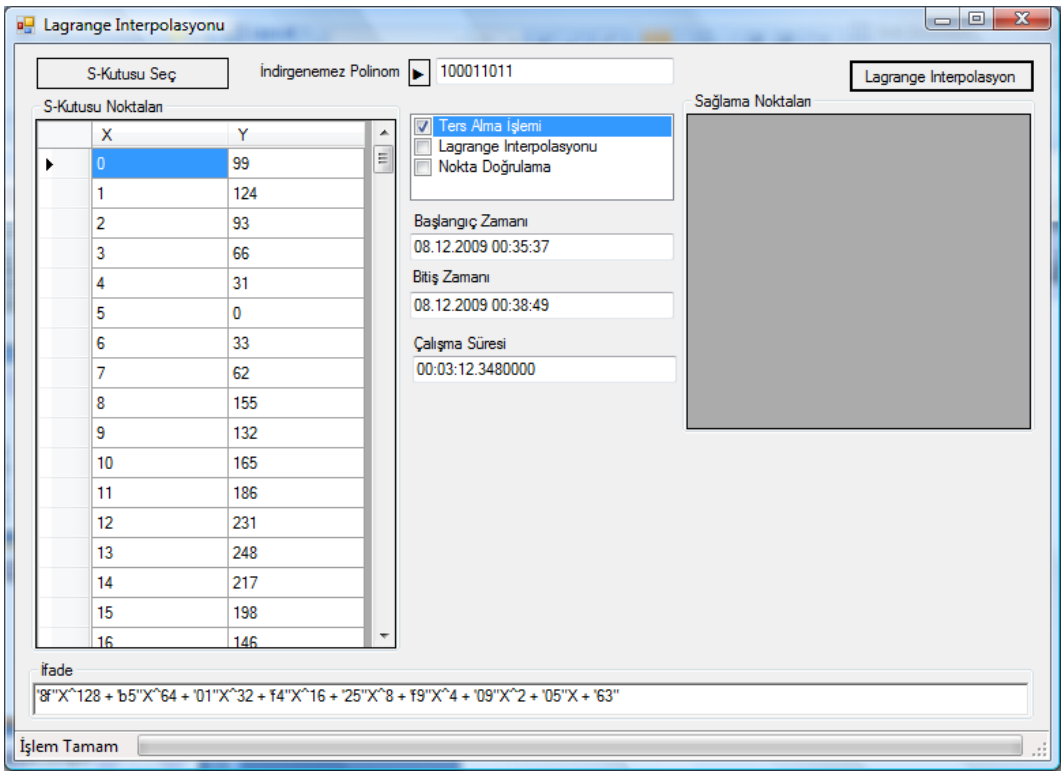
Aynı algoritma AES şifreleyicisinde kullanılan doğrusal dönüşüme de uygulanmış olup

$$A(x) = "63" + "05"\,x + "09"\,x^2 + "f9"\,x^4 + "25"\,x^8 + "f4"\,x^{16} + "01"\,x^{32} + "b5"\,x^{64} + "8f"\,x^{128}$$

cebirsel ifadesi elde edilmiştir. Şekil 5.12'de ekran görüntüsü görülmektedir.



Şekil 5.12. $x \rightarrow x^{254}$ Doğrusal Dönüşümün S-Kutusu Biçiminde Veri Tablosu



Şekil 5.13. AES Şifreleyicisinde Kullanılan Doğrusal Dönüşümün Cebirsel İfadesi

5.6.2 İz (Trace) Fonksiyonları ve Doğrusal Dönüşümün İz Fonksiyonları İle Elde Edilmesi

α, $GF(2^n)$ sonlu cismini üretmek için kulanılan ilkel eleman olmak üzere;

$$b_{n-1}\alpha^{n-1} + b_{n-2}\alpha^{n-2} + ... + b_0,\ b_i \in \{0,1\}$$

sonlu cisim elemanı $(b_{n-1}b_{n-2}...b_0)$ bitlerini içeren hexadecimal sayı olarak temsil edilebilir.

$F = GF(p),\ K = GF(p^n)$ ve $\lambda \in K$ olsun. O zaman alt cisim F 'in λ 'ya göre iz (trace) fonksiyonu

$$Tr_F^K(\lambda) = \lambda + \lambda^p + \lambda^{p^2} + ... + \lambda^{p^{n-1}}$$

şeklinde ifade edilebilir ve karışıklığın olmayacağı durumlarda Tr_F^K ifadesindeki alt ve üst indisler göz ardı edilebilir.

$\{\alpha_0.....\alpha_{n-1}\}$, $GF(2)$ üzerine $GF(2^n)$'in herhangi bir tabanı olmak üzere; $\{\beta_0.....\beta_{n-1}\}$ buna karşı gelen dual taban ve $f(x_0, x_1, ..., x_{n-1}) = (f_0(x), ..., f_{n-1}(x))$ ise $GF(2^n)$ üzerine bir permütasyon olsun. O zaman $g(x) = \sum_{i=0}^{n-1} \alpha_i f_i(x_0, ...x_{n-1})$ ' de $GF(2^n)$ üzerine tam eşleme (bijektif) bir haritadır. $f(x)$'in her çıkış koordinatı, $x = \sum_{i=0}^{n-1} x_i \alpha_i$ olmak üzere, (5.16) ifadesindeki gibi verilebilir [92], [94].

$$f_i(x) = Tr(g(x)\,\beta_i) \tag{5.16}$$

Buna ek olarak (5.16) ifadesinde β_i dual taban değerleri (5.17) ifadesinde gösterildiği gibi hesaplanabilir [92], [94].

$$\beta_i = \sum_{k=0}^{n-1} b_{ki}\alpha_k \tag{5.17}$$

(5.17) ifadesinde $B = \left[b_{ij} \right] = A^{-1}$ ve $A = \left[a_{ij} \right]$ olmak üzere $n \times n$ boyutundaki A matrisi elemanları,

$$a_{ij} = Tr(\alpha_i \alpha_j), \qquad 0 \le i, j \le n-1 \tag{5.18}$$

(5.18) ifadesindeki gibi gösterilebilir. $A = \left[a_{ij} \right]$ şeklindeki A matrisi (5.19) ifadesinde açık biçimde gösterilmiştir.

$$A = \begin{bmatrix} Tr(\alpha_0\alpha_0) & Tr(\alpha_0\alpha_1) & ..... & Tr(\alpha_0\alpha_{n-1}) \\ Tr(\alpha_1\alpha_0) & Tr(\alpha_1\alpha_1) & ..... & Tr(\alpha_1\alpha_{n-1}) \\ & & . & \\ & & . & \\ Tr(\alpha_{n-1}\alpha_0) & Tr(\alpha_{n-1}\alpha_1) & .... & Tr(\alpha_{n-1}\alpha_{n-1}) \end{bmatrix} \tag{5.19}$$

Böylece giriş bitlerine uygulanacak ters haritalama işleminden sonraki çıkış koordinatları ya da giriş bitlerine uygulanacak doğrusal dönüşüm işleminden sonraki çıkış koordinatları (5.20) ifadesi ile gösterilebilir.

$$f_i \ , \quad 0 \le i \le n\text{-}1 \tag{5.20}$$

Doğrusal dönüşüm çıkış bitlerini kullanarak doğrusal dönüşümün cebirsel ifadesi (5.21)' deki gibi elde edilebilir.

$$A(x) = \sum_{i=0}^{n-1} f_i \alpha^i \tag{5.21}$$

Örnek 5.1, yukarıdaki tanım ve matematik alt yapıyı kullanarak, AES S-kutusunun doğrusal dönüşümünün cebirsel ifadesinin elde edilişini göstermektedir.

Örnek 5.1. AES S-kutusunun tasarımında kullanılan doğrusal dönüşümü düşünelim (bknz. (5.12)). Bu doğrusal dönüşüm ikili bir dönüşümdür. $P(x) = x^8 + x^4 + x^3 + x + 1$ indirgenemez polinomu ile oluşturulan sonlu cisimde doğrusal dönüşümün cebirsel ifadesini bulmaya çalışalım. α, $P(x)$ polinomunun bir kökü olsun. $\beta = \alpha + 1$ ise ilkel elemanımız olsun (α, tüm cisim elemanlarını üretmemektedir). O zaman x_i giriş biti değerleri, β değerlerine bağlı olarak yukarıda verilen tanımlara göre, (5.22) ifadesindeki gibi elde edilebilir. Doğrusal matris çıkışı, koordinatlar $f_0, f_1, ..., f_7$ ise (5.23) ifadesindeki gibi elde edilebilir (β üs değerlerinin karşılıkları için Ekler bölümüne bakınız.).

$$
\begin{aligned}
x_0 &= Tr(\beta^{228}x), \quad x_4 = Tr(\beta^{73}x),\\
x_1 &= Tr(\beta^{204}x), \quad x_5 = Tr(\beta^{48}x),\\
x_2 &= Tr(\beta^{179}x), \quad x_6 = Tr(\beta^{23}x),\\
x_3 &= Tr(\beta^{2}x), \qquad x_7 = Tr(\beta^{253}x).
\end{aligned}
\tag{5.22}
$$

$$
\begin{aligned}
f_0 = x_0 + x_4 + x_5 + x_6 + x_7 \quad &= Tr\left(\beta^{228}x\right) + Tr\left(\beta^{73}x\right) + Tr\left(\beta^{48}x\right) + Tr\left(\beta^{23}x\right) + Tr\left(\beta^{253}x\right)\\
&= 29 + A6 + 53 + A4 + 52 = 2A\\
&= Tr\left(\beta^{166}x\right)\\
f_1 = x_0 + x_1 + x_5 + x_6 + x_7 + 1 &= Tr\left(\beta^{228}x\right) + Tr\left(\beta^{204}x\right) + Tr\left(\beta^{48}x\right) + Tr\left(\beta^{23}x\right) + Tr\left(\beta^{253}x\right) + 1\\
&= 29 + B0 + 53 + A4 + 52 = 53\\
&= Tr\left(\beta^{53}x\right) + 1\\
&\vdots\\
f_7 = x_3 + x_4 + x_5 + x_6 + x_7 \quad &= Tr\left(\beta^{2}x\right) + Tr\left(\beta^{73}x\right) + Tr\left(\beta^{48}x\right) + Tr\left(\beta^{23}x\right) + Tr\left(\beta^{253}x\right)\\
&= 05 + A6 + 53 + A4 + 52 = 06\\
&= Tr\left(\beta^{26}x\right)
\end{aligned}
$$

$$
\begin{aligned}
f_0 &= Tr(\beta^{166}x) + 1, \quad f_4 = Tr(\beta^{72}x),\\
f_1 &= Tr(\beta^{53}x) + 1, \quad f_5 = Tr(\beta^{76}x) + 1,\\
f_2 &= Tr(\beta^{36}x), \qquad f_6 = Tr(\beta^{51}x) + 1,\\
f_3 &= Tr(\beta^{11}x), \qquad f_7 = Tr(\beta^{26}x).
\end{aligned}
\tag{5.23}
$$

Doğrusal matris çıkış koordinatlarını kullanarak doğrusal dönüşümün cebirsel ifadesi polinom taban değerlerinin $\{1, \alpha, \alpha^2, ..., \alpha^7\}$ olduğu bilgisinden yola çıkarak (5.21) ifadesinde verildiği gibi,

$$A(X) = f_0 + \alpha f_1 + \alpha^2 f_2 + \alpha^3 f_3 + ... + \alpha^7 f_7$$

şeklinde tekrar yazılabilir. Bunun yanında,

$$
\begin{aligned}
&\alpha = \beta^{25}, \alpha^2 = \beta^{50}, \alpha^3 = \beta^{75}, \alpha^4 = \beta^{100}, \alpha^5 = \beta^{125},\\
&\alpha^6 = \beta^{150}, \alpha^7 = \beta^{175}
\end{aligned}
$$

şeklinde oluşturulan sonlu cisimde verilebileceğinden polinom taban değerleri $A(X)$ ifadesinde yerine konulur ve aşağıdaki ifade elde edilir.

$$Tr\left(\beta^{166}x\right) = \beta^{166}x + (\beta^{166})^2 x^2 + ... + (\beta^{166})^{128} x^{128}$$

$$
\begin{aligned}
A(x) = &\left(\beta^{166}x + (\beta^{166})^2 x^2 + ... + (\beta^{166})^{128} x^{128}\right) \\
&+ \beta^{25}(\beta^{53}x + (\beta^{53})^2 x^2 + ... + (\beta^{53})^{128} x \\
&+ \beta^{50}(\beta^{36}x + (\beta^{36})^2 x^2 + ... + (\beta^{36})^{128} x^{128}) \\
&+ \beta^{75}(\beta^{11}x + (\beta^{11})^2 x^2 + ... + (\beta^{11})^{128} x^{128}) \\
&+ \beta^{100}(\beta^{72}x + (\beta^{72})^2 x^2 + ... + (\beta^{72})^{128} x^{128}) \\
&+ \beta^{125}(\beta^{76}x + (\beta^{76})^2 x^2 + ... + (\beta^{76})^{128} x^{128}) \\
&+ \beta^{150}(\beta^{51}x + (\beta^{51})^2 x^2 + ... + (\beta^{51})^{128} x^{128}) \\
&+ \beta^{175}(\beta^{26}x + (\beta^{26})^2 x^2 + ... + (\beta^{26})^{128} x^{128}) \\
&+ '63'.
\end{aligned}
$$

