KOCAELİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI

DOKTORA TEZİ

MAKSİMUM UZAKLIKTA AYRILABİLEN MATRİSLERİN ELDE EDİLEBİLMESİ İÇİN YENİ BİR MATRİS FORMU VE BİR HAFİF SIKLET BLOK ŞİFREYE UYGULAMASI

MELTEM KURT PEHLİVANOĞLU

KOCAELİ 2018

KOCAELİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ

ANABİLİM DALI

DOKTORA TEZİ

MAKSİMUM UZAKLIKTA AYRILABİLEN MATRİSLERİN ELDE EDİLEBİLMESİ İÇİN YENİ BİR MATRİS FORMU VE BİR HAFİF SIKLET BLOK ŞİFREYE UYGULAMASI

MELTEM KURT PEHLİVANOĞLU

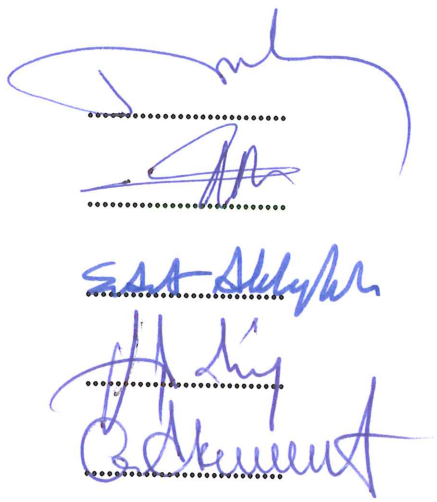
Prof. Dr. Nevcihan DURU
Danışman, Kocaeli Üniversitesi

Doç. Dr. Muharrem Tolga SAKALLI
Jüri Üyesi, Trakya Üniversitesi

Doç. Dr. Sedat AKLEYLEK
Jüri Üyesi, Ondokuz Mayıs Üniversitesi

Dr. Öğr. Üyesi Suhap ŞAHİN
Jüri Üyesi, Kocaeli Üniversitesi

Dr. Öğr. Üyesi Orhan AKBULUT
Jüri Üyesi, Kocaeli Üniversitesi



Tezin Savunulduğu Tarih: 13.07.2018

ÖNSÖZ VE TEŞEKKÜR

Bu tez çalışmasında, hafif sıklet blok şifreler için verimli yayılım tabakası tasarımı amaçlanmıştır. Bu amaçla yeni bir matris formu önerilerek, minimum donanım maliyetli doğrusal dönüşüm matrisleri üretilmiştir. Tez kapsamında geliştirilen özgün blok şifre bileşenleri ile yeni bir hafif sıklet blok şifre tasarlanmıştır.

Doktora eğitimim süresince desteklerini ve zamanını benden esirgemeyen, çalışmalarıma yön veren, bana güvenen ve yüreklendiren çok değerli Hocam ve danışmanım Sayın Prof. Dr. Nevcihan DURU'ya sonsuz teşekkürlerimi sunarım.

Tez çalışmalarım boyunca her türlü değerli akademik bilgi, tecrübe ve yardımlarını benden esirgemeyen, verdikleri destek ve değerli katkılarından dolayı çok değerli Hocalarım Doç. Dr. Muharrem Tolga SAKALLI ve Dr. Öğretim Üyesi Fatma BÜYÜKSARAÇOĞLU SAKALLI 'ya sonsuz teşekkürlerimi sunarım.

Araştırmalarım sırasında tez çalışmalarıma bilgi birikimleriyle çok büyük ölçüde katkıda bulunan, aynı zamanda değerli yönlendirmeleriyle akademik alanda bana her türlü destek olan saygı değer Hocam Doç. Dr. Sedat AKLEYLEK'e katkılarından dolayı sonsuz teşekkürlerimi sunarım.

Tez çalışmalarımız boyunca engin tecrübelerinden yararlanma fırsatı bulduğumuz, bize her türlü desteği veren Sayın Prof. Dr. Vincent RIJMEN'a teşekkürü borç bilirim.

Tez çalışmamız sırasında değerli bilgilerini bizimle paylaşan ve tez çalışmamıza yön veren çok değerli Dr. Tolga YALÇIN'a teşekkürü borç bilirim.

Tez çalışmalarıma değerli bilgi ve tavsiyeleri ile katkıda bulunan tez ilerleme jürim Dr. Öğretim Üyesi Suhap ŞAHİN'e teşekkürü borç bilirim.

Akademik çalışmalarım boyunca, bana her türlü desteği veren Arş. Gör. Burcu KIR SAVAŞ'a, Arş. Gör. Seda KUL'a ve Arş. Gör. Fulya AKDENİZ' e teşekkürü borç bilirim.

Hayatım boyunca sıkıntılarımı, mutluluklarımı paylaşan, benden maddi manevi desteklerini hiçbir zaman esirgemeyen en büyük destekçilerim, sevgili annem Hatice KURT, babam İbrahim KURT, kardeşim Görkem Mert KURT ve çok değerli eşim Ahmet PEHLİVANOĞLU'na sonsuz teşekkürlerimi sunarım.

Bu çalışmayı destekleyen Kocaeli Üniversitesi Bilimsel Araştırma Projeleri Koordinasyon Birimi'ne (BAP) (Proje No: 2017/078) teşekkürlerimi sunarım. Ayrıca tez çalışmasında geliştirilen GHadamard matris formu ile ilgili Türk Patent ve Marka Kurumu'na 2017/21073 Başvuru Numaralı Patent Başvurusu yapılmıştır.

Temmuz – 2018 Meltem KURT PEHLİVANOĞLU

İÇİNDEKİLER

ÖNSÖZ VE TEŞEKKÜR i
İÇİNDEKİLER ii
ŞEKİLLER DİZİNİ iv
TABLOLAR DİZİNİ v
SİMGELER VE KISALTMALAR DİZİNİ vii
ÖZET ix
ABSTRACT x
GİRİŞ 1
1. BLOK ŞİFRELER 8
1.1. Blok Şifreler ve Tasarım İlkeleri 8
1.2. Kriptanaliz 11
1.2.1. Doğrusal olmama kriteri ve doğrusal kriptanaliz 12
1.2.2. Diferansiyel kriptanaliz 15
1.3. Tasarım Mimarileri 17
1.3.1. SPN mimarisi 17
1.3.2. Feistel mimarisi 19
1.4. AES Şifreleme Algoritması 21
1.4.1. AES şifreleme adımları 22
1.4.2. AES şifre çözme adımları 24
1.4.3. AES anahtar planlama algoritması 25
1.5. Blok Şifre Çalışma Modları 27
1.6. AE (Authenticated Encryption-Kimlik Doğrulamalı Şifre) 30
1.7. Hafif Sıklet Blok Şifreler 33
1.7.1. Literatürde yer alan ve tez kapsamında kullanılan bazı hafif sıklet blok şifreler 36
1.7.2. Hafif sıklet blok şifre çalışma modları ve JAMBU AE çalışma modu 44
2. MATEMATİKSEL ALTYAPI 48
3. BLOK ŞİFRELERİN YAYILIM TABAKALARINDA KULLANILAN DOĞRUSAL DÖNÜŞÜMLER 61
3.1. MDS Matrislerin Tasarımı 64
4. SONLU CİSİMLERDE HADAMARD MATRİSLERİN GENELLEŞTİRİLMESİ 67
4.1. GHadamard Matris Formu 67
4.2. GHadamard Matris Formunun Farklı Boyutlardaki Diğer Matris Tiplerine Uygulanması 73
4.3. GHadamard Matris Formu İle Elde Edilen Sonuçlar 77
5. ELDE EDİLEN SONUÇLAR VE AES ŞİFRESİ KULLANILARAK BİR HAFİF SIKLET BLOK ŞİFRE UYGULAMASI 88
5.1. Geliştirilen Hafif Sıklet Blok Şifrenin Yapısı 88
5.1.1. SLayer adımı 91
5.1.2. SRows adımı 93
5.1.3. MC adımı 93

5.1.4. Anahtar planlama yapısı 94
5.1.5. Örnek şifreleme adımları ve test vektörleri 102
5.1.6. Önerilen hafif sıklet blok şifreleme algoritmasının 8-bit mikrodenetleyici üzerinde yazılım performansı analizi..... 111
5.1.7. Önerilen hafif sıklet blok şifreleme algoritmasının WeMos D1 ESP8266 tabanlıWiFi Modül üzerinde performans analizi 116
5.1.8. Önerilen hafif sıklet blok şifreleme algoritmasının doğrusal ve diferansiyel kriptanalize karşı gücünün test edilmesi..... 121
6. SONUÇLAR VE ÖNERİLER 123
KAYNAKLAR 127
KİŞİSEL YAYIN VE ESERLER 138
ÖZGEÇMİŞ 139

ŞEKİLLER DİZİNİ

Şekil 1.1. Blok şifre yapısı 8
Şekil 1.2. Ürün blok şifre yapısı 10
Şekil 1.3. SPN mimarisi döngü yapısı 18
Şekil 1.4. 1 döngülük SPN mimarisi örneği 19
Şekil 1.5. 2 döngülük Feistel mimarisi örneği 20
Şekil 1.6. AES şifreleme algoritması girdi durumunun oluşturulması 22
Şekil 1.7. Satırların ötelenmesi (Mix Columns) adımı 23
Şekil 1.8. AES-128 için anahtar planlama algoritması 26
Şekil 1.9. AE şifreleme ve şifre çözme blok şemaları 31
Şekil 1.10. AE tasarım yöntemleri 32
Şekil 1.11. Aygıt/cihaz spektrumu 33
Şekil 1.12. Kriptografik algoritmalar için oluşturulan profil özellikleri 35
Şekil 2.1. MDS kod üretimi 54
Şekil 3.1. MDS matris tasarım yöntemleri 65
Şekil 5.1. Önerilen hafif sıklet blok şifrenin şifreleme blok diyagramı 88
Şekil 5.2. Önerilen hafif sıklet blok şifrenin şifre çözme blok diyagramı 90
Şekil 5.3. Önerilen hafif sıklet blok şifre için durum matrisi 90
Şekil 5.4. Önerilen hafif sıklet blok şifre için SRows 93
Şekil 5.5. Önerilen anahtar planlama yapısı (64 bit) 94
Şekil 5.6. Önerilen anahtar planlama algoritmasına ait (64 bit) sözde kodu 95
Şekil 5.7. Önerilen anahtar planlama algoritmasına ait (64 bit) SAC testi sözde kodu 97
Şekil 5.8. 80-bit anahtar için alt anahtarların oluşturulması sözde kodu 99
Şekil 5.9. 128-bit anahtar için alt anahtarların oluşturulması sözde kodu 99

TABLOLAR DİZİNİ

Tablo 1.1. PRESENT S-kutusu 14
Tablo 1.2. PRESENT S-kutusunun LAT tablosu 15
Tablo 1.3. PRESENT S-kutusunun DDT tablosu 16
Tablo 1.4. Satırları öteleme dönüşümünde elde edilen byte permütasyonu 23
Tablo 1.5. AES-128 anahtar planlama algoritmasında kullanılan döngü sabiti değerleri 27
Tablo 1.6. PRESENT S-kutusunun tersi 37
Tablo 1.7. KLEIN S-kutusu 38
Tablo 1.8. KLEIN RotateNibbles adımından sonra 4-bitlik parçaların pozisyonları 39
Tablo 1.9. SIMON ve SPECK şifrelerinin genel özellikleri 41
Tablo 1.10. PRESENT, KLEIN, HIGHT, SIMON, SPECK şifrelerinin yazılım ve donanım performansları 43
Tablo 1.11. 2n-bit blok uzunluğundaki blok şifre üzerine AE modları uygulandığında durum bit uzunluklarının karşılaştırmaları 46
Tablo 2.1. n. dereceden asal polinomlar 50
Tablo 2.2. $GF(2^4)$ ve $GF(2^8)$ sonlu cisimleri için 4. ve 8. dereceden indirgenemez polinomlar 51
Tablo 2.3. $GF(2^4)$ / 0x13 sonlu cisminin elemanları 51
Tablo 2.4. $GF(2^4)$ üzerinde indirgenemez polinomlara göre elemanların çarpma işlemi için XOR sayıları 59
Tablo 3.1. Yayılım tabakasında MDS matris kullanan bazı blok/hafif sıklet blok şifrelerin doğrusal dönüşümleri 63
Tablo 4.1. $GF(2^4)$ / 0x13 sonlu cismi üzerinde 64 XOR sayılı GHadamard matris formunda $[GHad(a_0, a_1; b_1, a_2; b_2, a_3; b_3)]$ 4x4 tersi kendisine eşit MDS matrisler 79
Tablo 4.2. $GF(2^8)$ / 0x1c3 sonlu cismi üzerinde 680 XOR sayılı GHadamard matris formu ile üretilen 8x8 tersi kendisine eşit olmayan MDS matrisler ve b_i parametreleri 84
Tablo 4.3. $GF(2^4)$ ve $GF(2^8)$ sonlu cisimleri üzerinde 4x4 tersi kendisine eşit MDS ve 4x4 tersi kendisine eşit olmayan MDS matrislerin bilinen en iyi XOR sayısı metriğine göre karşılaştırılması 86
Tablo 4.4. $GF(2^4)$ ve $GF(2^8)$ sonlu cisimleri üzerinde 8x8 tersi kendisine eşit MDS ve 8x8 tersi kendisine eşit olmayan MDS matrislerin bilinen en iyi XOR sayısı metriğine göre karşılaştırılması 87
Tablo 5.1. Geliştirilen S-kutusu 91
Tablo 5.2. Geliştirilen S-kutusunun LAT tablosu 91
Tablo 5.3. Geliştirilen S-kutusunun DDT tablosu 92
Tablo 5.4. (27 F8 31 7F 09 AF EF 44) ana anahtarıyla üretilen alt anahtarlar 96
Tablo 5.5. Önerilen anahtar planlama algoritması için SAC testi sonuçları 98
Tablo 5.6. KLEIN-64 ve PRESENT-80 algoritmaları SAC testi

sonuçları 100
Tablo 5.7. Önerilen anahtar planlama yapısı için çığ etkisi analizi 101
Tablo 5.8. Örnek şifreleme anahtar üretme adımları 102
Tablo 5.9. Örnek şifreleme adımları 104
Tablo 5.10. Örnek şifreleme çözme adımları 107
Tablo 5.11. 80-bit için test vektörleri 111
Tablo 5.12. 128-bit için test vektörleri 112
Tablo 5.13. AVR ATmega128 mikrodenetleyici üzerinde yazılım performans karşılaştırmaları 114
Tablo 5.14. İstemci olarak belirlenen WeMos D1 ESP8266 IoT kartı üzerinde JAMBU modunda farklı şifreleme algoritmalarının şifreleme işlemi için gereken çalışma süreleri 117
Tablo 5.15. Sunucu olarak belirlenen WeMos D1 ESP8266 IoT kartı üzerinde JAMBU modunda farklı şifreleme algoritmalarının şifreleme işlemi için gereken çalışma süreleri 119

SİMGELER VE KISALTMALAR DİZİNİ

β	:	Dal Sayısı
$-$	:	Modüler Çıkarma
&	:	AND (Ve)
$\oplus$	:	Exclusive Or (Özel Veya)
+	:	Modüler Toplama
F_A	:	A matrisinin Sabit Nokta Sayısı
I	:	Birim Matris
wt	:	Hamming Ağırlığı
$\phi()$	:	Euler phi Fonksiyonu

Kısaltmalar

AD	:	Associated Data (İlişkili Veri)
AE	:	Authenticated Encryption (Kimlik Doğrulamalı Şifreleme)
AEAD	:	Authenticated Encryption with Associated Data (İlişkili Veri ile Kimlik Doğrulamalı Şifreleme)
AES	:	Advanced Encryption Standard (Gelişmiş Şifreleme Standardı)
ASIC	:	Application Specific Integrated Circuit (Uygulamaya Özel Tümleşik Devre)
BIC	:	Bit Independence Criterion (Bit Bağımsızlık Kriteri)
CBC	:	Cipher Block Chaining (Şifre Bloğu Zincirleme)
CCM	:	Counter with CBC-MAC Mode (CBC-MAC Sayaç Modu)
CFB	:	Cipher Feedback (Şifreyi Geri Besleme)
circ()	:	Circulant Matrix (Dairesel Matris)
CTR	:	Counter (Sayaç)
DDT	:	Difference Distribution Table-XOR Table (Fark Dağılım Tablosu-XOR Tablosu)
DES	:	Data Encryption Standard (Veri Şifreleme Standardı)
DP	:	Differential Probability (Diferansiyel Olasılık)
ECB	:	Electronic Code Book (Elektronik Kod Defteri)
FF	:	Finite Field (Sonlu Cisim)
GCM	:	Galois/Counter Mode (Galois/Sayaç Modu)
GCM-SIV	:	Galois/Counter Mode-Synthetic Initialization Vector (Galois/Sayaç Modu-Sentetik İlklendirme Vektörü)
GE	:	Gate Equivalent (Kapı Eşdeğerleri)
GF	:	Galois Field (Galois Cismi)
GFN	:	Generalised Feistel Network (Genelleştirilmiş Feistel Ağları)
Ghad()	:	Generalized Hadamard Matrix (Genelleştirilmiş Hadamard Matris)
had()	:	Hadamard Matrix (Hadamard Matris)
IoT	:	Internet of Things (Nesnelerin İnterneti)
IPSec	:	Internet Protocol Security (İnternet Protokolü Güvenliği)

ISO/IEC	:	International Organization for Standardization/International Electrotechnical Commission (Uluslararası Standartlar Teşkilâtı/Uluslararası Elektroteknik Komisyonu)
IV	:	Initialization Vector (İlklendirme Vektörü)
Kbps	:	Kilobit per Second (Saniye Başına Kilobit)
LAT	:	Lineer Approximation Table (Doğrusal Yaklaşım Tablosu)
MAC	:	Message Authentication Code (Mesaj Doğrulama Kodu)
MDBL	:	Maximum Distance Binary Linear Codes (Maksimum Uzaklıkta İkili Doğrusal Kodlar)
MDS	:	Maximum Distance Separable (Maksimum Uzaklıkla Ayrılabilen)
ms	:	Millisecond (Milisaniye)
NIST	:	National Institute of Standards and Technology (Ulusal Standartlar ve Teknoloji Enstitüsü)
NLM	:	Non-Linear Minimization (Doğrusal Olmama Minimizasyonu)
NSA	:	National Security Agency (Ulusal Güvenlik Ajansı)
OCB	:	Offset Code Book Mode (Ofset Kod Defteri)
OFB	:	Output Feedback (Çıkışı Geri Besleme)
OTP	:	OneTime Pad (Tek Kullanımlık Şerit)
PCBC	:	Propagating Cipher Block Chaining (Yayılan Şifre Bloğu Zincirleme)
RAM	:	Random Access Memory (Rastgele Erişimli Bellek)
RCon	:	Round Constant (Döngü Sabiti)
RFID	:	Radio Frequency Identification (Radyo Frekansı ile Tanımlama)
ROM	:	Read Only Memory (Sadece Okunabilir Bellek)
SAC	:	Strict Avalanche Criterion (Katı Çığ Kriteri)
S-box	:	Substitution Box (Yer Değiştirme Kutusu)
SPN	:	Substitution-Permutation Network (Yer Değiştirme-Permütasyon Ağı)
SRAM	:	Static Random Access Memory (Durağan Rastgele Erişimli Bellek)
SSH	:	Secure Shell (Güveli Kabuk)
SSL	:	Secure Sockets Layer (Güvenli Giriş Katmanı)
TLS	:	Transport Layer Secuity (Taşıma Katmanı Güvenliği)
VHDL	:	Very high speed integrated circuit Hardware Description Language (Yüksek Hızlı Tümleşik Devreler İçin Donanım Tanımlama Dili)
WSNs	:	Wireless Sensor Networks (Kablosuz Sensör Ağları)
XCBC	:	Extended Cipher Block Chaining Encryption (Genişletilmiş Şifre Bloğu Zincirleme)
XOR	:	Exclusive Or (Özel Veya)

MAKSİMUM UZAKLIKTA AYRILABİLEN MATRİSLERİN ELDE EDİLEBİLMESİ İÇİN YENİ BİR MATRİS FORMU VE BİR HAFİF SIKLET BLOK ŞİFREYE UYGULAMASI

ÖZET

Kaynak kısıtlı cihazların kullanımı gün geçtikçe artmaktadır. Ancak bu cihazların kullanımı bazı güvenlik risklerini de getirir. Son yıllarda, bu risklerle başa çıkabilmek için hafif sıklet olarak adlandırılan kriptosistemler geliştirilmiştir.

Hafif sıklet kriptografideki en önemli ve en maliyetli bileşenlerden biri yayılım tabakasıdır. Bu nedenle, minimum sayıda donanım elemanıyla, özellikle düşük XOR sayısı ile oluşturulan yayılım tabakalarının tasarımı literatürdeki açık problemlerden biridir.

Bu tez çalışmasında hafif sıklet blok şifrelerin yayılım tabakalarında kullanılabilecek, minimum XOR sayılı (tersi kendisine eşit) maksimum uzaklıkta ayrılabilen (MDS) matrislerin üretilmesi için Genelleştirilmiş Hadamard (Generalized Hadamard-GHadamard) olarak adlandırılan yeni bir matris formu geliştirilmiştir. GHadamard matris formu ile Hadamard matris formu genişletilmiş ve genelleştirilmiştir.

GHadamard matris formu kullanılarak, $GF(2^4)$ üzerinde 8×8 tersi kendisine eşit MDS ve tersi kendisine eşit olmayan MDS matrisler için bilinen en iyi XOR sayısı sonuçları iyileştirilmiştir. $GF(2^4)$ ve $GF(2^8)$ üzerinde bilinen en iyi XOR sayılı, yeni 4×4 tersi kendisine eşit MDS matrisler üretilmiştir. Dahası, $GF(2^8)$ üzerinde 8×8 GHadamard involutif olmayan MDS matrisler için bilinen en iyi XOR sayısı sonucu verilmiştir.

Tezin ikinci kısmında, yazılım verimli yeni bir hafif sıklet blok şifreleme algoritması tasarlanmıştır. Önerilen hafif sıklet blok şifreleme algoritmasının yayılım tabakası, karıştırma tabakası ve anahtar planlama algoritması için yeni bileşenler geliştirilmiştir. Önerilen hafif sıklet blok şifre kaynak kısıtlı cihaza güvenli iletişim için uygulanmıştır.

Anahtar Kelimeler: GHadamard Matris, Hadamard Matrisler, Hafif Sıklet Blok Şifreleme Algoritması, Hafif Sıklet MDS Matrisler, İnvolutif MDS Matrisler.

A NEW MATRIX FORM TO OBTAIN MAXIMUM DISTANCE SEPARABLE MATRICES AND ITS IMPLEMENTATION IN A LIGHTWEIGHT BLOCK CIPHER

ABSTRACT

The usage of resource-constrained devices has been increasing day by day. However, the usage of these devices also brings some security risks. In recent years, cryptosystems called lightweight have been developed to cope with these risks.

One of the most important and most costly components in a lightweight cryptography is the diffusion layer. Therefore, the design of the diffusion layers to be formed with a minimum number of hardware elements, especially with low exclusive OR (XOR) count, is one of the open problems in the literature.

In this thesis, a new matrix form called Generalized Hadamard (GHadamard) has been developed to generate new (involutory) Maximum Distance Separable (MDS) matrices with minimum XOR count which can be used in diffusion layers of lightweight the block ciphers. Hadamard matrix form has been extended and generalized with the GHadamard matrix form.

The best-known results of XOR count for 8×8 involutory MDS and non-involutory MDS matrices over GF(2^4) have been improved by using GHadamard matrix form. The new 4×4 involutory MDS matrices over GF(2^4) and GF(2^8) with the best-known XOR count have been generated. Moreover, the best-known result of XOR count for 8×8 GHadamard non-involutory MDS matrices over GF(2^8) has been given.

In the second part of the thesis, a new software-targeted lightweight block cipher has been designed. The new components have been developed for the diffusion layer, confusion layer and key scheduling algorithm of the proposed lightweight block cipher. The proposed lightweight block cipher has been applied for secure communication to the resource-constrained device.

Keywords: GHadamard Matrix, Hadamard Matrices, Lightweight Block Cipher Algorithm, Lightweight MDS Matrices, Involutory MDS Matrices.

GİRİŞ

Bilgisayarların ve iletişim teknolojilerinin hızlı değişim ve gelişimi modern hayatı etkileyen en önemli teknolojik gelişmelerden ikisidir. Özellikle hesaplama gücü yüksek, boyutu ve güç tüketimi düşük bilgisayarlar hayatımızın bir parçası haline gelmiştir. Tüm bu yenilikler, güvensiz haberleşme ortamında güvenli iletişimi sağlayan modern kriptolojinin temelini oluşturmakla kalmayıp, bu alanda yeni çalışmaların yapılmasına olanak sağlamıştır.

Yunanca "Kryptós", "lógos" kelimelerinden oluşan yani gizli bilgi anlamına gelen kriptoloji, verilerin güvenle, gizli biçimde iletimini ve saklanmasını inceleyen bilim dalıdır [1]. Kriptoloji iki temel kavramı içinde barındırır: kriptografi ve kriptanaliz. Kriptografi, matematiksel yöntemler kullanılarak bilginin gizliliğini sağlarken, kriptanaliz kriptografik algoritmaların zayıflığını bulmayı amaçlar.

"Bilgi güçtür" sözünden yola çıkılarak yüzyıllardır insanoğlu elindeki bilgiyi saklamak için birçok tekniğe başvurmuştur, bu nedenle kriptografinin tarihi antik dönemlerde bilgiyi saklayan geleneksel yöntemlerle başlayıp, 20. yüzyılın başlarında modern şifreleme teknikleriyle günümüze uzanmıştır. Özellikle yazının icadıyla birlikte bilgi saklanabilir hale gelmiştir. İlk gizli yazı formu, hiyerogliflerin kullanıldığı M.Ö. 4000'lerde Antik Mısır'a dayanır. M.Ö. 500'lü yıllarda Antik Yunan'da Spartalılar tarihteki ilk yer değiştirme şifresi olarak bilinen "scytale" adında etrafına parşömen kâğıdı sarılı bir silindir kullanmışlardır. M.Ö. 50'li yıllarda Julius Ceaser tarafından kullanılan Sezar şifresi ilk yerine koyma şifresi olarak bilinir. Bu şifreleme sisteminde her harf kendinden sonra gelen üçüncü harfle yer değiştirir. 1400'lü yıllarda Leon Alberti Sezar şifresine benzer yapıda, şifre çözmek için Alberti Diski'nin kullanıldığı bir şifreleme yöntemi geliştirmiştir. 1500'lü yılların sonunda Sezar şifresinden esinlenilerek geliştirilen Vigenere Sistemi, şifreleme işleminde bu şifre için aynı ismi taşıyan Vigenere Tablosu'nu kullanır, şifre 1800'lü yılların ortalarına kadar kırılamaz olarak kabul edilmiş ancak 1854 yılında frekans testi ile kırılmıştır. Geleneksel kriptografik algoritmalara bir başka tarihsel örnek İkinci

Dünya Savaşı sırasında kullanılan Alman Enigma makinesidir, savaş sırasında Enigma Polonyalı ve İngiliz kriptograflar tarafından kırılmış bu da Batı Müttefiklerinin savaş sırasında iletilen mesajları okumasına olanak sağlamıştır [2]. Modern kriptografik algoritmaların geliştirilmeye başlandığı dönemlerde bu algoritmalar genellikle askeri, diplomatik işlerde mesajları gizlemek için kullanılırken, 1970'lerde özellikle DES (Data Encryption Standard-Veri Şifreleme Standardı) [3] şifreleme algoritmasının yayınlanması ve simetrik şifrelemenin bulunmasıyla akademik alanda modern kriptografi ile ilgili çalışmalar yapılmaya başlanmıştır [2]. 1981'de ilk kişisel bilgisayarın tanıtılması, İnternetin 1990'ların başında yükselişi, artan IoT cihaz sayısı modern kriptografinin kitlesel kullanımı için bir basamak olmuştur.

Bugün milyarlarca kullanıcı birbirleriyle güvensiz haberleşme ortamı üzerinden sayısız bilgi paylaşmaktadır. Kriptografi, bu güvensiz ortamda bilginin tam, doğru ve eksiksiz olarak karşı tarafa ulaşılmasını garanti eder çünkü pratikte iletilen mesajı sadece gizli tutmak yeterli değildir, aynı zamanda mesajın içeriğinin değiştirilip değiştirilmediğinin belirlenmesi gerekir [2].

Gönderici-alıcı kanalları arasında güvenli bir haberleşme kanalının kurulması için güvenlik servislerinden faydalanılır, bilgi güvenliği servisleri matematiksel kriptografik algoritmaları kullanır. Güvenli bir iletişim kanalı kurulması için dört temel bilgi güvenliği servisi vardır bunlar gizlilik, bütünlük, kimlik doğrulama ve inkâr edememedir. Gizlilik, yalnızca yetkili kişilerin bilginin içeriğine erişebilmelerini sağlar böylece yetkisiz bir kullanıcının korunan bilgilerin içeriğine erişmesi önlenmiş olur. Bütünlük, verinin yetkisiz bir tarafça ekleme, silme, değiştirme işlemlerinin yapılmadığını garanti eder. Kimlik doğrulama en basit tanımıyla mesajı gönderenin kimliğinden emin olmayı sağlar. Kimlik doğrulama, veri kaynağı ve varlık kimlik doğrulama olarak ikiye ayrılır [4]. Veri kaynağı kimlik doğrulaması bir varlığın gönderilen bir mesajın orijinal kaynağı olduğuna dair güvence verir. Veri kaynağı kimlik doğrulaması bazen mesaj kimlik doğrulaması olarak adlandırılır ve dolaylı olarak veri bütünlüğünü sağlar. Varlık kimlik doğrulaması ile etkileşimde bulunulan ikinci bir varlığın kimliği hakkında güvence verilir ve varlık kimlik doğrulaması genellikle veri kaynağı kimlik doğrulaması gerektirir. İnkâr edememe yapılan bir eylemi veya taahhüdü reddetmeyi engelleyen

bir güvenlik hizmetidir, anlaşmazlıklara yol açabilecek durumlarda faydalıdır çünkü eylemle ilgili gerekli kanıtlar sunulabilir. Bu güvenlik servislerine ek olarak erişim kontrolü, anonimlik, sayısal imzalar gibi diğer güvenlik hizmetleri ile güvenli haberleşme ortamı sağlanabilir [4].

Kaynak kısıtlı cihazların kullanımı gün geçtikçe artmaktadır. Kaynak kısıtlıdan kasıt; önemli ölçüde azaltılmış güç tüketimi, daha az hesaplama gücü gereksinimi, daha az bellek kapasitesi tüketimi ve daha az güç kaynağı kullanımıdır. Ancak bu kısıtlar cihazların kullanımında bazı güvenlik risklerini de beraberinde getirir [5]. Bu risklerle başa çıkabilmek için kriptografik algoritmaların kullanımı kaçınılmazdır. Özellikle IoT (Internet of Things-Nesnelerin İnterneti), WSNs (Wireless Sensor Networks-Kablosuz Sensör Ağları), RFID (Radio Frequency Identification-Radyo Frekansı ile Tanımlama) etiketleri gibi kaynak kısıtlı cihazların güvenliğinin sağlanmasında hafif sıklet (lightweight) şifreleme algoritmaları kullanılmaktadır.

Bu tez çalışması kaynak kısıtlı cihazların güvenliğinde kullanılabilecek hafif sıklet blok şifrelerin bileşenlerinin geliştirilmesi üzerinedir. Bir hafif sıklet blok şifre üç temel bileşenden oluşur, bunlar; karıştırma tabakası, yayılım tabakası ve anahtar planlamadır. Blok şifredeki yayılma işlemi doğrusal dönüşümlerle gerçekleştirilir. Blok şifreler için en maliyetli (hesaplama gücü, güç tüketimi açısından) bileşen ise yayılım tabakalarıdır. Bu tez kapsamında yayılım elemanı olarak kullanılan doğrusal dönüşümler için düşük güç tüketimi özelliği "XOR (Exclusive Or - Özel Veya) sayısı" [6] anlamında ele alınmıştır. Çünkü düşük güç tüketimi gerektiren doğrusal dönüşümlerin elde edilmesi günümüzde üzerinde çalışılan açık bir problemdir. Gelecekte, hızla kullanımı artan kaynak kısıtlı cihazlar için, düşük güç tüketimiyle gerçeklenen şifre bileşenlerinin kullanımı gereklilik haline gelecektir.

Blok şifrelerin yayılım tabakalarının tasarımında kullanılan yapılar incelendiğinde üç temel yapı dikkat çeker bunlar çoklu permütasyon tabanlı yayılım tabakaları, bit permütasyonu ile oluşturulan yayılım tabakaları ve dalların karıştırılması (shuffles) ile oluşturulan yayılım tabakalarıdır [7]. Tez kapsamında özellikle çoklu permütasyon tabanlı yayılım tabakalarının tasarımında kullanılan MDS (Maximum Distance Separable-Maksimum Uzaklıkla Ayrılabilen) ve involutif (tersi kendisine eşit) MDS matrisler üzerine çalışılmıştır. Tersi kendisine eşit MDS matrislerin

kullanımı şifreleme ve şifre çözme işlemleri için aynı uygulama maliyeti sağlarken, aynı zamanda şifrenin kapladığı alandan tasarruf sağlar.

MDS matrislerin üretimi için literatürdeki yöntemler incelendiğinde mevcut yöntemlerin sınırlı olduğu açıktır; bunlar rastgele üretim yöntemi [8] ve özel matris formlarının [9-13] kullanıldığı yöntemlerdir. Rastgele üretim yönteminde üretilen matrislerin tüm alt kare matrislerinin de MDS özelliği kontrol edilir bu da arama maliyetini arttırır, ayrıca MDS matris elemanlarının aranacağı uzay çok büyük olduğu için verimlilik, alan, hız ve performans metrikleri açısından sistem kaynaklarının kısıtlı olduğu koşullarda bu yöntemin uygulanabilirliği neredeyse imkânsızdır. Özel matris formlarının kullanıldığı yöntemde ise rastgele üretim yönteminin aksine özel matris formları ve kodları kullanılarak MDS matrislerin aranacağı uzay küçültülür. Bu yöntemde Gabidulin [14] ve BCH (Bose-Chaudri-Hocquenghem) [15] kodları, Cauchy [8], Vandermonde [16] ve Companion [17] matrisleri kullanılarak MDS matrisler doğrudan üretilebilir, diğer bir deyişle arama maliyeti gerektirmez. Hadamard [11], Circulant (Dairesel) [9,10], Toeplitz [12], Circulant-like (Dairesel-benzeri) [13] özel matris formları ile üretilen MDS matrisler için ise arama maliyeti gerekir çünkü bu yapılar ile doğrudan MDS matrisler üretilemez. Tersi kendisine eşit MDS matrislerin üretilebilmesi için ise rastgele üretim yöntemi ve Hadamard matrislerin kullanımının dışında çok fazla bilinen yöntem yoktur. Ancak, hem Hadamard matrislerin kullanımı hem de rastgele üretim yöntemi ile üretilecek tersi kendisine eşit MDS matrisler için arama maliyeti gerekir. Bu da özellikle hafif sıklet tasarımlar için; tersi kendisine eşit MDS matrislerin üretilmesini sağlayan, arama maliyeti olmadan doğrudan üretim yapabilecek, bir taraftan da tersi kendisine eşit MDS matrislerin üretilmesini sağlayacak (Hadamard matris veya rastgele arama yöntemlerinden birinin kullanımı) yapıların bir arada kullanıldığı hibrit yapılara ihtiyaç duyulduğu açıktır. Literatürdeki çalışmalar [9,10] incelendiğinde arama maliyeti olmadan tersi kendisine eşit MDS matrislerin üretimini sağlayacak hibrit yapılara rastlanmamıştır. Öte yandan, yine hafif sıklet blok şifreler için donanım maliyeti açısından minimum XOR sayısı maliyetli (tersi kendisine eşit) MDS matrislerin üretilebilmesi için, genelleştirilmiş yöntem ya da form yapısının olmaması literatürdeki bir diğer açık problemlerden biridir [12,18].

Tez kapsamında (hafif sıklet) blok şifrelerin yayılım tabakalarında kullanılabilecek, minimum XOR sayısı maliyetli (tersi kendisine eşit) MDS matrislerin üretilebilmesi için Generalized Hadamard (GHadamard - Genelleştirilmiş Hadamard) [19] olarak adlandırılan bir matris formu geliştirilmiştir. Geliştirilen yapı hibrit bir yöntem olup, temelinde özel matris formlarından biri olan Hadamard matrisleri kullanır ve arama yapmadan doğrudan GHadamard formda yeni (tersi kendisine eşit) MDS matrisler üretir. Hadamard matrislerin kullanılmasındaki temel neden Hadamard formun tersi kendisine eşit MDS matrislerin üretiminin temelini oluşturmasıdır. GHadamard matris formu ile Hadamard matris tanımı genelleştirilmiş ve genişletilmiştir. Bu form ile MDS matrislerin üretilebilmesi için hibrit üretim yöntem eksikliği ve minimum XOR sayılı (tersi kendisine eşit) MDS matrislerin üretilebilmesi için genel yapıların eksikliği açık problemlerine çözüm önerisi getirilerek literatüre önemli bir katkı sağlanmıştır.

GHadamard matris formu ile elde edilen önemli yenilikler sıralandığında bunlar;

- GHadamard matris formu (tersi kendisine eşit) MDS matrislerin üretilmesi için önerilmiş bir matris formudur,
- Herhangi bir Hadamard (tersi kendisine eşit) MDS matristen GHadamard formu ile yeni (tersi kendisine eşit) MDS matrisler üretilebilir,
- k tek veya çift sayı olmak üzere, GHadamard matris formu $k \times k$ boyutunda matrislere uygulanabilir,
- I birim matris ve c sonlu cisim elemanı, H Hadamard matris olmak üzere; H Hadamard matris olduğundan $H^2 = c^2 I$ özelliğini sağlar, GHadamard matris formu ile üretilen matrisler bu özelliği korur. Bu nedenle GH GHadamard formda matris olmak üzere $(GH)^2 = c^2 I$'dır. Bu da GHadamard matris formu ile Hadamard matris formunun üretilebildiğini gösterir, diğer bir deyişle GHadamard matris formu ile Hadamard matris formu genelleştirilmiştir.
- $GF(2^4)$ üzerinde 4×4 tersi kendisine eşit MDS matrisler için bilinen en iyi XOR sayısı 64 [20] XOR sayısına eşit literatürde yer almayan GHadamard formda yeni matrisler üretilmiştir,
- $GF(2^8)$ üzerinde 4×4 tersi kendisine eşit MDS matrisler için bilinen en iyi XOR sayısı 160 XOR [12], 158'e düşürülmüştür,

- GF(2^4) üzerinde 8×8 tersi kendisine eşit MDS matrisler için bilinen en iyi XOR sayısı 512 XOR [18], 407'ye düşürülmüştür,
- GF(2^4) üzerinde 8×8 MDS matrisler için bilinen en iyi XOR sayısı 432 XOR [18], 380'e düşürülmüştür,
- GF(2^8) üzerinde 8×8 MDS matrisler için bilinen en iyi XOR sayısı 680 [12] XOR sayısına eşit literatürde yer almayan GHadamard formda yeni matrisler üretilmiştir.

Tez kapsamında GHadamard matris formu ile üretilen yeni yapılar, literatüre özgün bir katkı sağlayacak ve ileride yapılacak çalışmalar için bir rehber niteliği taşıyacaktır.

Bu tez çalışmasında hafif sıklet blok şifrelerin karıştırma tabakaları ve anahtar planlama yapıları da ele alınmıştır. Literatürde yer alan yapılar incelenerek, bu yapılar için eksiklikler analiz edilmiştir. Bu kapsamda karıştırma tabakası ve anahtar planlama yapıları için, literatürde yer almayan yeni yapılar önerilmiştir. Karıştırma tabakası için, yeni bir tersi kendisine eşit 4-bit S-kutusu (Substitution Box - Yer Değiştirme Kutusu) [21] verilmiştir. Anahtar planlama yapısı ise, AES (Advanced Encryption Standard - Gelişmiş Şifreleme Standardı) [22] şifreleme algoritmasının anahtar planlama yapısından esinlenerek tasarlanmıştır. Geliştirilen anahtar planlama algoritması, SAC (Strict Avalanche Criterion-Katı Çığ Kriteri) [21] testinden %94 başarıyla geçmiştir. Elde edilen tüm bu özgün yapılar kullanılarak, AES şifreleme algoritmasının tasarım yapısından ilham alınarak, IoT cihazlar için yazılım verimli hafif sıklet blok şifreleme algoritması geliştirilmiştir.

Tez çalışmasının ilk bölümünde blok şifrelerin tasarım ilkelleri, kriptanaliz kavramı, doğrusal ve diferansiyel kriptanaliz hakkında ayrıntılı bilgi verilmektedir. Aynı bölümde blok şifre tasarım mimarileri, AES şifreleme algoritmasının yapısı, blok şifre çalışma modları, kimlik doğrulamalı şifreleme, hafif sıklet blok şifreleme algoritmaları ile literatürde yer alan hafif sıklet blok şifreleme algoritmaları hakkında bilgi verilmektedir. İkinci bölümde, tez kapsamında kullanılacak matematiksel ifadelerin anlaşılır olması açısından matematiksel bir alt yapı verilmektedir. Tezin üçüncü bölümünde ise literatürde yer alan hafif sıklet blok şifreleme algoritmalarının

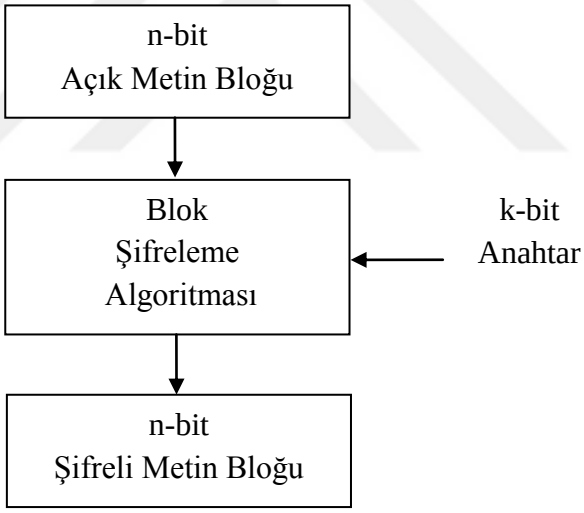
doğrusal dönüşüm yapıları incelenmektedir. Dördüncü bölümde ise hafif sıklet blok şifreler için doğrusal dönüşüm yapısı olarak kullanılabilecek GHadamard matris formu önerilmektedir. Tezin beşinci bölümünde AES şifreleme algoritmasından esinlenerek geliştirilmiş özgün bileşenlere sahip yazılım verimli hafif sıklet şifreleme algoritması verilmektedir. Bu bölümde önerilen yapı ile literatürde yer alan bazı hafif sıklet blok şifreleme algoritmalarının yazılım performansları AVR ATmega128 8-bit mikrodenetleyici üzerinde karşılaştırılmaktadır. Aynı zamanda geliştirilen şifreleme algoritması WeMos D1 ESP8266 tabanlı IoT cihaz üzerinde güvenli veri aktarımı için kullanılacaktır. Sonuçlar ve öneriler bölümünde yapılan çalışmalardan elde edilen sonuçlar analiz edilerek, önerilen yapıların bilime ve gelecekte yapılacak çalışmalara sağlayabileceği katkıları tartışılacaktır. Aynı bölümde gelecekte yapılabilecek çalışmalar için önerilerde bulunulacaktır.

Bu doktora tez çalışması kapsamında geliştirilen tüm yapılar ve çalışmalar özgün değere sahiptir. Tez kapsamında geliştirilen ve önerilen özel bir matris formu olan GHadamard matris formu, Hadamard matris formunu genelleştiren bir formdur. Aynı zamanda bu form ile üretilen matrisler hafif sıklet blok şifreler için minimum XOR sayısı maliyetli doğrusal dönüşüm yapılarıdır. Diğer taraftan önerilen form ile farklı özel matris formları da üretilmektedir. Literatürde yapılan çalışmalar incelendiğinde önerilen form yapısında benzer bir çalışma görülmemiştir. Bu nedenle, bu tez çalışmasının oldukça özgün bir değere sahip olduğu ve bu matris formuyla üretilen minimum XOR sayısı maliyetli doğrusal dönüşüm matrislerinin hafif sıklet blok şifreler için verimli yayılım tabakaları inşaa edeceği açıktır. Bu nedenle bu tezin literatüre ve gelecekte yapılacak çalışmalara önemli katkılar sağlayacağı düşünülmektedir.

1. BLOK ŞİFRELER

Simetrik şifreleme algoritmaları, şifreleme anahtarının şifre çözme anahtarından hesaplanabildiği, anahtarın gizliliği prensibine dayanan algoritmalardır. Simetrik algoritmalarda şifreleme ve şifre çözme anahtarları aynıdır. Gönderici ve alıcı tarafları güvenli iletişime başlamadan önce bir anahtar üzerinde anlaşmalı, iletişimin gizli kalması gerektiği sürece, anahtar gizli kalmalıdır. Simetrik şifreler, blok ve akan şifreler olmak üzere iki gruba ayrılır.

Blok şifreler n-bit uzunluğundaki açık metin bloklarını, k-bit anahtar yardımıyla, n-bit uzunluğundaki şifreli metin bloklarına dönüştürürler. Şekil 1.1'de bir blok şifrenin genel yapısı verilmiştir.



Şekil 1.1. Blok şifre yapısı

Blok şifrelere ait ayrıntılı bilgiler alt bölümlerde verilmiştir.

1.1. Blok Şifreler ve Tasarım İlkelleri

Auguste Kerckhoffs, 1883 yılında yayınladığı iki makalesinde [23,24] kriptosistemin altı aksiyomundan bahsetmiştir. Kriptografik sistem için bu aksiyomlar şu şekilde özetlenebilir:

- Sistem matematiksel ve fiziksel olarak çözülemez olmalı,

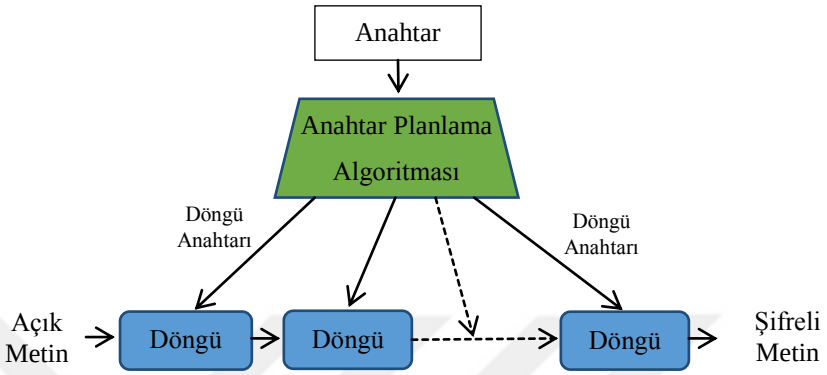
- Gizli algoritmalara veya parametrelere güvenilmemeli diğer bir deyişle yöntemin gizli tutulması gerekmez,
- Anahtar kolay (etkin) bir şekilde iletilmeli,
- Telekomünikasyona uygun olmalı,
- Basit uygulamalara izin vermeli,
- Kullanıcı dostu olmalı ve kullanımı teknik açıdan zor olmamalıdır.

İkinci aksiyom Kerckhoffs'un prensibi olarak bilinir. Bu prensip güvenliğin anahtarın gizliliğinden geldiğini tanımlar, daha sonra Claude Shannon bu prensibi "Düşman sistemi bilir" olarak formüle etmiştir bu söylem de Shannon'un Maxim'i olarak bilinir. Bu yaklaşım sadece blok şifreler için değil bütün modern şifreleme algoritmalarının (akan, simetrik, hash) tasarımı için önemli bir ilkedir [7]. Bu ilke ile tasarlanan bir şifre için, şifrenin saldırılara karşı direnci sadece anahtarın gizliliğine bağlıdır, şifreleme ve şifre çözme algoritmasının gizliliğine ihtiyaç yoktur çünkü anahtarı tahmin etmek zordur, modern şifreleme algoritmalarında anahtar aralığı oldukça büyüktür ve bu da saldırganın anahtarı bulmasını zorlaştırır. Büyük oranda açık metin/şifreli metin çiftleri bilinse bile anahtarı belirlemek mümkün değilse o şifre güvenli kabul edilir.

Bir blok şifre, sabit uzunluktaki blok (bit dizileri) kümelerine sırasıyla seçilen anahtarla, bir permütasyonu uygulayan ve tüm bu işlemleri geri alan şifreleme ve şifre çözme adı verilen bir çift kararlı (deterministik) algoritmadır: Şifreleme, açık metni şifreli metne, şifre çözme işlemi ise şifreli metni orijinal açık metne dönüştürür [7].

Bir blok şifre, tek başına kullanıldığında yetersiz olan ancak bir araya geldiğinde çok güçlü işlevleri oluşturan işlemlerin (işlemlerin tümüne bir döngü denir) art arda kullanımıyla oluşturulur, bu yapı product (ürün) şifre olarak adlandırılır. Shannon bu yapıyı ilk kez 1949 yılındaki çalışmasında tanımlamıştır [25]: bir ürün şifre şifreleme algoritması döngü adı verilen yapılardan oluşur, ilk döngünün girdisi açık metindir, her döngünün çıktısı kendisinden sonra gelen döngünün girdisini oluşturur ve son döngünün çıktısı şifreli metindir. Her döngüde, anahtar planlama algoritması kullanılarak şifreleme anahtarından üretilen döngü anahtarı eklenir. Döngü anahtarının kullanılmasındaki amaç, döngü fonksiyonu içinde kendisinden önce

yapılan adımların tekrarlanmasıyla elde edilecek yolların/izlerin (simetrinin) bozulmasını sağlamaktır. Şekil 1.2'de bir ürün blok şifre yapısı verilmiştir. Bir blok şifredeki istenilen kriptografik güçlülük sağlanana kadar döngü sayısı tekrarlanır bu da tekrarlanan ürün blok şifre yapısını oluşturur [25].



Şekil 1.2. Ürün blok şifre yapısı

Bir blok şifredeki döngüler inşa edilirken döngü yapısı tamamen doğrusal olmamalıdır, ayrıca döngü işlemleri içinde yüksek maliyet gerektiren işlemlerden kaçınılmalı diğer bir deyişle döngü etkin uygulanabilmeli ve yapı yapılan saldırılara karşı dirençli olmalıdır.

Shannon aynı makalesinde [25] bir döngünün confusion (karıştırma) ve diffusion (yayılma) özelliklerini sağlaması gerektiğini belirtmiştir. Karıştırma şifreli metin ile anahtar arasındaki ilişkiyi karmaşık hale getirmeyi amaçlar yani şifreli metnin her bir parçası mümkün olduğunca anahtara bağlıdır denir, yayılma ise açık metnin istatistiksel yapısının şifreli metin üzerinde yayılmasını sağlamak için kullanılır yani şifreli metnin bitleri mümkün olduğunca açık metnin bitlerine bağlıdır denir. Bir blok şifrede karıştırma yer değiştirme işlemleri ile gerçekleştirilirken yayılma ise doğrusal dönüşüm işlemleri ile gerçekleştirilir. Karıştırma ve yayılma özellikleri bir şifreyi istatistiksel saldırılara karşı korur. Döngü yapısının doğrusal olmama özelliğine ek olarak çığ etkisi özelliğini sağlaması da önemlidir. Çığ etkisi ile anahtar veya açık metin üzerindeki 1-bitlik pozisyon değişimi ile şifreli metin bloğunun $1/2$ olasılıkla değişmesi beklenir. Webster ve Tavares bu özelliği [21]'de SAC (Strict Avalanche Criterion - Katı Çığ Kriteri) ile formüle etmişlerdir bu özelliğe göre girdideki (anahtar veya açık metin) her 1-bitlik pozisyon değişimi, çıktıdaki her bir biti

yaklaşık 1/2 olasılıkla değiştirmelidir. Diğer bir ölçüt ise BIC'tir (Bit Independence Criterion - Bit Bağımsızlık Kriteri), bu kriterde girdideki herhangi bir bit değiştirildiğinde (ters çevrildiğinde) çıktı bitlerinin durumları bağımsız olarak değişmelidir. Shannon ayrıca çalışmasında [25] perfect secrecy (mükemmel gizlilik) kavramını ortaya koymuştur bu ancak bir şifrenin anahtar uzunluğu mesajın uzunluğuna eşit olduğu ve bir anahtar sadece bir kez kullanıldığı sürece (aksi halde aynı açık metni şifrelemek için kullanılan aynı anahtarlar bilgi sızıntısına neden olur) sağlanabilir. 19. yüzyılın sonlarına doğru geliştirilen OTP (OneTime Pad - Tek Kullanımlık Şerit) tabanlı şifrelemede anahtar uzunluğu mesajın uzunluğu kadar seçilir bu nedenle mükemmel gizlilik sağlar. Shannon ayrıca iki ayrı tasarım ilkesi önermiştir bunlardan ilki indirgemedir, sistem güvenliğini bilinen bazı zor problemlere indirgeme olarak tanımlanır ve genellikle simetrik şifrelemede kullanılır. Diğer bir prensip ise sistemi bilinen tüm saldırılara karşı güvenli hale getirmektir, bugün bu prensip özellikle gizli anahtarlı şifreleme sistemleri için bilinen en iyi tasarım prensibidir.

1.2. Kriptanaliz

Bruce Schneier'e göre iyi bir şifreleme algoritması tasarımı için iyi bir kriptanalist olmak gerekir [26]. Kriptanalizde saldırganın kriptosistemi bildiği varsayılarak açık metin veya anahtar elde edilmeye çalışılır. Saldırgan hesaplama gücü, depolama kapasitesi ve erişebileceği mesaj miktarıyla sınırlıdır bu nedenle uygulamada kullanılan bir şifre için hesaplanabilir güvenliği esas almak gerekir.

Blok şifrelerde kullanılan anahtar uzunluğu sınırlıdır bu nedenle saldırgan anahtar uzayını arayarak tüm anahtarları elde etmeye çalışır. Saldırganın bir sisteme saldırısı sonucunda sistem üzerinde elde edebileceği senaryolar azalan önem derecesine göre aşağıdaki gibi özetlenebilir [2]:

- Tamamen kırma: Saldırgan gizli anahtarı elde eder,
- Genel tümdengelim: Saldırgan şifreleme ve şifre çözme algoritmalarına eşdeğer bir algoritma bulur, bu onun herhangi bir şifreli metinden açık metni elde etmesini sağlar ancak anahtar seçimini bilmediğinden anahtara ulaşamaz,
- Yerel tümdengelim: Saldırgan daha önce görülmemiş bir şifreli metin veya açık metin çiftini üretebilir, bu durumda sadece tek bir çifte ait bilgi elde etmiş olur,

• Ayırıcı: Saldırganın rastgele anahtar veya rastgele seçilen açık metin/şifreli metin eşine erişimi vardır, burada saldırgan şifreli metnin blok şifreden mi yoksa rastgele eşleştirmeden mi geldiğine karar vermeye çalışır.

Saldırgan elindeki verileri kullanarak saldırı modellerini uygular. Bilinen saldırı modelleri aşağıda verilmiştir [1]:

• Sadece şifreli metin saldırısı: Saldırgan şifreli metin dizisine sahiptir,

• Bilinen açık metin saldırısı: Saldırgan açık metin ve şifreli metin dizilerine sahiptir,

• Seçilmiş açık metin saldırısı: Saldırgan bir açık metin dizini seçip bunların şifreli metinlerini oluşturabilir,

• Seçilmiş şifreli metin saldırısı: Saldırgan bir şifreli metin dizisi seçip bunların açık metinlerini oluşturabilir.

Bir şifreleme algoritması üzerinde uygulanan saldırı yöntemlerinin maliyeti ölçülürken önemli üç metrikten bahsedilebilir [2]; bunlardan ilki zamandır; bu metrik bir sistemin kırılması için gereken hesaplama gücünü temsil eder. Pratikte anahtarı elde etmek için gereken gerçek süre olarak ifade edilebilir ancak bu işlemler mevcut hesaplama gücüne de bağlı olduğundan, en genel ifadeyle başarılı olmak için gereken basit işlemlerin (komutların) sayısıdır denilebilir. Bu CPU üzerinde basit komutlar ya da blok şifreyi çağırma sayısı olabilir. Kişisel bilgisayarlarda bulunan tipik bir işlemci saniyede $\approx 2^{32}$ [2] işlem yapabilir. Diğer bir metrik ise bellektir; bir saldırı için gerekli depolama alanı önemlidir çünkü bir saldırının uygulanabilirliğini engelleyebilir. Gerekli olan bellek miktarının yanında belleğe erişim de bir maliyet gerektirir. Son metrik ise veridir; bu metrik başarılı saldırı yapmak için gereken veri miktarını temsil eder. Uygulamada, eğer bir saldırı çok özel mesajlar içeriyorsa büyük miktarda veri elde etmek çok zordur.

Blok şifreler üzerinde özellikle karıştırma fonksiyonu olan S-kutularını hedef alan doğrusal ve diferansiyel kriptanaliz yöntemleri geliştirilmiştir.

1.2.1. Doğrusal olmama kriteri ve doğrusal kriptanaliz

Bir şifredeki S-kutularının doğrusal olmaması istenir çünkü bu kriter S-kutuları için önemli bir güvenlik ölçütüdür.

f elemanları GF(2) sonlu cismi üzerinde tanımlı, n-bit'ten m-bit'e bir şifre olmak üzere, f şifresinin doğrusal olmama parametresi $NLM_f(z)$; $f : GF(2)^n \rightarrow GF(2)^m$, $a \in GF(2)^m$, $w \in GF(2)^n$ ve $c \in GF(2)$ buradan $z = (a, w, c) \in GF(2)^{n+m+1}$ olmak üzere, şifrenin tüm giriş değerleri için $P \in GF(2)^n$ affine fonksiyon $(w \cdot P \oplus c)$ ve sıfır haricindeki doğrusal kombinasyonlarını $(a \cdot f(P))$ birbirinden farklılaşması ile tanımlanır. Bir şifrenin tüm doğrusal olmama ölçüsü Eşitlik (1.1) ve Eşitlik (1.2)'deki gibi tanımlanabilir [27].

$$NLM_f(z) = \#\{P \mid a \cdot f(P) \neq w \cdot P \oplus c\} \tag{1.1}$$

$$NLM_f = \min_z NLM_f(z) \tag{1.2}$$

Doğrusal kriptanaliz S-kutularını doğrusal ifadelere dönüştürerek anahtar bitlerini elde etmeyi amaçlar. Şifreli metin bitleri ile açık metin bitleri arasındaki doğrusal ifadeleri elde etmeye çalışır. Bu saldırıda saldırganın kriptosistemi bildiği ve belli sayıda açık metin şifreli metin çiftlerine sahip olduğu varsayılır. Bu saldırıyı önlemenin en önemli yöntemlerinden ikisi şifredeki aktif S-kutusu (doğrusal ifade içinde olan S-kutusu) sayısını arttırılması ve doğrusal sapması küçük S-kutularının kullanımıdır. f şifresi üzerinde doğrusal kriptanalizin başarısız olabilmesi için NLM_f olabildiğince maksimum değer olan $NLM_{S\max} = 2^{n-1} - 2^{\frac{n}{2}-1}$ değerine yakın olması gerekir. Örneğin, Bent fonksiyonlarının maksimum doğrusal olmama değeri $2^{n-1} - 2^{\frac{n}{2}-1}$ değeridir [28]. NLM_f eğer 0'a yakın bir değer alıyorsa şifrenin affine fonksiyonlara yaklaştığını ve doğrusal kriptanalize karşı savunmasız olacağı anlamına gelir [27].

Bir S-kutusunun doğrusal kriptanalize karşı güvenliğinin ölçülmesinde LAT (Lineer Approximation Table-Doğrusal Yaklaşım Tablosu) tablosu önemli bir ölçüttür. S bir S-kutusu $S : GF(2)^n \rightarrow GF(2)^m$, w. satır ve c. sütun LAT(w,c) ve doğrusal olasılığı $LP^S(w, c)$ Eşitlik (1.3)'deki gibi tanımlanabilir. P giriş bitlerini ifade ederken S(P) S-kutusunun çıkış bitlerini ifade eder.

$$LAT(w, c) = \#\{P \in GF(2^n) \mid w \cdot P = c \cdot S(P)\} - 2^{n-1},$$

$$LP^S(w,c) = \left(\frac{\#\{P \in GF(2^n) \mid w \cdot P = c \cdot S(P)\}}{2^{m-1}} \right)^2 \quad (1.3)$$

LAT girişindeki en büyük değer ($\max_{w,c} |LAT_S(w,c)|$) önemlidir çünkü doğrusal kriptanalizin karmaşıklığı bu değerin genişliğine bağlıdır ayrıca bu değer Eşitlik (1.4)'deki S-kutusunun doğrusal olmama değerini hesaplarken kullanılır.

$$NLM_S = 2^{n-1} - \max_{w,c} |LAT_S(w,c)| \quad (1.4)$$

NLM_S değerinin $NLM_{S\max}$ değerine oranı ise o S-kutusunun doğrusal olmama değerini verir.

Doğrusal kriptanalize karşı dayanıklılık için S-kutusunun doğrusal yaklaşımlarının sapmaları olabildiğince düşük olmalı ve aktif S-kutusu sayısı ise mümkün olduğunca yüksek olmalıdır [27]. Tablo 1.1'de PRESENT [29] hafif sıklet blok şifresinde kullanılan S-kutusu ve devamında da PRESENT S-kutusu için elde edilen LAT değerleri verilmiştir.

Tablo 1.1. PRESENT S-kutusu [29]

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

Tablo 1.1'de PRESENT S-kutusuna ait giriş ve çıkış değerleri hexadecimal olarak verilmiştir. Bu S-kutusunun LAT tablosu değerleri (1.3) ifadesine göre hesaplanarak tüm değerler Tablo 1.2'de verilmiştir.

Tablo 1.2'den de görülebileceği gibi bu S-kutusuna ait ($\max_{w,c} |LAT_S(w,c)|$) değeri 4'tür. Buradan PRESENT S-kutusuna ait doğrusal olmama değerİ $NLM_S = 2^{4-1} - 4 = 4$ bulunur. 4 değeri bir S-kutusu için literatürde bilinen en iyi değerdir [29].

Tablo 1.2. PRESENT S-kutusunun LAT tablosu

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

1.2.2. Diferansiyel kriptanaliz

Bu saldırıda birçok açık metin ve şifreli metin çiftleri üretilerek açık metin çiftlerinden özel farkların şifreli metin üzerinde oluşturacağı farklar analiz edilir. Bu farka göre olası tüm anahtar değerleri denenir, yüksek olasılığı yakalayan anahtar değerleri şifre anahtarı olarak kabul edilir. Bir S-kutusunun diferansiyel kriptanalize karşı gücünü test etmek için DDT (Difference Distribution Table/XOR Table-Fark Dağılım Tablosu/XOR Tablosu) tablosundan faydalanılabilir. S, $GF(2)^n \rightarrow GF(2)^n$'e bijektif bir harita, a ve b sırasıyla n-bit büyüklüğünde giriş ve çıkış vektörleri olmak üzere S için diferansiyel olasılık $DP^S(a,b)$ Eşitlik (1.5)'deki gibi elde edilir.

$$DP^S(a,b) = \frac{\#\{x \in GF(2)^n \mid S(x) \oplus S(x \oplus a) = b\}}{2^n} \tag{1.5}$$

Tüm olası a ve b değerleri için Eşitlik (1.6)'daki denklem hesaplanırsa DDT tablosu elde edilir.

$$\#\{x \in GF(2)^n \mid S(x) \oplus S(x \oplus a) = b\} \tag{1.6}$$

Tablo 1.1'de verilen PRESENT S-kutusunun DDT tablosu değerleri hesaplanmış ve DDT tablosunu oluşturan bu değerler Tablo 1.3'de verilmiştir. $DP^S(a,b)$ değerleri ise Tablo 1.3'de verilen değerlerin 2^n'e bölünmesiyle elde edilir.

Tablo 1.3. PRESENT S-kutusunun DDT tablosu

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

Bir (hafif sıklet) blok şifrenin Diferansiyel kriptanalize karşı dayanıklılığı için kullanılan S-kutusunun diferansiyel yaklaşımlarının sapmaları olabildiğince düşük olmalı ve aktif S-kutusu sayısı ise mümkün olduğunca yüksek olmalıdır [27].

Örneğin AES şifreleme algoritmasında 4 döngü sonucunda 25 aktif S-kutusu bulunduğundan doğrusal ve diferansiyel kriptanaliz saldırıları 4 döngüden sonra etkisiz hale gelmektedir. Buna β^2 effect (etkisi) denir. β doğrusal dönüşümde kullanılan sütunları karıştırma (MixColumns) dönüşümünün dal sayısı (branch number) değerini ifade eder ve AES şifreleme algoritması için β değeri 5'dir. Not edilmelidir ki, β^2 etkisi MixColumns dönüşümüyle beraber satırları öteleme (ShiftRows) dönüşümünün katkısıyla sağlanır. AES S-kutusunun LAT tablosundaki en büyük değerin mutlak değeri 16, DDT tablosundaki en büyük değerin mutlak değeri ise 4'tür [30]. AES S-kutusunun maksimum diferansiyel ve doğrusal diferansiyel ve doğrusal olasalıklıları sırasıyla $DP^S = \frac{4}{2^8} = 2^{-6}$, $LP^S = \left(\frac{16}{2^7}\right)^2 = 2^{-6}$

olur. 4 döngü sonunda (25 aktif S-kutusu ile) şifrenin diferansiyel ve doğrusal karakteristiklerinin maksimum olasılıkları $(2^{-6})^{25} = 2^{-150}$ değerine ulaşır, bu değer 128-bit anahtar uzunlğuna sahip AES şifresi üzerinde doğrusal ve diferansiyel kriptanaliz saldırıları için sınır değeri olan 2^{-128} değerinden, küçük veya eşit olma koşulunu sağlayan $(2^{-150} \leq 2^{-128})$ minimum değerdir. Bu nedenle AES şifreleme algoritması 4 döngü sonunda diferansiyel ve doğrusal kriptanalize karşı dirençlidir denir. AES şifreleme algoritması ile ilgili kapsamlı bilgiler Bölüm 1.4 içinde verilmiştir.

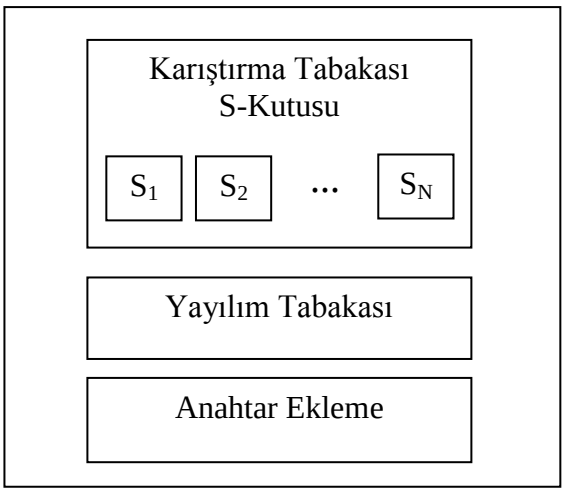
1.3. Tasarım Mimarileri

Shannon tarafından önerilen, iyi bir karıştırma ve yayılma sağlanabilmesi için lojik olarak ayrı iki katmanın kullanılması fikri farklı tasarım mimarilerinin geliştirilmesini sağlamıştır. Blok şifre tasarımında bilinen iki popüler mimari vardır bunlar SPN ve Feistel ağlarıdır.

1.3.1. SPN mimarisi

SPN mimarisinde tasarlanan bir blok şifre üç temel bileşenden oluşur bunlar; karıştırma; döngü fonksiyonunun doğrusal olmayan katmanıdır ve genellikle S-kutusu ile sağlanır, yayılma; doğrusal dönüşümler ile sağlanır, anahtar ekleme; karıştırma ve yayılma adımlarının tekrarlanmasıyla elde edilecek izlerin bozulması için her döngüde bir anahtar eklenir, döngü anahtarları anahtar planlama algoritması tarafından üretilir.

Şekil 1.3'te SPN mimarisindeki bir blok şifrenin döngü fonksiyonunda bulunan temel bileşenler verilmiştir. Döngü, döngü fonksiyonunda karıştırma işleminin sağlanması için doğrusal olmayan yapıların (örneğin S-kutuları, N kullanılacak S-kutusu sayısı olmak üzere şifre tasarımına göre değişir) yer aldığı karıştırma tabakası, yayılımın sağlanması için doğrusal dönüşümün/doğrusal dönüşümlerin kullanıldığı yayılım tabakası ve anahtar üretme mekanizması tarafından üretilen anahtarın eklenmesi adımlarından oluşur. Tüm bu bileşenler blok şifrenin gücünü belirler.

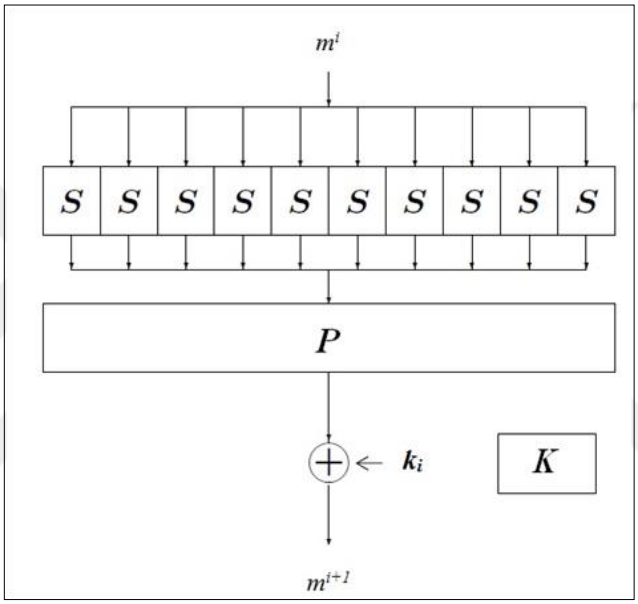


Şekil 1.3. SPN mimarisi döngü yapısı

S-kutusu bir blok şifredeki tek doğrusal olmayan bileşendir, bu bileşen olmadan bir şifre sadece bazı doğrusal işlemler içerir bu da sistemi saldırılara karşı güvensiz hale getirir. S-kutuları karıştırma işlemini n girdi bitini m çıktı bitine haritalama yaparak gerçekleştirir. Yayılım tabakaları ise kullanılan doğrusal dönüşümlerle sabit uzunlukta bir girdi bloğunu doğrusal olarak karıştırarak aynı uzunlukta çıkış bloklarını elde etmeyi sağlayan yapılardır. Şekil 1.3'de anahtar ekleme adımı yayılım tabakasından sonra eklenmiştir ancak döngünün başında eklenebilir. Anahtar ekleme işlemi basit işlemlerle gerçekleştirilir, bu işlem için genellikle XOR kullanılır, diğer bir işlem ise modüler toplamadır (GOST [31], RC5 [32], TEA [33], Twofish [34], SEA [35]). Yazılım maliyeti açısından, modüler toplama genellikle XOR işlemi ile aynı maliyetlidir, bu nedenle karıştırma ve yayılmayı sağlamak için ucuz bir yöntemdir denilebilir. Ancak bazı şifrelerde XOR kullanımına bağlı optimizasyonlar olabileceği için bu şifrelerde anahtar ekleme işlemi için modüler toplamanın kullanımı uygun olmayacaktır. Donanım maliyeti açısından XOR işlemi ucuzdur, çünkü modüler toplama işlemi kaynaklı gecikme CPU'nun boru hattı (pipeline) tarafından gizlenemez [7].

Anahtar bir blok şifredeki önemli bileşenlerden biridir, anahtar uzunluğu saldırılara karşı dayanıklı olacak şekilde seçilmelidir. İstatistiksel saldırılara karşı dayanıklı bir yapı için idealde n-bit bir blok şifrede olması gereken anahtar uzunluğu $n \cdot 2^n$ bit [36] olmalıdır. Ancak pratikte bu kadar büyük anahtar uzunluğuyla işlem yapmak mümkün zordur [36]. Uygun anahtar uzunluğu şifrenin kaba kuvvet (brute-force) saldırılarına karşı kırılabilirliğini zorlaştırır.

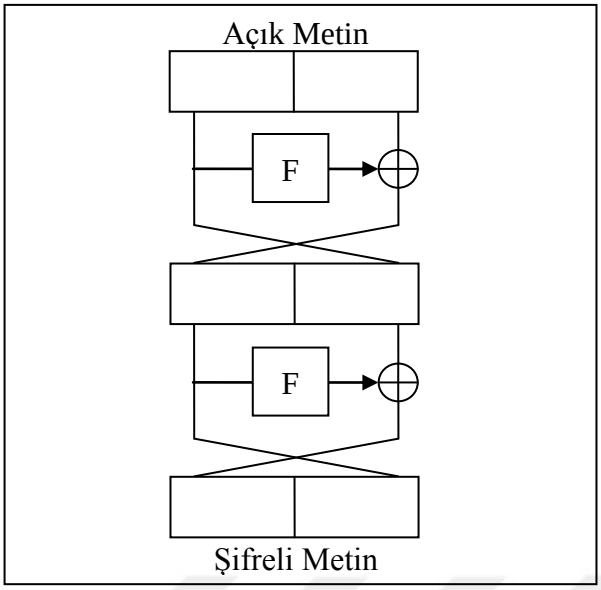
SPN mimarisinde bir blok şifre tasarlamanın birçok yöntemi vardır temelde her döngü art arda tekrarlanır. Anahtar ekleme işlemi K, karıştırma tabakası S, yayılım tabakası ise P ile ifade edilirse SPK adımlarını tekrar eden ve son döngüde SK adımıyla bir SPN mimarisinde blok şifre tasarlanabilir. K adımı genelde SP adımlarından sonra yapılır ancak bazı şifrelerde döngü başında anahtar ekleme işlemi yapılabilir bu durumda son döngüde sadece K adımı tekrarlanır. Literatürde farklı döngü yapısında tasarlanan blok şifreler vardır; KSKP [37], SPSPS-PKP [38]. Şekil 1.4'te 1 döngülük bir SPN mimarisi örneği verilmiştir.



Şekil 1.4. 1 döngülük SPN mimarisi örneği

1.3.2. Feistel mimarisi

Bu mimaride açık metin bloğu iki eşit parçaya (kaynak ve hedef parça) ayrılır, her bir döngü için üretilen döngü anahtarı ile bloğun sağ tarafına F fonksiyonu uygulanır. Bu işlem sonucu ile bloğun sol tarafı XOR işlemine tabi tutulur elde edilen sonuç bir sonraki döngünün sağ bloğunu oluştururken, bir sonraki bloğun sol bloğu ise bir önceki bloğun sağ bloğu olacak şekilde çaprazlanır. Yani her döngüde durumun yarısı değiştirilir bundan dolayı özellikle kaynak dalı karıştırma ve yayılma için önemli rol oynar. Bu ağlarda temelde üç işlem tekrar edilir açık metnin bitlerinin yer değiştirilmesi (karıştırma), basit doğrusal olmayan F fonksiyonun çağrılması (yayılma) ve XOR işlemiyle doğrusal karıştırma adımıdır. Şekil 1.5'de 2 döngülük Feistel mimarisi örneği verilmiştir.



Şekil 1.5. 2 döngülük Feistel mimarisi örneği

Literatürde birçok Feistel ağı tasarımı mevcuttu [7]: Bunlardan ilki klasik Feistel ağıdır bu ağlar dengelidir yani sağ ve sol dalları aynı boyuttadır, literatürde DES, GOST, Blowfish [39], TEA, Twofish ve SIMON [40] gibi blok şifreler bu topolojidedir. Diğer bir tasarım ise L (Left-Sol) ve R (Right-Sağ) ağıdır; Mitsuru Matsui'nin varyasyonu [41] bu tasarımda kullanılmıştır, L-ağında F fonksiyonu hedef dal üzerindedir, R-ağında ise F fonksiyonu kaynak dal üzerindedir, KASUMI [42] şifresi bu yapıya örnek verilebilir. Dengesiz Feistel ağı tasarımında girdi ve çıktı bit uzunlukları birbirinden farklıdır, bir döngü çıktısı bir sonraki döngünün girdisi olmadan önce yeniden bölümlenebilir ancak bölümlenmenin yapılmadığı tasarımlar da vardır. İki farklı F fonksiyonu kullanılır; 'genişleme' fonksiyonu uzunluğu küçük dallara uygulanırken, uzunluğu büyük dallara ise girdiyi küçültmek için 'sıkıştırma' fonksiyonu uygulanır, BEAR ve LION [43] blok şifreleri bu tasarıma örnek verilebilir. GFN (Generalised Feistel Network-Genelleştirilmiş Feistel Ağları) tasarımı dengesiz Feistel ağlarını tanımlamanın birçok yolundan biridir, farklı boyutlarda iki dal yerine açık metin ikiden fazla dala bölünebilir ancak ağ hala dengesiz olabilir. CAST-256 [44] şifresi bu tasarıma örnek verilebilir. Feistel ağları tasarım yöntemleriyle ilgili kapsamlı bilgiler [7] çalışmasından incelenebilir.

Feistel mimarisinde tasarlanmış bir blok şifre için blok uzunluğu, anahtar uzunluğu, döngü sayısı, anahtar üretme algoritması ve F fonksiyonu parametreleri önemlidir. Büyük blok uzunluğu ve anahtar uzunluğu daha fazla güvenlik sağlar ancak

şifreleme ve şifre çözme hızını azaltır, genellikle blok şifreler için kullanılan blok uzunluğu 64-bit tercih edilirken anahtar uzunluğu 128-bittir. Feistel ağları tek döngüde yetersiz güvenlik sağlarken, artan döngü sayısı güvenlik seviyesini de arttırır. Tipik döngü sayısı 16'dır. Anahtar planlama ve F fonksiyonu algoritmalarının karmaşıklıklarının arttırılması kriptanalize karşı direnç sağlar. Feistel mimarisinde blok şifreler tasarlanırken şifreleme/şifre çözme algoritmalarının yürütme hızları düşünülmelidir. Algoritmanın kriptanalize karşı olabildiğince dayanıklı olması istenirken bir taraftan da algoritmanın anlaşılabilir olması (kolay analiz edilmesi) gerekir çünkü bir algoritmanın analizi kolay açıklanabilirse kriptografik özellikleri o kadar iyi açıklanmış olur [36].

SPN ve Feistel ağları karşılaştırıldığında [7]: SPN mimaride şifreleme ve şifre çözme algoritmaları farklıdır, Feistel ağlarında ise şifre çözme algoritması için farklı algoritma kullanılması gerekmez. Şifre çözme işlemi için şifreli metin girdi olarak alınır ve döngü anahtarları tersinden (örneğin şifre çözme algoritasının ilk döngü anahtarı şifreleme algoritmasının son döngüsünde kullanılan anahtardır) kullanılır. SPN mimarisi daha hızlı karıştırma ve yayılma sağlar. SPN mimarisinde tasarlanmış bir blok şifre ile aynı blok ve anahtar uzunluğuna sahip Feistel ağı için, SPN mimarisi ile aynı güvenlik seviyesi sağlaması için daha fazla döngüye ihtiyaç duyar. Bu da işlem hacmini (throughput) (belirli bir frekansta gerçekleşen şifreleme/şifre çözme işlemi Kb/s (Kilobit/second) olarak ifade edilir) arttırır. Ancak Feistel ağları için artan döngü sayısı SPN mimariyle kıyaslandığında çok fazla maliyet gerektirmez çünkü şifre çözme algoritması neredeyse maliyetsizdir. SPN mimari özellikle yazılım/donanım optimize uygulamalar için daha fazla paralellik sağlar.

1.4. AES Şifreleme Algoritması

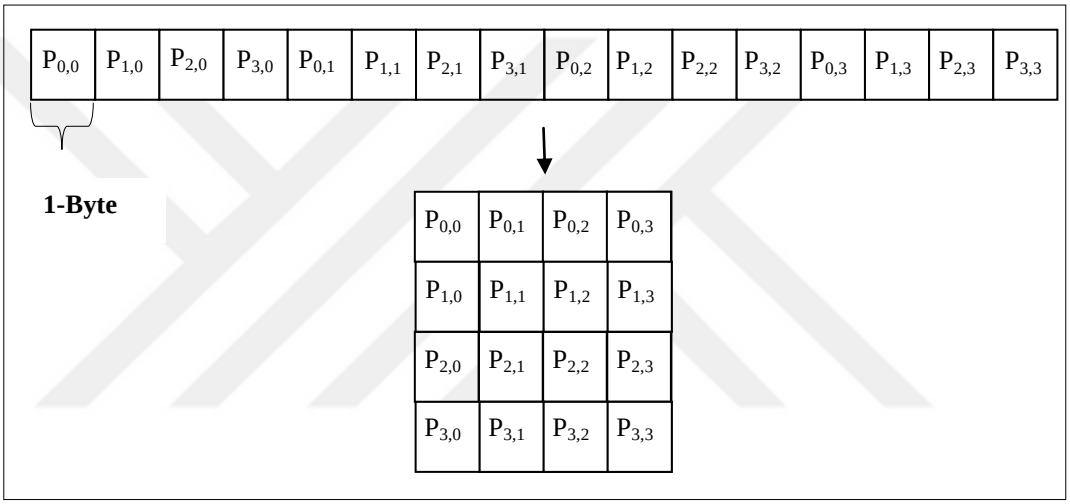
AES, SPN mimarisinde tasarlanmış, 128-bit blok uzunluğundaki verileri 128, 192, 256-bit anahtar uzunluklarıyla şifreleyen blok şifreleme algoritmasıdır. Algoritmanın döngü sayısı anahtar uzunluğuna bağlıdır, 128, 192, 256-bit anahtar uzunlukları için sırasıyla 10, 12 ve 14 döngüde şifreleme yapılır.

AES şifreleme algoritmasının döngü yapısı dört katmandan oluşur. Şifreleme işleminden önce mesaj 128-bitlik bloklara bölünür ve 128-bitlik (16-byte) veri 4×4 byte matrisine dönüştürülür, bu matrise sırasıyla byte'ların yer değiştirmesi

(SubBytes), satırların ötelenmesi (ShiftRows), sütunların karıştırılması (MixColumns) ve döngüye anahtar ekleme dönüşümleri (AddRoundKey) uygulanır [27].

1.4.1. AES şifreleme adımları

AES şifreleme algoritmasında döngü boyunca işlemler state (durum) adı verilen iki boyutlu (matris) byte dizisi üzerinde gerçekleştirilir [27]. Açık mesaj 128-bit (16-byte) bloklara ayrılır, her bloğa ait girdi durumu 4×4 matris ile ifade edilir ve her eleman (1-byte veri) Şekil 1.6'daki gibi matrise yerleştirilir.



Şekil 1.6. AES şifreleme algoritması girdi durumunun oluşturulması

Byte'ların yer değiştirmesi adımı diğer bir deyişle; SubBytes dönüşümü en basit tanımıyla doğrusal olmayan bir byte yer değiştirme işlemidir [27] ve bu işlem S-kutusu ile gerçekleştirilir. Byte matrisindeki her değer 8-bit giriş ve 8-bit çıkışlı S-kutusundaki ilgili değerle yer değiştirilir. AES şifreleme algoritmasında kullanılan S-kutusu Nyberg'in 1994 yılında önerdiği sonlu cisimde ters haritalama tabanlıdır [45]. S-kutusunun cebirsel ifadesi Eşitlik (1.7)'de [46] verilmiştir. Bu ifadeye göre, $GF(2^8)$ üzerinde tanımlı x elemanı f polinomu ile $x^8 + x^4 + x^3 + x + 1$ indirgenemez polinomu kullanılarak, aynı sonlu cisim üzerindeki tersine haritalanır.