$A(x)$ ifadesindeki x teriminin katsayısı A_0,

$$\beta^{166}, \beta^{(53+25) \bmod 255}, \beta^{(50+36) \bmod 255}, \beta^{(75+11) \bmod 255}, \beta^{(100+72) \bmod 255},$$
$$\beta^{(125+76) \bmod 255}, \beta^{(150+51) \bmod 255}, \beta^{(175+26) \bmod 255}$$

değerlerinin toplamı şeklinde ifade edilebileceğinden yola çıkarak,

$$A_0 = \beta^{166} + \beta^{78} + \beta^{86} + \beta^{86} + \beta^{172} + \beta^{201} + \beta^{201} + \beta^{201},$$

$$A_0 = "2A" + "78" + "DC" + "DC" + "7A" + "2D" + "2D" + "2D",$$

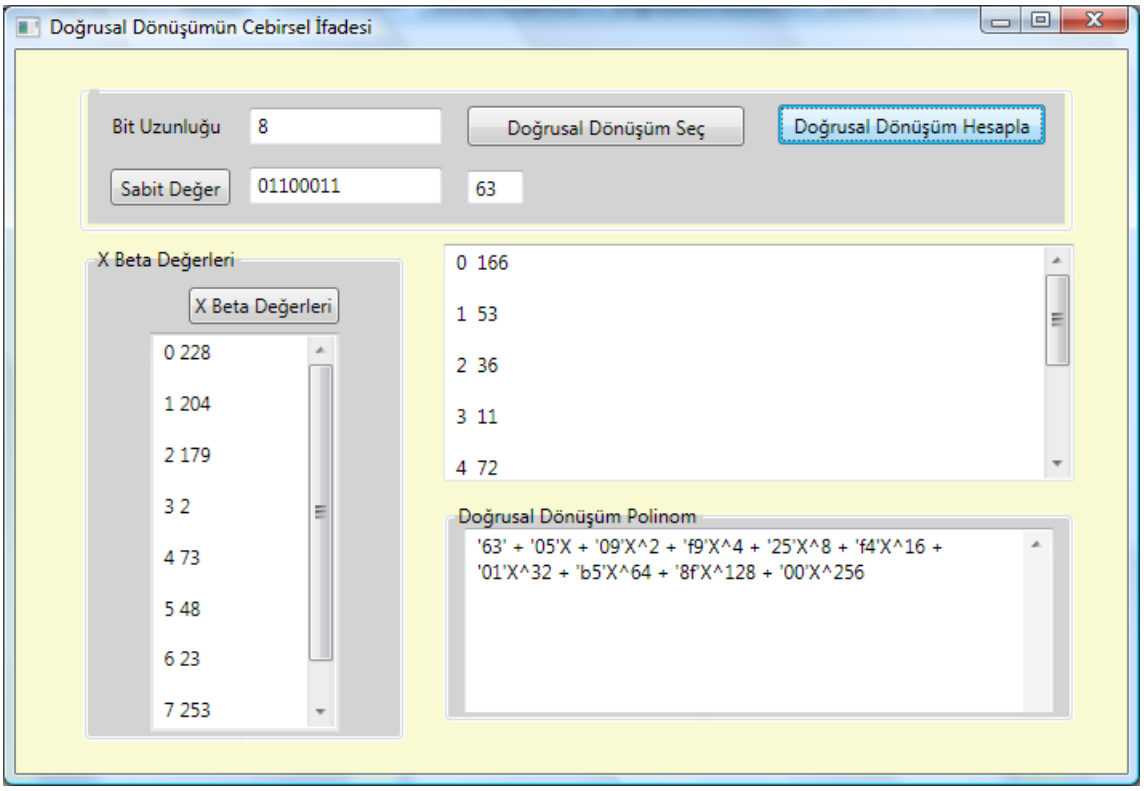
$$A_0 = "05"$$

şeklinde elde edebiliriz. Diğer terimlerin katsayıları da, aynı şekilde elde edildikten sonra sonuçlanan cebirsel ifade aşağıdaki gibidir.

$$A(x) = "63" + "05"x + "09"x^2 + "f9"x^4 + "25"x^8 + "f4"x^{16} + "01"x^{32} + "b5"x^{64} + "8f"x^{128}.$$

Doğrusal dönüşümün cebirsel ifadesini iz fonksiyonları ile elde etmek için geliştirilen uygulamada, (5.12) ifadesindeki gibi matris biçimindeki doğrusal dönüşüm, giriş değeri olarak alınır. Şekil 5.14' teki ekran görüntüsündeki "Bit Uzunluğu" sahasının değerinin 8 olarak verilmesi, işlemlerin $GF(2^8)$ sonlu cisminde gerçekleştirileceğini gösterir. $x_0, x_1, \ldots, x_7$ giriş değerlerinin iz fonksiyonu olarak değerleri, giriş değeri olarak algoritmaya verilir. Ekran görüntüsündeki "0 228" giriş değeri, $x_0 = Tr(\beta^{228})$ ifadesine denktir. Buradaki β_i dual taban değerleri Şekil 5.14'de ekran görüntüsü görülen uygulama aracılığı ile (5.17), (5.18), (5.19) ifadeleri kullanılarak hesaplanmıştır. Bu uygulamada öncelikle α üs değerleri bulunur. Daha sonra (5.19) ifadesindeki A matrisi iz fonksiyonlarının hesaplanması ile elde edilir. İz

fonksiyonları içerdikleri α üs değerlerinin ikili taban değerlerinin birbirleriyle toplanması ile elde edilir. Hesaplanan A matrisinin $GF(2)$'de tersi alınarak, β_i dual taban değerleri hesaplanır.



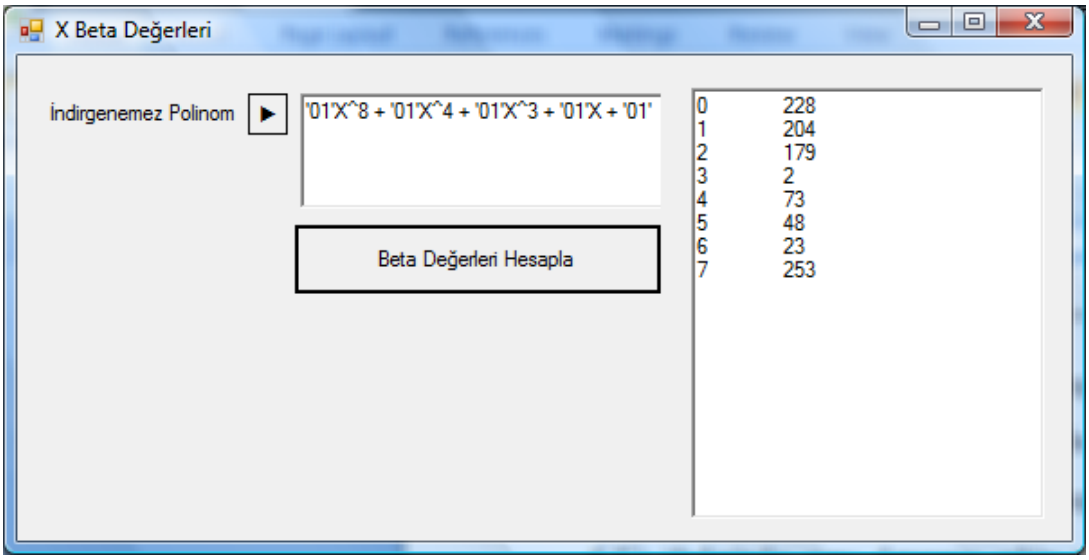
Şekil 5.14. Doğrusal Dönüşümün Cebirsel İfadesinin Trace Fonksiyonları İle Bulunması

Öncelikle $\beta = \alpha + 1$ ifadesi için, AES şifreleyicisinin indirgenemez polinomu olan $x^8 + x^4 + x^3 + x + 1$ kullanılarak, β üs değerleri hesaplanır. $f_0, f_1, ..., f_7$ terimleri ise bu terimlerin kullanılan doğrusal dönüşüm ile matris çarpımı sonucunda $x_0, x_1, \ldots, x_7$ terimleri cinsinden elde edilir. $x_0, x_1, \ldots, x_7$ terimlerinin karşılığı olan iz fonksiyonları $f_0, f_1, ..., f_7$ terimlerinin ifadelerinde yerlerine konur. İz fonksiyonlarının toplanarak, yani β üs değerleri birbirleriyle toplanarak elde edilen yeni iz fonksiyonu $f_0, f_1, ..., f_7$ terimlerinin karşılığıdır. Geliştirilen uygulamada bu toplama işlemi, β üs değerlerinin hexadecimal değeri elde edilip, birbiriyle toplandıktan sonra elde edilen toplam hex değerine karşılık gelen β üs değeri elde edilerek bulunur. β üs değerlerinin hex karşılığı, $\beta = \alpha + 1$ olarak ifade edilmesi, üs değerinin alınması ve AES indirgenemez polinomu $\alpha^8 + \alpha^4 + \alpha^3 + \alpha + 1$ indirgendikten sonra elde edilen

ifade içinde yer alan terimlerin, derecelerine göre 8 bit olarak ifade edilir. İfadede yer alan terim derecesi için 1, yer almayan terimler için 0 değeri verilir. Örneğin $\alpha+1=00000011$. Elde edilen bu değerler XOR işlemi ile toplanarak elde edilen değerin, β üs değeri bulunarak toplama işlemi sonlandırılır. $A(X)$ ile ifade edilen doğrusal dönüşüm, elde edilen $f_0, f_1,..., f_7$ iz fonksiyonlarının açılımı yapıldıktan sonra, aşağıdaki ifadede yerine koyularak elde edilir.

$$A(X) = f_0 + \alpha f_1 + \alpha^2 f_2 + \alpha^3 f_3 + ... + \alpha^7 f_7$$

Bu ifadedeki α değerinin β cinsinden karşılığı bulunur. İkili gösterimde $\alpha = 00000010$ olarak ifade edilebilir. Algoritmanın bir önceki adımında, tüm β üs değerleri bulunarak bir veri tablosunda saklanmıştı. (00000010) ikili değerine karşılık gelen β üs değeri (β^{25}) tablodan getirilir (bknz. Ekler). $A(X)$ ifadesinde aynı dereceye sahip terimlerin katsayıları toplanarak, derecelere göre indislenen bir tabloda saklanır. Daha sonra bu tablodaki verilere göre doğrusal dönüşüm polinomu Şekil 5.14'de gösterildiği gibi elde edilir.



Şekil 5.15. β_i Dual Taban Değerlerinin Bulunması

5.6.3 S-Kutusunun Cebirsel İfadesinin Elde Edilmesi İçin Yeni Yöntem

Tanım 5.1. $L(x)=\sum_{i=0}^{t}\beta_i x^{2^i}$ şeklinde ve $\beta_i, GF(2^n)$'in elemanı olmak üzere verilen özel biçime sahip polinoma, $GF(2^n)$ üzerine doğrusallaştırılmış polinom denir.

Tanım 5.2. Bir tamsayı d'yi içeren mod N'e göre cyclotomic koset,

$$C_d=\{d,dp,....,dp^{n-1}\} \pmod N \tag{5.24}$$

şeklinde bir kümedir ve d, $dp^n \equiv d \pmod N$ olacak şekilde en küçük tamsayıdır.

Önerme 5.1. $A, GF(2^n)$ üzerine doğrusal bir haritalama olsun. O zaman $A(x),\ x\in GF(2^n)$ olmak üzere $GF(2^n)$ üzerine doğrusallaştırılmış bir polinoma dayalı olarak,

$$A(x)=\sum_{i=0}^{n-1}\beta_i x^{2^i} \tag{5.25}$$

şeklinde ifade edilebilir.

Önerme 5.2. F_2^n üzerine tersi alınabilir doğrusal dönüşümlerin kümesi ile $GF(2^n)$ üzerine doğrusallaştırılmış polinomların kümesi arasında birebir ilişki vardır [87] .

Önerme 5.3. $GF(2^n)$'in bir fonksiyonu $F(x)=x^d$ olsun ve bu fonksiyon $f(x_1,...,x_n)=(f_1(x),...,f_n(x))$ boole fonksiyonuna karşılık gelsin. O zaman, $f(x_1,...,x_n)$'nin çıkış koordinatlarına uygulanan doğrusal bir dönüşüm ile elde edilen boole fonksiyonuna karşılık gelen $G(x)$, aşağıdaki şekilde ifade edilir.

$$G(x)=\sum_{i=0}^{n-1}b_i x^{d2^i},\quad c_i\in GF(2^n) \tag{5.26}$$

Eğer doğrusal dönüşüm affine bir dönüşüm ile yer değiştirilirse, o zaman $G(x)$,

$$G(x)=\sum_{i=0}^{n-1}b_i x^{d2^i}+c_n,\quad c_i\in GF(2^n) \tag{5.27}$$

şeklinde yazılabilir [87]. Gerçekte yukarıdaki tanım ve teoriler ışığı altında bir üs haritalamadan sonra uygulanacak doğrusal dönüşüm sonucunda, cebirsel ifadede üs

fonksiyonu d 'nin cyclotomic cosetinde bulunan terimler cebirsel ifade de yer alacaktır diyebiliriz.

Örnek 5.2. AES S-kutusunun cebirsel ifadesi, Örnek 5.1'de elde edilen doğrusal dönüşümün cebirsel ifadesinde x yerine x^{254} koyarak ve $x \in GF(2^n)$ için $x^a = x^{a \bmod 2^n - 1}$ olduğundan yola çıkarak,

$$S(x) = "63" + "05"\, x^{254} + "09"\, x^{254\times 2} + "f9"\, x^{254\times 4} + "25"\, x^{254\times 8} + "f4"\, x^{254\times 16} + "01"\, x^{254\times 32} + "b5"\, x^{254\times 64} + "8f"\, x^{254\times 128},$$

$$S(x) = "63" + 05"\, x^{254} + "09"\, x^{253} + "f9"\, x^{251} + "25"\, x^{247} + "f4"\, x^{239} + "01"\, x^{223} + "b5"\, x^{191} + "8f"\, x^{127}$$

şeklinde elde edilebilir.

Teorem 5.1. $GF(2^n)$'in bir fonksiyonu $F(x) = x^d$ olsun ve bu fonksiyon $f(x_1,...,x_n) = (f_1(x),...,f_n(x))$ boole fonksiyonuna karşılık gelsin. $G(x)$ ise $f(x_1,...,x_n)$ sabitlenirken $x_1,....,x_n$ giriş bitlerine doğrusal bir dönüşüm uygulanarak elde edilen bir boole haritasına karşılık gelen bir fonksiyon olsun. Bu durumda $G(x)$,

$$G(x) = \sum_{i=0}^{2^n-1} b_i x^i \qquad wt(i) > wt(d) \text{ için } b_i = 0 \tag{5.28}$$

şeklinde ifade edilir [95].

Teorem 5.1, doğrusal dönüşümün üs fonksiyonunun önüne uygulandığında oluşacak cebirsel ifadenin terimleri üzerindeki etkisini göstermektedir ve d üs fonksiyonunun Hamming ağırlığına eşit ve küçük Hamming ağırlığına sahip terimlerin cebirsel ifade de ortaya çıkacağını göstermektedir. Kısacası, Teorem 5.1 durum 2 için cebirsel ifadedeki terim sayısı hakkında bilgi vermektedir.

Şekil 5.16, S-kutusu tasarımında doğrusal dönüşümün yeri ile ilgili olarak olası durumların gösterimini yapmaktadır. Bu şekile dayalı olarak Durum 1, 2 ve 3 için cebirsel ifadenin hesaplanması aşağıdaki gibi özetlenebilir:

Durum 1 için;

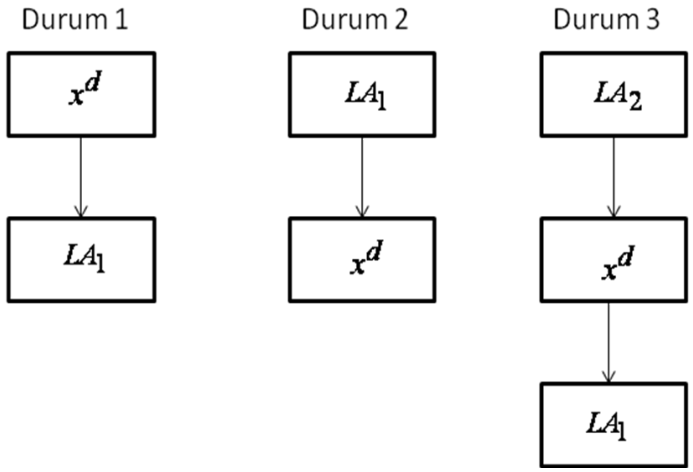
- Verilen teoriyi kullanarak, LA_1 doğrusal dönüşümüne karşılık gelen ve bu dönüşümün cebirsel ifadesi olan $LA_1(x)$'i hesapla,
- $LA_1(x)$'te x yerine x^d koyarak, S-kutusunun cebirsel ifadesini elde et.

Durum 2 için;

- Verilen teoriyi kullanarak LA_1 doğrusal dönüşümüne karşılık gelen ve bu dönüşümün cebirsel ifadesi olan $LA_1(x)$'i hesapla,
- x^d'de x yerine, $LA_1(x)$'i koy,
- S-kutusunun cebirsel ifadesini, $(LA_1(x))^d$ ifadesini hesaplayarak elde et.

Durum 3 için;

- Verilen teoriyi kullanarak, LA_1 ve LA_2 doğrusal dönüşümlerine karşılık gelen ve bu dönüşümlerin cebirsel ifadesi olan $LA_1(x)$ ve $LA_2(x)$'i hesapla,
- $LA_1(x^d)$'de x yerine $LA_2(x)$'i koy,
- S-kutusunun cebirsel ifadesini $LA_1(LA_2(x)^d)$ ifadesini hesaplayarak elde et.



Şekil 5.16. AES S-kutusu Benzeri S-kutuları Tasarımında Olası Durumların Gösterimi

Örnek 5.3. $n=8$ ve $GF(2^8)$, AES tanımlamalarında olduğu gibi $P(x)=x^8+x^4+x^3+x+1$ indirgenemez polinomu ile tanımlanmış olsun. Buna ek olarak LA_1 ve LA_2 doğrusal dönüşümlerinin cebirsel ifadesi aşağıdaki gibi olsun.

$$LA_1(x) = "63" + "05"\,x + "09"\,x^2 + "f9"\,x^4 + "25"\,x^8 + "f4"\,x^{16} + "01"\,x^{32} + "b5"\,x^{64} + "8f"\,x^{128}$$

$$LA_2(x) = "33" + "52"\,x + "77"\,x^2 + "13"\,x^4 + "e0"\,x^8 + "fe"\,x^{16} + "9e"\,x^{32} + "96"\,x^{64} + "27"\,x^{128}$$

S-kutusu $x \to x^{254}$ haritalaması ile birlikte durum 2'de olduğu gibi tasarlanırsa, cebirsel ifadesi aşağıdaki gibi elde edilebilir:

$$S(x) = (LA_1(x))^{254},$$

$$S(x) = \left("63" + "05"\,x + "09"\,x^2 + \cdots + "b5"\,x^{64} + "8f"\,x^{128}\right)^{254}.$$

S-kutusu $x \to x^{254}$ haritalaması ile birlikte durum 3'te olduğu gibi tasarlanırsa cebirsel ifadesi aşağıdaki gibi elde edilebilir:

$$\begin{aligned} S(x) = {} & "63" + 05"(LA_2(x))^{254} + "09"(LA_2(x))^{253} + "f9"(LA_2(x))^{251} + \\ & "25"(LA_2(x))^{247} + "f4"(LA_2(x))^{239} + "01"(LA_2(x))^{223} + "b5"(LA_2(x))^{191} + \\ & "8f"(LA_2(x))^{127}, \end{aligned}$$

$$\begin{aligned} S(x) = {} & 05"("33" + "52"\,x + "72"\,x^2 + \ldots + "9e"\,x^{32} + "96"\,x^{64} + "27"\,x^{128})^{254} + \\ & "09"("33" + "52"\,x + "72"\,x^2 + \ldots + "9e"\,x^{32} + "96"\,x^{64} + "27"\,x^{128})^{253} + \\ & "f9"("33" + "52"\,x + "72"\,x^2 + \ldots + "9e"\,x^{32} + "96"\,x^{64} + "27"\,x^{128})^{251} + \\ & . \\ & . \\ & "8f"("33" + "52"\,x + "72"\,x^2 + \ldots + "9e"\,x^{32} + "96"\,x^{64} + "27"\,x^{128})^{127} + "63". \end{aligned}$$

Tablo 5.7' de verilen S-kutusu [92] $x \to x^{254}$ üs haritalaması ve Örnek 5.13'de verilen LA_1 ve LA_2 doğrusal dönüşümleri ile birlikte durum 3'e göre tasarlanmıştır. Cebirsel ifadesi 255 terim içermekte olup, aşağıda kısaca gösterilmiştir:

$$S(x) = "1c"\,x^{254} + "1e"\,x^{253} + "16"\,x^{252} + "98"\,x^{251} + \ldots + "87"\,x^2 + "e7"\,x + "c3".$$

Gerek durum 2' de, gerekse durum 3'te cebirsel ifadenin elde edilmesinde gerekli olan 9 terimli doğrusal dönüşümün 254. kuvvetini almak, yüksek iş yükü olarak görülebilir.

Tablo 5.7. Örnekte verilen durum 3'e göre tasarlanan S-kutusu

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	C3	18	27	80	15	34	FD	F7	2B	FE	6B	77	F0	CA	D4	72
1	1A	1B	E3	D6	CF	6A	D1	B1	21	10	9D	40	85	D0	F9	9F
2	66	48	C1	57	8A	E8	78	B4	E9	CE	D9	98	68	8C	99	BB
3	0A	49	95	AC	08	6C	C8	4E	14	DE	2A	4F	17	CD	A7	19
4	89	E6	B0	0F	28	1E	E1	94	74	BD	1C	2E	F6	3E	61	9E
5	13	97	64	3D	0B	EE	60	88	F4	7A	8D	6D	24	32	C2	79
6	C9	59	9C	AF	AB	01	63	C5	E5	D8	36	26	05	C7	07	75
7	AA	4D	50	7F	F3	B6	51	F5	BE	4C	20	ED	5A	83	52	84
8	E7	A9	AE	56	91	62	3A	06	C4	73	44	0C	22	DC	B8	5E
9	BA	C6	8B	DD	86	B9	B5	03	41	16	42	A1	69	11	87	55
A	53	5B	58	CB	29	B3	2C	6E	45	A8	33	EF	92	8F	DA	FF
B	B7	CC	31	A5	EB	E2	23	96	AD	C0	47	82	F2	7B	67	D7
C	A3	38	D2	BC	3C	02	FB	43	3B	2F	A0	09	FC	00	39	4A
D	7C	6F	76	30	A4	A2	7D	FA	12	B2	9A	04	3F	93	F1	71
E	81	90	DB	46	5D	7E	EC	5F	D3	E4	5C	E0	D5	37	EA	65
F	F8	8E	DF	9B	54	2D	0D	BF	35	1D	0E	70	A6	25	1F	4B

Bunun için biz hızlı bir hesaplama yöntemi geliştirdik ve bu yöntem tüm $x \to x^{254}$ haritalama tabanlı tasarlanacak S-kutuları için uygulanabilir. Bu hesaplama yöntemi aşağıdaki gibi verilebilir (Algoritma 1):

- İlk olarak $LA_2(x)^2, LA_2(x)^4, LA_2(x)^8, LA_2(x)^{16}, LA_2(x)^{32}, LA_2(x)^{64}$ ifadelerini çarpma ve kare alma işlemlerini kullanarak elde et. Bu işlem 6 polinom çarpma işlemine denk düşer.
- $LA_2(x)^{127}$ ifadesini yukarıdaki ifadeleri çarparak elde et. Bu işlem de 5 polinom çarpma işlemine denk düşer.

$$LA_2(x)^{191} = LA_2(x)^{127}.LA_2(x)^{64},$$

$$LA_2(x)^{223} = LA_2(x)^{191}.LA_2(x)^{32},$$

$$LA_2(x)^{239} = LA_2(x)^{223}.LA_2(x)^{16},$$

$$LA_2(x)^{247} = LA_2(x)^{239}.LA_2(x)^{8},$$

$$LA_2(x)^{251} = LA_2(x)^{247}.LA_2(x)^{4},$$

$LA_2(x)^{253} = LA_2(x)^{251}.LA_2(x)^2$,

$LA_2(x)^{254} = LA_2(x)^{253}.LA_2(x)^1$ ifadelerini çarparak elde et. Bu işlemlerde 7 çarpma işlemi eder ve sonuç olarak 18 polinom çarpma işlemi ile istenen tüm doğrusal dönüşümlerin üsleri elde edilir (Algoritma 1).

Durum 2 için ise hesaplama yöntemi aşağıdaki gibi verilebilir:

- İlk olarak $LA_1(x)^2, LA_1(x)^4, LA_1(x)^8, LA_1(x)^{16}, LA_1(x)^{32}, LA_1(x)^{64}, LA_1(x)^{128}$ ifadelerini çarpma ve kare alma işlemlerini kullanarak elde et. Bu işlem 7 polinom çarpma işlemine denk düşer.
- $LA_1(x)^{254}$ ifadesini yukarıdaki ifadeleri çarparak elde et. Bu işlem de 6 polinom çarpma işlemine denk düşer. Bu işlemlerde, toplam 13 polinom çarpması ile sonuçlanır.

Algoritma 5.7'de, algoritmik gösterimi bulunan cebirsel ifadenin elde edilmesi üç bölümden oluşmaktadır:

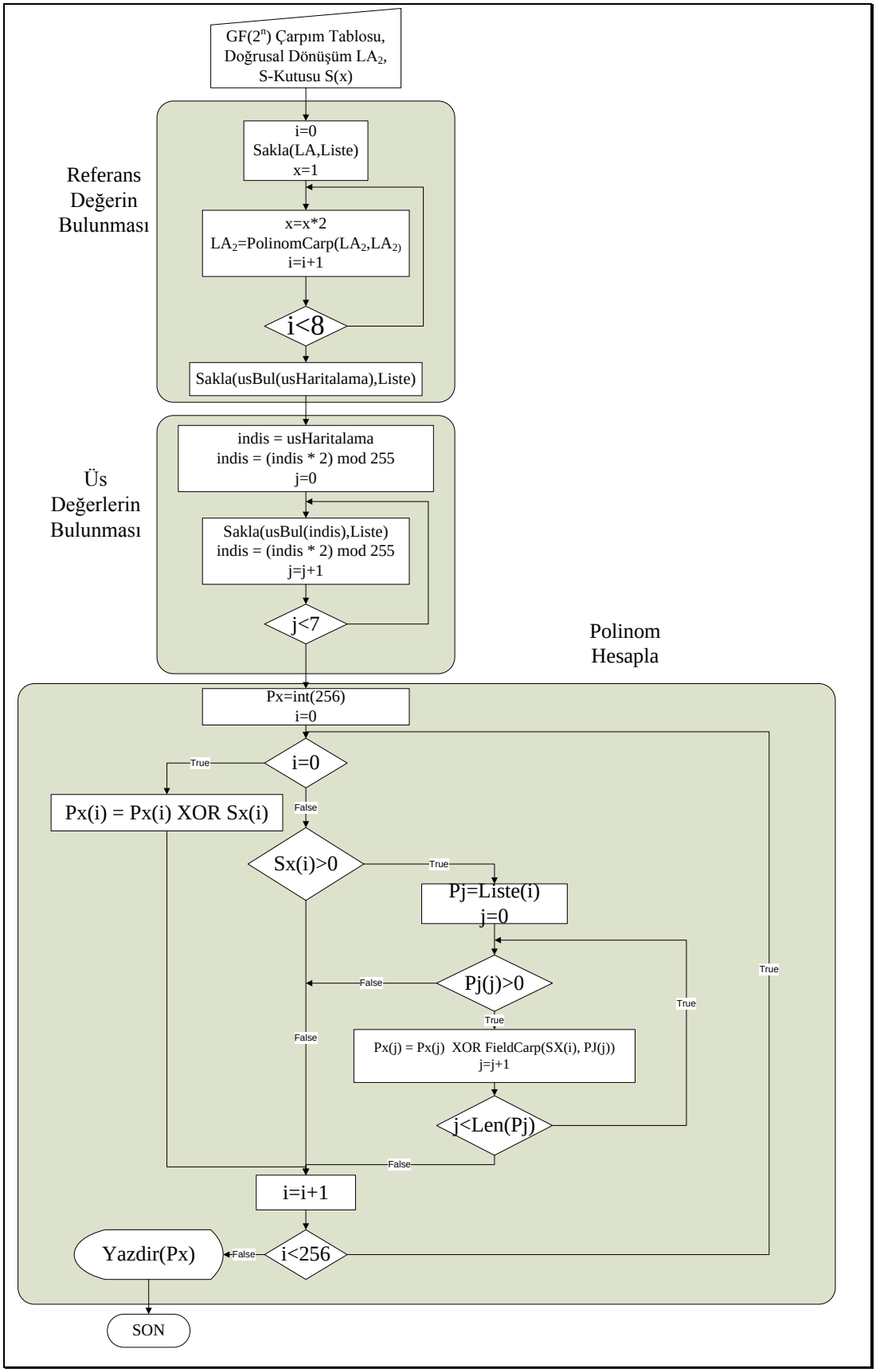
i. Referans Değerin Bulunması

ii. Üs Değerlerin Bulunması

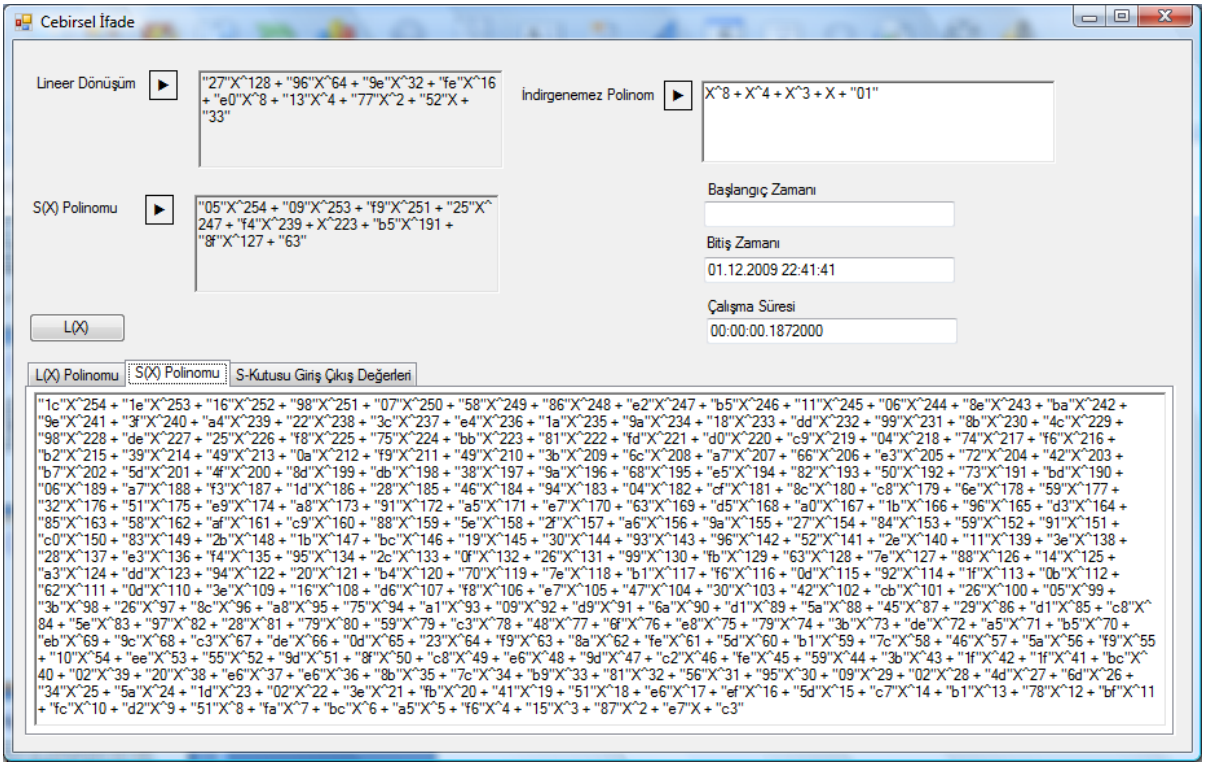
iii. Polinom Hesaplama

AES şifreleyicisinin S-kutusunda $x \to x^{254}$ üs haritalaması kullanılmaktadır. AES S-kutusunun cebirsel ifadesi AES S-kutusunda kullanılan doğrusal dönüşümde x yerine x^{254} koyarak elde edilebilir. Doğrusal dönüşümün cebirsel ifadesinde yer alan terimlerin derecesi 2'nin üsleri olduğuna göre, referans değerin bulunmasından $\left(x^{254}\right)$ sonra bu değerin modül 255 ile ardışık kareleri alınarak diğer terimler elde edilebilir.

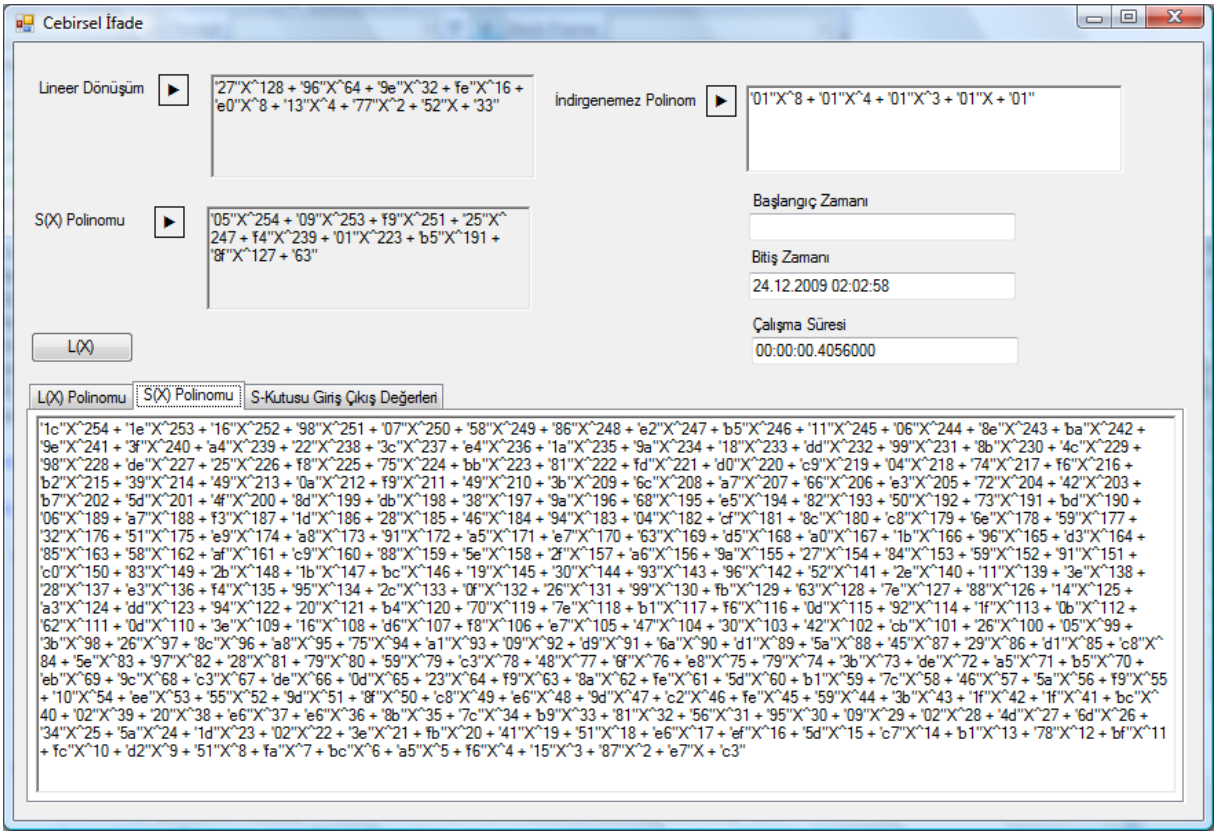
Referans değerin bulunması kısmında öncelikle 2'nin 255'den küçük olan tüm üs değerleri için polinomun üsleri bulunur ve sonra kullanılmak üzere bir listede saklanır. Daha sonra yukarıda ayrıntısı verilen hızlı üs bulma (fast exponentiation) algoritması ile $\left(x^{254}\right)$ üs değeri bulunur. Bu algoritmada polinomlar tamsayı olarak bir dizide, polinomda varolan terim dereceleri dizinin indisleri katsayılarıda bu indisdeki değerler olacak şekilde tutulur. Polinomda yer almayan terimlerin katsayıları sıfırdır.



Algoritma 5.7. Cebirsel İfade Akış Diyagramı



Şekil 5.17. Cebirsel İfade Ekran Görüntüsü- Çalışma Zamanı (Algoritma 1)



Şekil 5.18. Cebirsel İfade Ekran Görüntüsü- Çalışma Zamanı (Algoritma 2)

"Usbul" fonksiyonu bulunacak olan üs değerinin ikilik gösterimini bulur ve bit değerleri bir olan indisler için 2^{indis} üs değeri bulunur ve polinomun bu derecesi daha önce saklanan listeden getirilip, birbirleriyle polinom çarpma işlemine tabi tutularak polinom üs değeri bulunur. Bu algoritma hızlı üs bulma (fast exponentiation) olarak isimlendirilir.

Üs değerlerin bulunması kısmında referans polinom değerinin ardışık kareleri alınarak bir sonraki terim elde edilir. Bu algoritmadaki tüm çarpma ve kare alma işlemlerinde $x^{255}=1$ indirgemesi yapılır.

Polinom hesaplama kısmında elde edilen üs değerleri verilen $s(x)$ polinomunda yerine konularak cebirsel ifade elde edilir. Bu işlem sırasında $s(x)$ polinomundaki katsayılar ile elde edilen üs değer polinomlarının katsayılarının sonlu cisimde çarpma işlemine tabi tutulması ve oluşan aynı derecedeki terimlerin sonlu cisimde toplama işlemi (XOR) ile birbirleriyle toplanması gereklidir. Buradaki FieldCarp işlemi sonlu cisimde çarpma işlemini gerçekleştirmektedir.

Algoritma 2 – Fast Exponentiation Algoritması

- Bu algoritmada her bir üssün 8 bit olarak gösterimi yapılır. Örneğin 127 = (01111111).
- Daha sonra 1 olan bit indisleri için polinomun üsleri bulunur ve birbirleriyle çarpılarak üs değerleri hesaplanır.

$$(LA_2(x))^2 = LA_2(x)\times LA_2(x)$$
$$(LA_2(x))^4 = \left((LA_2(x))^2\right)^2$$
$$(LA_2(x))^8 = \left((LA_2(x))^4\right)^2$$
$$(LA_2(x))^{16} = \left((LA_2(x))^8\right)^2$$
$$(LA_2(x))^{32} = \left((LA_2(x))^{16}\right)^2$$
$$(LA_2(x))^{64} = \left((LA_2(x))^{32}\right)^2$$
$$(LA_2(x))^{127} = LA_2(x)\times(LA_2(x))^2\times(LA_2(x))^4\times(LA_2(x))^8\times (LA_2(x))^{16}\times(LA_2(x))^{32}\times(LA_2(x))^{64}$$

Algoritma 1 (bknz. Şekil 5.17) için, 187 milisaniyede işlem gerçekleşirken, Algoritma 2 (bknz. Şekil 5.18) için ise, 405 milisaniyede işlem gerçekleşmektedir.

Şekilllerden de görüleceği gibi, her iki algoritma ile de aynı polinom değeri elde edilmiştir.

Algoritma 2'deki hızlı üs bulma algoritmasında üsler bulunurken 7 polinom çarpma yapılır. Bu 7 polinom çarpma yukarıdan da görüldüğü gibi ardışık kare alma işlemlerinden ibarettir. Algoritmada üssün ikili gösterimde bit indis değerleri 1 olan kuvvetler birbiriyle çarpıldığı için, 254 üs değeri için yukarıda elde edilen 2,4,8,16,32,64,128. kuvvetler birbirleriyle çarpılır (6 polinom çarpma). Böylelikle çarpma işlemi sayısı 64516' dan 13'e indirilmiştir.

Çalışmamızda AES S-kutusu benzeri S-kutularının cebirsel ifadesinin hesaplanması için hızlı ve cebirsel ifadelerinde bulundurdukları terim sayılarını, tasarlandıkları üs haritalamasına göre gösteren bir yöntem geliştirilmiştir. Örneğin Tablo 5.3'de verilen ve $x \to x^{254}$ haritalaması tabanlı S-kutusunun cebirsel ifadesi 255 terim içermekte ve cebirsel ifadesinin derecesi 254'tür. Bu cebirsel ifade geliştirilen yöntemle 40 milisaniye civarında bir sürede 2 GHz işlemcili bir bilgisayar ile elde edilmiştir. Buna ek olarak diğer üslerle tasarlanacak S-kutuları için benzeri hesaplama yöntemleri geliştirilebilir ve görünürde ki hesaplama ile ilgili yüksek iş yükü çok makul seviyelere indirilebilir. Verilen hesaplama yönteminde, sonlu cisimde çarpma işlemleri çok hızlı olarak gerçeklenebileceği için polinomsal çarpma işlemi üzerinde durulmuştur. Diğer yandan S-kutusu durum 2 ya da durum 3 ile $x \to x^{7}$ haritalama yöntemi ile tasarlanmış olsaydı, cebirsel ifadesinin 93 terim içereceği ve derecesinin de 224 olacağı söylenebilirdi. Nitekim bu tip bir S-kutusu için de bu uygulanmış ve bahsedilen sonuçlar gözlenmiştir.

SONUÇLAR

Bu tezde, blok şifreleyiciler üzerinde önemli bir saldırı olan diferansiyel kriptanaliz ve gelişmiş türevleri kullanılarak, blok şifreleyicilerin gücü incelenmiş ve günümüzdeki en önemli blok şifreleyici olan AES şifreleyicisi temel alınarak, blok şifreleyicilerin cebirsel yapıları incelenmiş, interpolasyon saldırılarına karşı cebirsel blok şifreleyicilerinin zaafları belirlenmiş ve iyileştirme önerileri sunulmuştur.

Diferansiyel kriptanaliz, SPN şifreleyicisinin anahtarlarını geniş anahtar aramadan daha az zaman karmaşıklığında elde edebilmektedir. Dolayısıyla, blok şifreleyicilerin gücünün ölçülmesinde önemli bir tekniktir.

Bu çalışmada diferansiyel kriptanalizi dört, beş ve altı döngülük SPN şifreleyicisi üzerinde denedik. Dört döngülük ve beş döngülük SPN şifreleyicisinde, hedef anahtarın bir kısmını elde ederken, altı döngülük SPN şifreleyicisinde hedef anahtarı elde edemedik.

Diferansiyel kriptanaliz n döngülü bir SPN için, $n-1$ döngü diferansiyel karakteristik gerektirmektedir. Dört döngülü ve beş döngülük SPN'e saldırırken kullanmış olduğumuz diferansiyel karakteristik, hedef anahtarın 8 bitini sağlamaktadır. Altı döngülü SPN için kullanılan diferansiyel karakteristik ise, anahtarın 4 bitini vermektedir.

Yapılan analizler sonucunda altı döngülük SPN şifreleyicisinin diferansiyel kriptanalize direnç gösterdiği görülmüştür. Diferansiyel kriptanalizin, kesilmiş diferansiyel kriptanaliz, ilişkili anahtar diferansiyel kriptanaliz gibi gelişmiş yöntemleri kullanılarak, saldırı SPN'nin daha çok sayıda döngülü sürümlerine uygulanabilir.