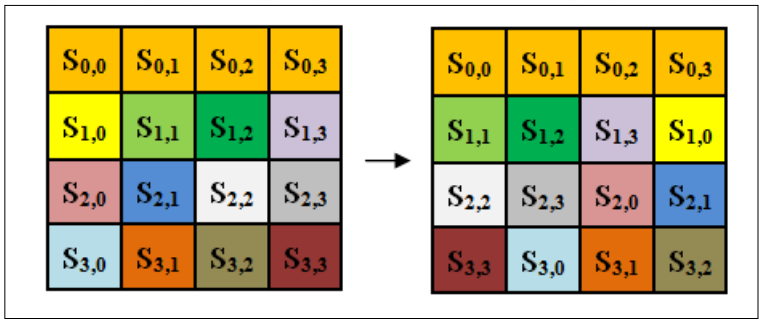
$$f(x) = x^{-1}, x \in GF(2^8), f(0) = 0 \tag{1.7}$$

(1.7)'de verilen ters haritalama işlemi basittir bu nedenle ters haritalama işleminden sonra ikili doğrusal dönüşüm uygulanır, böylece ifade karmaşık hale getirilir [46].

Doğrusal dönüşüm Eşitlik (1.8)'de verilmiştir, x değerleri 8-bitlik S-kutusunun ters alma işlemi sonucu çıkış bitlerini ifade ederken, y değerleri 8-bitlik çıktı bitlerini ifade eder.

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad (1.8)$$

Satırların ötelenmesi adımı, byte'ların yer değiştirmesi adımından sonra elde edilen durum matrisinin satırlarındaki elemanların dairesel olarak kaydırılması işlemlerinden oluşur. 1. satırdaki elemanlar yer değiştirilmez, 2, 3 ve 4. satırlar sırasıyla 1-byte, 2-byte ve 3-byte sola dairesel olarak kaydırılır. Şekil 1.7'de satırların ötelenmesi sonucu elde edilen durum matrisi verilmiştir.



Şekil 1.7. Satırların ötelenmesi (MixColumns) adımı

Bu dönüşüm sonucunda girdi bytle'larının permütasyonu incelendiğinde Tablo 1.4 elde edilir, bu tabloya göre örneğin byte'ların yer değiştirmesi adımından sonra elde edilen durum matrisinde 2. byte ile 11. byte yer değiştirilmiş olur.

Tablo 1.4. Satırları öteleme dönüşümünde elde edilen byte permütasyonu [27]

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

Sütunların yer değiştirmesi yani MixColumns dönüşümünde, satırların yer değiştirmesi sonucunda elde edilen durum matrisi üzerinde sütun sütun çarpma

işlemleri gerçekleştirir. Bu dönüşüm, doğrusal dönüşümdür, her bir sütun GF(2^8) üzerinde bir polinom gibi düşünülür ve Eşitlik (1.9)'da verilen doğrusal dönüşüm matrisi ile durum matrisinin sütunları çarpılarak $x^8 + x^4 + x^3 + x + 1$ indirgenemez polinomuna göre indirgenir ve yeni durum matrisinin sütunları elde edilir.

$$\begin{bmatrix} S'_{0,i} \\ S'_{1,i} \\ S'_{2,i} \\ S'_{3,i} \end{bmatrix} = \begin{bmatrix} 02_h & 03_h & 01_h & 01_h \\ 01_h & 02_h & 03_h & 01_h \\ 01_h & 01_h & 02_h & 03_h \\ 03_h & 01_h & 01_h & 02_h \end{bmatrix} . \begin{bmatrix} S_{0,i} \\ S_{1,i} \\ S_{2,i} \\ S_{3,i} \end{bmatrix} \tag{1.9}$$

Döngüye anahtar ekleme dönüşümünde, anahtar planlama algoritmasından üretilen döngü anahtarı, sütunların yer değiştirmesi adımından sonra elde edilen durum matrisi ile XOR'lama işlemine tabi tutulur.

AES şifreleme algoritması için her döngüde bu 4 temel işlem gerçekleştirilir ancak son döngüde sütunların ötelenmesi adımı kullanılmaz.

1.4.2. AES şifre çözme adımları

AES şifreleme algoritması için şifre çözme işlemi, şifreleme sırasında kullanılan tüm işlemlerin ters sırada kullanılmasıyla yapılır. Şifre çözme adımında yine aynı döngü anahtarları kullanılır, anahtar ekleme dönüşümü yine XOR işlemiyle sağlanır. Ayrıca döngü anahtarları ters sırayla kullanılır, yani son döngüde kullanılan döngü anahtarı şifre çözme algoritmasının ilk döngüsünde kullanılır.

Ters satırların ötelenmesi dönüşümünde durum matrisinin 1. satırındaki elemanlar yer değiştirilmez, 2, 3 ve 4. satırlar sırasıyla 1-byte, 2-byte ve 3-byte sağa dairesel olarak kaydırılır.

Ters bytle'ların yer değiştirmesi dönüşümünde, şifreleme adımında kullanılan S-kutusuna (1.8)'de verilen doğrusal dönüşümün tersi ve GF(2^8) üzerinde çarpma işlemine göre ters alma işlemlerinin sırayla uygulanması sonucunda elde edilen ters S-kutusu kullanılır. Kullanılan ters doğrusal dönüşüm Eşitlik (1.10)'da verilmiştir.

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (1.10)$$

Ters sütunları karıştırılması dönüşümünde, yine her bir sütun $GF(2^8)$ üzerinde bir polinom gibi düşünülür ve (1.9) ifadesinde verilen doğrusal dönüşüm matrisinin tersi Eşitlik (1.11)'de verilmiş olup, durum matrisinin sütunları bu ters dönüşüm matrisiyle çarpılarak $x^8 + x^4 + x^3 + x + 1$ indirgenemez polinomuna göre indirgenir ve yeni durum matrisinin sütunları elde edilir.

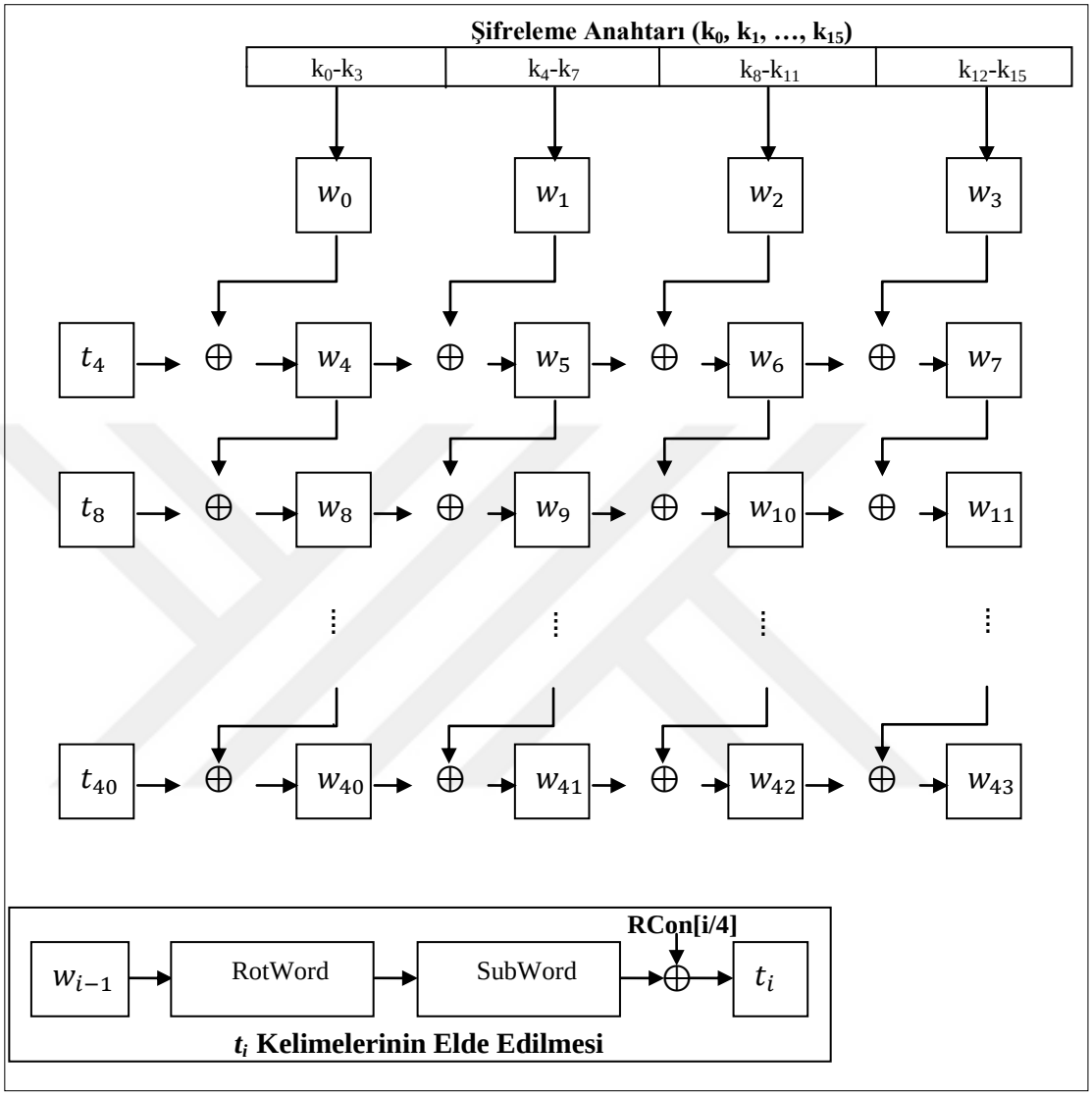
$$\begin{bmatrix} S_{0,i} \\ S_{1,i} \\ S_{2,i} \\ S_{3,i} \end{bmatrix} = \begin{bmatrix} 0e_h & 0b_h & 0d_h & 09_h \\ 09_h & 0e_h & 0b_h & 0d_h \\ 0d_h & 09_h & 0e_h & 0b_h \\ 0b_h & 0d_h & 09_h & 0e_h \end{bmatrix} \cdot \begin{bmatrix} S'_{0,i} \\ S'_{1,i} \\ S'_{2,i} \\ S'_{3,i} \end{bmatrix} \quad (1.11)$$

1.4.3. AES anahtar planlama algoritması

AES anahtar planlama diğer bir adıyla anahtar genişletme algoritmasında 128, 192 ve 256-bit olmak üzere üç farklı anahtar uzunluğu kullanılır. Bu uzunluklar için sırasıyla 10, 12, 14 döngüde şifreleme yapılır. Bu nedenle her döngü için döngü sayısı kadar anahtar üretilmesi gereklidir. Başlangıçta belirlenen tek bir gizli anahtardan (master key) anahtar planlama algoritması kullanılarak döngü anahtarları üretilir. Gizli anahtar sadece döngü başlamadan, diğer döngü anahtarları ise anahtar ekleme dönüşümünde kullanılır.

Anahtar planlama algoritması döngü anahtarlarını kelime kelime oluşturur. Döngü sayısı N_r ile ifade edilirse, anahtar planlama algoritması $4 \times (N_r + 1)$ kelime büyüklüğünde alt anahtarlar yaratır. Yani 128, 192, 256-bit anahtar uzunlukları için sırasıyla 44, 52, 60 kelime büyüklüğünde anahtarlar oluşturulur [47].

Şekil 1.8'de AES-128 bit anahtar planlama algoritmasının yapısı verilmiştir. 192, 256-bit anahtar uzunlukları ayrıntılı olarak [36]'da verilen çalışmadan incelenebilir.



Şekil 1.8. AES-128 için anahtar planlama algoritması [47]

Şekil 1.8'de genel yapısı verilen AES-128 anahtar planlama algoritmasının adımları incelendiğinde [47]:

- 128-bit şifreleme (master) anahtarı (k_0-k_{15}) her biri 32-bitlik 4 kelimeye ayrılır; w_0 (k_0-k_3), w_1 (k_4-k_7),w_2 (k_8-k_{11}), w_3 (k_{12}-k_{15}),
- Diğer kelimeler w_i (i=4'den 43'e kadar) aşağıdaki gibi bulunur:

Eğer $i \pmod 4 \neq 0$ ise $w_i = w_{i-1} \oplus w_{i-4}$ şeklinde bulunur, örneğin $w_5 = w_4 \oplus w_1$,

Eğer $i \pmod 4 = 0$ ise $w_i = t_i \oplus w_{i-4}$ şeklinde bulunur örneğin $w_4 = t_4 \oplus w_0$.

• t_i değerleri elde edilirken sırasıyla kelime döndürme (RotWord) ve kelime yer değiştirme (SubWord) ve döngü sabitleri (Round Constant-RCon) diye adlandırılan sabit ile XOR'lama adımları kullanılır. Yani, $t_i = \text{SubWord}(\text{RotWord}(w_{i-1})) \oplus \text{RCon}_{i/4}$'dir.

Kelime döndürme adımında, kelime dizisinin her byte'ı sola dairesel olarak kaydırılır. Kelime yer değiştirme adımında, kelime dizisindeki her byte değeri farklı bir byte değeriyle yer değiştirilir. Döngü sabiti ile XOR'lama adımında, her döngüde farklı sabit kullanılır. AES-128 için kullanılan döngü sabitleri Tablo 1.5'de verilmiştir.

Tablo 1.5. AES-128 anahtar planlama algoritmasında kullanılan döngü sabiti değerleri [47]

Döngü	**Döngü Sabiti (RCon)**	**Döngü**	**Döngü Sabiti (RCon)**
1	$(01\ 00\ 00\ 00)_{16}$	6	$(20\ 00\ 00\ 00)_{16}$
2	$(02\ 00\ 00\ 00)_{16}$	7	$(40\ 00\ 00\ 00)_{16}$
3	$(04\ 00\ 00\ 00)_{16}$	8	$(80\ 00\ 00\ 00)_{16}$
4	$(08\ 00\ 00\ 00)_{16}$	9	$(1B\ 00\ 00\ 00)_{16}$
5	$(10\ 00\ 00\ 00)_{16}$	10	$(36\ 00\ 00\ 00)_{16}$

AES şifreleme algoritmasının güvenliği incelendiğinde [27], SPN mimarisine sahip olması her döngüde veri bloklarını etkin işlenmesini sağlar [48]. Nyberg'in çalışmasında S-kutularının tasarımı için ortaya atılan doğrusal ve diferansiyel kriptanalize karşı LAT ve DDT tablolarında uniform dağılımın elde edilmesi fikri [45] temel alınarak tasarlanmış, sonlu cisim üzerinde ters alma işlemiyle cebirsel olarak elde edilen bir S-kutusuna sahiptir. MixColumns dönüşümü az sayıda S-kutusu kaynaklı ortaya çıkan doğrusal ve diferansiyel saldırıları zorlaştırır bu özelliğe geniş deneme stratejisi (wide trail strategy) [49] denir. AES anahtar planlama algoritması SAC testinden başarısız olurken, bit sızdırma saldırılarına karşı da güvenlik zafiyeti bulunmaktadır [47].

1.5. Blok Şifre Çalışma Modları

Çalışma modları blok şifrenin, tek bir anahtar ve bu anahtardan üretilen alt anahtarlar kullanılarak, güvenli bir şekilde tekrarlı kullanılmasını sağlayan yöntemlerdir [50].

Şifrelenecek açık mesaj uzunluğu, şifrenin blok uzunluğundan daha büyük olduğu durumlarda çalışma modlarından faydalanılır. Bu modlar, açık mesaj içindeki blokların şifreleme şeklini tanımlar.

Temelde şifrelenecek mesaj eşit uzunlukta bloklara bölünür, son blok şifrenin blok uzunluğuna uyacak şekilde tamamlanır. Bu işleme padding (tamamlama) denir, farklı yöntemler kullanılabilir. Açık metin blok uzunluğunun katı olacak şekilde son bloğa sıfır dizisi eklenir, ancak mesajın sonunda sıfır byte'ı yer alabilir bu nedenle şifre çözme işleminden önce açık mesajın uzunluğunun bilinmesi gerekir. Diğer bir tamamlama yöntemi ise tek adet '1' biti eklendikten sonra sıfır dizisinin eklenmesidir ancak eğer açık mesaj uzunluğu blok uzunluğunun tam katıysa yeni bir tamamlama bloğu eklenmiş olur bu da fazladan şifreli metin üretimine sebep olur. Bu yöntem DES şifresinde kullanılmıştır. Ayrıca Schneier ve Ferguson'un önerdiği iki yönteme göre [51], bloğa öncelikle $80_h (=128)$ değeri ekledikten sonra bloğun geri kalan değerleri sıfır byte'ı ile tamamlanır veya blok n adet n değeri ile tamamlanır.

Bazı çalışma modları şifreleme işlemini rassallaştırmak için IV (Initialization Vector-İlklendirme Vektörü) bit bloğunu kullanırlar, bu vektör aynı açık metnin tekrar şifrelendiği durumlarda, aynı anahtar ile farklı şifreli metin üretilmesini sağlar. IV gizli tutulmak zorunda değildir, ancak şifrelemede kullanılacak IV'ler birbirinden farklı olmalıdır. Çünkü açık metnin ilk bloğu hakkında ya da iki açık metnin aynı olan ilk blokları hakkında bilgi sızıntısına sebep olur.

• ECB (Electronic Code Book-Elektronik Kod Defteri): Açık metin bloklara ayrılır ve her blok ayrı ayrı şifrelenir. Bu yöntem tek değerlerin güvenli iletiminde kullanılabilir (örneğin tek bir şifreleme anahtarının karşı tarafa iletilmesi) [36]. Ancak bu yöntemde aynı açık metin blokları aynı şifreli metin bloklarını üretir bu nedenle kriptografik protokollerde kullanılması güvenli değildir. Bu modda şifreleme ve şifre çözme işlemleri paralel olarak gerçekleştirilir. ECB çalışma modunun şifreleme ve şifre çözme işlemlerinin matematiksel ifadeleri sırasıyla Eşitlik (1.12) ve (1.13)'de verilmiştir. P açık metni, C şifreli metni, E şifreleme algoritmasını, D şifre çözme algoritmasını ve ve N blok sayısını ifade eder.

$$C_j = E(K, P_j), \ j = 2, \ldots, N \tag{1.12}$$

$$P_j = D(K, C_j), \ j = 2, \ldots, N \quad (1.13)$$

• CBC (Cipher Block Chaining–Şifre Bloğu Zincirleme): Açık metin bloklara ayrılır ve bir mesajın aynı anahtar ile tekrar şifrelendiğinin anlaşılmaması için ilk blok IV ile XOR'lanır. Gönderici ve alıcı tarafları IV değerini bilir. Diğer açık metin blokları şifrelenmeden önce bir önceki şifreli metin bloğu ile XOR'lanır. Bu mod gizlilik sağlar ayrıca kimlik doğrulama için kullanılabilir [36]. Bu modda şifre çözme işlemi paralel olarak gerçekleştirilebilirken, şifreleme işlemi paralel olarak gerçekleştirilemez. CBC çalışma modunun şifreleme ve şifre çözme işlemlerinin matematiksel ifadeleri sırasıyla Eşitlik (1.14) ve (1.15)'de verilmiştir. P_1 açık metnin ilk bloğunu, C_1 şifreli metnin ilk bloğunu ve N blok sayısını ifade eder.

$$C_1 = E(K, [P_1 \oplus IV]), C_j = E(K, [P_j \oplus C_{j-1}]), \ j = 2, \ldots, N \quad (1.14)$$

$$P_1 = D(K, C_1) \oplus IV, P_j = D(K, C_j) \oplus C_{j-1}, \ j = 2, \ldots, N \quad (1.15)$$

• CFB (Cipher Feedback-Şifre Geri Beslemeli): Açık metin bloklara ayrılır, CBC modundaki gibi IV kullanılır. İlk blok başlangıçta IV ile şifrelenir ve elde edilen sonuç açık metnin ilk bloğuyla XOR'lanır. Diğer blokların girdileri bir önceki şifreli metin bloğudur. Sonrasında ilgili açık metin bloğu şifreleme adımından dönen sonuç ile XOR'lanır. Kimlik doğrulama için kullanılabilir [36]. Bu modda şifre çözme işlemi paralel olarak gerçekleştirilebilirken, şifreleme işlemi paralel olarak gerçekleştirilemez. CFB çalışma modunun şifreleme ve şifre çözme işlemlerinin matematiksel ifadeleri sırasıyla Eşitlik (1.16) ve (1.17)'de verilmiştir.

$$C_1 = E(K, IV) \oplus P_1, C_j = E(K, C_{j-1}) \oplus P_j), \ j = 2, \ldots, N \quad (1.16)$$

$$P_1 = D(K, IV) \oplus C_1, P_j = D(K, C_{j-1}) \oplus C_j), \ j = 2, \ldots, N \quad (1.17)$$

• OFB (Output Feedback-Çıkışı Geri Beslemeli): Açık metin bloklara ayrılır, CBC ve CFB modları gibi IV kullanılır. İlk blok başlangıçta IV ile şifrelenir ve elde edilen şifreleme sonucu bir sonraki bloğun girdisi olur. Ayrıca sonuç, açık metnin ilk bloğuyla XOR'lanır böylece ilk blok için şifreli metin bloğu elde edilir. Diğer blokların girdileri bir önceki bloğun açık metin ile XOR'lanmadan önceki durumudur. İlk blokla benzer şekilde giriş bloğu şifrelendikten sonra ilgili açık metin bloğuyla XOR'lanır ve şifreli metin bloğu elde edilir. Uydu haberleşmesi (iletimi) gibi gürültülü kanallar üzerinde veri iletiminde bu mod kullanılabilir [36]. Şifreleme ve şifre çözme işlemleri paralel olarak gerçekleştirilemez. OFB çalışma modunun

şifreleme ve şifre çözme işlemlerinin matematiksel ifadeleri sırasıyla Eşitlik (1.18) ve (1.19)'da verilmiştir.

$$C_1 = E(K, IV) \oplus P_1, C_j = E(K, O_{j-1}) \oplus P_j, O_{j-1} = E(K, O_{j-2}),\ j = 2, \ldots, N \quad (1.18)$$

$$P_1 = D(K, IV) \oplus C_1, P_j = D(K, Q_{j-1}) \oplus C_j, Q_{j-1} = D(K, Q_{j-2}),\ j = 2, \ldots, N \quad (1.19)$$

• CTR (Counter-Sayaç): Açık metin bloklara ayrılır, her 2n-bit blok için n-bit nonce (benzersiz IV) ve n-bit sayaç (counter) değerlerinden oluşan 2n-bit uzunluğunda IV değerleri şifreleme algoritmasına girdi olarak gönderilir. Sayaç değeri her açık metin bloğu için birbirinden farklı olmalıdır, genel olarak sıfırdan başlayarak birer birer arttırılır. Nonce değeri tüm bloklar için sabitken, bu değer gizli tutulmalıdır. Çıkış değerleri ile açık metin blokları XOR'lanarak şifreli metin blokları elde edilir. Yüksek hız gerektiren durumlarda kullanılabilir [36]. Bu modda şifreleme ve şifre çözme işlemleri paralel olarak gerçekleştirilir. CTR çalışma modunun şifreleme ve şifre çözme işlemlerinin matematiksel ifadeleri sırasıyla Eşitlik (1.20) ve (1.21)'de verilmiştir.

$$C_j = E(K, IV_j) \oplus P_j,\ j = 1, \ldots, N-1 \quad (1.20)$$

$$P_j = E(K, IV_j) \oplus C_j,\ j = 1, \ldots, N-1 \quad (1.21)$$

• PCBC (Propagating Cipher Block Chaining-Yayılan Şifre Bloğu Zincirleme): Şifreli metindeki değişikliklerin, şifre çözme adımında sonraki bloklara belirsiz bir şekilde yayılmasını sağlamak amaçlı geliştirilmiştir. Temelde bir bloğa ait açık metin ve ilgili bloğun şifreli metin blokları XOR'lanarak bir sonraki bloğun açık metni ile XOR'lama işlemine tabi tutulur. Bu modda şifreleme ve şifre çözme işlemleri paralel olarak gerçekleştirilemez. PCBC çalışma modunun şifreleme ve şifre çözme işlemlerinin matematiksel ifadeleri sırasıyla Eşitlik (1.22) ve (1.23)'de verilmiştir.

$$C_1 = E(K, [P_1 \oplus IV]), C_j = E(K, [P_j \oplus (P_{j-1} \oplus C_{j-1})]),\ j = 2, \ldots, N \quad (1.22)$$

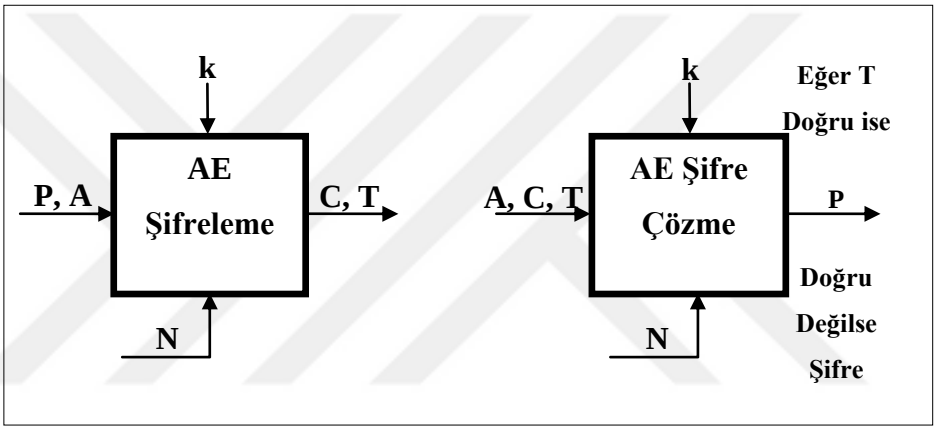
$$P_1 = D(K, C_1) \oplus IV, P_j = D(K, C_j) \oplus (C_{j-1} \oplus P_{j-1}),\ j = 2, \ldots, N \quad (1.23)$$

1.6. AE (Authenticated Encryption-Kimlik Doğrulamalı Şifreleme)

MAC (Message Authentication Code-Mesaj Doğrulama Kodu) mesajın bütünlüğünü sağlamaya yarayan diğer bir ifade ile mesajın içeriğinin alıcıya iletilene kadar değiştirilmediğini ve belirli bir kaynaktan çıktığını doğrulayan koddur. Alıcı ve

gönderici aynı anahtarı kullanır, şöyle ki gönderici etiketi hesaplamak için k anahtarını kullanırken alıcı da etiketi aynı k anahtarı ile doğrular [52].

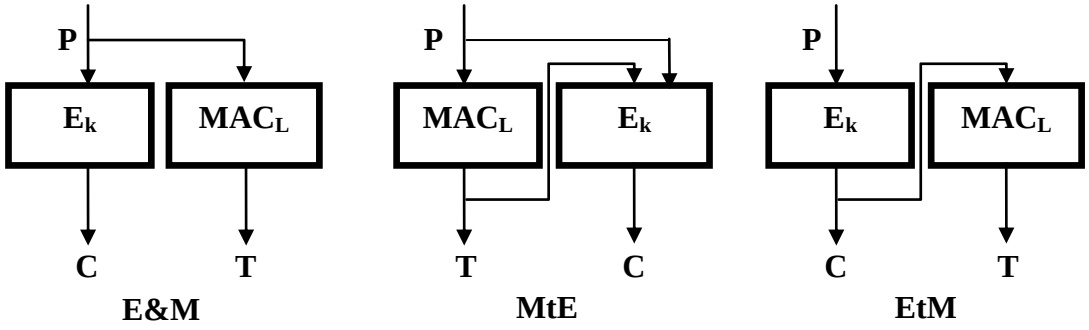
Bütünlük, gizlilik ve kimlik doğrulama servislerinin bir arada sağlandığı kriptografik sistemler AE veya AEAD (Authenticated Encryption with Associated Data - İlişkili Veri ile Kimlik Doğrulamalı Şifreleme) olarak adlandırılır. İlişkili veri gizli olmayan veridir [53]. AE'ye olan ihtiyacı tetikleyen iki faktör tanımlanabilir [53]; bunlardan ilki, sadece gizliliği sağlayan şifreleme şemaları ile MAC birbirine eklenmeye çalışıldığında sonucun oldukça zayıf olmasıdır. Diğer faktör ise, Bölüm 1.5'de verilen modların MAC ile birleştirildiğinde hataların ortaya çıkmasıdır.



Şekil 1.9. AE şifreleme ve şifre çözme blok şemaları

AE şifreleme ve şifre çözme adımlarının blok şemaları Şekil 1.9'da verilmiştir. P açık metin, A associated data, k şifre anahtarı ve N benzersiz IV (yapıyı rastgeleleştirir) olmak üzere, AE şifreleme algoritmasının girdi parametreleriyken, algoritmanın çıktı parametreleri ise C şifreli metni ve T etiketidir. T etiketi sayesinde açık metin ve associated veri için kimlik doğrulama sağlanmış olur. N her şifreleme adımı için farklı olmalıdır. AE şifre çözme algoritmasının girdi parametreleri A, C, T, k ve N iken çıktı parametresi koşullu olarak elde edilir, eğer etiket doğru ise açık metin elde edilir, aksi durumda şifre çözme işlemi yapılmaz.

AE, en genel tanımıyla MAC ve şifreleme algoritmasının birlikte kullanılması denilebilir [52]. AE tasarımı için Bellare ve Namprempre [52] üç yaklaşım önermişlerdir. Bu yaklaşımlar Şekil 1.10'da verilmiştir.



Şekil 1.10. AE tasarım yöntemleri

Tüm tasarımlar için $\mathrm{MAC_L}(\text{girdi}) = \text{girdi} \,\|\, \mathrm{T}$ (girdi bitlerinin sonuna T etiketinin bitleri eklenir) olmak üzere, ilk yaklaşımda $(\mathrm{E\,\&\,M})$ (Encrypt and MAC-Şifreleme ve MAC) şifreleme ve MAC oluşturma işlemleri aynı anda yapılır. SSH (Secure Shell-Güveli Kabuk) protokolünde kullanılır [36]. Ancak bu yaklaşım genel olarak güvensizdir. İkinci yaklaşımda (MtE) (MAC then Encrypt-MAC sonra Şifreleme) önce MAC oluşturulur, elde edilen etiket ve açık metin şifreleme algoritmasına girdi olarak gönderilir ve şifreli metin elde edilir. SSL (Secure Sockets Layer-Güvenli Giriş Katmanı)/TLS (Transport Layer Secuity-Taşıma Katmanı Güvenliği) protokollerinde kullanılır [36]. Bu yaklaşım hafif güvensiz olarak değerlendirilebilir. Son yaklaşımda (EtM) (Encrypt then MAC Şifreleme sonra MAC) açık metin bloğu önce şifrelenir daha sonra şifreli metin ile MAC oluşturulur yani şifreli metnin sonuna etiket değeri eklenir. IPSec (Internet Protocol Security-İnternet Protokolü Güvenliği) protokolünde kullanılır [36]. En güvenli tasarım budur, çünkü şifreli metnin bütünlüğü sağlanmış olur [52].

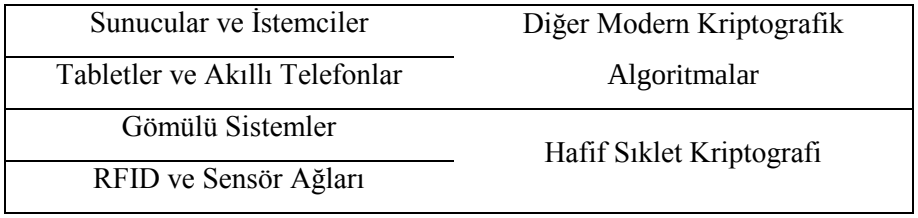
Bütünlük ve gizliliğin bir arada sağlandığı AE algoritmaları için de farklı çalışma modları mevcuttur, bu modlara XCBC (Extended Cipher Block Chaining Encryption-Genişletilmiş Şifre Bloğu Zincirleme), OCB (Offset Code Book Mode-Ofset Kod Defteri), OCB3, GCM (Galois/Counter Mode-Galois/Sayaç Modu), CCM (Counter with CBC-MAC Mode-CBC-MAC Sayaç Modu), EAX modları örnek verilebilir. Bu modların ve literatürde yer alan diğer AE modlarının ayrıntılı performans karşılaştırmaları [53-55] çalışmalarında verilmiştir. Özellikle 2015 yılında Gueron ve Lindell tarafından önerilen GCM-SIV (Galois/Counter Mode-Synthetic Initialization Vector-Galois/Sayaç Modu-Sentetik İlklendirme Vektörü)

modu [55] benzersiz IV değerinin tekrar kullanımına karşı dirençli ve diğer AE modlarına göre yüksek performanslıdır.

1.7. Hafif Sıklet Blok Şifreler

Özellikle kaynak kısıtlı sistemler üzerinde güvenlik ve verimlilik arasında dengenin sağlanması önemlidir. Hafif sıklet blok şifreler sistem kaynakları kısıtlı ya da sınırlı bu cihazların/sistemlerin (IoT cihazlar, RFID, sensör ağları, sağlık hizmetleri, otomotiv sistemleri, siber sistemler vb.) bilgi güvenlğini sağlamak amaçlı geliştirilmişlerdir. Kısıtlı kaynakla kasıt sınırlı alan (yazılım/donanım) kullanımı, düşük güç ve enerji tüketimidir.

Genel amaçlarla kullanılan bilgisayarlardan üzerinde yeni uygulamaların (insansız araçlar, sağlık izleme sistemleri vb.) gerçeklenebildiği özel kaynak kısıtlı cihazlara geçiş, güvenlik problemleri ve kaynak kısıtlı cihazlar için uygun kriptografik algoritmaların eksikliği konuları, NIST'i (National Institute of Standards and Technology-Ulusal Standartlar ve Teknoloji Enstitüsü) bu alanlarda birtakım çalışmalar yapmaya yöneltmiştir. Kurum "Lightweight Crypto Project" (Hafifsıklet Kripto Projesi) [38] altında bazı konferans, etkinlik ve atölyeler düzenlemiştir. Düzenlenen etkinliklerde elde edilen bulgular NIST tarafından analiz edilerek, 2017 yılında teknik rapor yayımlanmış [57], raporda hafif sıklet kriptografi tabanlı cihazlar, hafif sıklet algoritmaların tasarımı, hafif sıklet kriptografi için standartlar başlıkları altında bulgular ayrıntılı olarak verilmiştir.



Şekil 1.11. Aygıt/cihaz spektrumu [57]

Şekil 1.11'deki gibi tüm cihazlar bir spketrumda düşünülürse, sunucular, istemciler, tabletler ve akıllı telefonlar bu spektrumun üst katmanlarında yer alırken, gömülü sistemler, RFID ve sensör ağları sırasıyla spektrumun alt katmanlarında yer alır. Hafif sıklet kriptografi bu alt katmanlarda yer alan kısıtlı kaynaklara sahip cihazlara

odaklanırken, spektrumun üst katmanlarında yer alan cihazlar üzerinde ise diğer modern kriptografik algoritmalar iyi performans sağlar [57].

Hafif Sıklet kriptografi için hedef cihaz olan mikrodenetleyiciler geniş bir kullanım alanına sahiptir. 8-bit, 16-bit, 32-bit mikrodenetleyiciler yaygın kullanılan mikrodenetleyici türlerinden olsa da 4-bit mikrodenetleyiciler ultra-düşük maliyetli uygulamalar için kullanılır. Bu cihazlar çok az sayıda komut seti içerirler, bu da kriptografik aloritmaları çalıştırmak için çok sayıda döngüye ihtiyaç duyulmasına sebep olur. Bu nedenle bu cihazların güvenliği için sınırlı miktarda enerji kullanan hafif sıklet şifreleme algoritmaları kullanılır. RFID etiketleri pille çalışmazlar, çevreden sadece sınırlı miktarda güç elde edebilirler. Bu nedenle bu cihazlar, az sayıda donanımsal kapı (GE) (Gate Equivalent-Kapı Eşdeğerleri) kullanılarak tasarlanmış, zamanlama ve güç gereksinimlerini karşılayabilen kriptografik algoritmalara ihtiyaç duyarlar [57].

Hafif sıklet kriptografide bir algoritma tasarlanırken, performans ve kaynak kullanımı arasında bir denge olması hedeflenir. Performans güç ve enerji tüketimi, gecikme süresi (latency) ve işlem hacmi gibi terimlerle ifade edilebilir. Güç ve enerji tüketimi kaynak kısıtlı cihazların doğası gereği önemli metriklerdir. Güç tüketimi özellikle çevresindeki devre elemanlarından gücü çeken cihazlar için önemliyken, enerji tüketimi ise (yani belirli bir zaman dilimindeki güç tüketimi) özellikle sabit miktarda depolanmış enerjiye sahip batarya ile çalışan cihazlar için önemli bir metriktir. Bazı cihazlardaki pillerin bir kez yerleştirildikten sonra yeniden şarj edilmesi veya değiştirilmesi zor hatta imkânsız olabilir. Güç tüketimi kullanılan algoritmaya bağlı olmasının dışında, eşik gerilimi (threshold voltage), saat frekansı (clock frequency) ve uygulama için kullanılan teknoloji parametrelerine de bağlı olduğu unutulmamalıdır. Gecikme özellikle çok hızlı tepki sürelerinin gerekli olduğu uygulamalarda önemlidir. Bir işlemin ilk isteği ile üretilen çıktı arasındaki zamanın ölçüsü olarak tanımlanabilir. Örneğin bir şifreleme işleminin gecikmesi; bir bloğun şifrelenmesi için gereken saat çevrimi (clock cycle) sayısıdır. Verimlilik, yeni çıktıların (şifreli metin, kimlik doğrulama etiketleri vb.) üretilme hızı olarak tanımlanabilir. Diğer modern kriptografik algoritmaların aksine yüksek verim hafif sıklet şifreleme algoritmaları için bir tasarım hedefi olmayabilir ancak çoğu uygulamada orta verimlilik hedeflenir. Bir algoritmanın donanım uygulaması için,

kapı alanı (gate area), GEs veya mantık blokları (mantıksal elemanlar, uyarlamalı mantık modülleri) terimleri kullanılır. Bir uygulamanın kapladığı alan μm^2 ile ifade edilirken, kullanılan teknolojiye ve standart hücre kütüphanesine bağlıdır. Yazılım için ise, yazmaç (register), RAM (Random Access Memory-Rastgele Erişimli Bellek) ve ROM (Read Only Memory-Sadece Okunabilir Bellek) kullanımı önemli performans kriterleridir. Program kodu, S-kutusu ve döngü anahtarlarını saklamak için ROM kullanılır, ROM diğer bir deyişle şifrenin yani uygulamanın kod büyüklüğünü/boyutunu (code size) verir [58]. RAM ise hesaplamalarda kullanılan ara değerleri saklamak için kullanılır [57]. Hatzivasilis ve arkadaşlarının 2018 yılındaki çalışmasında [58] özellikle hafif sıklet blok şifrelerin yazılım ve donanım performans karşılaştırmaları için kullanılan metrikler ve hesaplamaları ayrıntılı olarak verilmiştir.

NIST aynı raporda [57] kriptografik algoritmaları tasarlarken üç temel özelliğin hedef alındığı bir profil oluşturmuştur, bunlar hedeflenen cihazların fiziksel özellikleri (karakteristikleri), uygulamaların uygulandığı performans özellikleri ve güvenlik özellikleridir. Tüm bu özellikler ve bu özelliklerin birimleri Şekil 1.12'de verilmiş olup, hedef platform için kriptografik algoritma geliştirilirken önem teşkil eder.

Fiziksel Özellikler	**Performans Özellikleri**	**Güvenlik Özellikleri**
-Alan (Ges, mm^2 veya sayısal bloklar) -Bellek (RAM/ROM) -Uygulama şekli (donanım, yazılım veya her ikisi) -Enerji (J)	-Gecikme (saat çevrimi, zaman periyodu) -İşlem Hacmi (Byte başına düşen saat çevrimi (clock cycle per byte)) -Güç (W)	-Minimum güvenlik gücü (Bit) -Saldırı modelleri (ilişkili anahtar (related key) saldırısı [59] vb.) -Yan kanal saldırısı [60] direnci gereksinimleri

Şekil 1.12. Kirptografik algoritmalar için oluşturulan profil özellikleri [57]

Geleneksel blok şifrelere göre hafif sıklet şifrelerin performans avantajları, aşağıdaki hafif tasarım seçenekleri kullanılarak elde edilir [57].

- Daha Küçük Blok Uzunluğu: Bellekten tasarruf etmek için hafif sıklet blok şifreler daha küçük blok uzunluğu kullanırlar, örneğin 128-bit yerine 64 veya 80-bit blok uzunluğu. Ancak geliştirilen şifreleme algoritmasına bağlı olarak, küçük blok

uzunluğu açık metin kurtarma (plaintext recovery) [61] veya anahtar kurtarma (key recovery) [62] gibi saldırılarına yol açabilir.

- Daha Küçük Anahtar Uzunluğu: Bazı hafif sıklet blok şifreler verimlilik için küçük anahtar uzunluğu (96-bitten daha az) kullanırlar, örneğin PRESENT şifreleme algoritması 80-bit anahtar uzunluğu kullanır. NIST tarafından önerilen minimum anahtar uzunluğu 112 bittir [63].
- Daha Basit Döngüler: Hafif sıklet şifreleme algoritmalarında kullanılan bileşenler ve işlemler diğer blok şifrelere göre daha basittir. Hafif sıklet tasarımda 8-bit S-kutuları yerine 4-bit S-kutuları tercih edilir, boyuttaki bu azalma önemli alan tasarrufu sağlar, örneğin AES şifreleme algoritmasının 8-bit S-kutusu 395 GE alan kaplarken, PRESENT şifreleme algoritmasının 4-bit S-kutusu 28 GE yer kaplar [29]. Donanım odaklı tasarımlar için, bit permütasyonları [29] veya rekürsif MDS matrisler (PHOTON [17], LED [64] şifrelerinde kullanılır) karmaşık yayılım tabakası tasarımlarına kıyasla daha çok tercih edilebilir. Döngü içindeki işlemler daha basit olduğunda, güvenliği sağlamak için daha fazla döngü sayısına ihtiyaç duyulur.
- Daha Basit Anahtar Planlama Algoritmaları: Karmaşık anahtar planlama algoritmaları bellek kullanımı, gecikme ve güç tüketimi parametrelerini arttırır. Bu nedenle hafif sıklet şifreleme algoritmalarının çoğu basit anahtar planlama yapılarına sahiptir. Bu da algoritmayı ilişkili anahtar saldırısı gibi saldırılara karşı savunmasız kılar. Literatürde bu saldırıların bazılarını önleyebilecek çalışmalar mevcuttur [63, 65, 66].
- Asgari (azaltılmış, minimuma indirgenmiş) Uygulamalar: Bazı uygulamalar bir aygıtın yalnızca şifreleme veya şifre çözme işlemlerinden birini desteklemesini gerektirebilir. Bir şifrenin sadece gerekli fonksiyonunu uygulamak, kaynak kullanımını azaltır.