Diferansiyel kriptanaliz ile SPN şifreleyicisi üzerinde yaptığımız çalışmalardan elde ettiğimiz bilgi birikimi ve sonuçları AES şifreleyicisi üzerinde denedik. AES şifreleyicisi doğrusal ve diferansiyel saldırılara dayanıklı olarak tasarlanmış bir

S-kutusuna sahip olduğu için ve bu saldırıyı zorlaştıran özellik gösteren MixColumn dönüşümünden dolayı, klasik diferansiyel saldırının, AES şifreleyicisi üzerinde etkin olma ihtimalinin düşük olacağının farkındaydık. Bu nedenle, AES şifreleyicisinin iki döngüsüne 1 döngülük diferansiyel karakteristik kullanarak diferansiyel kriptanaliz ile saldırdık; oysaki AES şifreleyicisi 14 döngüden oluşmaktadır. Klasik diferansiyel kriptanaliz saldırısı sonucunda, iki döngülük AES şifreleyicisi üzerinde dahi başarıya ulaşamadık.

(5.3.) bölümünde anlatılan DES şifreleleme algoritmasına uygulanan kesilmiş diferansiyel saldırı ile, anahtarın 18 biti yüksek bir olasılıkla elde edilebilmektedir. Doğru çiftlerin bulunması aşamasında, F fonksiyonunun tamamının hesaplanmasına gerek yoktur. Sadece S-kutularının yer aldığı hesaplamaların yapılması yeterli olmuştur. DES şifreleme algoritmasına uygulanan kesilmiş diferansiyel saldırıda karşılaşılan toplam karmaşıklık, 215 S-kutusu hesaplaması olmuştur. Bu değer yaklaşık olarak 500 6 döngülük DES hesaplamasına eşittir. Bu saldırıda kullanılan kesilmiş diferansiyelin olasılığı birdir. Farklı karakteristikleri sağlayan açık metinlerle gerçekleştirilecek benzer saldırılarla, anahtarın daha fazla bitini elde etmek mümkün olabilecektir.

İmkansız diferansiyel kriptanaliz, blok şifreleyiciler üzerindeki diferansiyel kriptanalizin bir çeşididir. Klasik diferansiyel kriptanaliz döngüler boyunca ilerletilen beklenen olasılığın üzerindeki farkları izlerken, imkansız diferansiyel kriptanaliz algoritmanın ara aşamalarında oluşan imkansız farklara odaklanır.

Bu tezde, Mini-AES şifreleyicisi üzerinde imkansız diferansiyel saldırıyı gerçeklemeye çalıştık. Mini-AES AES'in temel kavramlarının anlaşılması için eğitimsel amaçlı olarak geliştirilmiş olan, AES'in ilkel bir kopyasıdır.

Saldırıyı gerçeklemek için, öncelikle dört döngülük bir imkansız diferansiyel karakteristik belirledik. İlk nibble dışında eşit olan iki açık metin üzerinde çalıştık. Birinci döngüden sonra ilk kolonda iki aktif nibble'ı olan iki çıkış değeri elde ettik. İkinci döngü sonunda tüm nibble'ları aktif olan iki çıkış değerine sahip olduk. Sonra da şifreleyicinin diğer ucuna, dördüncü ,yani 4 döngülük Mini-AES şifreleyicisinin son döngüsünün çıkış değerlerine odaklandık. (5.4.2) bölümünden de görülebileceği gibi son iki döngü boyunca yapmış olduğumuz analiz sonuçları, ilk iki döngü boyunca yapmış olduğumuz analizlerden elde etmiş olduğumuz sonuçlarla çelişti. İlk iki

döngünün analizi sonucunda, ikinci döngünün çıkışında tüm nibble'ların aktif olduğu çıkarımına varılmışken, son iki döngünün analizi sadece iki nibble'ın aktif olduğu çıkarımını öne sürmüştür. Buradaki çelişkiden faydalanılarak imkansız diferansiyel oluşturulabilir.

Sadece bir nibble'ları farklı olan iki açık metin, 4 döngülük Mini-AES algoritması ile şifrelenirse, hiçbir zaman sadece kolon ve satırda, bir nibble'ları farklı olan şifreli metin çiftleri elde edilemez. Bu dört döngülük imkansız diferansiyelin kullanımı ile Mini-AES şifreleyicisine imkansız diferansiyel saldırı gerçekleştirmek mümkün olacaktır.

Bu çalışmada, imkansız diferansiyel ortadaki döngülere yerleştirilerek, dış döngülerdeki döngü anahtarları tahmin edilmiş ve bu anahtarlar kullanılarak imkansız diferansiyelin oluşup oluşmadığı doğrulanmıştır. Başarısızlıkla sonuçlanan doğrulamaların döngü anahtarları, yanlış olarak işaretlenip olası anahtarlar listesinden silinmiştir. Bu yetenek imkansız diferansiyel kriptanalizin arkasındaki büyük gücü oluşturmaktadır.

İmkansız diferansiyel kriptanaliz, AES şifreleyecisine uygulandığında teorik olarak 7. döngüye kadar saldırmak mümkün olmaktadır. Fakat, hesaplama gücünün yüksek olması nedeniyle gerçekleştirmiş olduğumuz uygulamada bu sonuçlara ulaşamadık. İmkansız diferansiyel kriptanalizin gelişmiş bir türevi olan ilişkili anahtar imkansız diferansiyel kriptanaliz tekniği kullanılarak, AES şifreleyicisinin daha fazla döngüsüne saldırmak mümkün olabilir.

S-kutuları blok ve akan şifreleme algoritmalarında kullanılan ve şifreye güvenliğini veren en önemli yapıdır. Bir S-kutusu n giriş bitinin farklı m çıkış bitine dönüşümünü yapar ve şifrede yer değiştirme görevini yerine getirir. Şifreye yapılan saldırılar S-kutularını hedef almaktadır ve S-kutusunun bu saldırılara karşı dayanıklılığı şifrenin de gücü ile ilişkilidir. Dolayısıyla saldırılara karşı dayanıklı S-kutusu tasarımları gerçekleştirilmelidir. 2001 yılında AES (Advanced Encryption Standard) olarak seçilen doğrusal ve diferansiyel saldırılara dayanıklı olan Rijndael şifresi Nyberg'in [2] önerdiği sonlu cisimde ters haritalama tabanlı 8-bit giriş ve 8-bit çıkışlı bir S-kutusunu kullanmaktadır ve cebirsel ifadesi aşağıdaki gibidir :

$$f(x) = x^{-1}, \quad \mathrm{x} \in GF(2^8), \quad f(0) = 0$$

Rijndael şifresinde kullanılan S-kutusunun en önemli sakıncası, yukarıda gösterilen cebirsel ifadenin basitliğidir. Bu basit cebirsel ifade interpolasyon saldırıları gibi bazı cebirsel saldırılara neden olabilmektedir. İnterpolasyon saldırılarına karşı, AES S-kutusunun tek terimden oluşmasının getirdiği zayıflık, ters haritalama işleminin sonuna eklenen bir doğrusal dönüşüm ile giderilmeye çalışılmıştır. Böylelikle aşağıdaki ifadeden de görüldüğü gibi, AES S-kutusunun cebirsel ifadesindeki terim sayısı, kullanılan doğrusal dönüşüm sayesinde 1'den 9'a çıkmıştır.

$$s(x) = "05"x^{254} + "09"x^{253} + "f9"x^{251} + "25"x^{247} + "f4"x^{239} + "01"x^{223} + "b5"x^{191} + "8f"x^{127} + "63"$$

Jakobsen ve Knudsen [3] tarafından sunulmuş olan interpolasyon saldırılarının karmaşıklığı, polinomsal ifadedeki terim sayısına ve elde edilen polinomun derecesine bağlıdır. (5.5)'de sunulan çalışmamızda herhangi bir 8-bit giriş ve 8-bit çıkışlı bir S-kutusunun $GF(2^8)$'de tasarlandığı indirgenemez polinoma göre cebirsel ifadesini hesaplayan Lagrange interpolasyonu uygulaması geliştirilmiştir. Geliştirilen uygulama ile, sonlu cisimde tasarlanan 8-bit giriş 8-bit çıkışlı bir S-kutusunun interpolasyon saldırılarına karşı dayanıklılığı incelenebilecektir. Buna ek olarak performans açısından daha iyi yöntemler geliştirmede referans olacak Lagrange interpolasyonu yönteminin performans değerlendirmesi de çalışmamızda verilmiştir.

S-kutusu tasarımında kullanılan üs haritalama, S-kutusunun cebirsel ifadesindeki terim sayısında etken olmaktadır. Buna ek olarak, kullanılan doğrusal dönüşüm yeri ve sayısı da, cebirsel ifadede hangi terimlerin bulunacağını belirlemektedir. AES şifreleyicisinde $x \to x^{254}$ üs haritalamasının kullanılması, 254'ün hamming ağırlığının 7 olması ve cebirsel ifadede yer alan terimlerin hamming ağırlığına eşit ve küçük olan üslü terimler olduğu savından yola çıkıldığında doğrudur. AES S-kutusunun cebirsel ifadesindeki terim sayısının az olması, doğrusal dönüşümün yeri ile ilgilidir. Doğrusal dönüşümün üs haritalamadan sonra kullanılması yerine, üs haritalamadan önce kullanılması veya hem önce hem sonra kullanılması durumunda, cebirsel ifadedeki terim sayısı 255'e çıkmaktadır. Böylelikle cebirsel saldırılara karşı iyileşme sağlanmış olur.

Klasik Lagrange interpolasyon algoritmasındaki polinom çarpımı sayısında yapılan iyileştirmelerle, çalışma zamanında iyileşmeler sağlanmıştır. Lagrange Paydalı olarak isimlendirilen yöntemde, önce *n* eleman birbiriyle bir defaya mahsus çarpılmış olup, $(i = j)$ için $(x - x_j)$ ifadesine bölünmüştür. Yapılan çarpma işlemi sayısı

azalmasına rağmen $(n-1)$ bölme işlemi eklenmiştir. Buna rağmen çalışma zamanında Tablo 5.6'dan da görüldüğü büyük bir iyileşme sağlanmıştır. (5.5) bölümünde ifade edildiği gibi $GF(2^n)$ sonlu cisminde Lagrange interpolasyonu algoritmasında paydadaki tüm terimlerin çarpımının sonucu 1 olacağından, paydanın çıkarılması sonucu yaklaşık olarak 2,5 saniyelik bir iyileşme daha sağlanmıştır.

Bu çalışma Lagrange interpolasyonu algoritmasında sonlu cisimde çarpma ve bölme işlemlerini hızlandıracak yöntemler geliştirilerek genişletilebilir. Ayrıca üs haritalamalı S-kutularının cebirsel ifadelerini, zaman karmaşıklığı ve çalışma zamanı açısından, daha hızlı şekilde elde edecek farklı yöntemler geliştirilebilir.

Bu fikirden yola çıkarak $GF\left(2^8\right)$'de olası tüm sonlu cisim çarpmalarını başka bir uygulama aracılığı ile gerçekleştirdik ve bir tabloda sakladık. Lagrange interpolasyonunda gerektiğinde, çarpma işlemini gerçekleştirmek yerine, bu tabloya ulaştık ve sonuçları buradan getirdik. Böylelikle işlem zamanında yarı yarıya kazanç sağladık.

(5.6) bölümünde sunulan çalışmada, AES S-kutusuna benzer S-kutularının cebirsel ifadesini Lagrange interpolasyonu yönteminden, hesaplama gücü ve zaman karmaşıklığı açısından daha iyi bir yöntemle elde ettik.

Bu yöntemle, $x \to x^{254}$ üs haritalaması kullanılarak tasarlanan ve cebirsel ifadesi 255 terim içeren S-kutusunun cebirsel ifadesi 40 milisaniye civarında bir sürede 2 GHz işlemcili bir bilgisayar ile elde edilmiştir. Buna ek olarak diğer üslerle tasarlanacak S-kutuları için benzeri hesaplama yöntemleri geliştirilebilir ve görünürde ki hesaplama ile ilgili yüksek iş yükü çok makul seviyelere indirilebilir. Verilen hesaplama yönteminde sonlu cisimde çarpma işlemleri çok hızlı olarak gerçeklenebileceği için, polinomsal çarpma işlemi üzerinde durulmuştur.

Bu çalışma sonucunda, AES S-kutusu ile ilgili elde edilen bilgiler, AES şifreleyecisine yapılacak olan olası bir cebirsel saldırıya ışık tutacaktır. AES S-kutusunda belirlenen olası zaafları giderecek iyileştirmelerin gerçeklenmesi, AES şifreleyicisini olası cebirsel saldırılara karşı koruyabilir.

TEZ SIRASINDA YAPILAN ÇALIŞMALAR

Uluslararası Kongre ve Sempozyum Bilgileri

i. Sakallı M.T., Aslan B., Buluş E, Şahin Mesut E., Büyüksaraçoğlu F., Karaahmetoğlu O., "On the Algebraic Expression of the AES S-box like S-boxes", yayında

Ulusal Kongre ve Sempozyum Bilgileri

i. Karaahmetoğlu O., Sakallı M.T., Buluş E., "GF (2^8) de Tasarlanan Bir S-Kutusunun Cebirsel İfadesinin Lagrange İnterpolasyonu Yöntemiyle Elde Edilmesi", IV. İletişim Teknolojileri Ulusal Sempozyumu, Adana-Türkiye, Ekim-2009.

ii. Karaahmetoğlu O., Sakallı M.T., Buluş E., "AES S-kutusuna Benzer S-kutularının Cebirsel İfadelerini Elde Etmek İçin Yeni Bir Yöntem", III. Ağ ve Bilgi Güvenliği Ulusal Sempozyumu,Ankara-Türkiye,Şubat-2010.

KISALTMALAR

AES (Advanced Encryption Standard)

AVAL (Avalanche Criterion)

CBC (Cipher Block Chaining)

CFB (Cipher Feedback)

DES (Data Encryption Standard)

DDT (Difference Distribution Table-XOR Table)

ECB (Electronic Codebook)

IP (Initial Permutation)

LAT (Linear Approximation Table)

OFB (Output Feedback)

SAC (Strict Avalanche Criterion)

SPN (Substitution-Permutation Networks)

UFN (Unbalanced Feistel Network)

KAYNAKÇA

[1] Mao Wenbo, *Modern Cryptography: Theory and Practice*.: Prentice Hall, 2004.

[2] C.E. Shannon, "A Mathematical Theory of Communication," *Bell Sys. Tech. Journal*, vol. 27, pp. 379–423, 623–656, July/Oct. 1948.

[3] A., Van Oorschot, P., Vanstone, S. Menezez, *Handbook of applied cryptography*, The CRC Press series on discrete mathematics and its applications ed.: CRC-Press, 1997.

[4] L.R. Knudsen, "The Number of Rounds in Block Ciphers", Public Reports of Nessie Projec May 12,2000.

[5] Federal Information Processing Standards, "Data Encryption Standard (DES)", 1999.

[6] Federal Information Processing Standards, "Advanced Encryption Standard (AES)", 1999.

[7] H. Heys, "A Tutorial on Linear and Differential Cryptanalysis", *Cryptologia*, vol. 26 No 3, pp. 189-221, 2002.

[8] Biham E. and Shamir A., *Differential Cryptanalysis of the Data Encryption Standard*.: Springer Verlag, 1993.

[9] Biryukov A. Biham E., "An Improvement of Davies' Attack on DES", *JOURNAL OF CRYPTOLOGY*, vol. 3, 1997.

[10] C. Shannon, "Communication theory of secrecy systems", *Bell System Technical Journal*, vol. 28(4), pp. 656-715, 1949.

[11] W.F. Friedman, "The index of coincidence and its applications in cryptology", Riverbank Laboratories, Publ 22.

[12] L.S. Hill, "Cryptography in an Algebraic Alphabet", *American Mathematical Monthly*, vol. 36, no. 6, pp. pp.306-312, 1929.

[13] H. Feistel, "Block Cipher Cryptographic System, US Patent 3,798,359", IBM, Filed June 30, 1971.

[14] B. Schneier, *Applied Cryptography, Second Edition: Protocols, Algorthms, and Source Code in C*.: John Wiley & Sons, Inc., 1996.

[15] L.R. Knudsen, "Contemporary Block Ciphers", *Lectures on Data Security.Modern Cryptology in Theory and Practice.Lecture Notes in Computer Science Tutorial, Ivan Damgård, Ed., LNCS 1561*, pp. 105-126, 1999.

[16] R.M. Hakala and K. Nyberg, Linear Distinguishing Attack on Shannon, 2008.

[17] M. Matsui, "Linear Cryptanalysis Method for DES Cipher", in *Advances in Cryptology–EUROCRYPT'93*, 1993, pp. 386-397.

[18] Biham, E. and Shamir, A., "Differential Cryptanalysis of DES-like Cryptosystems", *Journal of Cryptology*, vol. 4, No 1, pp. 3-72, 1991.

[19] Biham, E. and Shamir, A.;, "Differential Cryptanalysis of the full 16-round DES", in *Advances in Cryptology: Proceedings of CRYPTO'92*, 1993, pp. 487-496.

[20] D. Wagner, "The boomerang attack", *Lecture Notes in Computer Science*, vol. 1636, pp. 156-170, 1999.

[21] A. Biryukov, "The Boomerang Attack on 5 and 6-round Reduced AES⋆", *Lecture Notes in Computer Science*, vol. 3373, pp. 11-15, 2004.

[22] Winternitz, R. and Hellman, M., "Chosen-key attacks on a block cipher", *Cryptologia*, vol. 11(1), pp. 16-20, 1987.

[23] P. Junod, "Statistical Cryptanalysis of Block Ciphers", École Polytechnique Fédérale de Lausanne,Switzerland, phd thesis 3179, 2004.

[24] B., Hiltgen, A., Vaudenay, S. and Vuagnoux, M. Canvel, "Password interception in a SSL/TLS channel", in *Advances in Cryptology-Crypto 2003, 23rd Annual International Cryptology Conference*, Santa Barbara, California, USA, August 17-21, 2003, Proceedings, volume 2729 of Lecture Notes in Computer Science, pages 583-599. Springer-Verlag, 2003.

[25] H. and Heys, H. Handschuh, "A timing attack on RC5", in *Selected Areas in Cryptography: 5th Annual International Workshop, SAC'98*, Kingston, Ontario, Canada, August 1998, Proceedings, volume 1556 of Lecture Notes in Computer Science,pages 306-318. Springer-Verlag, 1999.

[26] F. and Quisquater, J.-J. Koeune, "A timing attack against Rijndael", Universite Catholique de Louvain, Louvain-la-Neuve, Belgium, Technical Report CG-1999/1 1999.

[27] P. Kocher, in *Advances in Cryptology-Crypto'96: 16th Annual International Cryptology Conference*, Santa Barbara, California, USA, August 18-22, 1996, Proceedings,volume 1109 of Lecture Notes in Computer Science, pages 104-113. Springer-Verlag, 1996.

[28] E. and Shamir, A. Biham, "Differential fault analysis of secret key cryptosystems", in *Advances in Cryptology - Crypto'97: 17th Annual International Cryptology Conference*, Santa Barbara, California, USA, August 1997, Proceedings, volume 1294 of Lecture Notes in Computer Science, pages 513-525. Springer-Verlag, 1997.

[29] L. Keliher, "Linear Cryptanalysis of Substitution-Permutation Networks", Queen's University, Queen's University, phd thesis.

[30] E. Biham, "How to forge DES-encrypted messages in 228 steps", Technion, Haifa,Israel, Technical Report CS-0884 1996.

[31] W. and Hellman, M. Diffie, "Exhaustive cryptanalysis of the NBS Data Encryption Standard", *Computer*, vol. 10(6), pp. 74-84, 1977.

[32] M.E. Hellman, "A cryptanalytic time-memory tradeoff", *IEEE Transactions on Information Theory*, vol. 26(4), pp. 401-406, 1980.

[33] E., Biryukov, A. and Shamir, A. Biham, "Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials", in *Advances in Cryptology-EUROCRYPT '99* , volume 1592 of Lecture Notes in Computer Science. Springer-Verlag, 1999.

[34] E., Biryukov, A., Shamir, A. Biham, "Miss in the Middle Attacks on Idea and Khufu", in *Proceedings of Fast Software Encryption'99*, 1999, LNCS 1636, pp. 124-138.

[35] Lai, "Higher order drivatives and differential cryptanalysis", *Communications and Cryptology*, pp. 227-233, 1994.

[36] L.R. Knudsen, "Truncated and Higher Order Differentials", in *Fast Software Encryption, 2nd International Workshop Proceedings*, 1995, pp. 196-211.

[37] D. Wagner, "The Boomerang Attack", in *Proceedings of the 6th International Workshop on Fast Software Encryption*, March 24-26, 1999, pp. 156-170, Lecture Notes In Computer Science; Vol. 1636 Springer-Verlag, 1999.

[38] G. and Desmedt, D. Jakimoski, "Related-Key Differential Cryptanalysis of 192-bit Key AES Variants, proceedings of Selected Areas in Cryptography 2003", *Lecture Notes in Computer Science 3006*, p. 208–221, 2004.

[39] E., Dunkellman, O., Keller, N. Biham, "Related-Key Boomerang and Rectangle Attacks, Advances in Cryptology, proceedings of EUROCRYPT 2005", *Lecture Notes in Computer Science*, vol. 3557, p. 507–525, 2005.

[40] J., Schneier, B., Wagner, D. Kelsey, "Related-Key Cryptanalysis of 3-WAY,Biham-DES, CAST, DES-X, NewDES, RC2, and TEA, proceedings of Information and Communication Security 1997", *Lecture Notes in Computer Science*, vol. 1334, p. 233–246, 1997.

[41] S., Kim, J., Kim, G., Lee, S., Preneel, B. Hong, "Related-Key Rectangle Attacks on Reduced Versions of SHACAL-1 and AES-192, proceedings of Fast Software Encryption 12", *Lecture Notes in Computer Science*, vol. 3557, p. 368–383, 2005.

[42] Y., Lee, C., Hong, S. and Lee, S. Ko, "Related Key Differential Cryptanalysis of Full-Round SPECTR-H64 and CIKS-1", vol. 3108, no. LNCS, pp. 137-148, 2004.

[43] L., Wagner, D. Knudsen, "Integral cryptanalysis (extended abstract),Fast Software Encryption: 9th International Workshop, FSE 2002, Leuven, Belgium, February 4-6, 2002. Revised Papers", *Lecture Notes in Computer Science*, pp. 112-127, 2002.

[44] Y., Zang, Y., Xiao, Y. Hu. (1999, August) Integral cryptanalysis of SAFER+. Electronic Letters,35(17):1458-1459.

[45] Knudsen L.R., Rijmen V. Daemen J., "The block cipher Square", *Fast Software Encryption, Lecture Notes in Computer Science*, vol. 1267, pp. 149-165, 1997.

[46] Buluş E., Şahin A., Büyüksaraçoğlu F. Sakallı M.T., "Bir blok şifreleme algoritmasına karşı square saldırısı", in *Ağ ve Bilgi Güvenliği Ulusal Sempozyumu-ABG2005*, İstanbul-TÜRKİYE, Haziran-2005.

[47] Knudsen L. Jakobsen T., "The interpolation attack on block ciphers", *Fast Software Encryption, Lecture Notes in Computer Science*, vol. 1267, pp. 28-40, 1997.

[48] Schroeppel R.,Whiting D. Ferguson N., "A simple algebraic representation of Rijndael", in *Selected Areas in Cryptography: 8th Annual International Workshop, SAC 2001*, Toronto, Ontario, Canada, August 16-17, 2001, Revised Papers, volume 2259 of Lecture Notes in Computer Science, pages 103-111. Springer-Verlag, 2001.

[49] Courtois N. and Pieprzyk J., "Cryptanalysis of block ciphers with overdefined systems of equations", in *Advances in Cryptology - Asiacrypt 2002: 8th*

International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, New Zealand, December 1-5, 2002, Proceedings, volume 2501 of Lecture Notes in Computer Science, pages 267-287. Springer-Verlag, 2002.

[50] Daemen J. and Rijmen V., "The Design of Rijndael", *Information Security and Cryptography*, 2002.

[51] Maurer U., "A provably-secure strongly-randomized cipher", in *Advances in Cryptology Eurocrypt'90: Workshop on the Theory and Application of Cryptographic Techniques*, Aarhus, Denmark, May 1990, Proceedings, volume 473 of Lecture Notes in Computer Science, pages 361-373. Springer-Verlag, 1991.

[52] Maurer U., "Conditionally-perfect secrecy and a provably-secure randomized cipher", *Journal of Cryptology*, vol. 5(1), pp. 53-66, 1992.

[53] Luby M. and Rackoff C., "How to construct pseudorandom permutations from pseudorandom functions", *SIAM Journal on Computing*, vol. 17(2), pp. 373-386 , 1988.

[54] Niederreiter H Lidl R., *Introduction to finite fields and their applications, Revised Edition*., 1994.

[55] Dawson M.H. and Tavares S.E., "An expanded set of S-box design criteria based on information theory and its relation to differential-like attacks", in *Advances in Cryptology - EUROCRYPT'91*, volume 547 of Lecture Notes in Computer Science, pages 352-367. Springer-Verlag, Berlin, Heidelberg, New York, 1991.

[56] Davida G.I. Kam J.B., "Structured Design of Substitution-Permutation Encryption Networks", *IEEE Transactions on Computers*, vol. 28, no. 10, pp. 747-753, 1979.

[57] Buluş E., Şahin A., Büyüksaraçoğlu F. Sakallı M.T., "Ters Haritalama Tabanlı S-kutularının Cebirsel Açıdan İyileştirilmesi", in *ISC'07 Uluslararası Katılımlı Bilgi Güvenliği ve Kriptoloji Konferansı*, Ankara-Türkiye, 13-14 Aralık 2007.

[58] Çeçen S., "Nonlinearity and Propagation Characteristics of Substitution Boxes", Middle East Technical University, Türkiye, M. S. Thesis 2001.

[59] Yücel D. Kavut S., "On Some Cryptographic Properties of Rijndael", *Lecture Notes in Computer Science: Information Assurance in Computer Networks, Methods, Models and Architectures for Network Security*, pp. 300-311, May 2001.

[60] Kim S., Lee S.,Sung S. H.,Yoon S. Chun K. (2002) Differential and linear cryptanalysis for 2-round SPNs. Information Processing Letters.

[61] Aslan Bora, "Boole Fonksiyonları Ve S-Kutularının Kriptografik Özelliklerinin İncelenmesi Ve Ters Haritalama Tabanlı Cebirsel Açıdan Güçlendirilmiş Bir S-Kutusu Önerisi ", Trakya Üniversitesi, Edirne,Türkiye, Yüksek Lisans Tezi 2008.

[62] Vergili I., "Statistics on Satisfaction of Security Criteria for Randomly Generated S-boxes", Middle East Technical University, Ankara, Türkiye, M.S. Thesis 2000.

[63] Aras E., "Analysis of Security Criteria for Block Ciphers", Ankara, Türkiye, M.S. Thesis 1999.

[64] Feistel H., "Cryptography and Computer Privacy", *Scientific American*, vol. 228, no. 5, pp. 15-23, 1973.

[65] Webster A. F. and Tavares S.H., "On the Design of S-boxes", in *Advances in Cryptology: Proceedings of CRYPTO'85*, New York, 1986, pp. 523-534.

[66] Soğukpınar İ. (2008) Ayrık Matematik Ders Notları.

[67] Buluş E., Büyüksaraçoğlu F. Sakallı M.T., "İki Döngülü Bir Blok Şifreleme Algoritmasının Lineer Kriptanaliz Uygulaması", in *ELECO 2004, Elektrik Elektronik Bilgisayar Mühendisliği Sempozyumu*, Bursa, 2004.

[68] Kelsey J. Schneier B., "Unbalanced Feistel networks and block-cipher design", in *Fast Software Encryption: Third International Workshop*, 1996, LNCS 1039, pp. 121-144, Springer-Verlag.

[69] Buluş E.,Sakallı M.T. Şahin A., "Modern Blok Şifreleme Algoritmalarının Gücünün İncelenmesi", in *2. Mühendislik Bilimleri Genç Araştırmacılar Kongresi MBGAK*, İstanbul, 2005.