1.7.1. Literatürde yer alan ve tez kapsamında kullanılan bazı hafif sıklet blok şifreler

Literatürde özellikle kaynak kısıtlı cihazlar için donanım, yazılım veya hem donanım hem yazılım verimli hafif sıklet blok şifreler geliştirilmiştir. Bu algoritmalardan tez çalışması kapsamında kullanılan algoritmalar ayrıntılı olarak verilmiştir.

PRESENT şifreleme algoritması; SPN mimarisinde, donanım hedefli (1570 GE alan kaplar), 2007 yılında tasarlanmış hafif sıklet blok şifredir. Blok uzunluğu 64-bit, anahtar uzunluğu 80 ve 128-bit, döngü sayısı ise 31'dir. PRESENT, 2012 yılında hafif sıklet kriptografi için standartlaştırılmış ilk hafif sıklet blok şifrelerden biridir: ISO/IEC 29192-2:2012 [67] (International Organization for Standardization/International Electrotechnical Commission-Uluslararası Standartlar Teşkilâtı/Uluslararası Elektroteknik Komisyonu). Güvenli ve etkin olmasının yanında basit (sade) yapıda olması amaçlanmıştır. Anahtar ekleme adımı için XOR işlemi kullanılır. Karıştırma tabakasında 4-bitlik S-kutusu kullanılır, Tablo 1.1'de bu S-kutusu verilmiştir. S-kutusu tersi kendisine eşit değildir bu nedenle, şifre çözme adımında Tablo 1.6'da verilen S-kutusu kullanılır.

Tablo 1.6. PRESENT S-kutusunun tersi [29]

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

Yayılım tabakasında ise donanım odaklı tasarım olduğu için en az sayıda işlem elemanı kullanımına gidilmiştir (örn. transistörler), bu da bit permütasyonu kullanımıyla sağlanmıştır, permütasyon işlemi durum bitlerinin pozisyonlarını değiştiren basit bir yapıdır. Bit permütasyonu P, $P^2 = P^{-1}$ dönüşümünü kullanır ve her i. bit için permütasyon değeri Eşitlik (1.24) ile ifade edilebilir.

$$P(i) = \begin{cases} 16 \cdot i \pmod{63}, & i \neq 63 \\ 63, & 63 \end{cases} \tag{1.24}$$

Anahtar planlama algoritması, 80-bit ve 128-bit versiyonlara sahiptir. 80-bit anahtar için yapı şöyledir, i. döngü için, ana anahtarın bitlerinin $(k_{79}k_{78}\ldots k_0)$ saklandığı K yazmaçının soldaki 64-biti $(k_{79}k_{78}\ldots k_{16})$, bir K_i $(K_{63}K_{62}\ldots K_0 = k_{79}k_{78}\ldots k_{16})$ yazmaçına atılır ve K_i yazmacındaki bu bitler döngü anahtarını oluşturur, döngü anahtarı oluşturulduktan sonra K yazmaçının içeriği aşağıdaki gibi güncellenir [29]:

- Bu yazmaçtaki bitler 61-bit sola kaydırılır: $[k_{79}k_{78}\ldots k_1k_0] = [k_{18}k_{17}\ldots k_{20}k_{19}]$,
- Ana anahtarın en soldaki 4-bitinin değerleri $(k_{79}k_{78}k_{77}k_{76})$ PRESENT S-kutusu kullanılarak değiştirilir: $[k_{79}k_{78}k_{77}k_{76}] = S[k_{79}k_{78}k_{77}k_{76}]$,

• Döngü sabiti değeri ile ana anahtarın $k_{19}k_{18}k_{17}k_{16}k_{15}$ bitleri XOR'lanır: $[k_{19}k_{18}k_{17}k_{16}k_{15}] = [k_{19}k_{18}k_{17}k_{16}k_{15}] \oplus \text{döngü_sabiti}$

Bu adımlar gerçekleştirildikten sonra K yazmaçının soldaki 64-biti K_i yazmacına atılarak döngü anahtarı oluşturulur, tüm işlemler her döngüde tekrarlanır.

PRESENT şifresi 25 döngü sonunda 50 aktif S-kutusu sayısına ulaşır. PRESENT S-kutusunun maksimum diferansiyel olasılığı 2^{-2} 'dir, PRESENT-80 için, 25 döngü sonunda şifrenin diferansiyel karakteristiğinin maksimum olasılığı $(2^{-2})^{50} = 2^{-100}$'ye ulaşır $2^{-100} \leq 2^{-80}$ olduğundan 2^{-2} şifre diferansiyel kriptanalize karşı güçlüdür. Şifrenin doğrusal kriptanalizine karşı 2^{84} açık metin/şifreli metin çiftlerinin bilinmesi gerekmektedir [29].

KLEIN şifreleme algoritması [68]; SPN mimarisinde, donanım ve yazılım hedefli, 2012 yılında tasarlanmış hafif sıklet blok şifredir. Blok uzunluğu 64-bit, anahtar uzunlukları 64, 80 ve 96-bit, döngü sayıları sırası ile 12, 16 ve 20'dir. Algoritmanın giriş ve çıkıları tek boyutlu byte dizileri olarak kabul edilir. Her döngü AddRounKey (Anahtar Ekleme), SubNibbles (Parçaların Yer Değiştirmesi), RotateNibbles (Parçaların Döndürülmesi) ve MixNibbles (Parçaların Karıştırılması) adımlarından oluşur. Döngü başlangıcında anahtar ekleme adımı XOR işlemi ile gerçekleştirilir. XOR sonucu her biri 16 adet 4-bitlik parçalara (nibbles) ayrılır. SubNibbles, yani karıştırma tabakasında 4-bitlik S-kutusu kullanılır, Tablo 1.7'de KLEIN S-kutusu verilmiştir. S-kutusu tersi kendisine eşittir bu nedenle şifre çözme işleminde aynı S-kutusu kullanılır. PRESENT şifreleme algoritmasının S-kutusu daha iyi çığ etkisi sağlar [29, 68] ancak tersi kendisine eşit değildir.

Tablo 1.7. KLEIN S-kutusu [68]

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

RotateNibbles, adımda ise 4 bitlik 16 parçaya döndürme işlemi uygulanır, bu adımda her döngüde 2 byte sola dairesel olarak kaydırılır. Sola kaydırma işleminden sonra her bir parçanın pozisyonu Tablo 1.8'de verilmiştir.

Tablo 1.8. KLEIN RotateNibbles adımından sonra 4-bitlik parçaların pozisyonları

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

MixNibbles yani yayılım tabakasında ise AES şifresinin MixColumns dönüşümünde kullanılan doğrusal dönüşüm matrisi kullanılır. PRESENT şifresinde olduğu gibi birçok hafif sıklet şifreleme algoritması yayılım tabakasında bit permütasyonu işlemlerini kullansada, KLEIN şifresinde, yayılma özelliği ve yazılım performansı arasında denge sağlamak için byte-tabanlı matris çarpımı tercih edilmiştir. Çünkü bit permütasyonları yazılım uygulamaları için verimliliğin düşmesine neden olur [68]. Anahtar planlama adımında ise tüm alt anahtarların önceden hesaplandığı geleneksel yöntemin aksine anında (on-the-fly) anahtar planlama tercih edilmiştir çünkü bu yapı donanım verimli uygulamalar için verimlilik sağlar [68]. Anahtar planlama algoritması, 64, 80 ve 128-bit versiyonlara sahiptir. Algoritma 4 temel adım içerir. İlk adımda alt anahtar iki eşit parçaya ayrılır, ikinci adımda bu parçaların her biri sola bir byte kaydırılır, üçüncü adımda Feistel yapısındaki gibi parçalar karıştırılarak dallar oluşturulur, son adımda ise sol dalın üçüncü byte'ı ile döngü sayacı i XOR'lanırken, sağ dalın ikinci ve üçüncü byte'ları KLEIN S-kutusundan geçirilir.

Yapı aşağıdaki gibi özetlenebilir [54]:

- i döngü sayacı, mk ana anahtar, ilk döngüde $i=1$ için, başlangıç alt anahtarı $sk^1 = mk = sk_0^1 \| sk_1^1 \| \cdots \| sk_t^1$ olmak üzere t=7/9/11 değerlerini sırasıyla 64/80/96-bit anahtar uzunlukları için alır. sk^i , i. alt anahtar olmak üzere, alt anahtar $a = (sk_0^i, sk_1^i, \cdots, sk_{\lfloor \frac{t}{2} \rfloor}^i)$ ve $b = (sk_{\lceil \frac{t}{2} \rceil}^i, sk_{\lceil \frac{t}{2} \rceil + 1}^i, \cdots, sk_t^i)$ şeklinde iki eşit parçaya ayrılır, örneğin 64 bit anahtar için $a = (sk_0^i, sk_1^i, sk_2^i, sk_3^i)$, $b = (sk_4^i, sk_5^i, sk_6^i, sk_7^i)$ 'dir.
- (a,b) pozisyonları bir byte sola kaydırılır, $a' = (sk_1^i, \cdots, sk_{\lfloor \frac{t}{2} \rfloor}^i, sk_0^i)$, $b' = (sk_{\lceil \frac{t}{2} \rceil + 1}^i, \cdots, sk_t^i, sk_{\lceil \frac{t}{2} \rceil}^i)$ olur. Örneğin 64 bit anahtar için $a' = (sk_1^i, sk_2^i, sk_3^i, sk_0^i)$, $b' = (sk_5^i, sk_6^i, sk_7^i, sk_4^i)$ 'dir.

- (a', b') çifti Feistel yapısındaki gibi değiştirilir şöyle ki; sol dal $a'' = b'$ olurken, sağ dal $b'' = a' \oplus b'$ olur. Örneğin 64 bit anahtar için $a'' = (sk_5^i, sk_6^i, sk_7^i, sk_4^i)$, $b'' = (sk_1^i, sk_2^i, sk_3^i, sk_0^i) \oplus (sk_5^i, sk_6^i, sk_7^i, sk_4^i)$ olur.
- a'' 'nın üçüncü byte'ı ile i XOR'lanır. b'''nin ikinci ve üçüncü byte'ları KLEIN S-kutusundan geçirilir. Örneğin 64 bit anahtar için $sk_2^{i+1} = sk_7^i \oplus i$, $sk_5^{i+1} = S[(sk_2^i \oplus sk_6^i)]$, $sk_6^{i+1} = S[(sk_3^i \oplus sk_7^i)]$ olur.

KLEIN şifresi 4 döngü sonunda 15 aktif S-kutusu sayısına ulaşır. KLEIN S-kutusunun maksimum diferansiyel olasılığı 2^{-2} 'dir, KLEIN-64 için, 12 döngü sonunda şifrenin diferansiyel karakteristiğinin maksimum olasılığı $(2^{-2})^{45} = 2^{-90}$ 'a ulaşır, $2^{-90} \leq 2^{-64}$ olduğundan şifre diferansiyel kriptanalize karşı güçlüdür. S-kutusunun doğrusal karakteristiği 2^{-2} olup, KLEIN-64 için, 12 döngü sonunda doğrusal yaklaşımın maksimum bias değeri $(2^{-2})^{45} = 2^{-90}$ olduğundan şifre doğrusal kriptanalizine karşı güçlüdür.

PRESENT ve KLEIN şifreleme algoritmalarının anahtar planlama algoritmaları tez kapsamında SAC testinden geçirilmiş, elde edilen bulgular dördüncü bölümde verilmiştir.

SPN mimarisinin yanı sıra Feistel mimarisinde tasarlanmış algoritmalar da tez çalışmasında performans karşılaştırmaları için kullanılmıştır.

HIGHT şifreleme algoritması [69]; iteratif yapıya sahip genelleştirilmiş Feistel mimarisinde, donanım hedefli (3048 GE alan kaplar), 2006 yılında tasarlanmış hafif sıklet blok şifredir. Blok uzunluğu 64-bit, anahtar uzunluğu 128-bit, döngü sayısı ise 32'dir. HIGHT XOR, modüler toplama, sola kaydırma gibi basit işlemlerden oluşur. Anahtar planlama, başlangıç çevrimi (initial transformation), döngü fonksiyonu ve bitiş çevrimi (final transformation) adımlarından oluşur. Anahtar planlama algoritması iki fonksiyondan oluşur bunlar anahtar beyazlatma [69] bytle'larını üreten WhiteningKeyGeneration (Anahtar Beyazlatma Üretimi) ve alt anahtarların üretildiği SubkeyGeneration (Alt Anahtar Üretimi) fonksiyonlarıdır. WhiteningKeyGeneration fonksiyonundan dönen anahtarlar başlangıç ve bitiş

çevrimlerinde kullanılır. SubkeyGeneration fonksiyonunda 128 alt anahtar üretilir. Şifreleme adımında her döngüde 4 alt anahtar kullanılır. Ayrıca bu fonksiyonun içinde döngü sabitlerinin üretildiği alt fonksiyon mevcuttur, bu fonksiyonda 7 bitlik 128 adet döngü sabiti üretilir. Başlangıç çevriminde, açık metin ve anahtar beyazlatma byte'ları kullanılarak 2^8'de modüler toplama veya XOR işlemleri ile dönüşüm gerçekleştirilir. Döngü fonksiyonunda iki yardımcı fonksiyon kullanılır bu fonksiyonlarda sola kaydırma işlemleri yapılır. Bitiş çevrimindeki işlemler başlangıç çevrimi ile aynıdır. Bitiş çevriminde çıktı olarak şifreli metin bitleri üretilir. Şifrenin doğrusal ve diferansiyel kriptanalize karşı direnci [69]'de ayrıntılı olarak verilmiştir.

SIMON ve SPECK [40] şifreleme algoritmaları; NSA (National Security Agency-Ulusal Güvenlik Ajansı) tarafından 2013 yılında tasarlanmış hafif sıklet blok şifrelerdir. SIMON donanım, SPECK yazılım için optimize edilmiş olup, her iki şifre de Feistel mimarisindedir ancak Feistel fonksiyonlarının tasarımı birbirinden farklıdır. Şifrelere ait blok, anahtar uzunlukları ve döngü sayıları Tablo 1.9'da verilmiştir.

Tablo 1.9. SIMON VE SPECK şifrelerinin genel özellikleri

SIMON			**SPECK**		
Blok Uzunluğu	**Anahtar Uzunluğu**	**Döngü Sayısı**	**Blok Uzunluğu**	**Anahtar Uzunluğu**	**Döngü Sayısı**
32	64	32	32	64	22
48	72/96	36	48	72/96	22/23
64	96/128	42/44	64	96/128	26/27
96	96/144	52/54	96	96/144	28/29
128	128/192/256	68/69/72	128	128/192/256	32/33/34

SIMON şifresinde şifreleme ve şifre çözme adımlarında n-bit kelime üzerinde XOR, AND (&) ve sola ve sağa dairesel kaydırma (S^j : j bit sola dairesel kaydırma, S^{-j} :j bit sağa dairesel kaydırma) işlemleri kullanılır. $k \in GF(2)^n$ ve k döngü anahtarı, $f(x) = (S^1 x \,\&\, S^8 x) \oplus S^2 x$ olmak üzere şifreleme adımında kullanılan anahtar bağımsız döngü fonksiyonu $R_k : GF(2)^n \times GF(2)^n \to GF(2)^n \times GF(2)^n$ bir Feistel haritalama ve $R_k(x,y) = (y \oplus f(x) \oplus k, x)$ olarak ifade edilirken, şifre çözme adımında kullanılan döngü fonksiyonunun tersi ise $R_k^{-1}(x,y) = (y, x \oplus f(y) \oplus k)$ şeklindedir. T döngü sayısı olmak üzere, anahtar planlama algoritması ana anahtardan $k_0, \ldots, k_{T-1}$ anahtar kelime dizisini üretir. Şifreleme haritası

$R_{k_{T-1}} \circ \cdots \circ R_{k_1} \circ R_{k_0}$ sağdan sola doğru döngü fonksiyonlarının birleşimi şeklinde yazılır. Şifrenin devre boyutunun arttırılmaması için anahtar beyazlatma adımları kullanılmamıştır. Anahtar planlama algoritması için, $m = \{2,3,4\}$, $c = 2^n - 4$, m tane anahtar kelime $(k_{m-1},\ldots,k_1,k_0)$, z_j sabit bir dizi, $0 \le i < T - m$ olmak üzere $(z_j)_i$ z_j nin i. bitini ifade eder. Döngü anahtarları Eşitlik (1.25) ile elde edilir.

$$k_{i+m} = \begin{cases} c \oplus (z_j)_i \oplus k_i \oplus (I \oplus S^{-1})S^{-3}k_{i+1}, & m = 2, \\ c \oplus (z_j)_i \oplus k_i \oplus (I \oplus S^{-1})S^{-3}k_{i+2}, & m = 3, \\ c \oplus (z_j)_i \oplus k_i \oplus (I \oplus S^{-1})(S^{-3}k_{i+3} \oplus k_{i+4}), & m = 4, \end{cases} \quad (1.25)$$

SPECK, SIMON gibi hem yazılım hem donanım uygulamaları üzerinde iyi performans sağlamak için tasarlanmış, ancak yazılım performansı için optimize edilmiştir. Şifreleme adımında n-bit kelime üzerinde XOR, 2^n'de modüler toplama $(+)$ ve çıkarma $(-)$, sola ve sağa dairesel kaydırma (S^j, S^{-j}) işlemleri kullanılır. k döngü anahtarı olmak üzere, şifreleme adımında kullanılan anahtar bağımsız döngü fonksiyonu $R_k : GF(2)^n \times GF(2)^n \to GF(2)^n \times GF(2)^n$ bir haritalamadır. Blok uzunluğu 32 için $\alpha = 7, \beta = 2$ diğer blok uzunluklarında $\alpha = 8, \beta = 3$ olmak üzere, döngü fonksiyonu $R_k(x,y) = ((S^{-\alpha}x + y) \oplus k, S^{\beta}y \oplus (S^{-\alpha}x + y) \oplus k)$ olarak ifade edilirken, döngü fonsiyonunun tersi $R_k^{-1}(x,y) = (S^{\alpha}((x \oplus k) - S^{-\beta}(x \oplus y)), S^{-\beta}(x \oplus y))$ olarak ifade edilir. SPECK aslında iki farklı Feistel–benzeri haritanın birleşimi olarak gerçekleştirilir, şöyleki $(x,y) \mapsto (y, (S^{-\alpha}x + y) \oplus k)$ ve $(x,y) \mapsto (y, S^{\beta}x \oplus y)$. Anahtar planlama algoritması k_i döngü anahtarlarını üretmek için döngü fonksiyonunu kullanır. $m = \{2,3,4\}$, $l_i, k_0 \in GF(2)^n$, $l_{i+m-1} = (k_i + S^{-\alpha}l_i) \oplus i$, $k_{i+1} = S^{\beta}k_i \oplus l_{i+m-1}$ olmak üzere, K anahtarı $K = (l_{m-2},\ldots l_0, k_0)$ olarak ifade edilir.

PRESENT, KLEIN, HIGHT, SIMON ve SPECK hafif sıklet blok şifrelerin donanım ve yazılım performans karşılaştırmaları Tablo 1.10'da verilmiştir [40].

Tablo 1.10. PRESENT, KLEIN, HIGHT, SIMON, SPECK şifrelerinin yazılım ve donanım performansları [40]

	Donanım			Yazılım		
Uzunluk (Blok/Anahtar)	**Şifre**	**Alan (GE)**	**İşlem Hacmi (Kbps)**	**Flash (byte)**	**SRAM (byte)**	**İşlem Hacmi (Kbps)**
48/96	SIMON	763	15.0	196	0	589
48/96	SPECK	884	12.0	134	0	943
64/80	PRESENT	1030	12.4	487	0	96
64/80	KLEIN	1478	23.6	766	18	168
64/96	SIMON	838	17.8	274	0	540
64/96	SPECK	984	14.5	182	0	888
64/96	KLEIN	1528	19.1	766	18	134
64/128	SIMON	1000	16.7	282	0	515
64/128	SPECK	1127	13.8	186	0	855
64/128	PRESENT	1339	12.1	487	0	96
64/128	HIGHT	3048	188.2	-	-	-
96/96	SIMON	984	14.8	454	0	454
96/96	SPECK	1134	13.8	276	0	866
128/128	SIMON	1317	22.9	732	0	342
128/128	SPECK	1396	12.1	396	2	768
128/128	AES	2400	56.6	943	33	445

Tablo 1.10'a göre [40], şifrelerin ASIC (Application Specific Integrated Circuit-Uygulamaya Özel Tümleşik Devre) uygulamaları VHDL (Very high speed integrated circuit Hardware Description Language-Yüksek Hızlı Tümleşik Devreler İçin Donanım Tanımlama Dili) kullanılarak geliştirilmiş, şifrelerin kapladıkları alan (GE) ve işlem hacmi değerleri (Kbps) donanım performans karşılaştırmasında kullanılmıştır. Yazılım performansı karşılaştırması için ise 8-bit AVR ATmega128 [70] mikrodenetleyicisi üzerinde yapılmıştır. Cihazın 128 Kilobyte programlanabilir flash bellek ve 4-Kilobyte SRAM (Static Random Access Memory-Durağan Rastgele Erişimli Bellek) kapasiteleri vardır. Bu belleklerin kullanım miktarı (byte) ve işlem hacmi metrikleri için verilen şifreler karşılaştırılmıştır. [40]'deki çalışmada HIGHT şifreleme algoritması için yazılım performans karşılaştırması yapılmamıştır. Tablodan da görüleceği gibi SIMON ve SPECK şifreleme algoritmaları hem yazılım hem donanım uygulamaları için en iyi sonuçları vermektedir. Literatürde yayınlanan hafif sıklet blok şifrelerin kapsamlı yazılım/donanım performans karşılaştırmalarının yapıldığı birçok çalışma mevcuttur [4, 5, 58, 71, 72]. Bu çalışmalardan farklı platformlar üzerinde hafif sıklet blok şifrelerin performansları incelenebilir.

1.7.2. Hafif sıklet blok şifre çalışma modları ve JAMBU AE çalışma modu

Bölüm 1.5'de verilen blok şifre çalışma modlarından özellikle CBC ve CTR modları hafif sıklet blok şifreler için de kullanılır.

Hafif sıklet blok şifreler ayrıca AE'de de kullanılır. Literatürde, özellikle kaynak kısıtlı cihazlar için kullanılabilecek hafif sıklet AE algoritmaları geliştirilmiştir. Bu algoritmalara Hummingbird-2 [73], ALE [74], Fides [75] AE algoritmaları örnek verilebilir. Ancak bu şifreleme algoritmalarının (şemaları) herbiri kendine özel tasarıma sahiptir bu nedenle bu algoritmalar, bir şifreleme şemasını AE şifreye dönüştürebilen bir şifreleme modu olarak kullanılamaz. Bu da güvenli bir hafif sıklet AE şifreleme algoritması geliştirmenin zorluğunu ortaya koyar. ALE ve Fides algoritmaları geliştirildikten kısa bir süre sonra, bu algoritmaların güvenlik zafiyetlerinin olduğu kanıtlanmıştır [76].

Wu ve Huang 2014 yılında ilk versiyonunu yayımladıkları JAMBU [76] olarak adlandırılan hafif sıklet AE çalışma modunu önermişlerdir. 2016 yılında çalışmanın yeni versiyonu JAMBU (v2.1) geliştirilmiştir [77]. Yazarlar, 2015 yılında düzenlenen CAESAR yarışmasına (çalışmaların AE için, güvenlik, uygulanabilirlik ve sağlamlık kategorilerinde değerlendirildiği bir yarışmadır) başvurmuşlar ancak çalışma 2018 Mart ayında gerçekleştirilen 4. turda elenmiş, finale kalamamıştır. Yapılan araştırmalarda JAMBU modundan farklı olarak, literatürde geliştirilen hafif sıklet AE çalışma moduna rastlanamamıştır.

JAMBU (v2.1) modu, 4 temel işlem adımı içerir, ilklendirme (initialization), AD'nin işlenmesi (processing the associated data), açık metnin işlenmesi (processing the plaintext) yani şifreleme, sonlandırma ve etiket üretimi (finalization and tag generation). Bu adımlara ait ayrıntılı bilgiler aşağıda verilmiştir [77].

K döngü anahtarı, E_K K anahtarını kullanarak bir bloğu şifreleme işlemi ve IV ilklendirme vektörü olmak üzere, JAMBU 2n-bit blok uzunluğu için n-bit IV kullanır. Başlangıç durumu $S_{-1} = (0^n, IV)$ olmak üzere (n-bit 0 vektörü ve n-bit IV birleşimi), ilklendirme adımında Eşitlik (1.26)'da verilen işlemler sırasıyla yapılır:

$$
\begin{aligned}
&1.\,(X_{-1}, Y_{-1},) = E_K(S_{-1});\\
&2.\qquad R_0 = X_{-1};\\
&3.\;\; S_0 = (X_{-1}, Y_{-1} \oplus 5).
\end{aligned}
\tag{1.26}
$$

AD'nin işlenmesi adımında AD n-bit bloklara bölünür ve ardışık olarak işlenir. Son blok için tamamlama yapılır. Bu işlem için başlangıçta "1" biti devamında AD'nin uzunluğu n-bit olacak şekilde "0" bitleri ardına eklenir, eğer AD'nin uzunluğu 0 ise $(1 \| 0^{n-1})$ olacak şekilde ekleme yapılır. N_A tamamlama işleminden sonra AD bloklarının sayısı, A_i AD'nin i. bloğundaki bitler olmak üzere, AD işlenmesi adımında $i = 0 : N_A - 1$'e kadar Eşitlik (1.27)'de verilen işlemler tekrarlanır:

$$
\begin{aligned}
&1.\quad (X_i, Y_i) = E_K(S_i),\\
&2.\quad U_{i+1} = X_i \oplus A_i,\\
&3.\quad V_{i+1} = Y_i \oplus R_i \oplus 1,\\
&4.\quad S_{i+1} = (U_{i+1}, V_{i+1}),\\
&5.\quad R_{i+1} = R_i \oplus U_{i+1}
\end{aligned}
\tag{1.27}
$$

JAMBU modunun şifreleme adımında ise açık metin n-bit bloklara bölünür ve son blok için eğer gerekiyorsa tamamlama işlemi yapılır. Tamamlama işlemi AD'nin işlenmesi adımıyla aynıdır. P_i, C_i sırasıyla i. açık metin ve şifreli metin blokları, N_P tamamlama adımından sonra açık metin bloklarının sayısı olmak üzere şifreleme adımında $i = N_A : N_A + N_P - 1$'e kadar Eşitlik (1.28)'de verilen işlemler tekrarlanır:

$$
\begin{aligned}
&1.\quad (X_i, Y_i) = E_K(S_i),\\
&2.\quad U_{i+1} = X_i \oplus P_{i-N_A},\\
&3.\quad V_{i+1} = Y_i \oplus R_i,\\
&4.\quad S_{i+1} = (U_{i+1}, V_{i+1}),\\
&5.\quad R_{i+1} = R_i \oplus U_{i+1},\\
&6.\quad C_{i-N_A} = P_{i-N_A} \oplus V_{i+1}
\end{aligned}
\tag{1.28}
$$

Eğer $i < N_A + N_P - 1$ veya son açık metin bloğu kısmi bloksa (son blok için tamamlama işlemi yapıldıysa) 6. adım hesaplanır aksi takdirde $C_{N_{P-1}}$ hesaplanmaz.

Sonlandırma ve etiket üretimi adımında, şifreleme adımından sonra elde edilen son durum S_{N+1} ve R_{N+1} $(N = N_A + N_P - 1)$ olmak üzere kimlik doğrulama etiketinin üretimi için Eşitlik (1.29)'da verilen işlemler sırasıyla yapılır:

$$
\begin{aligned}
&1.\quad (X_{N+1}, Y_{N+1}) = E_K(S_{N+1}),\\
&2.\quad U_{N+2} = X_{N+1},\\
&3.\quad V_{N+2} = Y_{N+1} \oplus R_{N+1} \oplus 3,\\
&4.\quad R_{N+2} = R_{N+1} \oplus X_{N+1},\\
&5.\quad S_{N+2} = (X_{N+2}, Y_{N+2}),\\
&6.\quad \text{Etiket} : T = R_{N+2} \oplus X_{N+2} \oplus Y_{N+2}
\end{aligned}
\tag{1.29}
$$

Şifre çözme ve doğrulama adımları, şifreleme ve kimlik doğrulama adımlarına benzer. Son blok için XOR işlemi gerçekleştirilmeden önce şifreli metin aynı tamamlama dizisiyle tamamlanır. Eğer şifreli metnin uzunluğu blok uzunluğunun katıysa diğer blok $1 \| 0^{n-1}$ bit dizisi ile tamamlanır. T' şifre çözme işleminden sonra hesaplanır eğer T' ile T birbirine eşit değilse açık metin elde edilemez.

JAMBU modunun donanım ve yazılım performansları diğer AE modlarla kıyaslandığında JAMBU'nun tasarımının donanım odaklı olduğu görülmüştür. 2n-bit blok uzunluğuna sahip bir blok şifreye JAMBU ve literatürde yer alan diğer AE çalışma modları uygulandığında, blok şifre uzunluğu için kullanılan yazmaçlara ek olarak gereken yazmaç uzunluğu (artış) Tablo 1.11'de verilmiştir. Tabloya göre JAMBU'nun en az artışa ihtiyaç duyduğu görülebilir [77].

Tablo 1.11. 2n-bit blok uzunluğundaki blok şifre üzerine AE modları uygulandığında durum bit uzunluklarının karşılaştırmaları [77]

AE Mod	**Durum Bit Uzunluğu (2n+Artış)**	**Artış**
CCM	4n	2n
GCM	6n	4n
OCB3	6n	4n
EAX	8n	6n
COPA	6n	4n
CPFB	6n	4n
ELmD	8n	6n
SILC	4n	2n
CLOC	4n	2n
JAMBU	3n	n

Tez çalışmasında geliştirilen hafif sıklet blok şifre algoritmasının performans analizi için JAMBU modu kullanılmıştır.

Tez çalışmasının bundan sonraki bölümünde tez çalışması kapsamında geliştirilen yapılarla ilgili kullanılan matematiksel ifadeler hakkında detaylı bilgi verilecektir.

2. MATEMATİKSEL ALTYAPI

Bu bölümde, tez kapsamında kullanılacak ifadelerin anlaşılırlığı açısından matematiksel bir altyapı sunulacaktır. Bölüm içerisinde şifreleme algoritmalarının tasarım temelini oluşturan özellikle sonlu cisimler teorisindeki temel kavramlar, tanım ve teoriler, özel matris formları verilmiş olup, ispatlarına [78-81]'den ulaşılabilir.

Tanım 2.1. Z_m toplama (+) ve çarpma (·) tabanlı aritmetik modulo kümesi ve a, b ∈ Z_m olmak üzere aşağıdaki aksiyomlar verilsin,

1. (+) işlemi üzerinde kapalılık özelliği: $a + b \in Z_m$

2. (·) işlemi üzerinde kapalılık özelliği: $a \cdot b \in Z_m$

3. (+) işlemi üzerinde değişme özelliği: $a + b = b + a$

4. (·) işlemi üzerinde değişme özelliği: $a \cdot b = b \cdot a$

5. (+) işlemi üzerinde geçişme özelliği: $(a + b) + c = a + (b + c)$

6. (·) işlemi üzerinde geçişme özelliği: $(a \cdot b) \cdot c = a \cdot (b \cdot c)$

7. (·) işlemi üzerinde dağılma özelliği: $(a \cdot b) \cdot c = a \cdot (b \cdot c)$,

$a \cdot (b + c) = a \cdot b + a \cdot c$

8. (+) işlemi üzerinde 0 birim eleman olmak üzere: $a + 0 = a$, $\forall a \in Z_m$

9. (·) işlemi üzerinde 1 birim eleman olmak üzere: $a \cdot 1 = a$, $a \cdot 0 = 0$, $\forall a \in Z_m$

10. (+) işlemi üzerinde a elemanının tersi: m−a'dır.

11. (·) işlemi üzerinde a elemanının tersi: a^{-1}'dir ve $a^{-1} \cdot a = 1$

Tanım 2.2. Z_m kümesi 1, 2, 5, 6 ve 7 numaralı aksiyomları sağlıyorsa (+) ve/veya (·) işlemlerine göre gruptur denir.

Tanım 2.3. Z_m kümesi grup olmak üzere 3, 4 numaralı aksiyomları da sağlıyorsa Z_m kümesine Abel /Değişmeli (Abelian Group) grup denir.

Tanım 2.4. Z_m kümesi Abelyen grup olmak üzere, 8, 9 numaralı aksiyomları da sağlıyorsa Z_m kümesine halka denir.

Tanım 2.5. Z_m kümesi halka olmak üzere 10, 11 numaralı aksiyomları da sağlıyorsa Z_m kümesine cisim denir. Bir cisim sadece sonlu elemanlardan oluşuyorsa bu cisme sonlu cisim denir.

Tanım 2.6. Z_m bir cisim olmak üzere set $Z_m[x] := \left\{ \sum_{i=0}^{n} a_i x^i : a_i \in Z_m, n \geq 0 \right\}$, Z_m üzerine bir polinom halka olarak isimlendirilir. $Z_m[x]$'in bir elemanı Z_m üzerine polinom olarak isimlendirilir. Pozitif dereceli bir polinom $f(x) = \sum_{i=0}^{n} a_i x^i$ için $\text{derece}(g(x)) < \text{derece}(f(x))$, $\text{derece}(h(x)) < \text{derece}(f(x))$ ve $f(x) = g(x) \cdot h(x)$ şartlarını sağlayacak şekilde iki polinom varsa $f(x)$ polinomu Z_m üzerine indirgenebilir aksi takdirde pozitif dereceli $f(x)$ polinomu Z_m üzerine indirgenemez polinom olarak tanımlanabilir.

Tanım 2.7. Sonlu cisim $GF(2^m)$, elemanları 0 veya 1'lerden oluşan GF(2) sonlu cisminin m. dereceden genişletilmiş bir cismidir denir ve m pozitif bir sayıyı olmak üzere, 2^m elemana sahiptir. GF(2) cismi $GF(2^m)$ cisminin taban cismidir. Sonlu cisim $GF(2^m)$'in 2^m elemanının her biri katsayıları GF(2) cisminden olan ve derecesi $m-1$ olan bir polinom şeklinde ifade edilebilir. Örneğin, x, $GF(2^m)$ cisminin bir elemanı ise o zaman x'in polinom tabanlı gösterimi Eşitlik (2.1) ifadesindeki gibi verilebilir.

$$x_{m-1}\alpha^{m-1} + x_{m-2}\alpha^{m-2} + x_1\alpha + x_0 \qquad (2.1)$$

(2.1) ifadesinde α, $GF(2^m)$ cismini oluşturmak için kullanılan ilkel elemanı temsil eder. $GF(2^m)$ cisminde iki elemanın toplama sonucu XOR işlemi ile elde edilirken çarpma sonucu, iki polinomun çarpım sonucu bir indirgenemez polinom ile indirgenmesi sonucu elde edilir.

Tanım 2.8. Taban cisminden genişletilmiş cismin tüm elemanlarını üretebilen polinoma asal polinom adı verilir.

$a \geq 1$ ve $m \geq 1$ olsun. Eğer $OBEB(a,m) = 1$ ise a ve m için aralarında asal denir. Z_m üzerinde m ile aralarında asal olan tamsayıların sayısı $\phi(m)$ ile gösterilir ve bu fonsiyona Euler phi fonksiyonu denir. $\phi(m)$ m'nin asal asal üslerinin çarpanlara ayrılması ile bulunur.

Teorem 2.1. q herhangi bir asal ya da asal üs olmak üzere, pozitif n değerleri için $GF(q)$ sonlu cismi üzerindeki n. dereceden asal polinomların sayısı $s_q(n) = \frac{\phi(q^n - 1)}{n}$ 'dir.

Örneğin $GF(2^4)$ sonlu cisminin 4. dereceden asal polinom sayısı $s_{2^4}(4) = \frac{\phi(2^4 - 1)}{4} \Rightarrow \frac{\phi(15)}{4} = \frac{(5^1 - 5^0)\cdot(3^1 - 3^0)}{4} = 2$ 'dir. Teorem 2.1 kullanılarak $GF(2^8)$ sonlu cismi için 8. dereceden asal polinom sayısı 16 bulunur. Tablo 2.1'de 5. 5. dereceye kadar asal polinomlar verilmiştir.

Tablo 2.1. n. dereceden asal polinomlar

n	Asal Polinomlar
1	$x + 1$
2	$x^2 + x + 1$
3	$x^3 + x + 1$
4	$x^4 + x + 1$, $x^4 + x^3 + 1$
5	$x^5 + x^2 + 1$, $x^5 + x^3 + 1$, $x^5 + x^3 + x^2 + x + 1$, $x^5 + x^4 + x^2 + x + 1$, $x^5 + x^4 + x^3 + x + 1$, $x^5 + x^4 + x^3 + x^2 + 1$

Tez kapsamında $GF(2^4)$ ve $GF(2^8)$ sonlu cisimleri kullanıldığından sırasıyla bu sonlu cisimler için mod 2'de 4. ve 8. dereceden indirgenemez polinomlar ve bu

polinomların hexadecimal (onaltılı) tabanda gösterimleri Tablo 2.2'de verilmiştir. Polinomlar vektörel olarak düşünüldüğünde, örneğin $x^4+x+1 \Rightarrow (00010011) = (0\times13)$ polinomunu ifade eder ve x^4+x+1 polinomu 0×13 gösterimi ile onaltılı tabanda ifade edilebilir.

Tablo 2.2. GF(2^4) ve GF(2^8) sonlu cisimleri için 4. ve 8. dereceden indirgenemez polinomlar

Sonlu Cisim	**Derece**	**İndirgenemez Polinomlar (Onaltılık Gösterimi)**
GF(2^4)	4.	$x^4 + x + 1$ (0×13), $x^4 + x^3 + 1$ (0×19), $x^4 + x^3 + x^2 + x + 1$ ($0\times1f$)
GF(2^8)	8.	$x^8 + x^4 + x^3 + x + 1$ ($0\times11b$), $x^8 + x^4 + x^3 + x^2 + 1$ ($0\times11d$), $x^8 + x^5 + x^3 + x + 1$ ($0\times12b$), $x^8 + x^5 + x^4 + x^3 + 1$ (0×139), $x^8 + x^6 + x^5 + x + 1$ (0×163), $x^8 + x^6 + x^5 + x^2 + 1$ (0×165), $x^8 + x^6 + x^5 + x^3 + 1$ (0×169), $x^8 + x^6 + x^4 + x^3 + x^2 + x + 1$ ($0\times15f$), $x^8 + x^6 + x^5 + x^4 + x^2 + x + 1$ (0×177), $x^8 + x^7 + x^3 + x + 1$ ($0\times18b$), $x^8 + x^7 + x^5 + x^4 + x^3 + x^2 + 1$ ($0\times1bd$), $x^8 + x^7 + x^6 + x + 1$ ($0\times1c3$), $x^8 + x^7 + x^6 + x^4 + x^2 + x + 1$ ($0\times1d7$), $x^8 + x^7 + x^6 + x^5 + x^2 + x + 1$ ($0\times1e7$), $x^8 + x^7 + x^6 + x^5 + x^4 + x + 1$ ($0\times1f3$)

Tez kapsamında çalışılan GF(2^4) ve GF(2^8) sonlu cisimleri için verimli sonuçların elde edildiği indirgenemez polinomlardan ikisi, GF(2^4) için 0×13 ve GF(2^8) için $0\times1c3$ 'tür. α ilkel eleman olmak üzere, α kullanılarak üretilen GF(2^4)/0×13 cisminin elemanları ve elemanların hexadecimal gösterimi Tablo 2.3'de verilmiştir.

Tablo 2.3. GF(2^4)/0×13 sonlu cisminin elemanları

$\alpha^0 = 1 \rightarrow 1_h$	$\alpha^5 = \alpha^2 + \alpha \rightarrow 6_h$	$\alpha^7 = \alpha^3 + \alpha + 1 \rightarrow B_h$
$\alpha^1 = \alpha \rightarrow 2_h$	$\alpha^{10} = \alpha^2 + \alpha + 1 \rightarrow 7_h$	$\alpha^6 = \alpha^3 + \alpha^2 \rightarrow C_h$
$\alpha^4 = \alpha + 1 \rightarrow 3_h$	$\alpha^3 = \alpha^3 \rightarrow 8_h$	$\alpha^{13} = \alpha^3 + \alpha^2 + 1 \rightarrow D_h$
$\alpha^2 = \alpha^2 \rightarrow 4_h$	$\alpha^{14} = \alpha^3 + 1 \rightarrow 9_h$	$\alpha^{11} = \alpha^3 + \alpha^2 + \alpha \rightarrow E_h$
$\alpha^8 = \alpha^2 + 1 \rightarrow 5_h$	$\alpha^9 = \alpha^3 + \alpha \rightarrow A_h$	$\alpha^{12} = \alpha^3 + \alpha^2 + \alpha + 1 \rightarrow F_h$

Yukarıda tanımlı sonlu cisim elemanları elde edilirken, standart polinom işlemleri uygulanır, elemanlar $\{0,1\}$ katsayılarına sahip polinomlar gibi ifade edilerek toplama işlemi modulo 2 aritmetiğine göre yapılır, bir başka ifade ile aynı dereceli elemanlara

XOR işlemi uygulanır. Örneğin $GF(2^4)/0\times13$ cismi üzerinde 9_h ile 5_h elemanları toplanırsa $9_h+5_h=\alpha^3+1+\alpha^2+1=\alpha^3+\alpha^2$ olur bu değer de C_h elemanına karşılık gelir.

Sonlu cisimler üzerinde çarpma işlemi ise polinom çarpımı ile yapılır, yüksek dereceli elemanlar indirgenemez polinom kullanılarak indirgenir. Örneğin $GF(2^4)/0\times13$ cismi üzerinde 9_h ile 5_h elemanlarının çarpımı $9_h\times5_h=(\alpha^3+1)\times(\alpha^2+1)=\alpha^5+\alpha^3+\alpha^2+1=\alpha^2+\alpha+\alpha^3+\alpha^2+1=\alpha^3+\alpha+1$ olur bu değer de B_h elemanına karşılık gelir.

Tanım 2.9. $\{0,1\}$ lerden oluşan vektör sıranın Hamming ağırlığı $wt(\alpha)=\alpha$ 'daki 1'lerin sayısını temsil eder.

Tanım 2.10. c kod kelimesinin Hamming ağırlığı $wt(c)=c$ olarak temsil edilebilir ve kod kelimesinin 0 olmayan elamanlarının sayısı olarak tanımlanır. Buna ek olarak $(GF(2^m))^n$ vektör uzayından n boyutlu iki vektör arasındaki Hamming uzaklığı da vektörlerin farklılaştığı pozisyon sayısı olarak tanımlanır [1].

Yayılım elemanları matris ile ifade edilebilen doğrusal dönüşümlerdir. Bu nedenle doğrusal dönüşümlerin analizi için aşağıdaki tanımlamalar fayda sağlayacaktır.

Tanım 2.11. Bir doğrusal dönüşüm A(x), $A:(\{0,1\}^m)^n\rightarrow(\{0,1\}^m)^n$ ve $A_{n\times n}$ matris, $x=(x_1,x_2,\ldots,x_n)^T, x_i\in\{0,1\}^m, i=1,\ldots,n$ olmak üzere Eşitlik (2.1)'deki gibi tanımlanabilir.

$$A(x)=A.x^T=\begin{pmatrix} a_{11} & a_{12} & . & . & . & a_{1n} \\ a_{21} & a_{22} & . & . & . & a_{2n} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ a_{n1} & a_{n2} & . & . & . & a_{nn} \end{pmatrix}.\begin{pmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{pmatrix} \quad (2.1)$$

Eğer doğrusal dönüşüm matrisinin elemanları $GF(2)$ üzerinden seçilirse bu doğrusal dönüşüme ikili doğrusal dönüşüm denir.

Tanım 2.12. m bit $GF(2^m)$ üzerinde doğrusal dönüşüm bloğuna giren değerler $A=(a_{ij})_{n\times n}, a_{ij}\in GF(2)$ veya $a_{ij}\in GF(2^m)$ ve $I_{n\times n}$ birim matris olmak üzere A matrisinin sabit nokta sayısı Eşitlik (2.2)'de verilmiştir.

$$F_A = 2^{m(n-\text{rank}(A+I))} \tag{2.2}$$

(2.2)'den de anlaşılacağı gibi $A+I$ matrisinin rankı ne kadar büyük olursa A matrisinin sabit nokta sayısı o kadar küçülür.

Tanım 2.13. $n\times n$ boyutunda bir A matrisinin diferansiyel dal sayısı Eşitlik (2.3)'deki gibi tanımlanabilir.

$$\beta_{\text{diferansiyel}}(A) = \min\{wt(x)+wt(A\cdot x^T) \mid x\in(\{0,1\}^m)^n, x\neq 0\} \tag{2.3}$$

Tanım 2.14. $n\times n$ boyutunda bir A matrisinin doğrusal dal sayısı Eşitlik (2.4)'deki gibi tanımlanabilir.

$$\beta_{\text{doğrusal}}(A) = \min\{wt(x)+wt(A^T\cdot x^T) \mid x\in(\{0,1\}^m)^n, x\neq 0\} \tag{2.4}$$

Tanım 2.15. q bir asal sayının kuvveti olmak üzere, q elemanlı bir $GF(q)=\{\alpha_0,\alpha_1,\ldots\alpha_{q-1}\}$ cismi tanımlansın. $V(n,q)$, $GF(q)$ üzerinde n uzunluğundaki bütün vektörlerin kümesi olsun bu durumda $V(n,q)$ bir vektör uzayıdır. $C\subset V(n,q)$ yani C, $V(n,q)$ vektör uzayının k boyutlu bir alt vektör uzayı ise, C n uzunluğunda, k boyutunda bir doğrusal koddur denir ve $[n,k]$ ile gösterilir.

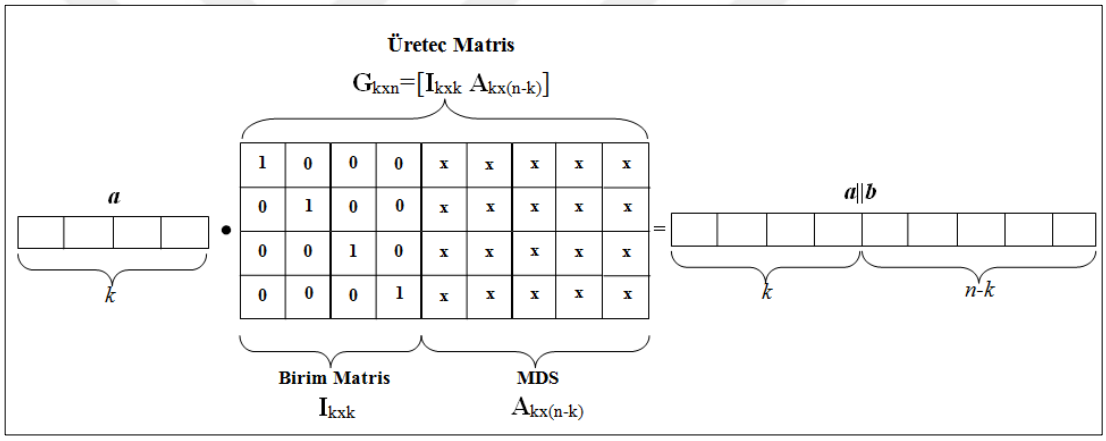
C nin elemanlarının her birine kodsöz denir. Bir kodsözdeki sıfırdan farklı bileşenlerinin sayısı ilgili kodsözün Hamming ağırlığını verir. İki kodsöz arasındaki Hamming uzaklığı ise bu kodsözlerin farklarının Hamming ağırlığına eşittir. C'nin tüm kodsözlerinin sıfırdan farklı en küçük ağırlığına C'nin Hamming ağırlığı denir ve $wt(C)$ ile gösterilir, C'nin sıfırdan farklı en küçük Hamming uzaklığına is C'nin minimum uzaklığı denir [82].

Tanım 2.16. $GF(2^m)$ üzerinde bir $[n,k,d]$ kod, vektör uzayı $(GF(2^m))^n$'in k boyutlu bir alt uzayıdır ve n elemanlı iki vektör arasındaki Hamming uzaklığı minimum d dir.

Bu özellik ile d en büyük değerdir. Doğrusal bir $[n,k,d]$ kod C için bir G üreteç matris satırları c için bir taban oluşturan $k \times n$ boyutunda bir matristir. Doğrusal $[n,k,d]$ kodlar Singleton sınırı olan $d \leq n-k+1$ eşitsizliğini sağlar [1].

Önerme 2.1. Bir doğrusal $[n,k,d]$ kod Singleton sınırı olan $d \leq n-k+1$ eşitsizliğini sağlıyor ise MDS koddur. Alternatif olarak, bir matrisin MDS matris olabilmesi için satır ve sütunlarından oluşturulan tüm alt matrislerinin determinantının 0'dan farklı olması gerekir [1].

Tanım 2.17. $n \times n$ boyutlu A matrisinin determinantı 0'a eşit değilse, A matrisinin tersi vardır ve rankı n'dir. Bu durumda A matrisine tekil olmayan (non-singular) matris denir.



Şekil 2.1. MDS kod üretimi [83]

Şekil 2.1'de MDS kod üretimi verilmiştir. C bir $[n,k,d]$ doğrusal kod ve $G_{k\times n}=[I_{k\times k}\,|\,A_{k\times(n-k)}]$ üreteç matrisi olmak üzere, C ancak ve ancak, A matrisinin tüm kare alt matrisleri tekil değilse MDS koddur, bu durumda A matrisi MDS matrisi ifade eder [83].

Elemanları $GF(2^m)$ üzerinde tanımlı $n \times n$ boyutlu bir matrisin alt matris determinantlarının sayısı Eşitlik (2.5)'de verilmiştir.

$$\sum_{k=1}^{n-2}\left[\binom{n}{n-k}\right]^2 \tag{2.5}$$

MDS matrislerin önemli özellikleri aşağıdaki gibi verilebilir:

- $n \times n$ boyutlu M matrisi ancak ve ancak her kare alt matrisleri tekil olmayan matrisse ve M matrisinin elemanları 0'a eşit değilse M MDS matristir.
- MDS özelliği, transpoz (devrik) alma işleminde korunur. Bu nedenle $n \times n$ boyutlu M matrisi MDS ise M^T 'de MDS matristir.
- MDS özelliği, $n \times n$ boyutlu M matrisinin herhangi bir satır veya sütunu bir c $(c \in GF(2^m), c \neq 0)$ sabitiyle çarpıldığında korunur.

Tanım 2.18. A matrisi dairesel (Circulant) matris olmak üzere $A = \text{circ}(a_0, a_1, \ldots a_{n-1})$ gösterimi ile gösterilir. Matrisin satırları, bir önceki satırın elemanlarını 1 eleman sağa kaydırılmasıyla oluşturulur. $n \times n$ boyutlu dairesel A matrisi Eşitlik (2.6)'daki gibi ifade edilir.

$$A = \text{circ}(a_0, a_1, \ldots a_{n-1}) = \begin{bmatrix} a_0 & a_1 & \cdots & a_{n-1} \\ a_{n-1} & a_0 & \cdots & a_{n-2} \\ \vdots & \vdots & \cdots & \vdots \\ a_1 & a_2 & \cdots & a_0 \end{bmatrix} \tag{2.6}$$

$n \times n$ dairesel matris $A = \text{circ}(a_0, a_1, \ldots a_{n-1})$ polinom ifadesi $A = a_0 I + a_1 P + a_2 P^2 + \ldots a_{n-1} P^{n-1}$ şeklindedir $(P = \text{circ}(0,1,0,\ldots 0))$.

Tanım 2.19. A $n \times n$ Hadamard matris, $n = 2^t, t > 0$ olmak üzere H matrisi $GF(2^m)$ üzerinde Eşitlik (2.7)'deki gibi ifade edilir (A_0 ve A_1 $2^{t-1} \times 2^{t-1}$ Hadamard matris).

$$H = \text{had}(A_0, A_1) = \begin{bmatrix} A_0 & A_1 \\ A_1 & A_0 \end{bmatrix} \tag{2.7}$$

(2.7)'de verilen matris formu finite field Hadamard (Sonlu Cisim Hadamard) matris formudur. Literatürde birçok makalede finite field Hadamard ifadesi yerine kısaca Hadamard ifadesi kullanılır. Bu nedenle tez kapsamında finite field Hadamard formu kısaca Hadamard olarak adlandırılmış ve elemanları $GF(2^m)$ üzerinde tanımlı $n \times n$ boyutlu Hadamard matris $\text{had}(a_0, a_1, \ldots a_{n-1})$ gösterimi ile ifade edilmiştir $0 \leq i \leq n-1$ ve tüm $a_i \neq 0$'dır.

Bu bağlamda 4×4 Hadamard matris Eşitlik (2.8)'deki gibi verilebilir.

$$A = had(a_0, a_1, a_2, a_3) = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_3 & a_0 & a_1 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \tag{2.8}$$

Tanım 2.20. Eğer bir A matrisinin tersi kendisine eşitse ($A = A^{-1}$) bu matrise involutif (tersi kendisine eşit) matris denir.

Elemanları GF(2^m) üzerinde tanımlı $n\times n$ boyutunda bir Hadamard matris H bazı önemli özelliklere sahiptir, bu özellikler aşağıdaki gibidir;

- $H_{i,j} = a_{i\oplus j}$'dir ve a_i H Hadamard matrisinin ilk satır elemanlarını temsil eder.
- H iki yönlü değişmeli (bisimetrik) matristir yani $H = H^T$ ve $HJ = JH$ (J matrisi $n\times n$ boyutunda bir exchange matristir yani $J_{i,n-i+1} = 1$ diğer elemanları 0'dır)
- $H^2 = c^2 I$'dır. ($c = \bigoplus_{i=0}^{n-1} a_i$ ve I $n\times n$ boyutunda birim matristir)

Önerme 2.2. Elemanları GF(2^m) üzerinde tanımlı $n\times n$ boyutunda bir Hadamard matris eğer tersi kendisine eşit ise, ilk satırın elemanlarının toplamı (XOR) 1'e eşittir.

İspat. i satır, j sütun olmak üzere H^2 matrisinin elemanları $H^2_{i,j}$ ile ifade edilebilir. H Hadamard tersi kendisine eşit matris olduğu için $H^2_{i,i} = \bigoplus_{i=0}^{n-1} a_i^2$'dir buradan $(\bigoplus_{i=0}^{n-1} a_i)^2 = 1$ ve $\bigoplus_{i=0}^{n-1} a_i = 1$'dir.

Önerme 2.3. A elemanları GF(2^n) üzerinde tanımlı, 4×4 Hadamard matris olmak üzere $A = had(a_0, a_1, a_2, a_3)$ A matrisinin elemanları birbirinden ve 0'dan farklı olsun. Eğer $\sum_{i=0}^{3} a_i = 1$ ve her kare alt matrislerinin determinantları 0'dan farklı ise A matrisi tersi kendisine eşit MDS matristir denir.

İspat. Önerme 2.1'den, A matrisinin her kare alt matrislerinin determinantlarının 0'dan farklı olması A'nın MDS olması için yeterli koşuldur. A matrisi tersi kendisine

eşit ise, Tanım 2.20'den $A = A^{-1} \rightarrow A^2 = I$ olmalıdır bu koşul ise $\sum_0^3 a_i^2 = 1$ olduğunda sağlanır.

$$A^2 = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_3 & a_0 & a_1 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} . \begin{bmatrix} a_0 & a_1 & a_2 & a_3 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_3 & a_0 & a_1 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} = \begin{bmatrix} \sum_0^3 a_i^2 & 0 & 0 & 0 \\ 0 & \sum_0^3 a_i^2 & 0 & 0 \\ 0 & 0 & \sum_0^3 a_i^2 & 0 \\ 0 & 0 & 0 & \sum_0^3 a_i^2 \end{bmatrix}$$

Önerme 2.4. $m/2 \times m/2$ boyutunda tekil olmayan matrisler kullanılarak GF(2^m) üzerinde $m \times m$ tersi kendisine eşit matrisler üretilebilir. M elemanları GF(2^m)'de tanımlı $m/2 \times m/2$ boyutunda tekil olmayan matris olsun. $m \times m$ boyutunda RM matris $RM = \begin{bmatrix} M & M^{-1} \\ M^3 + M & M \end{bmatrix}$, GF($2^m$) üzerinde tersi kendisine eşit matristir [8].

Tanım 2.21. Bir matris üzerinde eğer soldan sağa doğru her azalan köşegeni sabitse o matris Toeplitz matris olarak adlandırılır. $n \times n$ Toeplitz matrisin genel formu Eşitlik (2.9)'da verilmiştir.

$$T = \begin{bmatrix} a_0 & a_1 & a_2 & \cdots & a_{n-2} & a_{n-1} \\ a_{-1} & a_0 & a_1 & \cdots & a_{n-3} & a_{n-2} \\ a_{-2} & a_{-1} & a_0 & \cdots & a_{n-4} & a_{n-3} \\ a_{-3} & a_{-2} & a_{-1} & \cdots & a_{n-5} & a_{n-4} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{-(n-1)} & a_{-(n-2)} & a_{-(n-3)} & \cdots & a_{-1} & a_0 \end{bmatrix} \quad (2.9)$$

Toeplitz matris ilk satırı ve ilk sütunu bir arada yazılarak $Toep(a_0, a_1, \ldots a_{n-1}, a_{-1}, a_{-2}, \ldots, a_{-(n-1)})$ gösterimi ile ifade edilir. i satır, j sütun olmak üzere, Eşitlik (2.9)'da verilen T matrisi elemanları $T = [m_{i,j}]$, $m_{i,j} = a_{j-i}$ olarak tanımlanabilir.

Tanım 2.22. A bir Vandermonde matris ve $A = van_d(a_0, a_1, \ldots a_{m-1})$ gösterimi ile ifade edilir. $m \times d$ A Vandermonde matrisi $a_0, a_1, \ldots a_{m-1}$ elemanları kullanılarak Eşitlik (2.10)'da verildiği gibidir.

$$A = van_d(a_0, a_1, \ldots, a_{m-1}) = \begin{bmatrix} 1 & a_0 & a_0^2 & \cdots & a_0^{d-1} \\ 1 & a_1 & a_1^2 & \cdots & a_1^{d-1} \\ \vdots & & \ddots & & \\ 1 & a_{m-1} & a_{m-1}^2 & \cdots & a_{m-1}^{d-1} \end{bmatrix} \quad (2.10)$$

Eşitlik (2.10)'da verilen Vandermonde A matrisi elemanları i satır, j sütun olmak üzere $A = [m_{i,j}],\ m_{i,j} = a_i^{j-1}$ olarak olarak tanımlanabilir.

Tanım 2.23. A bir Cauchy matris olmak üzere $n \times n$ A matrisi elemanları $A = [a_{i,j}]$, $a_{i,j} = \frac{1}{x_i + y_j},\ x_i \neq y_j,\ 0 \leq i, j \leq n-1$'dir. A matrisi Eşitlik (2.11)'de verildiği gibidir.

$$A = \begin{bmatrix} \frac{1}{x_0 + y_0} & \frac{1}{x_0 + y_1} & \cdots & \frac{1}{x_0 + y_{n-1}} \\ \frac{1}{x_1 + y_0} & \frac{1}{x_1 + y_1} & \cdots & \frac{1}{x_1 + y_{n-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{x_{n-1} + y_0} & \frac{1}{x_{n-1} + y_1} & \cdots & \frac{1}{x_{n-1} + y_{n-1}} \end{bmatrix} \quad (2.11)$$

Tanım 2.24. $n \times n$ ikili (binary) matrislerin maksimum dal sayısı ikili [2n, n] doğrusal kodların maksimum uzaklığına eşittir. İkili bir doğrusal dönüşümün maksimum dal sayısına sahip olabilmesi için bu kodlar için maksimum uzaklığın alt ve üst sınırlarının eşit olması gerekir. Alt ve üst sınırlarının maksimum uzaklığı birbirine eşit kodlara Maksimum uzaklıkta ikili doğrusal kodlar (Maximum Distance Binary Linear Codes-MDBL) denir [1]. MDBL kodlarla üretilen matrislere de MDBL matris denir.

Tanım 2.25. $p(x)$ $GF(2^m)$ üzerinde indirgenemez polinom, $a \in GF(2^m)/p(x)$ ve $b \in GF(2^m)/p(x)$ olmak üzere; XOR(a), a elemanını keyfi bir b elemanıyla çarpmak için gerekli XOR sayısını ifade eder.

Örneğin $GF(2^4)/0\times13$ cismi üzerinde $a = E_h = 14$ ile çarpma işlemi için kaç XOR sayısına ihtiyaç duyulduğu hesaplanırsa;

(a_3, a_2, a_1, a_0) a elemanının, (b_3, b_2, b_1, b_0) ise keyfi seçilen b elemanının ikili dönüşümü olsun, buradan $a = E_h \rightarrow a = (1,1,1,0)$ ikili dönüşümü ile ifade edilebilir. $(1,1,1,0) \cdot (b_3, b_2, b_1, b_0) = (b_0 \oplus b_1 \oplus b_2 \oplus b_3, b_0 \oplus b_1 \oplus b_2, b_0 \oplus b_1, b_1 \oplus b_2 \oplus b_3)$ olur. Buna göre $GF(2^4)/0\times13$ ismi üzerinde E_h ile çarpma işlemi için gereken XOR sayısı 8'dir (XOR(E_h) = 8).

XOR sayısı ayrıca blok şifrelerin yayılım tabakalarında kullanılacak yayılım matrislerinin donanımsal maliyetini ölçer. Bir sonlu cisim elemanının XOR sayısı kullanılan indirgenemez polinoma göre değişir [11]. Aynı çalışmada [11], XOR sayısı dağılımı bakımından en yüksek standart sapmaya sahip indirgenemez polinomların seçiminin doğru olacağı, çünkü bu polinomların düşük XOR sayılı daha fazla elemana sahip olduğu gösterilmiştir.

Tablo 2.4. $GF(2^4)$ üzerinde indirgenemez polinomlara göre elemanların çarpma işlemi için XOR sayıları [11]

	Sonlu Eleman	**0×13**	**0×19**	**0×1f**
0	0	0	0	0
0×1	1	0	0	0
0×2	α	1	1	3
0×3	$\alpha+1$	5	3	5
0×4	α^2	2	3	3
0×5	α^2+1	6	5	5
0×6	$\alpha^2+\alpha$	5	2	6
0×7	$\alpha^2+\alpha+1$	9	6	6
0×8	α^3	3	6	3
0×9	α^3+1	1	8	5
0×a	$\alpha^3+\alpha$	8	5	6
0×b	$\alpha^3+\alpha+1$	6	9	6
0×c	$\alpha^3+\alpha^2$	5	1	6
0×d	$\alpha^3+\alpha^2+1$	3	5	6
0×e	$\alpha^3+\alpha^2+\alpha$	8	6	5
0×f	$\alpha^3+\alpha^2+\alpha+1$	6	8	3
Standart Sapma		2,68	2,68	1,7075
Aritmetik Ortalama		4,5	4,5	4,5

Tablo 2.4'de $GF(2^4)$ sonlu cismi üzerinde üç faklı indirgenemez polinom için, her bir eleman ile çarpma işlemi için gereken XOR sayısı ve bu XOR sayısı dağılımlarının standart sapmaları verilmiştir. Standart sapması en yüksek olan

polinomlar 0×13 ve 0×19 polinomlarıdır. [11]'de $GF(2^4)$ ve $GF(2^8)$ üzerindeki tüm indirgenemez polinomlar için elemanların XOR sayıları verilmiştir. Tez kapsamında özellikle hafif sıklet blok şifreler için üretilen yayılım matrisleri için standart sapması en yüksek polinomlar seçilmiştir, bunlar $GF(2^4)/0\times13$ ve $GF(2^8)/0\times1c3$ polinomlarıdır. $0\times1c3$ indirgenemez polinomu $GF(2^8)$ üzerinde 7,4634 değeri ile standart sapması en yüksek polinomlardan biridir.

Tezin çalışmasının bundan sonraki bölümünde blok şifrelerin yayılım tabakalarında kullanılan doğrusal dönüşümler incelenecek özellikle MDS matrisler hakkında ayrıntılı bilgi verilecektir.

3. BLOK ŞİFRELERİN YAYILIM TABAKALARINDA KULLANILAN DOĞRUSAL DÖNÜŞÜMLER

Tezin bu bölümünde blok/hafif sıklet blok şifrelerin yayılım tabakalarında kullanılan doğrusal dönüşüm yapıları incelenerek literatürde yer alan açık problemler ele alınmıştır.

Bir blok şifrede perfect diffusion (iyi yayılımın) sağlanması amaçlanır ancak blok şifrelerdeki yayılım tabakaları için çok fazla (bilinen) genel yapıların olmaması literatürde yer alan açık problemlerden biridir [84]. En basitinden bu yapılar için iki temel yaklaşım tanımlanır bunlar: geçici (ad-hoc) (örn. Serpent [85]) ve geniş-iz (wide-trail) (örn. AES, LED) stratejileridir. Geçici yaklaşım çok güvenli ve verimli algoritmaların tasarımını sağlayabilir ancak bilimsel açıdan tatmin edici değildir. Bu yaklaşım en iyi (optimal) izlerin aranması için bilgisayar destekli araçlara ihtiyaç duyar [84]. Geniş-iz stratejisi, Daemen and Rijmen tarafından önerilmiştir [49] . Bir şifrenin güvenliği hakkında fikir sahibi olmak için güçlü bir araç sağlar. Ancak bu yaklaşım verimli yayılım tabakalarının (uygun sayıda aktif S-kutusu ile) tasarımı açısından yardımcı olmaz. Geniş-iz stratejisinde yayılımı ölçmek için kullanılan en önemli yöntem dal sayısıdır [49]. Dal sayısı ardışık iki döngü için minimum aktif S-kutusu sayısını belirtir, yayılım hızını temsil eder aynı zamanda doğrusal ve diferansiyel kritanalize (Eşitlik (2.3), Eşitlik (2.4)) karşı güvenliği ölçmek için kullanılır. Dal sayısının yanında, sabit nokta sayısı da (Eşitlik (2.2)) bir doğrusal dönüşümün gücünü belirleyen önemli metriklerden biridir.

Literatürde yer alan blok/hafif sıklet blok şifreler incelendiğinde yayılım tabakalarının tasarımında kullanılan önemli üç yapı vardır bunlar [7];

- Bit permütasyonu ile oluşturulan yayılım tabakaları,
- Çoklu permütasyon tabanlı yayılım tabakaları,
- Dalların karıştırılması (shuffles) ile oluşturulan yayılım tabakaları

Bit permütasyonu ile oluşturulan yayılım tabakalarında (örn. Serpent, PRESENT, PRINTcipher [86]) yayılımın sağlanması için, S-kutusu çıktı bitleri basit bit

permütasyonu işlemleri ile değiştirilir. Basit bit permütasyonları özellikle donanım verimli uygulamalar için tercih edilir ancak bazı istisnai durumlarda [7] yazılım uygulamalarına da verimli bir şekilde uygulanabilir. Bir çıktı biti, şifrenin bir sonraki katmanının tek bir giriş bitini etkileyebilir bu da bit permütasyonu uygulamalarının problemidir bu nedenle yayılma diğer doğrusal dönüşümlerden daha yavaştır ve daha fazla döngü sayısına ihtiyaç duyar.

Çoklu permütasyon tabanlı yayılım tabakalarında yayılım elemanları kod tabanlıdır. Bu yaklaşımda iki yöntem dikkat çeker bunlar MDS matrisler, diğeri ise MDBL matrislerdir. MDBL matrisler ikili doğrusal dönüşümlerdir, Camellia [87], ARIA [88] ve PRIDE [89] şifreleri bu doğrusal dönüşümleri yayılım tabakasında kullanırlar. Bu matrisler maksimum dal sayısına sahiptir. $8\times8, 16\times16, 32\times32$ ikili matrisler için bilinen maksimum dal sayısı sırasıyla 5, 8, 12'dir. Vaudenay, 1994 yılındaki çalışmasında çoklu permütasyonu (perfect diffusion) sağladığı için MDS matrislerin kriptografik yapılarda kullanılabileceği fikrini ortaya koymuş, ayrıca çoklu permütasyonun kullanılmadığı yayılım katmanlarının kriptoanalizini çalışmasında göstermiştir [16, 90]. Vaudenay'ın kullandığı bu ifade Daemon tarafından dal sayısı olarak isimlendirmiş, en etkili yayılım katmanının, maksimum dal sayısına ulaşılan MDS matrislerin kullanıldığı yayılım katmanlarıyla sağlanacağı ileri sürülmüştür [90]. MDS matrislerden Bölüm 3.1'de ayrıntılı olarak bahsedilecektir.

Dalların karıştırılması ile oluşturulan yayılım tabakaları Feistel ağları/GFN'de kullanılır ve genellikle basit dairesel kaydırma işlemleridir (örn. SAFER). Farklı tipteki karıştırma işlemlerinin bir araya getirilerek daha iyi yayılım sağlanması hala literatürdeki açık problemlerden biridir [7].

Feistel ağlarındaki yayılma SPN mimarisine göre daha karmaşıktır çünkü Feistel ağlarında yayılma F fonksiyonu seviyesinde ve Feistel ağı seviyesinde olmak üzere iki farklı seviyede gerçekleşir. F fonksiyonu seviyesinde yayılım için bir giriş bitindeki değişim çıkışa dağıtılmalıdır, bu seviyede çoğu durumda yayılım sadece kısmidir. Feistel ağı seviyesinde yayılımda durumun bir dalındaki değişikler, izleyen döngülerdeki diğer dallara uygulanır böylece F fonksiyonu seviyesinde yayılan değişim tüm ağa yayılmış olur [7], bu iki seviyenin varlığı yayılmanın neden Feistel

ağlarında genellikle daha yavaş olduğunu ve daha fazla döngü sayısına ihtiyaç duyduğunu açıklar.

Tablo 3.1. Yayılım tabakasında MDS matris kullanan bazı blok/hafif sıklet blok şifrelerin doğrusal dönüşümleri

Şifre	**Yayılım Tabakasında Kullanılan MDS Matrisler ve Özellikleri**		
	Matris	**Polinom**	**Özellik**
AES	$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}$	$GF(2^8)$ /0×11b	Dairesel
Twofish	$\begin{bmatrix} 01 & EF & 5B & 5B \\ 5B & EF & EF & 01 \\ EF & 5B & 01 & EF \\ EF & 01 & EF & 5B \end{bmatrix}$	$GF(2^8)$ /0×14d	-
Khazad [91]	$\begin{bmatrix} 01 & 03 & 04 & 05 & 06 & 08 & 0B & 07 \\ 03 & 01 & 05 & 04 & 08 & 06 & 07 & 0B \\ 04 & 05 & 01 & 03 & 0B & 07 & 06 & 08 \\ 05 & 04 & 03 & 01 & 07 & 0B & 08 & 06 \\ 06 & 08 & 0B & 07 & 01 & 03 & 04 & 05 \\ 08 & 06 & 07 & 0B & 03 & 01 & 05 & 04 \\ 0B & 07 & 06 & 08 & 04 & 05 & 01 & 03 \\ 07 & 0B & 08 & 06 & 05 & 04 & 03 & 01 \end{bmatrix}$	$GF(2^8)$ /0×11d	Tersi kendisine eşit Hadamard
ANUBIS [92]	$\begin{bmatrix} 01 & 02 & 04 & 06 \\ 02 & 01 & 06 & 04 \\ 04 & 06 & 01 & 02 \\ 06 & 04 & 02 & 01 \end{bmatrix}$	$GF(2^8)$ /0×11d	Tersi kendisine eşit Hadamard
CURUPIRA-1 [83], CURUPIRA-2 [93]	$\begin{bmatrix} 03 & 02 & 02 \\ 04 & 05 & 04 \\ 06 & 06 & 07 \end{bmatrix}$	$GF(2^8)$ /0×14d	-
Clefia [94]	$M_0 = \begin{bmatrix} 01 & 02 & 04 & 06 \\ 02 & 01 & 06 & 04 \\ 04 & 06 & 01 & 02 \\ 06 & 04 & 02 & 01 \end{bmatrix}, M_1 = \begin{bmatrix} 01 & 08 & 02 & 0A \\ 08 & 01 & 0A & 02 \\ 02 & 0A & 01 & 08 \\ 0A & 02 & 08 & 01 \end{bmatrix}$	$GF(2^8)$ /0×11d	Hadamard (2 doğrusal dönüşüm matrisi kullanır)
Piccolo [95]	$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$	$GF(2^4)$ /0×13	Dairesel
LED	$\begin{bmatrix} 4 & 1 & 2 & 2 \\ 8 & 6 & 5 & 6 \\ B & E & A & 9 \\ 2 & 2 & F & B \end{bmatrix}$	$GF(2^4)$ /0×13	-

Tez kapsamında maksimum yayılma gücüne sahip MDS matrisleri kullanan verimli yayılım tabakalarının tasarımı üzerine çalışıldığından Tablo 3.1'de yayılım

tabakasında MDS matrisleri kullanan blok şifreler ve özellikleri kronolojik sırada verilmiştir.

İlerleyen bölümde (tersi kendisine eşit) MDS matrisler için tasarım yöntemleri literatürde yer alan açık problemler ayrıntılı olarak ele alınmıştır.

3.1. MDS Matrislerin Tasarımı

Özellikle hafif sıklet blok şifreler için MDS matrisleri kullanan (tersi kendisine eşit) yayılım tabakalarının tasarımı önemlidir. Tersi kendisine eşit yayılım tabakaları şifreleme ve şifre çözme işlemlerinde aynı yapıyı kullandığından alandan kazanç sağlar. Bu nedenle düşük maliyetli tersi kendisine eşit yayılım tabakalarının tasarımı özellikle donanım hedefli hafif sıklet kriptosistemler için literatürde çalışılan güncel konulardandır çünkü hafif sıklet bir şifreye uygun MDS veya tersi kendisine eşit MDS matrisin seçilmesi, tasarımcının uygulama maliyetindeki kısıtlamaları göz önünde bulundurması gerektiğinden zorlu bir iştir [20]. Ancak tersi kendisine eşit MDS matrislerin üretilmesi için Hadamard matrislerin ve rastgele üretimin [8] dışında çok fazla yöntem bulunmamaktadır. MDS matrislerin üretilmesi için ise özel matris formları kullanılır.

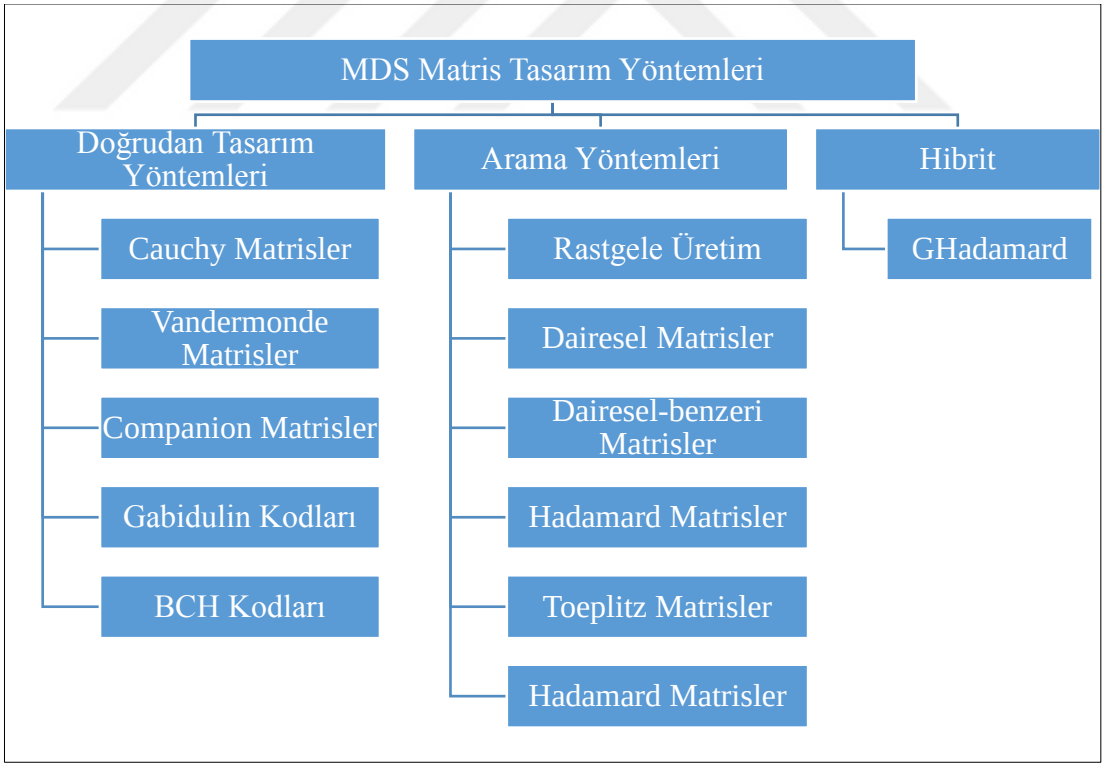
Şekil 3.1'de MDS matrislerin tasarımı (üretimi) için literatürde yer alan çalışmalar incelenerek, tez kapsamında geliştirilen yöntem ile birlikte çalışmalar üç ana başlık altında toplanmıştır. Bunlar; doğrudan tasarım, arama ve hibrit tasarım yöntemleridir.

Doğrudan tasarım yöntemi özel kodlar ve matris formlarıyla doğrudan MDS matrislerin üretilebildiği yapılardır. Bu yöntem altında literatürde yer alan çalışmalar incelendiğinde, yöntemde Cauchy [8], Vandermonde [16, 96, 97], Companion [17, 98] matrisleri ile Gabidulin [89] ve BCH (Bose-Chaudri-Hocquenghem) [15, 99] kodları kullanılır. Ancak Cauchy matrisler ile elde edilen katsayılar çok karmaşıktır, bu nedenle bu karmaşık katsayılar verimli uygulamaları olumsuz etkiler [83].

Arama yönteminde, rastgele üretim ve bazı özel matris formlarını içeren yapılar yer alır. Rastgele üretim için; sonlu cisim üzerinde elemanlar rastgele seçilerek bu elemanlarla oluşan matrisler içinden MDS matrisler aranır [8]. Bu yöntem bir MDS matrisin tüm alt kare matrislerinin tekil olmayan matris olduğunun kontrol edilmesi

ve doğrulanması tabanlıdır. Ayrıca Dairesel [9, 10], Hadamard [11], Toeplitz [12, 20], Dairesel-benzeri [13] özel matris formları ile arama uzayı küçültülerek MDS matrisler aranır.

Hibrit yöntem ise doğrudan tasarım ve arama yöntemlerinin bir arada kullanıldığı tasarım yöntemidir. Hibrit yapıların ihtiyacı özellikle kaynak kısıtlı hafif sıklet uygulamalar için, (tersi kendisine eşit) MDS matrislerin arama maliyetini düşürmekten gelir. Hibrit yapı, temeline özel matris formlarını alır ve bu özel matris formları kullanılarak yeni (tersi kendisine eşit) MDS matrislerin üretilebilmesi için doğrudan tasarım yöntemlerinin kullandığı doğrudan matris üretimi fikri kullanıllır. Literatürde böyle yapılara rastlanmamıştır. Tez kapsamında geliştirilen GHadamard matris formu temeline Hadamard matrisleri alır; çünkü bu özel matris formu tersi kendisine eşit MDS matrisleri üretmenin sınırlı yollarından biridir. Gelişitirilen GHadamard matris formu ile (tersi kendisine eşit) MDS matrisler kulllanılarak yeni (tersi kendisine eşit) MDS matrisler doğrudan arama yapmadan üretilir.



Şekil 3.1. MDS matris tasarım yöntemleri

Khoo ve arkadaşları 2014 yılında yayılım matrislerinin donanım maliyetini XOR sayısı metriği ile ifade etmişlerdir [6]. Bu fikir, sabit bir sonlu cisim elemanıyla

çarpma işlemi için gereken XOR sayısı (Tanım 2.25) hesabına dayanır. Ayrıca bu metrik düşük Hamming ağırlıklı yayılım tabakası matrislerinin, donanım uygulamalarının her zaman düşük maliyetli olmadığını gösterir [12].

Elemanları $GF(2^m)$ üzerinde tanımlı $n \times n$ boyutunda bir matrisinin donanım uygulaması için gerekli XOR sayısı Eşitlik (3.1)'de verilmiştir. i satır j sütun olmak üzere, $\gamma_{i,j}$ matrisin ilgili elemanı için gerekli XOR sayısını, $C(M)$ ise matrisin tüm elemanları için gereken XOR sayıları toplamını ifade eder. $(n-1)\cdot m$, $(n-1)$ tane m bitlik elemanın her satır için toplanmasından gelir.

$$\sum_{i=0}^{n-1}\left(\sum_{j=0}^{n-1}\gamma_{i,j}+(n-1)\cdot m\right)=C(M)+n\cdot(n-1)\cdot m \tag{3.1}$$

Düşük XOR sayısı, bir matris için gereken GE sayısını da azaltır. Bu nedenle hafif sıklet kriptografi için sonlu cisimler üzerinde düşük XOR sayılı (tersi kendisine eşit) MDS matrislerin aranması güncel ve açık problemlerden biridir [12]. Bu nedenle tez kapsamında geliştirilen GHadamard formu ile bu problem ele alınarak, (tersi kendisine eşit) MDS matrisler için literatürde bilinen en iyi XOR sayısından daha düşük XOR sayılı yeni matrisler üretilmiştir.

Tez çalışmasının ilerleyen bölümünde, düşük XOR sayılı (tersi kendisine eşit) MDS matrislerin üretilmesi için, Hadamard matrislerin genelleştirilmesi üzerine geliştirilen ve hibrit bit yapı olan GHadamard matris formu ayrıntılı olarak verilecektir.

4. SONLU CİSİMLERDE HADAMARD MATRİSLERİN GENELLEŞTİRİLMESİ

Tezin bu bölümünde hafif sıklet kriptografi için verimli yayılım tabakalarında kullanılabilecek düşük XOR sayılı (tersi kendisine eşit) MDS matrislerin üretimi için yeni bir yöntem geliştirilmiştir.

Hadamard matrisler çalışmanın merkezine alınmış ve bunun nedeni tersi kendisine eşit yayılım matrisleri üretmenin ana fikir olmasıdır. GHadamard matris formu ile Hadamard matris tanımı genişletilmiş ve genelleştirilmiştir. Tüm hesaplamalar MAGMA [100] yazılımı kullanılarak elde edilmiştir.

4.1. GHadamard Matris Formu

Tanım 4.1. $GF(2^m)$ üzerinde $H = had(a_0, a_1)$ 2×2 Hadamard matris olmak üzere, H matrisine karşılık gelen $GF(2^m)$ üzerinde 2×2 GHadamard matrisi Eşitlik (4.1)'deki gibi ifade edilir ($b_1 \in GF(2^m), b_1 \neq 0$).

$$GH = Ghad(a_0, a_1; b_1) = \begin{bmatrix} a_0 & a_1 b_1 \\ a_1 b_1^{-1} & a_0 \end{bmatrix} \tag{4.1}$$

2×2 GHadamard matris $Ghad(a_0, a_1; b_1)$ Tanım 4.1'de verilmiştir. Bu form Bölüm 2'de verilen MDS özelliğini koruma dönüşümleri kullanılarak elde edilmiştir (MDS özelliği, $n\times n$ boyutlu M matrisinin herhangi bir satır veya sütunu bir c $(c \in GF(2^m), c \neq 0)$ sabitiyle çarpıldığında korunur). Ayrıca elde edilen önemli özelliklerden biri şudur; bir Hadamard matrise, $GF(2^m)$ üzerinde tanımlı sıfıra eşit olmayan elemanlar kullanılarak özel bir çarpım kombinasyonu (hem satır hem sütun çarpımı) uygulandığında Hadamard olma özelliği korunur ($H^2 = c^2 I$, $c = \bigoplus_{i=0}^{n-1} a_i$ ve I birim matris).