[70] Sakallı M.T., "Modern Şifreleme Yöntemlerinin Gücünün İncelenmesi", Trakya Üniversitesi, Doktora Tezi 2006.

[71] Shamir A. Biham E., "Differential Cryptanalysis of Snefru, Khafre, REDOC II, LOKI, and Lucifer", in *Advances in Cryptology, CRYPTO '91 Proceedings*, 1992, pp. 156-171.

[72] Buluş E., Şahin A., Büyüksaraçoğlu F. Sakallı M.T., "Differential Cryptanalysis for a 3-Round SPN", in *4th Internternational Conference on Electrical and Electronics Engineering ELECO'2005*, Bursa-TURKİYE, Aralık-2005, pp. 297-301.

[73] Stinson D., *Cryptography Theory and Practice*, 2nd ed.: CRC Press, Inc., Fubruary 2002.

[74] Tavares S. Heys H., "Substitution Permutation Networks Resistant to Differential and Linear Cryptanalysis", *JOURNAL OF CRYPTOLOGY*, vol. 9, no. 1, pp. 1-19, 1996.

[75] Song B. and Seberry J., "Consistent Differential Patterns of Rijndael", Centre for Computer Security Research,.

[76] Massey J.L., "SAFER K-64: A Byte-Oriented Block-Ciphering Algorithm", in *Fast Software Encryption*, 1993, pp. 1-17.

[77] Berson T.A. Knudsen L.R., "Truncated Differentials of SAFER", in *Fast Software Encryption*, 1996, pp. 15-26.

[78] Phan W. Chung R., "Impossible Differential Cryptanalysis of Mini-AES", *Cryptologia*, vol. 27, no. 4, October 2003.

[79] Phan W. Chung R., "Mini Advanced Encryption Standard (Mini-AES): A Testbed for Cryptanalysis Students", *Cryptologia*, vol. 26, no. 4.

[80] E., Keller, N. Biham, "Cryptanalysis of Reduced Variants of Rijndael", in *Proceedings of 3rd AES Conference*, 2000.

[81] Kim M., Kim K., Lee J. Y., Kang S. Cheon J. H., "Improved Impossible Differential Cryptanalysis of Rijndael and Crypton", in *In International Conference on Information Security and Cryptology (ICISC)*, 2001, pp. 39-49, (Lecture Notes in Computer Science No. 2288).

[82] Phan W., Siddiqi M.U. Chung R., Generalised impossible Differentials of Advanced Encryption standard, 2001.

[83] R. C. W. Phan, Classes of Impossible Differentials of Advanced Encryption Standard, 2002, IEE Electronics Letters. 38(11): 508-510.

[84] Phan W. Chung R., Impossible Differential Cryptanalysis of 7-round AES, 2004.

[85] Leiserson C.E., Rivest R.L., and Stein C. Cormen T.H., *Introduction to Algorithms, Second Edition*.: MIT Press and McGraw-Hill, 2001.

[86] Gong G. Youssef A.M., "On the Interpolation Attacks on Block Ciphers", in *7 the International Workshop on Fast Software Encryption*, 2000, p. 109–120.

[87] Tavares S. E., Gong G. Youssef A. M., "On Some probabilistic approximations for AES-like s-boxes", *Discrete Mathematics, Elsevier*, 2006.

[88] Ichikawa T., Kanda M., Matsui M., Moriai S., Nakajima J., Tokita T. Aoki K., "Camellia: a 128-bit block cipher suitable for multiple platforms-design and analysis", in *Proceedings of Seventh Annual International Workshop on Selected Areas in Cryptography, SAC'2000 Lecture Notes in Computer Science*, vol. 2012, Berlin, 2001, pp. 39-56.

[89] Sakallı M.T., Buluş E. Aslan B., "Üs Haritalama Tabanlı Cebirsel 8-bit giriş 8-bit çıkışlı S-kutularının Sınıflandırılması", in *Ağ ve Bilgi Ulusal Sempozyumu 2*, Girne-Kıbrıs, 2008.

[90] Sakallı M.T., Buluş E. Aslan B., "Classifying 8-bit to 8-bit S-boxes based on Power Mappings from the point of DDT and LAT Distributions", in *International Workshop on the Arithmetic of Finite Fields, WAIFI 2008, Lecture Notes in Computer Science*, Siena-Italy , 2008.

[91] Buluş E., Şahin A., Büyüksaraçoğlu F. Sakallı M.T., "Sonlu Cisim Teorisini Kullanarak Ters Haritalama Tabanlı Bir S-kutusunun Cebirsel ifadesini Elde Etme - Obtaining Algebraic expression of an S-box Based on Inversion Mapping Using Finite Field Theory", in *IEEE 15. Sinyal İşleme ve İletişim Uygulamaları Kurultayı-SİU 2007*, Eskişehir-TÜRKİYE, Haziran-2007.

[92] Buluş E., Şahin A., Büyüksaraçoğlu F. Sakallı M.T., "Aes S-Kutusuna Alternatif Cebirsel Olarak Kuvvetlendirilmiş Bir S-Kutusu Önerisi", in *Ağ ve Bilgi Güvenliği Ulusal Sempozyumu-ABG'08*, Girne-Kuzey Kıbrıs Türk Cumhuriyeti, Mayıs-2008.

[93] Daemen J., Preneel B., Bosselaers A., De Win E. Rijmen V., "The cipher Shark", *Fast Software Encryption, Lecture Notes in Computer Science*, vol. 1039, p. 99–112, 1996.

[94] Tavares S.E. Youssef A. M., "Affine equivalence in the AES round function", *Discrete Applied Mathematics*, 2005.

[95] Gong G. Youssef A. M., "On the Interpolation Attacks on Block Ciphers", in *7 the International Workshop on Fast Software Encryption*, 2000, p. 109–120.

ÖZGEÇMİŞ

Osman KARAAHMETOĞLU, 25 Nisan 1974 tarihinde Trabzon'da doğdu. İlk ve Orta öğenimini İstanbul Güngören'de tamamladıktan sonra, 1991 yılında Yıldız Teknik Üniversitesi Bilgisayar Bilimleri Mühendisliği bölümünü kazandı. Biri İngilizce Hazırlık olmak üzere 5 yıldan oluşan Bilgisayar Bilimleri Mühendisliği Bölümü'nden, 1996 yılında mezun oldu. Yine aynı yıl, İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü Kontrol ve Bilgisayar Mühendisliği Anabilim dalında yüksek lisansa başladı. Yüksek lisansını 2001 yılında başarıyla bitirdi. Yeditepe Üniversitesi Sosyal Bilimler Enstitüsü'nde 1999 yılında başladığı İngilizce İşletme yüksek lisansını 2002 yılında tamamladı. 2003 yılında Trakya Üniversitesi Bilgisayar Mühendisliği Bölümü'nde doktora çalışmalarına başladı. 1996 yılından beri çeşitli firmalarda yazılım geliştirme görevlerinde bulundu. Şu anda bir bankanın bilgi işlem firmasında yazılım ekip lideri olarak çalışmaktadır. İyi derecede İngilizce bilmektedir.

EKLER

Beta Değerleri

Beta Üsleri	Polinomsal Gösterim	Hex
β^0	1	01
β^1	$\alpha+1$	03
β^2	α^2+1	05
β^3	$\alpha^3+\alpha^2+\alpha+1$	0f
β^4	α^4+1	11
β^5	$\alpha^5+\alpha^4+\alpha+1$	33
β^6	$\alpha^6+\alpha^4+\alpha^2+1$	55
β^7	$\alpha^7+\alpha^6+\alpha^5+\alpha^4+\alpha^3+\alpha^2+\alpha+1$	ff
β^8	$\alpha^4+\alpha^3+\alpha$	1a
β^9	$\alpha^5+\alpha^3+\alpha^2+1$	2e
β^{10}	$\alpha^6+\alpha^5+\alpha^4+\alpha$	72
β^{11}	$\alpha^7+\alpha^4+\alpha^2+\alpha$	96
β^{12}	$\alpha^7+\alpha^5+1$	a1
β^{13}	$\alpha^7+\alpha^6+\alpha^5+\alpha^4+\alpha^3$	f8
β^{14}	$\alpha^4+\alpha+1$	13
β^{15}	$\alpha^5+\alpha^4+\alpha^2+1$	35
β^{16}	$\alpha^6+\alpha^4+\alpha^3+\alpha^2+\alpha+1$	5f
β^{17}	$\alpha^7+\alpha^6+\alpha^5+1$	e1
β^{18}	$\alpha^5+\alpha^4+\alpha^3$	38
β^{19}	$\alpha^6+\alpha^3$	48
β^{20}	$\alpha^7+\alpha^6+\alpha^4+\alpha^3$	d8
β^{21}	$\alpha^6+\alpha^5+\alpha^4+\alpha+1$	73
β^{22}	$\alpha^7+\alpha^4+\alpha^2+1$	95
β^{23}	$\alpha^7+\alpha^5+\alpha^2$	a4
β^{24}	$\alpha^7+\alpha^6+\alpha^5+\alpha^4+\alpha^2+\alpha+1$	f7
β^{25}	α	02
β^{26}	$\alpha^2+\alpha$	06
β^{27}	$\alpha^3+\alpha$	0a
β^{28}	$\alpha^4+\alpha^3+\alpha^2+\alpha$	1e
β^{29}	$\alpha^5+\alpha$	22
β^{30}	$\alpha^6+\alpha^5+\alpha^2+\alpha$	66

β^{31}	$\alpha^7+\alpha^5+\alpha^3+\alpha$	aa
β^{32}	$\alpha^7+\alpha^6+\alpha^5+\alpha^2+1$	e5
β^{33}	$\alpha^5+\alpha^4+\alpha^2$	34
β^{34}	$\alpha^6+\alpha^4+\alpha^3+\alpha^2$	5c
β^{35}	$\alpha^7+\alpha^6+\alpha^5+\alpha^2$	e4
β^{36}	$\alpha^5+\alpha^4+\alpha^2+\alpha+1$	37
β^{37}	$\alpha^6+\alpha^4+\alpha^3+1$	59
β^{38}	$\alpha^7+\alpha^6+\alpha^5+\alpha^3+\alpha+1$	eb
β^{39}	$\alpha^5+\alpha^2+\alpha$	26
β^{40}	$\alpha^6+\alpha^5+\alpha^3+\alpha$	6a
β^{41}	$\alpha^7+\alpha^5+\alpha^4+\alpha^3+\alpha^2+\alpha$	be
β^{42}	$\alpha^7+\alpha^6+\alpha^4+\alpha^3+1$	d9
β^{43}	$\alpha^6+\alpha^5+\alpha^4$	70
β^{44}	$\alpha^7+\alpha^4$	90
β^{45}	$\alpha^7+\alpha^5+\alpha^3+\alpha+1$	ab
β^{46}	$\alpha^7+\alpha^6+\alpha^5+\alpha^2+\alpha$	e6
β^{47}	$\alpha^5+\alpha^4+1$	31
β^{48}	$\alpha^6+\alpha^4+\alpha+1$	53
β^{49}	$\alpha^7+\alpha^6+\alpha^5+\alpha^4+\alpha^2+1$	f5
β^{50}	α^2	04
β^{51}	$\alpha^3+\alpha^2$	0c
β^{52}	$\alpha^4+\alpha^2$	14
β^{53}	$\alpha^5+\alpha^4+\alpha^3+\alpha^2$	3c
β^{54}	$\alpha^6+\alpha^2$	44
β^{55}	$\alpha^7+\alpha^6+\alpha^3+\alpha^2$	cc
β^{56}	$\alpha^6+\alpha^3+\alpha^2+\alpha+1$	4f
β^{57}	$\alpha^7+\alpha^6+\alpha^4+1$	d1
β^{58}	$\alpha^6+\alpha^5+\alpha^3$	68
β^{59}	$\alpha^7+\alpha^5+\alpha^4+\alpha^3$	b8
β^{60}	$\alpha^7+\alpha^6+\alpha^4+\alpha+1$	d3
β^{61}	$\alpha^6+\alpha^5+\alpha^3+\alpha^2+\alpha$	6e
β^{62}	$\alpha^7+\alpha^5+\alpha^4+\alpha$	b2
β^{63}	$\alpha^7+\alpha^6+\alpha^3+\alpha^2+1$	cd
β^{64}	$\alpha^6+\alpha^3+\alpha^2$	4c
β^{65}	$\alpha^7+\alpha^6+\alpha^4+\alpha^2$	d4
β^{66}	$\alpha^6+\alpha^5+\alpha^2+\alpha+1$	67
β^{67}	$\alpha^7+\alpha^5+\alpha^3+1$	a9
β^{68}	$\alpha^7+\alpha^6+\alpha^5$	e0

β^{69}	$\alpha^5+\alpha^4+\alpha^3+\alpha+1$	3b
β^{70}	$\alpha^6+\alpha^3+\alpha^2+1$	4d
β^{71}	$\alpha^7+\alpha^6+\alpha^4+\alpha^2+\alpha+1$	d7
β^{72}	$\alpha^6+\alpha^5+\alpha$	62
β^{73}	$\alpha^7+\alpha^5+\alpha^2+\alpha$	a6
β^{74}	$\alpha^7+\alpha^6+\alpha^5+\alpha^4+1$	f1
β^{75}	α^3	08
β^{76}	$\alpha^4+\alpha^3$	18
β^{77}	$\alpha^5+\alpha^3$	28
β^{78}	$\alpha^6+\alpha^5+\alpha^4+\alpha^3$	78
β^{79}	$\alpha^7+\alpha^3$	88
β^{80}	$\alpha^7+\alpha+1$	83
β^{81}	$\alpha^7+\alpha^4+\alpha^3+\alpha^2+1$	9e
β^{82}	$\alpha^7+\alpha^5+\alpha^4+\alpha^3+1$	b9
β^{83}	$\alpha^7+\alpha^6+\alpha^4$	d0
β^{84}	$\alpha^6+\alpha^5+\alpha^3+\alpha+1$	6b
β^{85}	$\alpha^7+\alpha^5+\alpha^4+\alpha^3+\alpha^2+1$	bd
β^{86}	$\alpha^7+\alpha^6+\alpha^4+\alpha^3+\alpha^2$	dc
β^{87}	$\alpha^6+\alpha^5+\alpha^4+\alpha^3+\alpha^2+\alpha+1$	7f
β^{88}	α^7+1	81
β^{89}	$\alpha^7+\alpha^4+\alpha^3$	98
β^{90}	$\alpha^7+\alpha^5+\alpha^4+\alpha+1$	b3
β^{91}	$\alpha^7+\alpha^6+\alpha^3+\alpha^2+\alpha$	ce
β^{92}	$\alpha^6+\alpha^3+1$	49
β^{93}	$\alpha^7+\alpha^6+\alpha^4+\alpha^3+\alpha+1$	db
β^{94}	$\alpha^6+\alpha^5+\alpha^4+\alpha^2+\alpha$	76
β^{95}	$\alpha^7+\alpha^4+\alpha^3+\alpha$	9a
β^{96}	$\alpha^7+\alpha^5+\alpha^4+\alpha^2+1$	b5
β^{97}	$\alpha^7+\alpha^6+\alpha^2$	c4
β^{98}	$\alpha^6+\alpha^4+\alpha^2+\alpha+1$	57
β^{99}	$\alpha^7+\alpha^6+\alpha^5+\alpha^4+\alpha^3+1$	f9
β^{100}	α^4	10
β^{101}	$\alpha^5+\alpha^4$	30
β^{102}	$\alpha^6+\alpha^4$	50
β^{103}	$\alpha^7+\alpha^6+\alpha^5+\alpha^4$	f0
β^{104}	$\alpha^3+\alpha+1$	0b
β^{105}	$\alpha^4+\alpha^3+\alpha^2+1$	1d
β^{106}	$\alpha^5+\alpha^2+\alpha+1$	27