Tanım 4.2. $k = 2^t$ ve $t > 0$ ve GH $GF(2^m)$ üzerinde $k \times k$ matris olmak üzere, $GH_{i,j}$ GH matrisinin i. satır, j. sütun elemanını temsil eder. $k \times k$ GHadamard matrisin elemanları $GH_{i,j} = a_{i \oplus j} b_i^{-1} b_j$ olarak ifade edilir. ($0 \le i \le k-1$, $1 \le j \le k-1$ ve a_i, b_i $GF(2^m)$'de sıfıra eşit olmayan elemanlar, b_i^{-1} ise $GF(2^m)$'de b_i'nin çarpma işlemine göre tersidir ve $b_0 = b_0^{-1} = 1$)

Önerme 4.1'de herhangi bir $k \times k$ $(k = 2^t, t > 0)$ boyutunda Hadamard matrise MDS özelliği koruyucu dönüşümler uygulandığında yeni tipte (tersi kendisine eşit) MDS matrisleri elde etmek için bir yöntem verilmiştir.

Önerme 4.1. $k = 2^t$ ve $t > 0$ ve A $k \times k$ GHadamard matris olmak üzere $[Ghad(a_0, a_1; b_1, a_2; b_2, \ldots a_{k-1}; b_{k-1})]$, A matrisi $\bigoplus_{i=0}^{k-1} a_i = 1$ ve $b_i \neq 0$ olduğunda tersi kendisine eşittir.

İspat. 2×2 matrisler için önerilen yöntem kullanılarak, 2×2 Hadamard matrislerden 2×2 GHadamard matrisleri üretmenin detayları verilecektir. Daha büyük boyutlu matrisler için (örn. 4×4, 8×8, 16×16 vb.) ispat benzer şekildedir.

H, 2×2 boyutunda $H = had(a_0, a_1)$ Hadamard matris, $0 \le i_1, i_2 \le 2^m - 2$, α $GF(2^m)$ 'de ilkel eleman olmak üzere, H matrisinin birinci ve ikinci sütunları sırasıyla α^{i_1}, α^{i_2} elemanları ile çarpılır, daha sonra birinci ve ikinci satırları sırasıyla α^{-i_1}, α^{-i_2} elemanları ile çarpılır. Elde edilen 2×2 boyutunda H_m matrisi Eşitlik (4.2)'deki gibidir.

$$H_m = \begin{bmatrix} a_0 & a_1 \alpha^{i_2 - i_1} \\ a_1 \alpha^{i_1 - i_2} & a_0 \end{bmatrix} \tag{4.2}$$

2×2 boyutunda $Ghad(a_0, a_1; b_1)$ matrisi, H_m matrisinde $\alpha^{i_2 - i_1}$, $\alpha^{i_1 - i_2}$ elemanlarının sırasıyla b_1 ve b_1^{-1} ile ifade edilmesiyle elde edilir.

GH matrisinin karesinin $(GH)^2 = c^2 I$ özelliğini koruduğu açıkça görülebilir:

$$(GH)^2 = \begin{bmatrix} a_0 & a_1b_1 \\ a_1b_1^{-1} & a_0 \end{bmatrix} \cdot \begin{bmatrix} a_0 & a_1b_1 \\ a_1b_1^{-1} & a_0 \end{bmatrix} = \begin{bmatrix} a_0{}^2 \oplus a_1{}^2 b_1 b_1^{-1} & a_0a_1b_1 \oplus a_1b_1a_0 \\ a_1b_1{}^{-1}a_0 \oplus a_0a_1b_1^{-1} & a_1b_1{}^{-1}a_1b_1 \oplus a_0{}^2 \end{bmatrix}$$ olur

buradan $GF(2^m)$ 'de bir elemanın tersi ile çarpımı 1'e eşit olduğundan $(GH)^2 = \begin{bmatrix} a_0^2 \oplus a_1^2 & 0 \\ 0 & a_0^2 \oplus a_1^2 \end{bmatrix}$ olur. $c = a_0 \oplus a_1$ ve I 2×2 birim matris olmak üzere buradan $(GH)^2 = \begin{bmatrix} c^2 & 0 \\ 0 & c^2 \end{bmatrix} = c^2 \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = c^2 I$ olur.

Sonuç 4.1. Herhangi bir $k\times k$ $(k = 2^t, t > 0)$ GHadamard matris $Ghad(a_0, a_1; b_1, a_2; b_2, \ldots a_{k-1}; b_{k-1})$, $\bigoplus_{i=0}^{k-1} a_i = 1$ ve I $k\times k$ birim matris olduğu sürece $(GH)^2 = c^2 I$ özelliğini sağlar.

$Ghad(a_0, a_1; b_1, a_2; b_2, a_3; b_3)$ GHadamard matris formu, Eşitlik (4.3)'de verilmiştir.

$$GH = \begin{bmatrix} a_0 & a_1b_1 & a_2b_2 & a_3b_3 \\ a_1b_1^{-1} & a_0 & a_3b_1^{-1}b_2 & a_2b_1^{-1}b_3 \\ a_2b_2^{-1} & a_3b_2^{-1}b_1 & a_0 & a_1b_2^{-1}b_3 \\ a_3b_3^{-1} & a_2b_3^{-1}b_1 & a_1b_3^{-1}b_2 & a_0 \end{bmatrix} \quad (4.3)$$

8×8 $Ghad(a_0, a_1; b_1, a_2; b_2, a_3; b_3, a_4; b_4, a_5; b_5, a_6; b_6, a_7; b_7)$ GHadamard matris formu, Eşitlik (4.4)'de verilmiştir.

$$GH = \begin{bmatrix} a_0 & a_1b_1 & a_2b_2 & a_3b_3 & a_4b_4 & a_5b_5 & a_6b_6 & a_7b_7 \\ a_1b_1^{-1} & a_0 & a_3b_1^{-1}b_2 & a_2b_1^{-1}b_3 & a_5b_1^{-1}b_4 & a_4b_1^{-1}b_5 & a_7b_1^{-1}b_6 & a_6b_1^{-1}b_7 \\ a_2b_2^{-1} & a_3b_2^{-1}b_1 & a_0 & a_1b_2^{-1}b_3 & a_6b_2^{-1}b_4 & a_7b_2^{-1}b_5 & a_4b_2^{-1}b_6 & a_5b_2^{-1}b_7 \\ a_3b_3^{-1} & a_2b_3^{-1}b_1 & a_1b_3^{-1}b_2 & a_0 & a_7b_3^{-1}b_4 & a_6b_3^{-1}b_5 & a_5b_3^{-1}b_6 & a_4b_3^{-1}b_7 \\ a_4b_4^{-1} & a_5b_4^{-1}b_1 & a_6b_4^{-1}b_2 & a_7b_4^{-1}b_3 & a_0 & a_1b_4^{-1}b_5 & a_2b_4^{-1}b_6 & a_3b_4^{-1}b_7 \\ a_5b_5^{-1} & a_4b_5^{-1}b_1 & a_7b_5^{-1}b_2 & a_6b_5^{-1}b_3 & a_1b_5^{-1}b_4 & a_0 & a_3b_5^{-1}b_6 & a_2b_5^{-1}b_7 \\ a_6b_6^{-1} & a_7b_6^{-1}b_1 & a_4b_6^{-1}b_2 & a_5b_6^{-1}b_3 & a_2b_6^{-1}b_4 & a_3b_6^{-1}b_5 & a_0 & a_1b_6^{-1}b_7 \\ a_7b_7^{-1} & a_6b_7^{-1}b_1 & a_5b_7^{-1}b_2 & a_4b_7^{-1}b_3 & a_3b_7^{-1}b_4 & a_2b_7^{-1}b_5 & a_1b_7^{-1}b_6 & a_0 \end{bmatrix} \quad (4.4)$$

Literatürde tersi kendisine eşit matrisleri üretmenin bir diğer yolu da rastgele üretim yöntemidir, Önerme 2.4'de verilen yapının [8], GHadamard matris formu ile de elde

edildiği görülebilir. Şöyle ki; $a_0 \in GF(2^m), a_0 \neq 0$ olmak üzere 2×2 $RM_1 = \begin{bmatrix} a_0 & a_0^{-1} \\ a_0^3 \oplus a_0 & a_0 \end{bmatrix}$ matrisi tersi kendisine eşittir.

Önerme 4.2'de rastgele üretimle elde edilen 2×2 tersi kendisine eşit MDS matrislerin, $Ghad(a_0, a_1; b_1)$ formundan üretilen alt-sınıflardan biri olduğu verilmiştir.

Önerme 4.2. $RM_1 = \begin{bmatrix} a_0 & a_0^{-1} \\ a_0^3 \oplus a_0 & a_0 \end{bmatrix}$ 'den üretilen 2×2 tersi kendisine eşit MDS matrisler $a_0 \in GF(2^m), a_0 \neq 0$ ve $a_0^3 \oplus a_0 \neq 0$ olmak koşuluyla $Ghad(a_0, a_1; b_1)$ matrisi ile üretilen alt-sınıflardan biridir.

İspat. Önerme 4.1'den yola çıkılarak $GH = Ghad(a_0, a_1; b_1) = \begin{bmatrix} a_0 & a_1 b_1 \\ a_1 b_1^{-1} & a_0 \end{bmatrix}$ için $(a_0 \oplus a_1)^2 = 1$ ve $a_0 \oplus a_1 = 1$'dir buradan a_1 çarpımsal tersi olduğu sürece $a_1 = a_0 \oplus 1$ olur. $a_1 b_1 = a_0^{-1}$'e eşit olduğunu varsayarsak buradan $b_1 = a_1^{-1} a_0^{-1}$ ve $a_1 b_1^{-1} = a_1^2 a_0$'dir, buradan da $a_1 b_1^{-1} = a_0 (a_0 \oplus 1)^2 = a_0^3 \oplus a_0$ olur.

Önerme 4.3. H $k\times k$ Hadamard tersi kendisine eşit MDS matris olmak üzere, GHadamard matrisi $Ghad(a_0, a_1; b_1, a_2; b_2, \ldots a_{k-1}; b_{k-1})$ H matrisine karşılık gelen tersi kendisine eşit MDS matristir.

İspat. Önerme 4.1 gereği H matrisi tersi kendisine eşit ise GH matrisi de tersi kendisine eşit matristir. MDS özelliği, sıfır olmayan sabit bir sonlu cisim elemanı ile satır / sütun çarpımı altında değişmez.

Yardımcı Önerme 4.1. $GF(2^m)$ üzerinde tanımlı $k\times k$ boyutunda Hadamard formda tersi kendisine eşit MDS matristen, GHadamard matris formu kullanılarak, $(2^m - 1)^{k-1}$ farklı tersi kendisine eşit MDS matris üretilebilir. Daha detaylı olarak;

Arama yöntemi ile $GF(2^4)$ ve $GF(2^8)$ üzerinde sırasıyla 1512 ve 15,937,992 tane 4×4 Hadamard tersi kendisine eşit MDS matris üretilebilir. GHadamard matris

formu kullanılarak ise GF(2^4) ve GF(2^8) üzerinde sırasıyla $1512\times(2^4-1)\simeq 2^{22.28}$ ve $15,937,992\times(2^8-1)\simeq 2^{47,90}$ tane 4×4 tersi kendisine eşit MDS matris üretilebilir.

Literatürde XOR sayısı metriği tanıtıldıktan sonra, düşük XOR sayılı (tersi kendisine eşit) MDS matrislerin bulunması için çalışmalar başlamıştır. Bu çalışmalar genelde arama tabanlıdır. Sim ve arkadaşlarının 2015 yılındaki [20] ve Sarkar ve Syed'in 2016 yılındaki [11] çalışmaları dikkat çekici sonuçlar ortaya koymuştur. Bu çalışmalar hafif sıklet (tersi kendisine eşit) MDS matrislerin bulunmasına odaklanmışlardır. Sarkar ve Syed çalışmalarında [11] tersi kendisine eşit MDS matrislerin bir sınıfı için teorik bir yapı önermişler, GF(2^4) ve GF(2^8) üzerinde 4×4 tersi kendisine eşit ve MDS matrisler için minimum XOR sayısına sahip iki matris geliştirmişlerdir. Bu çalışma ile yazarlar GF(2^4) üzerinde 4×4 tersi kendisine eşit MDS matrisler için bilinen en iyi (en düşük) XOR sayısına (64 XOR) ulaşmışlardır. [11]'de verilen iki tersi kendisine eşit MDS matris, 4×4 GHadamard matris formu ile üretilmektedir (Örnek 4.1, Örnek 4.2).

Örnek 4.1. GF(2^4) üzerinde α ilkel eleman, 0×13 indirgenemez polinomu altında 4×4 Hadamard tersi kendisine eşit MDS $H_1=\text{had}(0\times1,0\times2,0\times7,0\times5)=\text{had}(1,\alpha,\alpha^{10},\alpha^8)$ matrisi olsun. H_1 matrisi üzerinde $b_1=1,\quad b_2=\alpha^7$ ve $b_3=\alpha^7$ parametreleri ile üretilen GHadamard $GH_1=\text{Ghad}(1,\alpha;1,\alpha^{10};\alpha^7,\alpha^8;\alpha^7)$ tersi kendisine eşit MDS matrisi Eşitlik (4.5)'de verilmiştir.

$$GH_1=\begin{bmatrix}1 & \alpha & \alpha^2 & 1\\ \alpha & 1 & 1 & \alpha^2\\ \alpha^3 & \alpha & 1 & \alpha\\ \alpha & \alpha^3 & \alpha & 1\end{bmatrix} \tag{4.5}$$

Eşitlik (4.5)'de verilen GH_1 matrisinin XOR sayısı, GF(2^4)'de minimum XOR sayısı olan 64'e eşittir ($=16+4\times3\times4$). Ayrıca $A_0=\begin{bmatrix}1 & \alpha\\ \alpha & 1\end{bmatrix}, A_1=\begin{bmatrix}\alpha^2 & 1\\ 1 & \alpha^2\end{bmatrix}$ olmak üzere $GH_1=\begin{bmatrix}A_0 & A_1\\ \alpha A_1 & A_0\end{bmatrix}$'de aynı matrisi ifade eder.

Örnek 4.2. GF(2^8) üzerinde α ilkel eleman, 0×165 indirgenemez polinomu altında 4×4 Hadamard tersi kendisine eşit MDS $H_2 = \text{had}(0\times01, 0\times02, 0\times b2, 0\times b0) = \text{had}(1, \alpha, \alpha^{254}, \alpha^{210})$ matrisi olsun. H_2 matrisi üzerinde $b_1 = 1$, $b_2 = \alpha$ ve $b_3 = \alpha$ parametreleri ile üretilen GHadamard $GH_2 = \text{Ghad}(1, \alpha; 1, \alpha^{254}; \alpha, \alpha^{210}; \alpha)$ tersi kendisine eşit MDS matrisi Eşitlik (4.6)'da verilmiştir.

$$GH_2 = \begin{bmatrix} 1 & \alpha & 1 & \alpha^{211} \\ \alpha & 1 & \alpha^{211} & 1 \\ \alpha^{253} & \alpha^{209} & 1 & \alpha \\ \alpha^{209} & \alpha^{253} & \alpha & 1 \end{bmatrix} \tag{4.6}$$

Eşitlik (4.6)'da GF(2^8) üzerinde verilen GH_2 matrisinin XOR sayısı 160'tır $(= 64 + 4\times3\times8)$. Ayrıca $A_0 = \begin{bmatrix} 1 & \alpha \\ \alpha & 1 \end{bmatrix}, A_1 = \begin{bmatrix} 1 & \alpha^{211} \\ \alpha^{211} & 1 \end{bmatrix}$ olmak üzere $GH_2 = \begin{bmatrix} A_0 & A_1 \\ \alpha^{-2}A_1 & A_0 \end{bmatrix}$'de aynı matrisi ifade eder.

Minimum sayıda farklı eleman kullanarak MDS matrislerin elde edilmesi [101]'deki çalışmada ele alınmıştır. Ancak tersi kendisine eşit MDS matrisler için bilinen sınır yoktur. Tez kapsamında ilk kez, tersi kendisine eşit MDS matrisler için minimum farklı eleman sayısı verilmiştir (Önerme 4.4).

Önerme 4.4. GHadamard matris formu [$\text{Ghad}(a_0, a_1; b_1, a_2; b_2, \ldots a_{k-1}; b_{k-1})$] ile $k\times k$ tersi kendisine eşit MDS matris üretmek için gereken minimum sayıda farklı eleman sayısı k'dır.

İspat. $k\times k$ boyutunda M matrisi ancak ve ancak her kare alt matrisleri tekil olmayan matris ise MDS'dir koşulundan yola çıkarak, Tanım 4.1 ve Önerme 4.1 kullanılarak, $a_0 \oplus a_1 \neq 0$ ve $a_0 \neq a_1$ sonuçları çıkar bu gösterir ki; 2×2 tersi kendisine eşit MDS matrisler için gereken minimum farklı eleman sayısı 2'dir. Buradan bir GHadamard matrisin alt matrislerinin her biri en az iki farklı elemandan oluşur şöyle ki; $a_0 \neq a_1 \neq a_2 \neq \ldots \neq a_{k-1}$, $a_i \neq a_j (0 \leq i, j \leq k-1)$.

Yardımcı Önerme 4.2'de Hadamard matrislerin avantajları tartışılmış, GHadamard yapısı tarafından bu avantajlardan hangilerinin korunduğu verilmiştir.

Yardımcı Önerme 4.2. MDS matrisleri oluştururken ve uygularken Hadamard matrisler üç ana avantaj sağlar;

- $k \times k$ Hadamard matris k tane farklı katsayıya sahiptir $(k = 2^t, t > 0)$. Bu her satırın, ilk satırın permütasyonu olduğu anlamına gelir.
- İnvolutif MDS matrisler kolayca üretilebilir (Önerme 2.2). Bu nedenle alan gereksinimi açısından avantaj sağlar.
- Rastgele üretilen bir kare matrise oranla daha yüksek olasılıkla MDS matris bulunabilir (Toeplitz ve Companion matris gibi birçok matrise oranla).

GHadamard matrisinin ana avantajı, $k \times k$ (tersi kendisine eşit) MDS Hadamard matrisler kullanılarak, yeni $k \times k$ (tersi kendisine eşit) MDS matrislerin üretilmesidir. Üstelik bu fikir $k \neq 2^t$ olduğu $k \times k$ (tersi kendisine eşit) MDS matrislere de uygulanabilir (Tanım 4.3 ve Önerme 4.5). Bu bağlamda GHadamard matrisler, gereken XOR sayısının sayılması kaynaklı zaman karmaşıklığı metriği üzerinde, avantajlı olabilecek yeni tipteki (tersi kendisine eşit) MDS matrislerin üretilmesi için kullanılabilir. Diğer yandan GHadamard matrislerin temelinde Hadamard matrislerin yatmasının sebebi, Hadamard matrisler sadece k farklı eleman ile uygulanabildiğinden bu matrisler alan karmaşıklığı açısından avantaj sağlar.

4.2. GHadamard Matris Formunun Farklı Boyutlardaki Diğer Matris Tiplerine Uygulanması

Bu bölümde GHadamard matris formunun farklı boyutlardaki diğer matris tiplerine uygulanması ayrıntılı olarak verilmiştir.

Yeni tipteki MDS matrislerin üretilebileceği fikrinin verildiği Önerme 4.1, $k \neq 2^t$ olduğu durumda $k \times k$ boyutlu MDS matrisler için genelleştirilebilir. Diğer bir deyişle önerilen formla, Hadamard veya GHadamard matris formlarının dışındaki matrislerden biri de üretilebilir. Unutulmamalıdır ki; Yardımcı Önerme 4.1 aynı zamanda $k \times k$ $(k \neq 2^t)$ boyutlu matrislerin üretilmesi için de geçerlidir çünkü bu

matris formuyla değil sadece boyut ve uzantı derecesiyle (extension degree) ilgilidir. Tanım 4.2. $k \neq 2^t$ koşulu için Tanım 4.3'de genelleştirilmiştir.

Tanım 4.3. $k \neq 2^t$ ve $t > 0$ ve GM $GF(2^m)$ üzerinde $k \times k$ matris olmak üzere, $GM_{i,j}$ GM matrisinin i. satır, j. sütun elemanını temsil eder. $k \times k$ GHadamard matrisin elemanları $GM_{i,j} = a_{ki+j} b_i^{-1} b_j$ olarak ifade edilir. $(0 \leq i \leq k-1, 1 \leq j \leq k-1,$ ve a_i, b_i $GF(2^m)$'de sıfıra eşit olmayan elemanlar, b_i^{-1} ise $GF(2^m)$'de b_i elemanının çarpma işlemine göre tersidir ve $b_0 = b_0^{-1} = 1)$

Örnek 4.3'de verilen 5×5 (tersi kendisine eşit) MDS matrislerden, 5×5 (tersi kendisine eşit) MDS matrisleri üreten matris formu verilmiştir.

Örnek 4.3. M, $GF(2^m)$'de 5×5 matris olmak üzere Eşitlik (4.7)'de M matrisi verilmiştir.

$$M = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 & a_4 \\ a_5 & a_6 & a_7 & a_8 & a_9 \\ a_{10} & a_{11} & a_{12} & a_{13} & a_{14} \\ a_{15} & a_{16} & a_{17} & a_{18} & a_{19} \\ a_{20} & a_{21} & a_{22} & a_{23} & a_{24} \end{bmatrix} \tag{4.7}$$

Eğer $GF(2^m)$'de 5×5 boyutlu M matrisi tersi kendisine eşit MDS ise, Eşitlik (4.8)'de verilen GM matrisi kullanılarak $(2^m - 1)^4$ (Yardımcı Önerme 4.1 ve Önerme 4.5) tane tersi kendisine eşit MDS üretilir.

$$GM = \begin{bmatrix} a_0 & a_1 b_1 & a_2 b_2 & a_3 b_3 & a_4 b_4 \\ a_5 b_1^{-1} & a_6 & a_7 b_1^{-1} b_2 & a_8 b_1^{-1} b_3 & a_9 b_1^{-1} b_4 \\ a_{10} b_2^{-1} & a_{11} b_2^{-1} b_1 & a_{12} & a_{13} b_2^{-1} b_3 & a_{14} b_2^{-1} b_4 \\ a_{15} b_3^{-1} & a_{16} b_3^{-1} b_1 & a_{17} b_3^{-1} b_2 & a_{18} & a_{19} b_3^{-1} b_4 \\ a_{20} b_4^{-1} & a_{21} b_4^{-1} b_1 & a_{22} b_4^{-1} b_2 & a_{23} b_4^{-1} b_3 & a_{24} \end{bmatrix} \tag{4.8}$$

Önerme 4.5 GHadamard matris formunun tüm $k \times k$ matrislere uygulanabileceğini gösterir, k boyutu 2'nin kuvveti olmak zorunda değildir.

Önerme 4.5. M, $GF(2^m)$'de $k \times k$ $(k \neq 2^t)$ (tersi kendisine eşit) MDS matris olmak üzere, Tanım 4.3'den elde edilen (M matrisi kullanılarak) GM matrisi de (tersi kendisine eşit) MDS matristir.

İspat. $k \times k$ (tersi kendisine eşit) MDS M matrisi Eşitlik (4.9)'daki gibi verilsin. Tanım 4.3'den, GM matrisi önerilen yöntemden elde edilerek M matrisinden türetilir, GM matrisi Eşitlik (4.10)'da verilmiştir.

$$M = \begin{bmatrix} a_0 & a_1 & \cdots & a_{k-2} & a_{k-1} \\ a_k & a_{k+1} & \cdots & a_{2k-2} & a_{2k-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{(k-2)k} & a_{(k-2)k+1} & \cdots & a_{(k-2)k+k-2} & a_{(k-2)k+k-1} \\ a_{(k-1)k} & a_{(k-1)k+1} & \cdots & a_{(k-1)k+k-2} & a_{(k-1)k+k-1} \end{bmatrix} \tag{4.9}$$

$$GM = \begin{bmatrix} a_0 & a_1 b_1 & \cdots & a_{k-2} b_{k-2} & a_{k-1} b_{k-1} \\ a_k b_1^{-1} & a_{k+1} & \cdots & a_{2k-2} b_1^{-1} b_{k-2} & a_{2k-1} b_1^{-1} b_{k-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{(k-2)k} b_{k-2}^{-1} & a_{(k-2)k+1} b_{k-2}^{-1} b_1 & \cdots & a_{(k-2)k+k-2} & a_{(k-2)k+k-1} b_{k-2}^{-1} b_{k-1} \\ a_{(k-1)k} b_{k-1}^{-1} & a_{(k-1)k+1} b_{k-1}^{-1} b_1 & \cdots & a_{(k-1)k+k-2} b_{k-1}^{-1} b_{k-2} & a_{(k-1)k+k-1} \end{bmatrix} \tag{4.10}$$

Buradan M^2 ve $(GM)^2$ (kısaca GM^2 şeklinde ifade edilmiştir) matrisleri sırasıyla Eşitlik (4.11) ve (4.12)'de verilmiştir.

$$M^2 = \begin{bmatrix} c_0 & c_1 & \cdots & c_{k-2} & c_{k-1} \\ c_k & c_{k+1} & \cdots & c_{2k-2} & c_{2k-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ c_{(k-2)k} & c_{(k-2)k+1} & \cdots & c_{(k-2)k+k-2} & c_{(k-2)k+k-1} \\ c_{(k-1)k} & c_{(k-1)k+1} & \cdots & c_{(k-1)k+k-2} & c_{(k-1)k+k-1} \end{bmatrix} \tag{4.11}$$

$$GM^2 = \begin{bmatrix} c_0 & c_1 b_1 & \cdots & c_{k-2} b_{k-2} & c_{k-1} b_{k-1} \\ c_k b_1^{-1} & c_{k+1} & \cdots & c_{2k-2} b_1^{-1} b_{k-2} & c_{2k-1} b_1^{-1} b_{k-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ c_{(k-2)k} b_{k-2}^{-1} & c_{(k-2)k+1} b_{k-2}^{-1} b_1 & \cdots & c_{(k-2)k+k-2} & c_{(k-2)k+k-1} b_{k-2}^{-1} b_{k-1} \\ c_{(k-1)k} b_{k-1}^{-1} & c_{(k-1)k+1} b_{k-1}^{-1} b_1 & \cdots & c_{(k-1)k+k-2} b_{k-1}^{-1} b_{k-2} & c_{(k-1)k+k-1} \end{bmatrix} \tag{4.12}$$

Eğer M tersi kendisine eşit matris ise; I $k \times k$ birim matris olmak üzere, $M^2 = I$ $c_0 = c_{k+1} = \ldots = c_{(k-1)k+k-1} = 1$ ve diğer elemanlar 0'dır. Buradan GM matrisi de tersi kendisine eşit matristir $GM^2 = I$. Eğer M matrisi MDS ise GM matrisi de MDS matristir (satır/sütunun sıfıra eşit olmayan bir sonlu cisim elemanıyla çarpılması sonucu elde edilen matris de MDS'dir özelliği kullanılarak).

Yardımcı Önerme 4.3. $k = 2^t$ olmak üzere, çalışmada $k \times k$ (tersi kendisine eşit) MDS matrisler için herhangi bir üretim yöntemi verilmemiştir. Bu nedenle, bu boyutlarda (Önerme 4.5) var olan (tersi kendisine eşit) MDS matrisler ile yeni (tersi kendisine eşit) MDS matrisler üretilir.

Örnek 4.4'de $GF(2^4)/0\times13$ sonlu cismi üzerinde 4×4 tersi kendisine eşit MDS matris M_1 verilmiştir, bu matris Hadamard veya GHadamard matris formu ile değil arama yöntemi kullanılarak elde edilmiştir. İnvolutif MDS M_1 matrisi kullanılaraK, $(2^4 - 1)^3$ (Yardımcı Önerme 4.1) tersi kendisine eşit MDS matris üretilebilir.

Örnek 4.4. $GF(2^4)/0\times13$ sonlu cismi üzerinde α ilkel eleman olmak üzere, 4×4 tersi kendisine eşit MDS M_1 matrisi Eşitlik (4.13)'de verilmiştir.

$$M_1 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \alpha & \alpha^2 & \alpha^5 \\ \alpha^7 & \alpha^{10} & \alpha & \alpha^5 \\ \alpha^9 & \alpha^2 & \alpha^{10} & 1 \end{bmatrix} \tag{4.13}$$

Yardımcı Önerme 4.4. $k \times k$ boyutunda GHadamard matris formu ile $k \times k$ boyutunda Toeplitz formda matrisler üretilebilir.

İspat. Dairesel matris formu (Eşitlik (2.6)), Toeplitz matris formunun (Eşitlik (2.9)) özel bir durumudur [102]. 4×4 matrisler için önerilen yöntem kullanılarak, 4×4 dairesel formda matrislerden yararlanılarak, Toeplitz formda matrisler üreten 4×4 GHadamard matrislerin detayları verilecektir. Daha büyük boyutlu matrisler için (örn. 8×8, 16×16 vb.) ispat benzer şekildedir.

Dairesel formda 4×4 boyutunda A matrisi Eşitlik (4.14)'deki gibi verilsin. GA matrisi GHadamard matris formundan elde edilerek A matrisinden türetilir, GA matrisi Eşitlik (4.15)'de verilmiştir.

$$A=\begin{bmatrix} a_0 & a_1 & a_2 & a_3 \\ a_3 & a_0 & a_1 & a_2 \\ a_2 & a_3 & a_0 & a_1 \\ a_1 & a_2 & a_3 & a_0 \end{bmatrix} \tag{4.14}$$

$$GA=\begin{bmatrix} a_0 & a_1b_1 & a_2b_2 & a_3b_3 \\ a_3b_1^{-1} & a_0 & a_1b_1^{-1}b_2 & a_2b_1^{-1}b_3 \\ a_2b_2^{-1} & a_3b_2^{-1}b_1 & a_0 & a_1b_2^{-1}b_3 \\ a_1b_3^{-1} & a_2b_3^{-1}b_1 & a_3b_3^{-1}b_2 & a_0 \end{bmatrix} \tag{4.15}$$

GA matrisi Toeplitz matris formuna benzetilmek üzere, Eşitlik (2.9)'da verilen yapı ele alındığında 1. satır ve 2. satır elemanları için; $a_1b_1=a_1b_1^{-1}b_2$ ve $a_2b_2=a_2b_1^{-1}b_3$ olmalıdır, buradan $b_1=b_1^{-1}b_2$, $b_2=b_1^{-1}b_3$ olarak ifade edilir. Matris tümüyle ele alındığında $b_3=b_1^3$, $b_2=b_1^2$ olur. GA matrisi elemanları üzerinde bu eşitlikler yerine konulduğunda elde edilen GA_1 matrisi Eşitlik (4.16)'da verilmiştir.

$$GA_1=\begin{bmatrix} a_0 & a_1b_1 & a_2b_1^2 & a_3b_1^3 \\ a_3b_1^{-1} & a_0 & a_1b_1 & a_2b_1^2 \\ a_2b_1^{-2} & a_3b_1^{-1} & a_0 & a_1b_1 \\ a_1b_1^{-3} & a_2b_1^{-2} & a_3b_1^{-1} & a_0 \end{bmatrix} \tag{4.16}$$

GA_1 matrisi bu kısıtlar altında Toeplitz formda matrisler üretir ve $Toep(a_0, a_1b_1, a_2b_1^2, a_3b_1^3, a_3b_1^{-1}, a_2b_1^{-2}, a_1b_1^{-3})$ gösterimi ile ifade edilir.

4.3. GHadamard Matris Formu İle Elde Edilen Sonuçlar

Bu bölümde öncelikle GHadamard matris formu ile $GF(2^4)$ ve $GF(2^8)$ üzerinde minimum XOR sayılı 4×4 tersi kendisine eşit MDS matrislerin üretilmesine odaklanılmıştır. $GF(2^4)$ üzerinde 4×4 tersi kendisine eşit MDS matrisler için, [20]'de de verilen, bilinen en düşük XOR sayısı 64'e $(=16+4\times3\times4)$ eşit matrisler

üretilmiştir. GF(2^8) üzerinde 4×4 boyutlu 158 $(=62+4\times3\times8)$ XOR sayılı tersi kendisine eşit MDS matris üretilmiştir.

Ayrıca, önerilen GHadamard matris formu, GF(2^4) üzerinde minimum XOR sayılı 8×8 tersi kendisine eşit ve tersi kendisine eşit olmayan MDS matrisler üretmek için kullanılmıştır. GF(2^4) üzerinde 8×8 tersi kendisine eşit MDS matrisler için [94]'de verilen ve bilinen en düşük XOR sayısı 512'den $(=288+4\times7\times8)$ 407'ye $(=183+4\times7\times8)$ iyileştirilmiştir (düşürülmüştür). GF(2^4) 'de 8×8 tersi kendisine eşit olmayan MDS matrisler için yine [11]'de verilen ve bilinen en düşük XOR sayısı 432'den $(=208+4\times7\times8)$, 380'e $(=156+4\times7\times8)$ düşürülmüştür.

Minimum XOR sayılı GF(2^4) üzerinde 4×4 tersi kendisine eşit MDS matrisler elde etmek için, tüm olası tersi kendisine eşit MDS matrisler ($[1512\times(2^4-1)^3 \simeq 2^{22,28}]$ Yardımcı Önerme 4.1 kullanılarak) üretilmiş ve minimum XOR sayısı kontrol edilmiştir. GF(2^4) üzerinde 64 XOR sayılı, 48 farklı 4×4 tersi kendisine eşit MDS matris tespit edilmiştir. Bu matrislerden biri Örnek 4.5'de verilmiştir.

Örnek 4.5. GF(2^4) üzerinde α ilkel eleman, 0×13 indirgenemez polinomu altında 4×4 Hadamard tersi kendisine eşit MDS $H_3 = \text{had}(0\times1, 0\times2, 0\times5, 0\times7) = \text{had}(1, \alpha, \alpha^8, \alpha^{10})$ matrisi olsun. H_3 matrisi üzerinde $b_1 = \alpha$, $b_2 = \alpha^8$ ve $b_3 = \alpha^8$ parametreleri ile üretilen GHadamard $GH_3 = \text{Ghad}(1, \alpha; \alpha, \alpha^8; \alpha^8, \alpha^{10}; \alpha^8)$ tersi kendisine eşit MDS matrisi Eşitlik (4.17)'de verilmiştir.

$$GH_3 = \begin{bmatrix} 1 & \alpha^2 & \alpha & \alpha^3 \\ 1 & 1 & \alpha^2 & 1 \\ 1 & \alpha^3 & 1 & \alpha \\ \alpha^2 & \alpha & \alpha & 1 \end{bmatrix} \tag{4.17}$$

GHadamard matris formuyla üretilen minimum XOR sayılı (64 XOR) tüm tersi kendisine eşit MDS matrisler Tablo 4.1'de verilmiştir. Bu matrisler H Hadamard matrisleri üzerinden GHadamard matris formu ile üretilmiştir.

$\{c_1, c_2, c_3\} \in \{\alpha, \alpha^8, \alpha^{10}\}$ ve $\{c_4, c_5, c_6\} \in \{1, \alpha^8, \alpha^{10}\}$ olmak üzere, H matrisleri $had(1, c_1, c_2, c_3)$ ve $had(\alpha, c_4, c_5, c_6)$ formundadır.

Tablo 4.1. GF(2^4)/0×13 sonlu cismi üzerinde 64 XOR sayılı GHadamard matris formunda [Ghad($a_0, a_1; b_1, a_2; b_2, a_3; b_3$)] 4×4 tersi kendisine eşit MDS matrisler

a_0	a_1	b_1	a_2	b_2	a_3	b_3	a_0	a_1	b_1	a_2	b_2	a_3	b_3	a_0	a_1	b_1	a_2	b_2	a_3	b_3
1	a	1	a^8	a^7	a^{10}	a^7	1	a	1	a^{10}	a^7	a^8	a^7	a	1	1	a^8	a^7	a^{10}	a^7
1	a	1	a^8	a^7	a^{10}	a^8	1	a	1	a^{10}	a^7	a^8	a^8	a	1	1	a^8	a^7	a^{10}	a^8
1	a	1	a^8	a^8	a^{10}	a^7	1	a	1	a^{10}	a^8	a^8	a^7	a	1	1	a^{10}	a^7	a^8	a^7
1	a	1	a^8	a^8	a^{10}	a^8	1	a	1	a^{10}	a^8	a^8	a^8	a	1	1	a^{10}	a^7	a^8	a^8
1	a	a	a^8	a^8	a^{10}	a^8	1	a	a	a^{10}	a^8	a^8	a^8	a	a^8	a^7	1	1	a^{10}	a^7
1	a	a^{-1}	a^8	a^7	a^{10}	a^7	1	a	a^{-1}	a^{10}	a^7	a^8	a^7	a	a^8	a^8	1	1	a^{10}	a^8
1	a^8	a^7	a	1	a^{10}	a^7	1	a^8	a^7	a^{10}	a^7	a	1	a	a^8	a^7	a^{10}	a^7	1	1
1	a^8	a^7	a	1	a^{10}	a^8	1	a^8	a^7	a^{10}	a^7	a	a^{-1}	a	a^8	a^8	a^{10}	a^8	1	1
1	a^8	a^7	a	a^{-1}	a^{10}	a^7	1	a^8	a^7	a^{10}	a^8	a	1	a	a^{10}	a^7	1	1	a^8	a^7
1	a^8	a^8	a	1	a^{10}	a^7	1	a^8	a^8	a^{10}	a^7	a	1	a	a^{10}	a^8	1	1	a^8	a^8
1	a^8	a^8	a	1	a^{10}	a^8	1	a^8	a^8	a^{10}	a^8	a	1	a	a^{10}	a^7	a^8	a^7	1	1
1	a^8	a^8	a	a	a^{10}	a^8	1	a^8	a^8	a^{10}	a^8	a	a	a	a^{10}	a^8	a^8	a^8	1	1
1	a^{10}	a^7	a	1	a^8	a^7	1	a^{10}	a^7	a^8	a^7	a	1	-	-	-	-	-	-	-
1	a^{10}	a^7	a	1	a^8	a^8	1	a^{10}	a^7	a^8	a^7	a	a^{-1}	-	-	-	-	-	-	-
1	a^{10}	a^7	a	a^{-1}	a^8	a^7	1	a^{10}	a^7	a^8	a^8	a	1	-	-	-	-	-	-	-
1	a^{10}	a^8	a	1	a^8	a^7	1	a^{10}	a^8	a^8	a^7	a	1	-	-	-	-	-	-	-
1	a^{10}	a^8	a	1	a^8	a^8	1	a^{10}	a^8	a^8	a^8	a	1	-	-	-	-	-	-	-
1	a^{10}	a^8	a	a	a^8	a^8	1	a^{10}	a^8	a^8	a^8	a	1	-	-	-	-	-	-	-

Önerme 4.6. GF(2^4)/p(x) sonlu cismi üzerinde 4×4 invloutif MDS matrisler için gereken en az XOR sayısı 64'tür $(=16+4\times3\times4)$.

İspat. Önerme 4.4'den her eleman birbirinden farklıdır, en düşük XOR sayıları $1, \alpha, \alpha^3+1, \alpha^2$ elemanları için XOR sayıları sırasıyla 0, 1, 1 ve 2'dir.

GF(2^8) üzerinde düşük XOR sayılı tersi kendisine eşit MDS matrisleri kontrol etmek ve belirlemek zordur çünkü çok sayıda $[15,937,992\cdot(2^8-1)^3 \simeq 2^{47,90}]$ (Yardımcı Önerme 4.1) tersi kendisine eşit MDS matris vardır. Bu nedenle GF(2^4) üzerinde 4×4 tersi kendisine eşit MDS matrislerin üretilmesi işlemlerinde edinilen tecrübeye dayanarak, GF(2^8)/1c3 üzerinde tanımlı bazı özel Hadamard tersi kendisine eşit MDS matrislerine karşılık gelen GHadamard matrisleri oluşturulmuştur. Bu özel Hadamard tersi kendisine eşit MDS matrisler şu şekildedir: $had(1, \alpha, d_1, d_2)$, $had(1, \alpha^2, d_3, d_4)$, $had(1, \alpha^3, d_5, d_6)$, $had(1, \alpha^{252}, d_7, d_8)$, $had(1, \alpha^{253}, d_9, d_{10})$, $had(1, \alpha^{254}, d_{11}, d_{12})$, $1 \le i \le 12$ olmak üzere tüm d_i elemanları GF(2^8)/1c3 üzerinde tanımlıdır. Sonuç olarak, GF(2^8)/1c3 sonlu cismi üzerinde

$\sim 2^{34,53}$ tane 4×4 tersi kendisine eşit MDS matris kontrol edilmiştir. Önerilen yöntem Örnek 4.6'dan görülebilir. Burada GF(2^8)/1c3 üzerinde 158 XOR sayılı 4×4 tersi kendisine eşit MDS matris verilmiştir, bu matris GF(2^8) üzerinde 4×4 tersi kendisine eşit MDS matrisler için bilinen en iyi (düşük) XOR sayılı matristir.

Örnek 4.6. GF(2^8) üzerinde α ilkel eleman, $0\times1c3$ indirgenemez polinomu altında 4×4 Hadamard tersi kendisine eşit MDS $H_4 = \text{had}(0\times01, 0\times02, 0\times9d, 0\times9f) = \text{had}(1, \alpha, \alpha^{26}, \alpha^{127})$ matrisi olsun. H_4 matrisi üzerinde $b_1 = \alpha$, $b_2 = \alpha^{128}$ ve $b_3 = \alpha^{127}$ parametreleri ile üretilen GHadamard $GH_4 = \text{Ghad}(1, \alpha; \alpha, \alpha^{26}; \alpha^{128}, \alpha^{127}; \alpha^{127})$ tersi kendisine eşit MDS matrisi Eşitlik (4.18)'de verilmiştir.

$$GH_4 = \begin{bmatrix} 1 & \alpha^2 & \alpha^{154} & \alpha^{254} \\ 1 & 1 & \alpha^{254} & \alpha^{152} \\ \alpha^{153} & 1 & 1 & 1 \\ 1 & \alpha^{155} & \alpha^2 & 1 \end{bmatrix} \tag{4.18}$$

GH_4 matrisi XOR sayısı 158 $(=62+4\times3\times8)$ tersi kendisine eşit MDS matristir. Her bir eleman için gereken XOR sayıları ise $1, \alpha^2, \alpha^{152}, \alpha^{153}, \alpha^{154}, \alpha^{155}, \alpha^{254}$ elemanları için sırasıyla 0, 5, 13, 12, 11, 10, 3 XOR'dur. Farklı sonlu cisimler üzerinde 158 XOR sayısından daha düşük XOR sayılı matrisler elde edilebilir.

Önerme 4.7. $GH = \text{Ghad}(a_0, a_1; b_1, a_2; b_2, \ldots a_{k-1}; b_{k-1})$ GHadamard matrisi $k\times k$ Hadamard $H = \text{had}(a_0, a_1, a_2, \ldots a_{k-1})$ matrisinden elde edilsin. H matrislerinden (H matrisleri sabit a_0 elemanı ve $(a_1, a_2, \ldots a_{k-1})$ elemanlarının permütasyonları ile oluşturulmak üzere) elde edilen GH matrisleri ile üretilen matrislerin XOR sayısı dağılımları aynıdır.

Önerme 4.7, daha büyük boyutlu matrisler üzerinde düşük XOR sayılı matrisleri bulmak için yarar sağlar. Örneğin, GF(2^4) sonlu cismi üzerinde tanımlı sekiz elemanın permütasyonu ile üretilen 1344 tane Hadamard formda 8×8 tersi kendisine eşit MDS matris vardır. Diğer bir deyişle; 8! permütasyonun 1344 tanesi GF(2^4)

üzerinde 8×8 tersi kendisine eşit MDS matris üretmede başarılı olmuştur. GF(2^4) üzerinde α ilkel eleman, 0×13 indirgenemez polinomu altında sekiz eleman $(\alpha,\alpha^2,\alpha^3,\alpha^4,\alpha^6,\alpha^8,\alpha^9,\alpha^{12})$ elemanlarıdır. Önerme 4.7'den, sekiz Hadamard tersi kendisine eşit MDS matris (bu matrislerin ilk elemanları a_0 birbirinden farklı) seçildiğinde, GF(2^4)$/0\times13$ sonlu cisminde GHadamard matris formunda toplamda $8\times(2^4-1)^7 \simeq 2^{30,35}$ tane 8×8 tersi kendisine eşit MDS matris üretilir. Bu matrisler içinden bazıları GF(2^4) sonlu cismi üzerinde 8×8 tersi kendisine eşit MDS matrisler için bilinen en iyi XOR sayısı 512'den $(=288+8\times7\times4)$ [11] daha düşük XOR sayılı matrislerdir. GF(2^4)$/0\times13$ sonlu cismi üzerinde elde edilen bu 8×8 tersi kendisine eşit MDSlerin XOR sayısı 407'dir $(=183+8\times7\times4)$. Bu matrisler, GF(2^4) üzerinde 8×8 tersi kendisine eşit MDS matrisler için bilinen en düşük XOR sayılı matrislerdir. Önerilen yöntem Örnek 4.7'de verilmiştir.

Örnek 4.7. GF(2^4) üzerinde α ilkel eleman, 0×13 indirgenemez polinomu altında 8×8 Hadamard tersi kendisine eşit MDS $H_5=\text{had}(0\times2,0\times\text{f},0\times\text{c},0\times5,0\times\text{a},0\times4,0\times8,0\times3)=\text{had}(\alpha,\alpha^{12},\alpha^6,\alpha^8,\alpha^9,\alpha^2,\alpha^3,\alpha^4)$ matrisi olsun. H_5 matrisi üzerinde $b_1=\alpha^2$, $b_2=\alpha^9$, $b_3=\alpha^7$, $b_4=\alpha^6$, $b_5=\alpha^{11}$, $b_6=\alpha^3$ ve $b_7=\alpha^{13}$ parametreleri ile üretilen GHadamard $GH_5=\text{Ghad}(\alpha,\alpha^{12};\alpha^2,\alpha^6;\alpha^9,\alpha^8;\alpha^7,\alpha^9;\alpha^6,\alpha^2;\alpha^{11},\alpha^3;\alpha^3,\alpha^4;\alpha^{13})$ tersi kendisine eşit MDS matrisi Eşitlik (4.19)'de verilmiştir.

$$GH_5=\begin{bmatrix}
\alpha & \alpha^{14} & 1 & 1 & 1 & \alpha^{13} & \alpha^6 & \alpha^2 \\
\alpha^{10} & \alpha & 1 & \alpha^{11} & \alpha^6 & \alpha^3 & \alpha^5 & \alpha^{14} \\
\alpha^{12} & \alpha & \alpha & \alpha^{10} & 1 & \alpha^6 & \alpha^3 & \alpha^6 \\
\alpha & \alpha & \alpha^{14} & \alpha & \alpha^3 & \alpha^7 & \alpha^{13} & 1 \\
\alpha^3 & \alpha^{13} & \alpha^6 & \alpha^5 & \alpha & \alpha^2 & \alpha^3 & 1 \\
\alpha^6 & 1 & \alpha^2 & \alpha^{14} & \alpha^7 & \alpha & 1 & \alpha^8 \\
1 & \alpha^3 & 1 & \alpha^6 & \alpha^9 & \alpha & \alpha & \alpha^7 \\
\alpha^6 & \alpha^7 & \alpha^{13} & \alpha^3 & \alpha & \alpha^4 & \alpha^2 & \alpha
\end{bmatrix} \tag{4.19}$$

GH_5 matrisi, $GF(2^4)/0\times13$ sonlu cismi üzerinde 8×8 tersi kendisine eşit MDS matrisler için 407 $(=183+8\times7\times4)$ XOR sayısıyla literatürde bilinen en iyi XOR sayılı matristir.

Önerilen GHadamard matris formu ayrıca tersi kendisine eşit olmayan MDS matrislerin üretilmesi için de kullanılabilir. Tez kapsamında yapılan çalışma doğrultusunda $GF(2^4)$ ve $GF(2^8)$ sonlu cisimleri üzerinde minimum XOR sayılı 8×8 tersi kendisine eşit olmayan MDS matrislerin aranması üzerine de çalışılmıştır. Bu kapsamda öncelikle $GF(2^4)$ üzerinde 8×8 Hadamard tersi kendisine eşit olmayan MDS matrisler için arama yapılmıştır. Buradan, $GF(2^4)$ üzerinde 14 farklı grup/sınıf (her grup/sınıf sekiz farklı eleman içerir) ile 18,816 tane $(=14\times1344)$ 8×8 Hadamard tersi kendisine eşit olmayan MDS matris üretilmiştir. Dahası (0×1) (1) elemanını içeren sekiz farklı grup/sınıf sınıflandırılmıştır. Daha sonra her sekiz farklı grup/sınıftan, sekiz Hadamard MDS (bu matrislerin ilk elemanları a_0 1'e eşit olmak üzere) matris seçilmiştir. Son olarak bu matrislere karşılık gelen GHadamard matrisleri üretilmiştir. Sonuç olarak $GF(2^4)/0\times13$ üzerinde, GHadamard matris formunda $8\times(2^4-1)^7 \simeq 2^{30,35}$ tane 8×8 tersi kendisine eşit olmayan MDS matris üretilmiştir. Bu matrisler içinden bazıları $GF(2^4)$ sonlu cismi üzerinde 8×8 tersi kendisine eşit olmayan MDS matrisler için bilinen en iyi XOR sayısı 394'ten [12] $(=170+8\times7\times4)$ daha düşük XOR sayılı matrislerdir. $GF(2^4)/0\times13$ sonlu cismi üzerinde elde edilen bu 8×8 tersi kendisine eşit olmayan MDSlerin XOR sayısı 380'dir $(=156+8\times7\times4)$. Bu matrisler, $GF(2^4)$ üzerinde 8×8 tersi kendisine eşit olmayan MDS matrisler için bilinen en iyi (düşük) XOR sayılı matrislerdir. Önerilen yöntem Örnek 4.8'de verilmiştir.

Örnek 4.8. $GF(2^4)$ üzerinde α ilkel eleman, 0×13 indirgenemez polinomu altında 8×8 Hadamard tersi kendisine eşit olmayan MDS $H_6 = \text{had}(0\times1, 0\times2, 0\times4, 0\times\text{b}, 0\times3, 0\times7, 0\times9, 0\times\text{c}) = \text{had}(1, \alpha, \alpha^2, \alpha^7, \alpha^4, \alpha^{10}, \alpha^{14}, \alpha^6)$ matrisi olsun. H_6 matrisi üzerinde $b_1=\alpha^7$, $b_2=1$, $b_3=\alpha^8$, $b_4=1$, $b_5=\alpha^8$, $b_6=1$ ve $b_7=\alpha^7$ parametreleri ile üretilen GHadamard

$GH_6 = \text{Ghad}(1,\alpha;\alpha^7,\alpha^2;1,\alpha^7;\alpha^8,\alpha^4;1,\alpha^{10};\alpha^8,\alpha^{14};1,\alpha^6;\alpha^7)$ tersi kendisine eşit MDS matrisi Eşitlik (4.20)'de verilmiştir.

$$GH_6 = \begin{bmatrix} 1 & \alpha^8 & \alpha^2 & 1 & \alpha^4 & \alpha^3 & \alpha^{14} & \alpha^{13} \\ \alpha^9 & 1 & 1 & \alpha^3 & \alpha^3 & \alpha^5 & \alpha^{14} & \alpha^{14} \\ \alpha^2 & \alpha^{14} & 1 & \alpha^9 & \alpha^{14} & \alpha^{14} & \alpha^4 & \alpha^2 \\ \alpha^{14} & \alpha & \alpha^8 & 1 & \alpha^{13} & \alpha^{14} & \alpha^2 & \alpha^3 \\ \alpha^4 & \alpha^2 & \alpha^{14} & \alpha^{14} & 1 & \alpha^9 & \alpha^2 & \alpha^{14} \\ \alpha^2 & \alpha^3 & \alpha^{13} & \alpha^{14} & \alpha^8 & 1 & \alpha^{14} & \alpha \\ \alpha^{14} & \alpha^{13} & \alpha^4 & \alpha^3 & \alpha^2 & 1 & 1 & \alpha^8 \\ \alpha^{14} & \alpha^{14} & \alpha^3 & \alpha^5 & 1 & \alpha^3 & \alpha^9 & 1 \end{bmatrix} \quad (4.20)$$

GH_6 matrisi, $GF(2^4)/0\times13$ sonlu cismi üzerinde 8×8 tersi kendisine eşit olmayan MDS matrisler için 380 $(=156+8\times7\times4)$ XOR sayısıyla literatürde bilinen en iyi XOR sayılı matristir.

Tez kapsamında, $GF(2^8)$ üzerinde minimum XOR sayılı 8×8 tersi kendisine eşit olmayan MDS matrislerin aranması üzerine de çalışılmıştır. $GF(2^8)$ üzerinde düşük XOR sayılı tersi kendisine eşit MDS matrisleri kontrol etmek ve belirlemek zordur çünkü çok fazla sayıda tersi kendisine eşit olmayan MDS matris vardır. Bu nedenle $GF(2^4)$ ve $GF(2^8)$ üzerinde 4×4 tersi kendisine eşit MDS matrislerin üretilmesi işlemlerinde edinilen tecrübeye dayanarak, $GF(2^8)/1c3$ üzerinde tanımlı özel Toeplitz tersi kendisine eşit olmayan MDS matrisine karşılık gelen GHadamard matrisleri oluşturulmuştur. Bu özel Toeplitz tersi kendisine eşit olmayan MDS matris şu şekildedir: $\text{Toep}(1,1,\alpha,\alpha^{253},1,\alpha^{253},\alpha^{252},\alpha^{157},\alpha^{158},\alpha^{253},\alpha^{254},\alpha,\alpha^{254},\alpha^2,\alpha)$ [12]. Bu matris $GF(2^8)$ üzerinde 8×8 tersi kendisine eşit olmayan MDS matrisler için bilinen en iyi XOR sayısı 680 $(=232+8\times7\times8)$ XOR'a sahiptir. Bu bağlamda, [12]'de verilen Toeplitz tersi kendisine eşit olmayan MDS matrisine karşılık gelen GHadamard matrisler üretilmiştir. Bilinen en iyi XOR sayısı ile aynı (680 XOR) GHadamard formda $GF(2^8)$ üzerinde yedi tane 8×8 tersi kendisine eşit olmayan MDS matris elde edilmiştir. Önerilen yöntem Örnek 4.9'da verilmiştir.

Örnek 4.9. GF(2^8) üzerinde α ilkel eleman, $0\times1c3$ indirgenemez polinomu altında 8×8 Toeplitz tersi kendisine eşit olmayan MDS $T = \text{Toep}(0\times01, 0\times01, 0\times02, 0\times91, 0\times01, 0\times91, 0\times a9, 0\times03, 0\times06, 0\times91, 0\times e1, 0\times02, 0\times e1, 0\times04, 0\times02) = \text{Toep}(1,1,\alpha,\alpha^{253},1,\alpha^{253},\alpha^{252},\alpha^{157},\alpha^{158},\alpha^{253},\alpha^{254},\alpha,\alpha^{254},\alpha^2,\alpha)$ matrisi olsun. T matrisi üzerinde $b_1=1,\ b_2=1,\ b_3=1,\ b_4=1,\ b_5=1,\ b_6=1$ ve $b_7=\alpha$ parametreleri ile üretilen GHadamard formundaki GT matrisi Eşitlik (4.21)'de verilmiştir.

$$GT = \begin{bmatrix} 1 & 1 & \alpha & \alpha^{253} & 1 & \alpha^{253} & \alpha^{252} & \alpha^{158} \\ \alpha^{158} & 1 & 1 & \alpha & \alpha^{253} & 1 & \alpha^{253} & \alpha^{253} \\ \alpha^{253} & \alpha^{158} & 1 & 1 & \alpha & \alpha^{253} & 1 & \alpha^{254} \\ \alpha^{254} & \alpha^{253} & \alpha^{158} & 1 & 1 & \alpha & \alpha^{253} & \alpha \\ \alpha & \alpha^{254} & \alpha^{253} & \alpha^{158} & 1 & 1 & \alpha & \alpha^{254} \\ \alpha^{254} & \alpha & \alpha^{254} & \alpha^{253} & \alpha^{158} & 1 & 1 & \alpha^2 \\ \alpha^2 & \alpha^{254} & \alpha & \alpha^{254} & \alpha^{253} & \alpha^{158} & 1 & \alpha \\ 1 & \alpha & \alpha^{253} & 1 & \alpha^{253} & \alpha^{252} & \alpha^{157} & 1 \end{bmatrix} \tag{4.21}$$

GT matrisi, GF(2^8)/1c3 sonlu cismi üzerinde 8×8 tersi kendisine eşit olmayan MDS matrisler için 680 ($=232+8\times7\times8$) XOR sayısıyla literatürde bilinen en iyi XOR sayılı matrislerden biridir. T matrisi üzerinde GHadamard formu ile üretilen diğer matrisler ve bu matrisler için $b_i=(b_1,b_2,\ldots b_7)$ parametreleri Tablo 4.2'de verilmiştir.

Tablo 4.2. GF(2^8)/1c3 sonlu cismi üzerinde 680 XOR sayılı GHadamard matris formu ile üretilen 8×8 tersi kendisine eşit olmayan MDS matrisler ve b_i parametreleri

$\begin{bmatrix} 1 & 1 & \alpha & \alpha^{253} & 1 & \alpha^{253} & \alpha^{253} & \alpha^{158} \\ \alpha^{158} & 1 & 1 & \alpha & \alpha^{253} & 1 & \alpha^{254} & \alpha^{253} \\ \alpha^{253} & \alpha^{158} & 1 & 1 & \alpha & \alpha^{253} & \alpha & \alpha^{254} \\ \alpha^{254} & \alpha^{253} & \alpha^{158} & 1 & 1 & \alpha & \alpha^{253} & \alpha \\ \alpha & \alpha^{254} & \alpha^{253} & \alpha^{158} & 1 & 1 & \alpha^2 & \alpha^{254} \\ \alpha^{254} & \alpha & \alpha^{254} & \alpha^{253} & \alpha^{158} & 1 & \alpha & \alpha^2 \\ \alpha & \alpha^{253} & 1 & \alpha^{253} & \alpha^{252} & \alpha^{157} & 1 & 1 \\ 1 & \alpha & \alpha^{253} & 1 & \alpha^{253} & \alpha^{252} & \alpha^{158} & 1 \end{bmatrix}$ $(b_1,b_2,\ldots,b_7)=(1,1,1,1,1,\alpha,\alpha)$	$\begin{bmatrix} 1 & 1 & \alpha & \alpha^{253} & 1 & \alpha^{254} & \alpha^{253} & \alpha^{158} \\ \alpha^{158} & 1 & 1 & \alpha & \alpha^{253} & \alpha & \alpha^{254} & \alpha^{253} \\ \alpha^{253} & \alpha^{158} & 1 & 1 & \alpha & \alpha^{254} & \alpha & \alpha^{254} \\ \alpha^{254} & \alpha^{253} & \alpha^{158} & 1 & 1 & \alpha^2 & \alpha^{254} & \alpha \\ \alpha & \alpha^{254} & \alpha^{253} & \alpha^{158} & 1 & \alpha & \alpha^2 & \alpha^{254} \\ \alpha^{253} & 1 & \alpha^{253} & \alpha^{252} & \alpha^{157} & 1 & 1 & \alpha \\ \alpha & \alpha^{253} & 1 & \alpha^{253} & \alpha^{252} & \alpha^{158} & 1 & 1 \\ 1 & \alpha & \alpha^{253} & 1 & \alpha^{253} & \alpha^{253} & \alpha^{158} & 1 \end{bmatrix}$ $(b_1,b_2,\ldots,b_7)=(1,1,1,1,\alpha,\alpha,\alpha)$

Tablo 4.2. (Devam) GF(2^8)/1c3 sonlu cismi üzerinde 680 XOR sayılı GHadamard matris formu ile üretilen 8×8 tersi kendisine eşit olmayan MDS matrisler ve b_i parametreleri

$\begin{bmatrix} 1 & 1 & \alpha & \alpha^{253} & \alpha & \alpha^{254} & \alpha^{253} & \alpha^{158} \\ \alpha^{158} & 1 & 1 & \alpha & \alpha^{254} & \alpha & \alpha^{254} & \alpha^{253} \\ \alpha^{253} & \alpha^{158} & 1 & 1 & \alpha^{2} & \alpha^{254} & \alpha & \alpha^{254} \\ \alpha^{254} & \alpha^{253} & \alpha^{158} & 1 & \alpha & \alpha^{2} & \alpha^{254} & \alpha \\ 1 & \alpha^{253} & \alpha^{252} & \alpha^{157} & 1 & 1 & \alpha & \alpha^{253} \\ \alpha^{253} & 1 & \alpha^{253} & \alpha^{252} & \alpha^{158} & 1 & 1 & \alpha \\ \alpha & \alpha^{253} & 1 & \alpha^{253} & \alpha^{253} & \alpha^{158} & 1 & 1 \\ 1 & \alpha & \alpha^{253} & 1 & \alpha^{254} & \alpha^{253} & \alpha^{158} & 1 \end{bmatrix}$ $(b_1, b_2, \ldots, b_7) = (1,1,1,\alpha,\alpha,\alpha,\alpha)$	$\begin{bmatrix} 1 & 1 & \alpha & \alpha^{254} & \alpha & \alpha^{254} & \alpha^{253} & \alpha^{158} \\ \alpha^{158} & 1 & 1 & \alpha^{2} & \alpha^{254} & \alpha & \alpha^{254} & \alpha^{253} \\ \alpha^{253} & \alpha^{158} & 1 & \alpha & \alpha^{2} & \alpha^{254} & \alpha & \alpha^{254} \\ \alpha^{253} & \alpha^{252} & \alpha^{157} & 1 & 1 & \alpha & \alpha^{253} & 1 \\ 1 & \alpha^{253} & \alpha^{252} & \alpha^{158} & 1 & 1 & \alpha & \alpha^{253} \\ \alpha^{253} & 1 & \alpha^{253} & \alpha^{253} & \alpha^{158} & 1 & 1 & \alpha \\ \alpha & \alpha^{253} & 1 & \alpha^{254} & \alpha^{253} & \alpha^{158} & 1 & 1 \\ 1 & \alpha & \alpha^{253} & \alpha & \alpha^{254} & \alpha^{253} & \alpha^{158} & 1 \end{bmatrix}$ $(b_1, b_2, \ldots, b_7) = (1,1,\alpha,\alpha,\alpha,\alpha,\alpha)$
$\begin{bmatrix} 1 & 1 & \alpha^{2} & \alpha^{254} & \alpha & \alpha^{254} & \alpha^{253} & \alpha^{158} \\ \alpha^{158} & 1 & \alpha & \alpha^{2} & \alpha^{254} & \alpha & \alpha^{254} & \alpha^{253} \\ \alpha^{252} & \alpha^{157} & 1 & 1 & \alpha & \alpha^{253} & 1 & \alpha^{253} \\ \alpha^{253} & \alpha^{252} & \alpha^{158} & 1 & 1 & \alpha & \alpha^{253} & 1 \\ 1 & \alpha^{253} & \alpha^{253} & \alpha^{158} & 1 & 1 & \alpha & \alpha^{253} \\ \alpha^{253} & 1 & \alpha^{254} & \alpha^{253} & \alpha^{158} & 1 & 1 & \alpha \\ \alpha & \alpha^{253} & \alpha & \alpha^{254} & \alpha^{253} & \alpha^{158} & 1 & 1 \\ 1 & \alpha & \alpha^{254} & \alpha & \alpha^{254} & \alpha^{253} & \alpha^{158} & 1 \end{bmatrix}$ $(b_1, b_2, \ldots, b_7) = (1,\alpha,\alpha,\alpha,\alpha,\alpha,\alpha)$	$\begin{bmatrix} 1 & \alpha & \alpha^{2} & \alpha^{254} & \alpha & \alpha^{254} & \alpha^{253} & \alpha^{158} \\ \alpha^{157} & 1 & 1 & \alpha & \alpha^{253} & 1 & \alpha^{253} & \alpha^{252} \\ \alpha^{252} & \alpha^{158} & 1 & 1 & \alpha & \alpha^{253} & 1 & \alpha^{253} \\ \alpha^{253} & \alpha^{253} & \alpha^{158} & 1 & 1 & \alpha & \alpha^{253} & 1 \\ 1 & \alpha^{254} & \alpha^{253} & \alpha^{158} & 1 & 1 & \alpha & \alpha^{253} \\ \alpha^{253} & \alpha & \alpha^{254} & \alpha^{253} & \alpha^{158} & 1 & 1 & \alpha \\ \alpha & \alpha^{254} & \alpha & \alpha^{254} & \alpha^{253} & \alpha^{158} & 1 & 1 \\ 1 & \alpha^{2} & \alpha^{254} & \alpha & \alpha^{254} & \alpha^{253} & \alpha^{158} & 1 \end{bmatrix}$ $(b_1, b_2, \ldots, b_7) = (\alpha,\alpha,\alpha,\alpha,\alpha,\alpha,\alpha)$

Tez kapsamında geliştirilen ve önerilen GHadamard matris formu, simetrik anahtarlı şifrelerin yayılım tabakalarında maksimum yayılımın sağlanması ve çığ etkisinin güçlendirilmesi amaçlı kullanılan MDS matrislerin üretilmesini sağlayan bir formdur. Bu yapı ile özellikle hafif sıklet yayılım tabakalarında kullanılabilecek, literatürde yer almayan minimum XOR sayılı (tersi kendisine eşit) MDS matrislerin üretildiği görülmüştür. Bu matrisler, yeni tasarlanacak hafif sıklet blok şifreler için özgün ve güçlü yayılım tabakaları oluşturacaklardır.

GHadamard matris formu ile elde edilen sonuçlar literatürde yer alan diğer çalışmalar ile karşılaştırılmış, sonuçlar Tablo 4.3 ve Tablo 4.4'de verilmiştir. Tablolarda tersi kendisine eşit matrisler kısaca "TKE", tersi kendisine eşit olmayan matrisler ise "TKEO" şeklinde verilmiştir.

Tablo 4.3'de GF(2^4) ve GF(2^8) sonlu cisimleri üzerinde 4×4 tersi kendisine eşit MDS ve tersi kendisine eşit olmayan MDS matrisler XOR sayısı metriği kullanılarak karşılaştırılmıştır. GHadamard form ile üretilen, GF(2^4) üzerinde 4×4 tersi

kendisine eşit MDS ve $GF(2^8)$ üzerinde 4×4 tersi kendisine eşit MDS matrisleri literatürde yer alan minimum XOR sayılı matrislerdir.

Tablo 4.3. $GF(2^4)$ ve $GF(2^8)$ sonlu cisimleri üzerinde 4×4 tersi kendisine eşit MDS ve 4×4 tersi kendisine eşit olmayan MDS matrislerin bilinen en iyi XOR sayısı metriğine göre karşılaştırılması

	Referans	**Sonlu Cisim**	**Matris Formu**	**XOR sayısı**
		$GF(2^4)$		
TKE	Örnek 4.5, Tablo 4.1	$GF(2^4)/0\times13$	**GHadamard**	$64\,(=16+4\times3\times4)$
TKE	[20]	$GF(2^4)/0\times13$	Form (14)	$64\,(=16+4\times3\times4)$
TKE	[18]	$GF(2^4)/0\times13$	Hadamard	$72\,(=24+4\times3\times4)$
TKEO	[20]	$GF(2^4)/0\times19$	Toeplitz	$58\,(=10+4\times3\times4)$
TKEO	[6]	$GF(2^4)/0\times13$	Serial-Circulant	$60\,(=12+4\times3\times4)$
TKEO	[64]	$GF(2^4)/0\times13$	Serial	$64\,(=16+4\times3\times4)$
TKEO	[18]	$GF(2^4)/0\times13$	Hadamard	$68\,(=20+4\times3\times4)$
		$GF(2^8)$		
TKE	Örnek 4.6	$GF(2^8)/0\times1c3$	**GHadamard**	$158\,(=62+4\times3\times8)$
TKE	[18]	$GF(2^8)/0\times165$	Hadamard	$160\,(=64+4\times3\times8)$
TKE	[20]	$GF(2^8)/0\times165$	Form (14)	$160\,(=64+4\times3\times8)$
TKEO	[20]	$GF(2^8)/0\times1c3$	Toeplitz	$123\,(=27+4\times3\times8)$
TKEO	[10]	$GF(2^8)/0\times1c3$	Left-Circulant	$128\,(=32+4\times3\times8)$
TKEO	[6]	$GF(2^8)/0\times11d$	Serial-Circulant	$132\,(=36+4\times3\times8)$
TKEO	[18]	$GF(2^8)/0\times11d$	Hadamard	$148\,(=52+4\times3\times8)$

Tablo 4.4'de $GF(2^4)$ ve $GF(2^8)$ sonlu cisimleri üzerinde 8×8 tersi kendisine eşit MDS ve tersi kendisine eşit olmayan MDS matrisler XOR sayısı metriği kullanılarak karşılaştırılmıştır. Tabloya göre GHadamard form ile üretilen, $GF(2^4)$ üzerinde 8×8 tersi kendisine eşit MDS ve tersi kendisine eşit olmayan MDS matrisleri, literatürde yer alan minimum XOR sayılı matrislerdir. Ayrıca $GF(2^8)$ sonlu cismi üzerinde 8×8 tersi kendisine eşit olmayan MDS matrisler için literatürde yer alan minimum XOR sayılı matrisler üretilmiştir.

Her iki tablodan da açıkça görülebilir ki; Hadamard matrisler minimun XOR sayılı tersi kendisine eşit MDS matrisleri üretmek için kullanılan önemli özel matris formlarından biriyken, Toeplitz matrisler de minimun XOR sayılı tersi kendisine eşit olmayan MDS matrisleri üretmek için kullanılan önemli matris formlarından biridir.

Bu nedenle Hadamard matrislerin çalışmanın merkezine alınması ve önerilen GHadamard formunun özel matris tiplerine (Toeplitz, Dairesel) de uygulanabilir olması bu formu minimum XOR sayılı (tersi kendisine eşit) MDS matrislerin üretimi açısından çok başarılı kılmıştır.

Tablo 4.4. $GF(2^4)$ ve $GF(2^8)$ sonlu cisimleri üzerinde 8×8 tersi kendisine eşit MDS ve 8×8 tersi kendisine eşit olmayan MDS matrislerin bilinen en iyi XOR sayısı metriğine göre karşılaştırılması

	Referans	**Sonlu Cisim**	**Matris Formu**	**XOR sayısı**
		$GF(2^4)$		
TKE	Örnek 4.7	$GF(2^4)/0\times13$	**GHadamard**	407 $(=183+8\times7\times4)$
TKE	[18]	$GF(2^4)/0\times13$	Hadamard	512 $(=288+8\times7\times4)$
TKEO	Örnek 4.8	$GF(2^4)/0\times13$	**GHadamard**	380 $(=156+8\times7\times4)$
TKEO	[12]	$GF(2^4)/0\times13$	Toeplitz	394 $(=170+8\times7\times4)$
TKEO	[18]	$GF(2^4)/0\times13$	Hadamard	432 $(=208+8\times7\times4)$
		$GF(2^8)$		
TKE	[18]	$GF(2^8)/0\times1c3$	Hadamard	494 $(=46+8\times7\times8)$
TKE	[91]	$GF(2^8)/0\times11d$	Hadamard	546 $(=98+8\times7\times8)$
TKE	[16]	$GF(2^8)/0\times11b$	Hadamard-Cauchy	570 $(=122+8\times7\times8)$
TKEO	Örnek 4.9 Tablo 4.2	$GF(2^8)/0\times1c3$	**GHadamard**	680 $(=232+8\times7\times8)$
TKEO	[12]	$GF(2^8)/0\times1c3$	Toeplitz	680 $(=232+8\times7\times8)$
TKEO	[10]	$GF(2^8)/0\times1c3$	Circulant	688 $(=240+8\times7\times8)$
TKEO	[18]	$GF(2^8)/0\times1c3$	Hadamard	768 $(=320+8\times7\times8)$

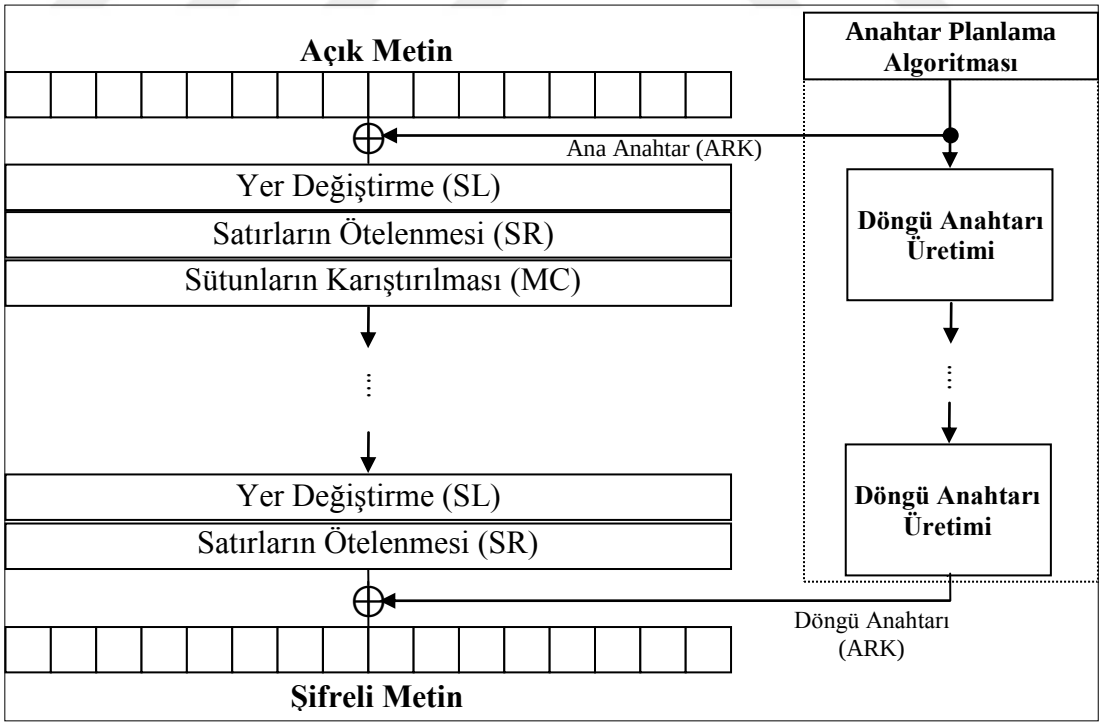
Tez çalışmasının ilerleyen bölümünde, tez kapsamında AES şifreleme algoritmasından esinlenilerek geliştirilen hafif sıklet blok şifrenin yapısı ayrıntılı olarak verilecektir.

5. ELDE EDİLEN SONUÇLAR VE AES ŞİFRESİ KULLANILARAK BİR HAFİF SIKLET BLOK ŞİFRE UYGULAMASI

Tezin bu bölümünde tez kapsamında geliştirilen hafif sıklet blok şifreleme algoritmasının yapısı ayrıntılı olarak verilecektir.

5.1. Geliştirilen Hafif Sıklet Blok Şifrenin Yapısı

Önerilen hafif sıklet blok şifreleme algoritması SPN mimarisinde, 64-bit blok uzunluğunda, 64-bit, 80-bit ve 128-bit anahtar uzunlukları için sırasıyla 12, 14, 16 döngü kullanır. Döngü yapısı AES şifreleme algoritmasına benzer yapıda tasarlanmış olup, her döngü sırasıyla Yer Değiştirme (SL), Satırların Ötelenmesi (SR), Sütunların Karıştırılması (MC) ve Döngü Anahtarı Ekleme (ARK) adımlarından oluşur. Algoritmanın döngü yapısı Şekil 5.1'de verilmiştir. Son döngüde tıpkı AES şifreleme algoritmasındaki gibi sütunların karıştırılması MC adımı uygulanmamıştır.



Şekil 5.1. Önerilen hafif sıklet blok şifrenin şifreleme blok diyagramı

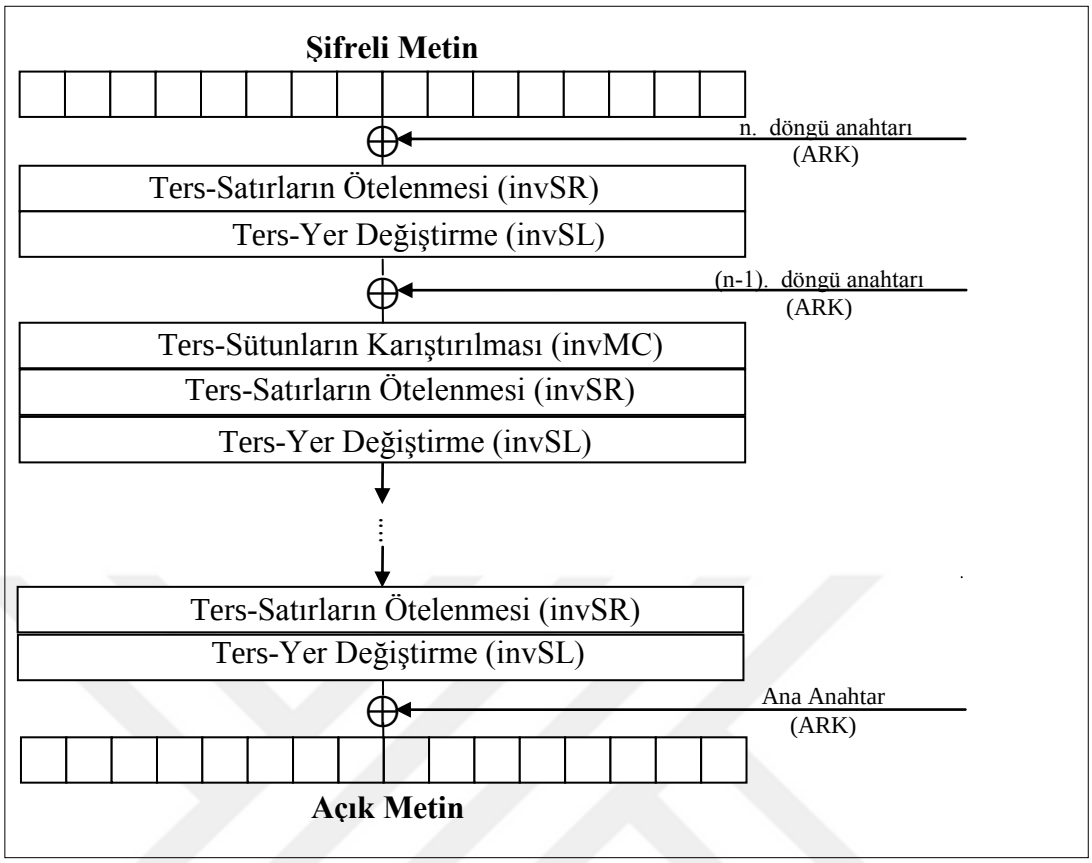
Şifreleme işlemlerinde bir döngü içindeki adımlar aşağıdaki işlemleri içerir:

- SLayer: Bu dönüşümde giriş bitleri S-kutusundan geçirilerek başka bit değerlerine haritalanır. Bu yapıyla karıştırma (confusion) özelliğinin sağlanması amaçlanır. S-kutusu olarak optimal özelliklere sahip literatürde yer almayan tersi kendisine eşit bir S-kutusu kullanılmıştır.
- SR: Bu dönüşümde AES şifreleme algoritmasında yer alan satırların ötelenmesi adımına çok benzer. Elde edilen durum matrisinin ilk satırı değiştirilmeden kalır, 2, 3 ve 4. satırlar sırasıyla 4-bit, 8-bit ve 12-bit sola dairesel olarak kaydırılır.
- MC: Bu dönüşümde GHadamard matris formuyla elde edilen 4×4 tersi kendisine eşit MDS yayılım matrisi SR adımından gelen durum bitleri ile çarpılır. Bu matrisin tersi kendisine eşit olması şifre çözme adımında da aynı matrisin kullanılmasını sağlar.
- ARK: Şifrenin anahtar planlama algoritmasından gelen 64-bit, 80-bit veya 128-bit döngü anahtarı (alt anahtarlar) giriş bitleri ile XOR'lanır.

Şifre çözme adımında yapılan işlemler, şifreleme adımlarında yapılan işlemlerin tersidir. Yine aynı döngü anahtarları kullanılır, ARK yine XOR işlemiyle sağlanır. Ayrıca döngü anahtarları ters sırayla kullanılır, yani son döngüde (n. döngü) kullanılan döngü anahtarı şifre çözme algoritmasının ilk döngüsünde kullanılır. Diğer işlemler ise Ters-Yer Değiştirme (invSL), Ters-Satırların Ötelenmesi (invSR) ve Ters-Sütunların Karıştırılması (invMC) işlemleridir.

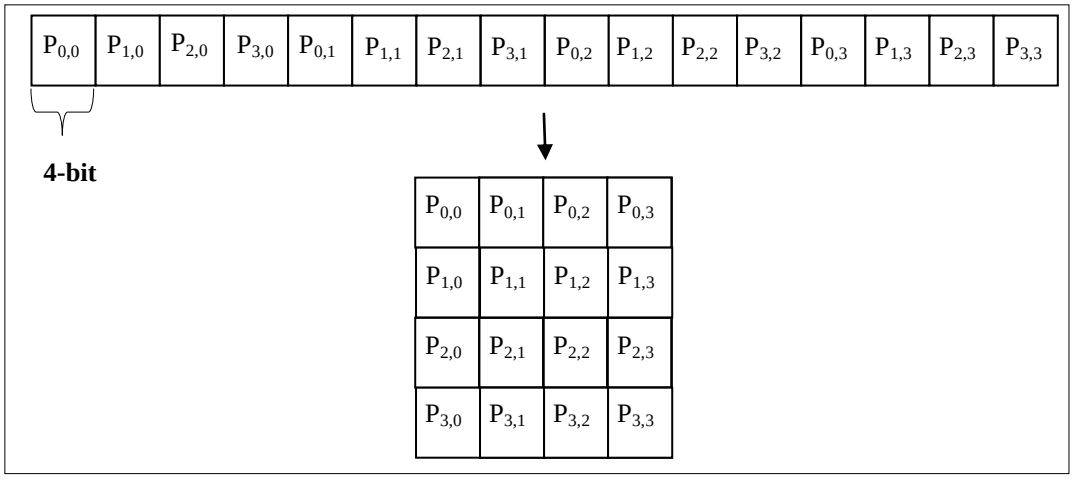
- invSL: Bu dönüşümde SL dönüşümünde kullanılan S-kutusunun tersi kullanılır. Durum bitleri bu ters S-kutusundan geçirilerek başka bit değerlerine haritalanır. Geliştirilen S-kutusu involtutif olduğundan bu adımda da SL'de kullanılan S-kutusu kullanılır.
- invSR: Bu dönüşümde elde edilen durum matrisinin ilk satırı değiştirilmeden kalır, 2, 3 ve 4. satırlar sırasıyla 4-bit, 8-bit ve 12-bit sağa dairesel olarak kaydırılır.
- invMC: Bu dönüşümde MC adımında kullanılan 4×4 tersi kendisine eşit doğrusal dönüşüm matrisi ile sonlu cisim üzerinde çarpma işlemi yapılır.

Önerilen hafif sıklet blok şifrenin şifre çözme adımı Şekil 5.2'de verilmiştir.



Şekil 5.2. Önerilen hafif sıklet blok şifrenin şifre çözme blok diyagramı

Şifreleme işleminden önce 64-bit blok uzunluğundaki açık metin bitleri ile Şekil 5.3'deki gibi durum matrisi oluşturulur.