β^{107}	$\alpha^6+\alpha^5+\alpha^3+1$	69
β^{108}	$\alpha^7+\alpha^5+\alpha^4+\alpha^3+\alpha+1$	bb
β^{109}	$\alpha^7+\alpha^6+\alpha^4+\alpha^2+1$	d6
β^{110}	$\alpha^6+\alpha^5+1$	61
β^{111}	$\alpha^7+\alpha^5+\alpha+1$	a3
β^{112}	$\alpha^7+\alpha^6+\alpha^5+\alpha^4+\alpha^3+\alpha^2+\alpha$	fe
β^{113}	$\alpha^4+\alpha^3+1$	19
β^{114}	$\alpha^5+\alpha^3+\alpha+1$	2b
β^{115}	$\alpha^6+\alpha^5+\alpha^4+\alpha^3+\alpha^2+1$	7d
β^{116}	$\alpha^7+\alpha^2+\alpha+1$	87
β^{117}	$\alpha^7+\alpha^4+\alpha$	92
β^{118}	$\alpha^7+\alpha^5+\alpha^3+\alpha^2+1$	ad
β^{119}	$\alpha^7+\alpha^6+\alpha^5+\alpha^3+\alpha^2$	ec
β^{120}	$\alpha^5+\alpha^3+\alpha^2+\alpha+1$	2f
β^{121}	$\alpha^6+\alpha^5+\alpha^4+1$	71
β^{122}	$\alpha^7+\alpha^4+\alpha+1$	93
β^{123}	$\alpha^7+\alpha^5+\alpha^3+\alpha^2+\alpha$	ae
β^{124}	$\alpha^7+\alpha^6+\alpha^5+\alpha^3+1$	e9
β^{125}	α^5	20
β^{126}	$\alpha^6+\alpha^5$	60
β^{127}	$\alpha^7+\alpha^5$	a0
β^{128}	$\alpha^7+\alpha^6+\alpha^5+\alpha^4+\alpha^3+\alpha+1$	fb
β^{129}	$\alpha^4+\alpha^2+\alpha$	16
β^{130}	$\alpha^5+\alpha^4+\alpha^3+\alpha$	3a
β^{131}	$\alpha^6+\alpha^3+\alpha^2+\alpha$	4e
β^{132}	$\alpha^7+\alpha^6+\alpha^4+\alpha$	d2
β^{133}	$\alpha^6+\alpha^5+\alpha^3+\alpha^2+1$	6d
β^{134}	$\alpha^7+\alpha^5+\alpha^4+\alpha^2+\alpha+1$	b7
β^{135}	$\alpha^7+\alpha^6+\alpha$	c2
β^{136}	$\alpha^6+\alpha^4+\alpha^3+\alpha^2+1$	5d
β^{137}	$\alpha^7+\alpha^6+\alpha^5+\alpha^2+\alpha+1$	e7
β^{138}	$\alpha^5+\alpha^4+\alpha$	32
β^{139}	$\alpha^6+\alpha^4+\alpha^2+\alpha$	56
β^{140}	$\alpha^7+\alpha^6+\alpha^5+\alpha^4+\alpha^3+\alpha$	fa
β^{141}	$\alpha^4+\alpha^2+1$	15
β^{142}	$\alpha^5+\alpha^4+\alpha^3+\alpha^2+\alpha+1$	3f
β^{143}	α^6+1	41
β^{144}	$\alpha^7+\alpha^6+\alpha+1$	c3

β^{145}	$\alpha^6+\alpha^4+\alpha^3+\alpha^2+\alpha$	5e
β^{146}	$\alpha^7+\alpha^6+\alpha^5+\alpha$	e2
β^{147}	$\alpha^5+\alpha^4+\alpha^3+\alpha^2+1$	3d
β^{148}	$\alpha^6+\alpha^2+\alpha+1$	47
β^{149}	$\alpha^7+\alpha^6+\alpha^3+1$	c9
β^{150}	α^6	40
β^{151}	$\alpha^7+\alpha^6$	c0
β^{152}	$\alpha^6+\alpha^4+\alpha^3+\alpha+1$	5b
β^{153}	$\alpha^7+\alpha^6+\alpha^5+\alpha^3+\alpha^2+1$	ed
β^{154}	$\alpha^5+\alpha^3+\alpha^2$	2c
β^{155}	$\alpha^6+\alpha^5+\alpha^4+\alpha^2$	74
β^{156}	$\alpha^7+\alpha^4+\alpha^3+\alpha^2$	9c
β^{157}	$\alpha^7+\alpha^5+\alpha^4+\alpha^3+\alpha^2+\alpha+1$	bf
β^{158}	$\alpha^7+\alpha^6+\alpha^4+\alpha^3+\alpha$	da
β^{159}	$\alpha^6+\alpha^5+\alpha^4+\alpha^2+1$	75
β^{160}	$\alpha^7+\alpha^4+\alpha^3+\alpha^2+\alpha+1$	9f
β^{161}	$\alpha^7+\alpha^5+\alpha^4+\alpha^3+\alpha$	ba
β^{162}	$\alpha^7+\alpha^6+\alpha^4+\alpha^2+1$	d5
β^{163}	$\alpha^6+\alpha^5+\alpha^2$	64
β^{164}	$\alpha^7+\alpha^5+\alpha^3+\alpha^2$	ac
β^{165}	$\alpha^7+\alpha^6+\alpha^5+\alpha^3+\alpha^2+\alpha+1$	ef
β^{166}	$\alpha^5+\alpha^3+\alpha$	2a
β^{167}	$\alpha^6+\alpha^5+\alpha^4+\alpha^3+\alpha^2+\alpha$	7e
β^{168}	$\alpha^7+\alpha$	82
β^{169}	$\alpha^7+\alpha^4+\alpha^3+\alpha^2+1$	9d
β^{170}	$\alpha^7+\alpha^5+\alpha^4+\alpha^3+\alpha^2$	bc
β^{171}	$\alpha^7+\alpha^6+\alpha^4+\alpha^3+\alpha^2+\alpha+1$	df
β^{172}	$\alpha^6+\alpha^5+\alpha^4+\alpha^3+\alpha$	7a
β^{173}	$\alpha^7+\alpha^3+\alpha^2+\alpha$	8e
β^{174}	$\alpha^7+\alpha^3+1$	89
β^{175}	α^7	80
β^{176}	$\alpha^7+\alpha^4+\alpha^3+\alpha+1$	9b
β^{177}	$\alpha^7+\alpha^5+\alpha^4+\alpha^2+\alpha$	b6
β^{178}	$\alpha^7+\alpha^6+1$	c1
β^{179}	$\alpha^6+\alpha^4+\alpha^3$	58
β^{180}	$\alpha^7+\alpha^6+\alpha^5+\alpha^3$	e8
β^{181}	$\alpha^5+\alpha+1$	23
β^{182}	$\alpha^6+\alpha^5+\alpha^2+1$	65

β^{183}	$\alpha^7+\alpha^5+\alpha^3+\alpha^2+\alpha+1$	af
β^{184}	$\alpha^7+\alpha^6+\alpha^5+\alpha^3+\alpha$	ea
β^{185}	$\alpha^5+\alpha^2+1$	25
β^{186}	$\alpha^6+\alpha^5+\alpha^3+\alpha^2+\alpha+1$	6f
β^{187}	$\alpha^7+\alpha^5+\alpha^4+1$	b1
β^{188}	$\alpha^7+\alpha^6+\alpha^3$	c8
β^{189}	$\alpha^6+\alpha+1$	43
β^{190}	$\alpha^7+\alpha^6+\alpha^2+1$	c5
β^{191}	$\alpha^6+\alpha^4+\alpha^2$	54
β^{192}	$\alpha^7+\alpha^6+\alpha^5+\alpha^4+\alpha^3+\alpha^2$	fc
β^{193}	$\alpha^4+\alpha^3+\alpha^2+\alpha+1$	1f
β^{194}	α^5+1	21
β^{195}	$\alpha^6+\alpha^5+\alpha+1$	63
β^{196}	$\alpha^7+\alpha^5+\alpha^2+1$	a5
β^{197}	$\alpha^7+\alpha^6+\alpha^5+\alpha^4+\alpha^2$	f4
β^{198}	$\alpha^2+\alpha+1$	07
β^{199}	α^3+1	09
β^{200}	$\alpha^4+\alpha^3+\alpha+1$	1b
β^{201}	$\alpha^5+\alpha^3+\alpha^2+1$	2d
β^{202}	$\alpha^6+\alpha^5+\alpha^4+\alpha^2+\alpha+1$	77
β^{203}	$\alpha^7+\alpha^4+\alpha^3+1$	99
β^{204}	$\alpha^7+\alpha^5+\alpha^4$	b0
β^{205}	$\alpha^7+\alpha^6+\alpha^3+\alpha+1$	cb
β^{206}	$\alpha^6+\alpha^2+\alpha$	46
β^{207}	$\alpha^7+\alpha^6+\alpha^3+\alpha$	ca
β^{208}	$\alpha^6+\alpha^2+1"$	45
β^{209}	$\alpha^7+\alpha^6+\alpha^3+\alpha^2+\alpha+1$	cf
β^{210}	$\alpha^6+\alpha^3+\alpha$	4a
β^{211}	$\alpha^7+\alpha^6+\alpha^4+\alpha^3+\alpha^2+\alpha$	de
β^{212}	$\alpha^6+\alpha^5+\alpha^4+\alpha^3+1$	79
β^{213}	$\alpha^7+\alpha^3+\alpha+1$	8b
β^{214}	$\alpha^7+\alpha^2+\alpha$	86
β^{215}	$\alpha^7+\alpha^4+1$	91
β^{216}	$\alpha^7+\alpha^5+\alpha^3$	a8
β^{217}	$\alpha^7+\alpha^6+\alpha^5+\alpha+1$	e3
β^{218}	$\alpha^5+\alpha^4+\alpha^3+\alpha^2+\alpha$	3e
β^{219}	$\alpha^6+\alpha$	42
β^{220}	$\alpha^7+\alpha^6+\alpha^2+\alpha$	c6

β^{221}	$\alpha^6+\alpha^4+1$	51
β^{222}	$\alpha^7+\alpha^6+\alpha^5+\alpha^4+\alpha+1$	f3
β^{223}	$\alpha^3+\alpha^2+\alpha$	0e
β^{224}	$\alpha^4+\alpha$	12
β^{225}	$\alpha^5+\alpha^4+\alpha^2+\alpha$	36
β^{226}	$\alpha^6+\alpha^4+\alpha^3+\alpha$	5a
β^{227}	$\alpha^7+\alpha^6+\alpha^5+\alpha^3+\alpha^2+\alpha$	ee
β^{228}	$\alpha^5+\alpha^3+1$	29
β^{229}	$\alpha^6+\alpha^5+\alpha^4+\alpha^3+\alpha+1$	7b
β^{230}	$\alpha^7+\alpha^3+\alpha^2+1$	8d
β^{231}	$\alpha^7+\alpha^3+\alpha^2$	8c
β^{232}	$\alpha^7+\alpha^3+\alpha^2+\alpha+1$	8f
β^{233}	$\alpha^7+\alpha^3+\alpha$	8a
β^{234}	$\alpha^7+\alpha^2+1$	85
β^{235}	$\alpha^7+\alpha^4+\alpha^2$	94
β^{236}	$\alpha^7+\alpha^5+\alpha^2+\alpha+1$	a7
β^{237}	$\alpha^7+\alpha^6+\alpha^5+\alpha^4+\alpha$	f2
β^{238}	$\alpha^3+\alpha^2+1$	0d
β^{239}	$\alpha^4+\alpha^2+\alpha+1$	17
β^{240}	$\alpha^5+\alpha^4+\alpha^3+1$	39
β^{241}	$\alpha^6+\alpha^3+\alpha+1$	4b
β^{242}	$\alpha^7+\alpha^6+\alpha^4+\alpha^3+\alpha^2+1$	dd
β^{243}	$\alpha^6+\alpha^5+\alpha^4+\alpha^3+\alpha^2$	7c
β^{244}	$\alpha^7+\alpha^2$	84
β^{245}	$\alpha^7+\alpha^4+\alpha^2+\alpha+1$	97
β^{246}	$\alpha^7+\alpha^5+\alpha$	a2
β^{247}	$\alpha^7+\alpha^6+\alpha^5+\alpha^4+\alpha^3+\alpha^2+1$	fd
β^{248}	$\alpha^4+\alpha^3+\alpha^2$	1c
β^{249}	$\alpha^5+\alpha^2$	24
β^{250}	$\alpha^6+\alpha^5+\alpha^3+\alpha^2$	6c
β^{251}	$\alpha^7+\alpha^5+\alpha^4+\alpha^2$	b4
β^{252}	$\alpha^7+\alpha^6+\alpha^2+\alpha+1$	c7
β^{253}	$\alpha^6+\alpha^4+\alpha$	52
β^{254}	$\alpha^7+\alpha^6+\alpha^5+\alpha^4+\alpha^2+\alpha$	f6
β^{255}	1	01