Şekil 5.3. Önerilen hafif sıklet blok şifre için durum matrisi

5.1.1. SLayer adımı

Önerilen hafif sıklet blok şifrede kullanılan S-kutusu literatürde yer almayan $x \to x^{-1}$ üs haritalaması ile oluşturulmuş $S: GF(2)^4 \to GF(2)^4, S(x) = x^{-1}$ tersi kendisine eşit S-kutusudur. S-kutusu Tablo 5.1'de verilmiştir.

Tablo 5.1. Geliştirilen S-kutusu

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

Tablo 5.2'de geliştirilen S-kutusunun LAT tablosu verilmiştir.

Tablo 5.2. Geliştirilen S-kutusunun LAT tablosu

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

LAT tablosundaki değerler incelendiğinde 0, 2, 4 ve 8 bias değerlerinden sırasıyla 105, 120, 30, 1 tane yer aldığı görülür. 8 değeri dikkate alınmaz bu nedenle tablodaki maksimum bias değeri 4'tür. Buradan S-kutusunun maksimum doğrusal olasılığı

$LP^S = \left(\frac{4}{2^3}\right)^2 = 2^{-2}$ olduğu görülür. Diğer bir deyişle doğrusal yaklaşımın maksimum mutlak (absolute) bias değeri 2^{-2} 'dir denir. Buradan mutlak bias değerine (4 değerine) sahip 30 doğrusal yaklaşım vardır.

Tablo 5.3'de ise geliştirilen S-kutusuna ait DDT tablosu verilmiştir.

Tablo 5.3. Geliştirilen S-kutusunun DDT tablosu

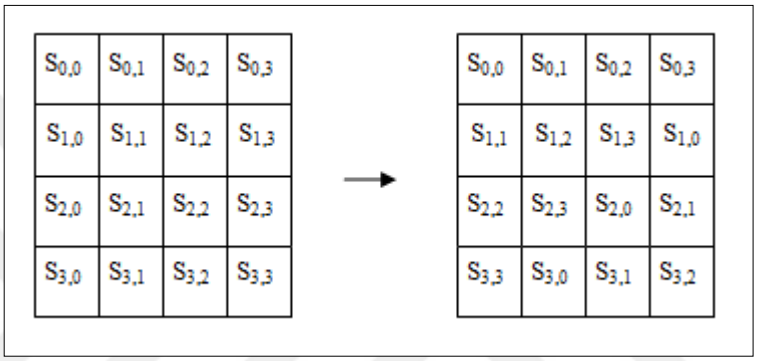
		Çıkış Toplamı															
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
Giriş Toplamı	**0**	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	2	0	0	2	2	2	4	0	2	0	2	0	0	0
	2	0	2	0	0	4	2	0	0	0	2	2	2	2	0	0	0
	3	0	0	0	4	2	0	2	0	2	2	0	0	2	0	0	2
	4	0	0	4	2	2	0	0	0	2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

DDT tablosundaki değerler incelendiğinde 0, 2, 4 ve 16 diferansiyellerinden sırasıyla 150, 90, 15, 1 tane yer aldığı görülür. 0 için 16 değeri dikkate alınmaz bu nedenle tablodaki maksimum değerli diferansiyel 4'tür. Buradan S-kutusunun maksimum diferansiyel olasılığı $\frac{4}{2^4} = 2^{-2}$ olduğu görülür. 2^{-2} olasılıklı 15 diferansiyel vardır. Ayrıca sıfır olmayan bileşen fonksiyonlarının her birinin cebirsel derecesi (algebraic degree) 3'tür. [104, 105]'de verilen bilgilere göre tüm bu özellikleri sağlayan sekiz tane S-kutusu sınıfı vardır. Geliştirilen S-kutusu da bu sekiz sınıftan birine aittir, yani

4×4 $S:GF(2)^4 \rightarrow GF(2)^4$ için en iyi tersi kendisine eşit S-kutularından biridir. Ayrıca S-kutusu 14,5 GE alana sahiptir. Ayrıca S-kutusu tersi kendisine eşit olduğundan şifre çözme adımında kullanılan invSLayer adımında da aynı S-kutusu kullanılır.

5.1.2. SRows adımı

SLayer dönüşümünden gelen çıktı bitleri bu dönüşüm için girdi bitlerini oluşturur. Bu dönüşüm AES şifreleme algoritmasının satırların ötelenmesi adımına çok benzer.



Şekil 5.4. Önerilen hafif sıklet blok şifre için SRows

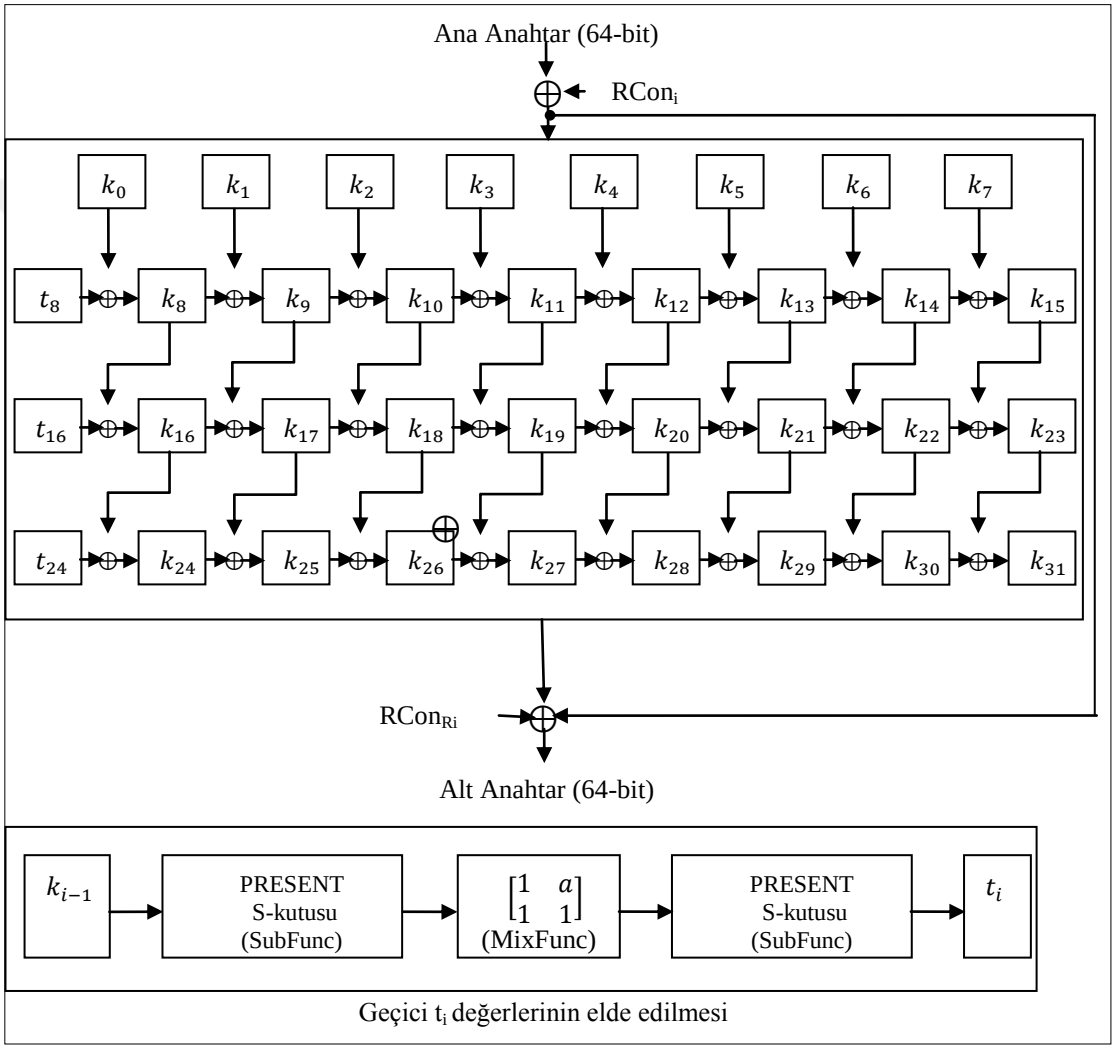
Şekil 5.4'de önerilen hafif sıklet blok şifreleme algoritması için SRows dönüşümü verilmiştir. Bu dönüşümde durum matrisinin ilk satırı değiştirilmeden kalır, 2, 3 ve 4. satırlar sırasıyla 4-bit, 8-bit ve 12-bit sola dairesel olarak kaydırılır. invsRows dönüşümünde ise matrisinin ilk satırı değiştirilmeden kalır, 2, 3 ve 4. satırlar sırasıyla 4-bit, 8-bit ve 12-bit sağa dairesel olarak kaydırılır.

5.1.3. MC adımı

Bu dönüşümde GHadamard matris formuyla $GF(2^4)/0\times13$ sonlu cismi üzerinde üretilen, mininmum XOR sayılı (64 XOR) ve Eşitlik (4.17)'de verilen tersi kendisine eşit MDS $GH_3 = \begin{bmatrix} 1 & \alpha^2 & \alpha & \alpha^3 \\ 1 & 1 & \alpha^2 & 1 \\ 1 & \alpha^3 & 1 & \alpha \\ \alpha^2 & \alpha & \alpha & 1 \end{bmatrix}$ matrisi önerilen hafif sıklet blok şifreleme algoritmasında doğrusal dönüşüm matrisi olarak kullanılmıştır. invMC dönüşmünde de tersi kendisine eşit özelliğinden dolayı aynı matris kullanılır.

5.1.4. Anahtar planlama yapısı

64, 80 ve 128-bit anahtar uzunlukları için sırasıyla 12, 14 ve 16 döngü seçenekleri ile anahtar planlama algoritması geliştirilmiştir. Geliştirilen yapı AES şifreleme algoritmasının yavaş yayılım özelliği (ilişkili anahtar saldırılarına açık) ve bit sızıntısı problemlerini iyileştirmek amaçlı tasarlanmıştır. Algoritmanın yapısı Şekil 5.5'de (64-bit) verilmiştir.



Şekil 5.5. Önerilen anahtar planlama yapısı (64-bit)

Şekil 5.5'de 64-bitlik anahtar planlama algoritması verilmiştir. 80 ve 128-bit için de yapı benzerdir. Her döngü için, 64-bitlik ana anahtar (master key) ilgili döngüdeki RCon$_i$ (i=1,2,..,12) döngü sabitiyle (bu sabit her döngü için rastgele oluşturulur ve birbirinden farklıdır) XOR'lanarak elde edilen değer sekiz bitlik (1 byte) bloklara (k_j, j=0,..,7) ayrılır. AES şifreleme algoritmasındaki yavaş yayılım probleminin

giderilmesi için geçici t_i (i=8, 16, 24) değerleri elde edilirken bir yayılım elemanının daha eklenmesi gereklidir [105] bu nedenle Şekil 5.4'de verilen geçici t_i değerlerinin elde edilmesi adımında elemanları $GF(2^4)/0\times13$ sonlu cismi üzerinde tanımlı 2×2 boyutunda $\begin{bmatrix}1 & \alpha\\ 1 & 1\end{bmatrix}$ MDS matrisi ile çarpma adımı (MixFunc) eklenmiştir. Ayrıca bu geçici değerlerin elde edilmesinde Yer Değiştirme-Yayılım-Yer Değiştirme (Subtitution-Diffusion-Substitution/SDS) mimarisi kullanılmıştır. Yer değiştirme adımı için PRESENT şifreleme algoritmasında kullanılan S-kutusu seçilmiştir çünkü bu S-kutusu hem donanım verimli hem de diferansiyel ve doğrusal kriptoanalize karşı dirençlidir (SubFunc adımı). Tüm işlemler yapıldıktan sonra elde edilen $[t_{24}k_{24}k_{25}k_{26}k_{27}k_{28}k_{29}k_{30}k_{31}]$ bit dizisi, k_j (j=0,..,7) bit dizisi ve $RCon_{Ri}$ değerleri XOR'lanarak ilgili döngüye ait alt anahtar elde edilmiş olur. Şekil 5.6'da tüm bu işlemlere ait sözde kod gösterilmiştir.

Algoritma 1: Önerilen Anahtar Planlama Algoritması için Alt Anahtar Üretimi

```
yeniK[i]: yenianahtar değişkeninin i. byte değerleri;
r: Döngü sayısı  r = 1,2,…,12 ;
Master Key: Şifre ana anahtarı;
Rcon [r]: r. döngü sabiti;
RconR [r]: r. döngü sabitinin tersi;
sub key: alt anahtar;
for r = 1 to 12
   yenianahtar = Master Key ⊕ Rcon [r];
   for i = 0 to 8 do
      k [i] = yeniK [i];
   end
   for j = 8 to 32 do
      temp = k [j-1];
          if (j mod 8 = 0)
             temp = SubFunc (MixFunc (SubFunc (temp) ) );
          end
      k [j] = k [j-8] ⊕ temp;
      l = j;
   end
   sub temp = (k [l-7], k [l-6], k [l-5], k [l-4], k [l-3], k [l-2], k [l-1], k [l] );
   sub key [r] = yenianahtar ⊕ RconR [r] ⊕ sub temp;
   return sub key [r];
End
```

Şekil 5.6. Önerilen anahtar planlama algoritmasına ait (64-bit) sözde kodu

Tablo 5.4'de (27 F8 31 7F 09 AF EF 44) ana anahtarından üretilen alt (döngü) anahtarları, kullanılan $RCon_i$ döngü sabitleri ve $RCon_{Ri}$ döngü sabitlerinin ters değerleri ve ara adımlarda elde edilen geçici t_i değerleri verilmiştir.

Tablo 5.4. (27 F8 31 7F 09 AF EF 44) ana anahtarıyla üretilen alt anahtarlar

Döngü	**t_i Değerleri (t_8, t_{16}, t_{24})**	**$RCon_i$ Değeri**	**$RCon_{Ri}$ Değeri**	**Üretilen Alt (Döngü) Anahtarı**
1	AB, 3F, F1	42 FF 60 74 EB 43 30 1C	C1 03 34 BE 47 06 FF 24	A4 3C FD 1E 4C DC 8B 50
2	61, 53, 01	CE 5A FE 9B 0D FD 79 28	82 97 DF D0 B9 EF A5 EC	B1 1E 2F 1E 87 C0 C8 50
3	C0, 8A, D8	58 4B B2 A8 02 C5 F8 6C	C6 8F 5C 20 8A 2B B4 85	54 E8 BD 7B B0 F4 BC 1E
4	98, 3F, 37	79 9B 9E A5 D1 20 E3 D5	5D 3E 02 1D 5A E9 B9 97	CD CF 69 85 4E A3 F2 D9
5	78, D2, 8D	99 B3 2A 2C F2 A6 B3 0B	B0 3B 6A 2F C2 A2 3B 99	97 70 7E B9 08 59 3D 00
6	B7, 7B, 9D	4F 3A DE A4 DE C0 EE 67	76 EE 0C ED 4A ED A3 F4	27 AC 28 9F A8 BA 07 5C
7	07, 4B, 96	F3 98 33 95 0C F5 0E 80	08 E0 5F C0 59 33 89 3F	D2 A5 E2 54 BD 13 6C A0
8	7A, E3, F9	3D 46 B0 78 F5 CD 9B DF	FD B9 DC 5F 87 0B 64 D3	9D 20 78 27 FA D0 23 D8
9	71, 12, 8E	85 CB EE D4 30 A1 0B EA	AE B0 1A 03 4D EE BC 58	43 ED B5 52 A9 B9 C2 46
10	13, D8, 8E	54 AF 44 32 45 28 E8 2B	B2 8E 82 54 23 44 FA 45	F7 60 83 AF 58 B1 09 F4
11	38, 5B, 07	32 6B 4A 08 9C BA B9 2A	A2 9B AB C9 80 A4 B6 23	C6 B1 64 B5 86 88 17 7E
12	E3, AC, C3	95 76 78 D0 AE EF E7 6D	D6 7E FE EA 0D 87 67 59	5A EC 1F 60 9C 3C 8F 74

Bir anahtar üretim yapısında üretilen alt anahtarlar ile ana anahtar arasındaki ilişkinin tespiti için bazı testler mevcuttur bunlardan en önemlisi ana anahtarın tüm bit pozisyonlarının 1-bit değişimiyle alt anahtarlarda etkilenen bitleri gösteren Katı Çığ Etkisi (Strict Avalanche Criterion-SAC) [21] testidir. Bu test temelde bit yayılım özelliğinin ölçülmesinde kullanılır. Bu teste göre girdideki her 1-bitlik pozisyon değişimi, çıktıdaki her bir biti yaklaşık $1/2$ olasılıkla değiştirmesi gerekmektedir. SAC testinde, seçilen bir n değeri kadar test tekrar ettirilir ve bu n sayısına bağlı olarak bir güven aralığı ($\text{GüvenAralığı} = \pm\sqrt{n}$) oluşturulur. Güven aralığının dışındaki değerler sapma olarak değerlendirilir. Bir anahtar planlama algoritmasının bu algoritma için elde edilen bit değerlerinin yüksek oranda güven aralığında olması istenir. Şekil 5.7'de SAC testine ait sözde kod verilmiştir.

Algoritma 2: SAC Testi

```
r: Döngü sayısı  r = 1, 2, …, 12 ;
test_sayisi: n değeri;
Master Key: Şifre ana anahtarı;
counter [ ][ ][ ]: Güven aralığında yer alan bitlerin sayısı;
for t = 1 to test_sayisi
   64-bit Ana anahtar Master Key = round ( ) fonksiyonuyla rastgele üretilir;
   for i = 1 to 64 do
      Master Key değerinin i. biti değiştirilir K1 = degistir_bit (Master Key, i);
      Algoritma 1'e göre Master Key'in alt anahtarlarını hesapla SubKey_Master
      değişkenine ata;
      Algoritma 1'e göre K1'in alt anahtarlarını hesapla SubKey_K1 değişkenine
      ata;
   end
      for r = 1 to 12 do
         for j = 1 to 64 do
            if ( SubKey_Master [j] != SubKey_K1 [j] )
               counter [i] [r] [j] ++;
         end
      end
  end
End
```

Şekil 5.7. Önerilen anahtar planlama algoritmasına ait (64-bit) SAC testi sözde kodu

Geliştirilen anahtar planlama algoritması, n farklı ana anahtar değeri için SAC testinden geçirilmiştir. Bu n değerleri için elde edilen sonuçlar analiz edildiğinde, önerilen 64-bit anahtar planlama algoritması (12 döngü) için, her n değerinde SAC testi ile üretilen 49152 $(= 64 \times 12 \times 64)$ değerin yaklaşık % 94'ü güven aralığındadır. Bu da önerilen anahtar planlama algoritmasının, anahtar bitleri üzerinde iyi yayılım ve karıştırma sağlandığını garanti eder. Farklı n tane ana anahtar için SAC testi ile elde edilen ayrıntılı sonuçlar Tablo 5.5'de verilmiştir.

Tablo 5.5'de önerilen anahtar planlama algoritması için n farklı ana anahtardan üretilen değerler için; güven aralığında ve güven aralığının dışında (güven aralığından küçük veya büyük) yer alan değerlere ilişkin sonuçlar ayrıntılı olarak verilmiştir. Ayrıca güven aralığındaki değerlerden küçük olan değerler için; bu değerlerin arasındaki en küçük ve en büyük değer ile bu değerlerden kaç tane olduğu bilgileri tablodaki ilgili sütunlarda verilmiştir. Güven aralığında yer alan sonuçlar incelendiğinde 12 döngü sonunda bile algoritmanın SAC testinden başarılı olduğu görülmüştür. 12 döngü sonunda her bir anahtar için toplamda 49152 $(= 64 \times 12 \times 64)$

değer elde edilir. 11 farklı n değeri için birbirinden farklı n adet ana anahtar üretilir, algoritma SAC testinden geçirildiğinde 49152 değerden ortalama 46005 tanesinin bu testten geçtiği görülebilir.

Tablo 5.5. Önerilen anahtar planlama algoritması için SAC testi sonuçları

n Değeri	**Güven Aralığı Değeri: γ X: Alt Sınır Y:Üst Sınır** $X \le \gamma \le Y$	**Güven Aralığındaki Değerlerin Sayısı** $X \le \gamma \le Y$	**Üst Sınırdan Büyük Değerlerin Sayısı** $(\gamma > Y)$	**Alt Sınırdan Küçük Değerlerin Sayısı** $(\gamma < X)$	**Alt Sınırdan Küçük Değerler** $\beta = (\gamma < X)$			
					En Küçük β Değeri $\min(\beta)$		**En Büyük β Değeri** $\max(\beta)$	
					Değer	**Sayısı**	**Değer**	**Sayısı**
25	$7{,}5 \le \gamma \le 17{,}5$	47020	1017	1115	4	24	7	724
50	$18 \le \gamma \le 32$	47502	763	887	12	6	17	471
100	$40 \le \gamma \le 60$	47306	949	897	31	8	39	384
250	$109 \le \gamma \le 141$	47290	944	918	94	4	108	224
500	$228 \le \gamma \le 272$	46806	1116	1230	207	6	227	218
750	$348 \le \gamma \le 402$	46290	1343	1519	315	4	347	250
1000	$468 \le \gamma \le 532$	46670	1087	1395	433	4	347	250
2000	$955 \le \gamma \le 1045$	46062	1483	1607	915	4	954	157
3000	$144 \le \gamma \le 1555$	45422	1720	2010	1389	4	1444	132
5000	$2429 \le \gamma \le 2571$	44234	2229	2689	2310	4	2428	129
10000	$4900 \le \gamma \le 5100$	41450	3459	4243	4726	4	4899	110
Ortalama		**46005**	**1465**	**1683**				

80-bit ve 128-bit anahtar planlama algoritmaları için, Algoritma 1 kullanılarak sırasıyla 12 ve 8 adet anahtar üretilir. Bu anahtarlar 64-bitlik parçalara ayrılarak 80-bit için 14, 128-bit için 16 adet döngü anahtarı oluşturulur.

80-bit anahtar planlama algoritması için, ilk anahtarın 64-biti 1. döngü anahtarını, ilk anahtarın son 16-biti ve ikinci anahtarın ilk 48-biti birleştirilerek 2. döngü anahtarı oluşturulur. İkinci anahtarın son 32-biti ve üçüncü anahtarın ilk 32-biti birleştirilerek ise 3. döngü anahtarı oluşturulur. Tüm işlemler benzer şekilde 14. döngü anahtarı oluşturulana kadar devam eder. Son anahtarın (80-bitlik 12. anahtar) ilk 16-biti kullanılır geri kalan 64-biti ise kullanılmaz. 80-bit için 14 adet alt anahtarların üretimi ve anahtar parçalama işlemleri için sözde kod Şekil 5.8'de verilmiştir.

Algoritma 3: 80-bit Anahtar Kullanılarak Alt Anahtarların Oluşturulması

```
PreSubKey [12][8]: Alt anahtarları oluşturacak 64 bitlik 12 elemanlı anahtar dizisi;
SubKey [14][8]: 64 bitlik 14 elemanlı döngü alt anahtarları dizisi;
Algoritma 1'e göre (k0 – k9) olacak şekilde 80-bit 12 adet anahtarı üret ve üretilen
anahtarları PreSubKey [12][8] dizisine ata;
j = 1;
index = 1;
for i = 1 to 3
   SubKey [j][ ] = PreSubKey[index][ ] elemanının ilk 64 biti;
   SubKey [j+1][ ]= birleştir (PreSubKey[index][ ] elemanının son 16 biti,
   PreSubKey [index+1][ ] elemanının ilk 48 biti);
   SubKey [j+2][ ] = birleştir (PreSubKey[index+1][ ] elemanının son 32 biti,
   PreSubKey [index+2][ ] elemanının ilk 32 biti);
   SubKey [j+3][] = birleştir (PreSubKey[index+2] elemanının son 48 biti,
   PreSubKey [index+3][ ] elemanının ilk 16 biti);
     if ( i != 3 )
        SubKey [j+4][ ] = PreSubKey[index+3][ ] elemanının son 64 biti;
     end
j++;
index++;
End
return SubKey [ ][ ];
```

Şekil 5.8. 80-bit anahtar için alt anahtarların oluşturulması sözde kodu

Algoritma 4: 128-bit Anahtar Kullanılarak Alt Anahtarların Oluşturulması

```
PreSubKey [8][8]: Alt anahtarları oluşturacak 64 bitlik 8 elemanlı anahtar dizisi;
SubKey [16][8]: 64 bitlik 16 elemanlı döngü alt anahtarları dizisi;
Algoritma 1'e göre (k0 – k15) olacak şekilde 128-bit 8 adet anahtarı üret ve üretilen
anahtarları PreSubKey [8][8] dizisine ata;
j = 1;
for i = 1 to 8
   SubKey [j][] = PreSubKey[i][ ] elemanının ilk 64 biti;
   SubKey [j+1][]= PreSubKey[i][ ] elemanının son 64 biti;
  j++;
End
return SubKey [][];
```

Şekil 5.9. 128-bit anahtar için alt anahtarların oluşturulması sözde kodu

128-bit anahtar planlama algoritması için, tek bir 128-bit anahtardan iki adet 64-bit alt anahtar üretilir, şöyleki; anahtarın ilk 64-biti ilk döngü anahtarını oluştururken, son 64-biti de ikinci döngü anahtarını oluşturur. 128-bit için 16 adet alt anahtarların üretimi ve anahtar parçalama işlemleri için sözde kod Şekil 5.9'de verilmiştir.

80-bit ve 128-bit anahtarlar için şifreleme işlemleri için ana anahtar ile XOR'lama adımında her iki yapı için de ana anahtarların ilk 64-biti kullanılır.

Literatürde yer alan KLEIN-64 ve PRESENT-80 hafif sıklet blok şifreleme algoritmalarının anahtar planlama yapıları SAC testinden geçirilmiş ancak bu algoritmaların başarısız olduğu görülmüştür. Bu algoritmalara ait SAC testi sonuçları Tablo 5.6'da verilmiştir.

Tablo 5.6. KLEIN-64 ve PRESENT-80 algoritmaları SAC testi sonuçları

n Değeri: **Güven Aralığı Değeri:** γ **X: Alt Sınır** **Y:Üst Sınır** $X \leq \gamma \leq Y$	**KLEIN-64**			**PRESENT-80**		
	$X \leq \gamma \leq Y$	$(\gamma > Y)$	$(\gamma < X)$	$X \leq \gamma \leq Y$	$(\gamma > Y)$	$(\gamma < X)$
25: $7{,}5 \leq \gamma \leq 17{,}5$	2309	1203	45640	5234	2967	190199
50: $18 \leq \gamma \leq 32$	2130	1363	45659	4568	3358	190474
100: $40 \leq \gamma \leq 60$	2038	1445	45669	4231	3492	190677
500: $228 \leq \gamma \leq 272$	1871	1539	45742	3924	3623	190853
Ortalama	2087	1388	45678	4489	3360	190551

Tablo 5.6'dan da görüleceği gibi her iki algoritmanın da anahtar planlama yapısı SAC testinden başarısız olmuştur. KLEIN-64 şifreleme algoritması için 12 döngü sonunda her bir anahtar için toplamda 49152 $(=64\times12\times64)$ tane değer elde edilir. 4 farklı n değeri için birbirinden farklı n adet ana anahtar üretilir, algoritma SAC testinden geçirildiğinde 49152 değerden ortalama 2087 tanesinin bu testten geçtiği görülebilir bu da algoritmanın SAC testi başarısının yaklaşık % 4,25 olduğunu gösterir. PRESENT-80 şifreleme algoritması için, 31 döngü sonunda her bir anahtar için toplamda 198400 $(=80\times31\times80)$ tane değer elde edilir. 4 farklı n değeri için birbirinden farklı n adet ana anahtar üretilir, algoritma SAC testinden geçirildiğinde 198400 değerden ortalama 4489 tanesinin bu testten geçtiği görülebilir bu da algoritmanın SAC testi başarısının yaklaşık % 2,26 olduğunu gösterir.

Önerilen anahtar planlama algoritması çığ etkisi kriterine göre analiz edildiğinde, 64-bit için algoritmanın daha ilk döngüden itibaren çıkış bitlerinin yarısından fazlasını

değiştirdiği görülebilir. Tablo 5.7'de verilen RCon ve $RCon_R$ değerleri için 64-bit (00 01 02 03 04 05 06 07) ana anahtarı kullanılarak alt anahtarlar elde edilmiştir. Anahtar planlama yapısının çığ etkisini ölçmek için ana anahtarın ilk byte dizisinin en önemsiz biti 1-bit değiştirilerek (01 01 02 03 04 05 06 07) ana anahtarı için de alt anahtarlar elde edilmiştir. İlgili döngü için elde edilen anahtarların bit pozisyonları değişimi hesaplanarak her döngüdeki değişen bit sayıları elde edilmiştir.

Tablo 5.7. Önerilen anahtar planlama yapısı için çığ etkisi analizi

Döngü	**$RCon_i$ ve $RCon_{Ri}$ Değerleri**	**Ana Anahtar: '00 01 02 03 04 05 06 07'**	**Ana Anahtar: '01 01 02 03 04 05 06 07'**	**Değişen Bit Sayısı**
		Alt Anahtar	**Alt Anahtar**	
1	7F CE 54 AC 89 01 24 7B B7 42 10 98 CA 45 EC F7	66 EA 00 17 CB AF AE F5	13 00 75 FC BF 45 DB 1E	**41**
2	9E DD 32 01 76 BA 9C F3 3F C9 AB 67 10 23 DD E9	5B D5 AB 6D E8 91 52 7B	B9 00 49 B9 0B 44 B0 AF	**35**
3	F2 03 25 0C 8D 7B 6E 34 43 E6 B7 D8 C0 52 30 2F	06 1F EC E8 78 20 32 78	FB A4 11 52 84 9B CF C2	**49**
4	1D 6B 39 2E 41 3E 62 0C C0 26 E3 14 E2 93 B6 D1	17 CD 77 F0 05 57 22 7C	69 38 09 04 7A A2 5C 88	**47**
5	DD 3E 09 FF 68 3A A5 48 84 5A A3 86 FF 90 E3 DD	3C BA 5C C8 66 23 28 CC	87 E1 E7 92 DC 78 93 96	**41**
6	5B 1F CC 8B 45 6E 2C D9 9D C2 E6 54 B8 CC F1 B5	3A 21 55 28 CC 70 E7 6B	93 51 FC 59 64 00 4E 1A	**29**
7	8C 2B 76 91 BC 6A A2 D1 1D 2A A6 CB 19 67 B2 C8	66 21 30 FF 7C FE 3F CA	4B 46 1D 99 50 99 12 AC	**33**
8	14 DE B8 9F C7 D0 2A FE EF A2 0D 7C F9 8B ED 41	6F 44 26 40 F1 71 A9 CD	03 E8 4A ED 9C DD C5 60	**35**
9	CD 18 8A A3 AA B8 F5 49 94 5F 8B AA 3A A8 81 DC	D3 A3 56 90 10 E3 69 B5	90 DF 15 ED 52 9F 2A C8	**33**
10	5F 6B 7E 56 43 10 D7 3A A3 7D 01 34 65 E7 B6 F5	BF 0E 73 60 73 B9 AD 25	66 40 AA 2F AB F7 74 6A	**37**
11	5E 6A 4E 21 52 34 E4 5B B5 4E 43 25 12 E4 A6 E5	E4 CD 40 8D 3F 5A D8 8D	62 CB C6 8A B8 5C 5E 8A	**23**
12	16 39 D3 41 A3 C2 B2 55 55 2B 2C 3A 14 3D 93 61	7E C5 75 59 6B 4C DC F4	20 EC 2B 71 34 65 82 DC	**31**
Ortalama Bit Değişimi				**36**

AES 128-bit anahtar planlama algoritması için, ana anahtar üzerinde 1 bit değişim ile 2. döngüde 58-biti, 3. döngüde ise 67-biti değiştirdiği gözlemlenmiştir [36], benzer şekilde DES şifreleme algoritması için bitlerin neredeyse yarısının değişimi ancak 3. döngü sonunda elde edilmektedir [36]. AES şifreleme algoritmasının çığ etkisi DES şifreleme algoritmasına göre daha güçlüdür. Ancak önerilen hafif sıklet şifreleme algoritması için geliştirilen anahtar planlama yapısı hem AES hem de DES'den daha güçlü çığ etkisine sahiptir.

5.1.5. Örnek şifreleme adımları ve test vektörleri

64-bit açık metin için 64-bit anahtar kulanılarak önerilen şifreleme algoritması kullanılarak şifreleme ve şifre çözme adımları ayrıntılı olarak aşağıdaki tablolarda verilmiştir.

Açık Metin: E6 59 64 FE 0A C1 43 16
Ana (Master) Anahtar: (75 35 B1 CB C7 33 ED 39)

Tablo 5.8. Örnek şifreleme anahtar üretme adımları

Döngü	**t_i Değerleri (t_8, t_{16}, t_{24})**	**$RCon_i$ ve $RCon_{Ri}$ Değerleri**	**k_i Değerleri**	**Üretilen Alt (Döngü) Anahtarı**
1	66, 53, A2	$RCon_i$ = 61 DC 45 E9 7A E3 95 CC	$k_0,\ldots k_7$ = 14 E9 F4 22 BD D0 78 F5	**5B 89 26 F1 3F D0 F1 1A**
			$k_8,\ldots k_{15}$ = 72 9B 6F 4D F0 20 58 AD	
		$RCon_{Ri}$ = CC 59 3E A7 9E 54 CD 16	$k_{16},\ldots k_{23}$ = 21 BA D5 98 68 48 10 BD	
			$k_{24},\ldots k_{31}$ = 83 39 EC 74 1C 54 44 F9	
2	3D, E9, 5F	$RCon_i$ = 6F CF 6D C0 6E 21 3B C8	$k_0,\ldots k_7$ = 1A FA DC 0B A9 12 D6 F1	**07 CB 5E 65 96 FD 7E A8**
			$k_8,\ldots k_{15}$ = 27 DD 01 0A A3 B1 67 96	
		$RCon_{Ri}$ = 8C B3 12 E6 0C D6 FC F6	$k_{16},\ldots k_{23}$ = CE 13 12 18 BB 0A 6D FB	
			$k_{24},\ldots k_{31}$ = 91 82 90 88 33 39 54 AF	
3	42, 46, EA	$RCon_i$ = E4 FD C9 FB 42 92 47 A7	$k_0,\ldots k_7$ = 91 C8 78 30 85 A1 AA 9E	**94 4D 4D B6 F0 E8 62 46**
			$k_8,\ldots k_{15}$ = D3 1B 63 53 D6 77 DD 43	
		$RCon_{Ri}$ = 7A 74 29 24 BF 9C DF 4E	$k_{16},\ldots k_{23}$ = 95 8E ED BE 68 1F C2 81	
			$k_{24},\ldots k_{31}$ = 7F F1 1C A2 CA D5 17 96	
4	57, 8C, CD	$RCon_i$ = C8 42 26 00 C8 7F 98 75	$k_0,\ldots k_7$ = BD 77 97 CB 0F 4C 75 4C	**41 AE C1 D6 60 3D C9 68**
			$k_8,\ldots k_{15}$ = EA 9D 0A C1 CE 82 F7 BB	
		$RCon_{Ri}$ = 57 89 F7 8C 00 62 24 8C	$k_{16},\ldots k_{23}$ = 66 FB F1 30 FE 7C 8B 30	
			$k_{24},\ldots k_{31}$ = AB 50 A1 91 6F 13 98 A8	
5	76, 31, C1	$RCon_i$ = 84 59 FD 57 5B 30 97 E8	$k_0,\ldots k_7$ = F1 6C 4C 9C 9C 03 7A D1	**08 3F 9F 38 9E 69 46 23**
			$k_8,\ldots k_{15}$ = 87 EB A7 3B A7 A4 DE 0F	

Tablo 5.8. (Devam) Örnek şifreleme anahtar üretme adımları

		$RCon_{Ri}$ = 8E 79 03 B5 75 DF 95 48	$k_{16},\ldots k_{23}$ = B6 5D FA C1 66 C2 1C 13	
			$k_{24},\ldots k_{31}$ = 77 2A D0 11 77 B5 A9 BA	
6	25, 00, A0	$RCon_i$ = 9F 77 C5 79 80 6E CA 5A	$k_0,\ldots k_7$ = EA 42 74 B2 47 5D 27 63	**20 C3 04 DC 4A 36 BC B8**
			$k_8,\ldots k_{15}$ = CF 8D F9 4B 0C 51 76 15	
		$RCon_{Ri}$ = A5 AC E6 08 97 5C 77 F9	$k_{16},\ldots k_{23}$ = CF 42 BB F0 FC AD DB CE	
			$k_{24},\ldots k_{31}$ = 6F 2D 96 66 9A 37 EC 22	
7	10, 0A, 50	$RCon_i$ = 81 56 06 DD D1 80 CB 42	$k_0,\ldots k_7$ = F4 63 B7 16 16 B3 26 7B	**6E 08 31 FA 75 A1 58 CF**
			$k_8,\ldots k_{15}$ = E4 87 30 26 30 83 A5 DE	
		$RCon_{Ri}$ = 24 BC 08 1D DD 60 65 18	$k_{16},\ldots k_{23}$ = EE 69 59 7F 4F CC 69 B7	
			$k_{24},\ldots k_{31}$ = BE D7 8E F1 BE 72 1B AC	
8	A5, 98, CE	$RCon_i$ = C2 4D 49 FB B5 0E E1 49	$k_0,\ldots k_7$ = B7 78 F8 30 72 3D 0C 70	**67 C2 CE 6D CB 42 3F 26**
			$k_8,\ldots k_{15}$ = 12 6A 92 A2 D0 ED E1 91	
		$RCon_{Ri}$ = 94 1E E0 5B BF 94 D4 2C	$k_{16},\ldots k_{23}$ = 8A E0 72 D0 00 ED 0C 9D	
			$k_{24},\ldots k_{31}$ = 44 A4 D6 06 06 EB E7 7A	
9	E0, AD, 8B	$RCon_i$ = EB F4 05 7C B5 60 D5 D5	$k_0,\ldots k_7$ = 9E C1 B4 B7 72 53 38 EC	**9B A8 E1 64 28 16 4F 0E**
			$k_8,\ldots k_{15}$ = 7E BF 0B BC CE 9D A5 49	
		$RCon_{Ri}$ = 5D 5D 06 5B C7 50 4F BE	$k_{16},\ldots k_{23}$ = D3 6C 67 DB 15 88 2D 64	
			$k_{24},\ldots k_{31}$ = 58 34 53 88 9D 15 38 5C	
10	AB, 7A, 64	$RCon_i$ = 63 C4 B2 B7 48 4F EE 61	$k_0,\ldots k_7$ = 16 F1 03 7C 8F 7C 03 58	**A3 37 1B E3 A4 8C DC 2E**
			$k_8,\ldots k_{15}$ = BD 4C 4F 33 BC C0 C3 9B	
		$RCon_{Ri}$ = 16 EE F4 84 7B 2B 4C 36	$k_{16},\ldots k_{23}$ = C7 8B C4 F7 4B 8B 48 D3	
			$k_{24},\ldots k_{31}$ = A3 28 EC 1B 50 DB 93 40	
11	F5, 6A, D9	$RCon_i$ = B3 34 D9 34 F6 F2 C4 72	$k_0,\ldots k_7$ = C6 01 68 FF 31 C1 29 4B	**61 A6 9D DE 3C 47 58 5C**
			$k_8,\ldots k_{15}$ = 33 32 5A A5 94 55 7C 37	
		$RCon_{Ri}$ = 27 4C 2F 6F 43 9D 43 3B	$k_{16},\ldots k_{23}$ = 59 6B 31 94 00 55 29 1E	
			$k_{24},\ldots k_{31}$ = 80 EB DA 4E 4E 1B 32 2C	

Tablo 5.8. (Devam) Örnek şifreleme anahtar üretme adımları

<table>
<tr><td rowspan="4">12</td><td rowspan="4">7A, B8, AF</td><td rowspan="2">$RCon_i$ = 52 DC F1 8C 37 22 D0 A2</td><td>$k_0,\ldots k_7$ = 27 E9 40 47 F0 11 3D 9B</td><td rowspan="4">47 FF DC 9C C5 F4 62 B0</td></tr>
<tr><td>$k_8,\ldots k_{15}$ = 5D B4 F4 B3 43 52 6F F4</td></tr>
<tr><td rowspan="2">$RCon_{Ri}$ = 2A 0D 22 73 C8 1F CD 25</td><td>$k_{16},\ldots k_{23}$ = E5 51 A5 16 55 07 68 9C</td></tr>
<tr><td>$k_{24},\ldots k_{31}$ = 4A 1B BE A8 FD FA 92 0E</td></tr>
</table>

Tablo 5.9. Örnek şifreleme adımları

Döngü	**Adımlar**	
	E6 59 64 FE 0A C1 43 16 ⊕ 75 35 B1 CB C7 33 ED 39 = 93 6C D5 35 CD F2 AE 2F	
1	**SL:**	**15 F0 23 53 02 6D EA D6**
	SR:	$\begin{bmatrix} 1 & 2 & 0 & E \\ 3 & 2 & A & 5 \\ 6 & D & F & 5 \\ 6 & 0 & 3 & D \end{bmatrix}$
	MC:	$\begin{bmatrix} 4 & 3 & 8 & 1 \\ F & 1 & 0 & 1 \\ 0 & C & 6 & C \\ 8 & 5 & 9 & 0 \end{bmatrix}$
	ARK:	**4F 08 31 C5 80 69 11 C0 ⊕ 5B 89 26 F1 3F D0 F1 1A = 14 81 17 34 BF B9 E0 DA**
2	**SL:**	**94 B9 97 54 86 81 AC 2E**
	SR:	$\begin{bmatrix} 9 & 9 & 8 & A \\ 7 & 6 & C & 4 \\ 8 & 2 & B & 5 \\ E & 9 & 4 & 1 \end{bmatrix}$
	MC:	$\begin{bmatrix} C & 2 & E & B \\ 6 & E & A & 8 \\ 3 & F & 1 & B \\ 1 & 3 & C & D \end{bmatrix}$
	ARK:	**C6 31 2E F3 EA 1C B8 BD ⊕ 07 CB 5E 65 96 FD 7E A8 = C1 FA 70 96 7C E1 C6 15**
3	**SL:**	**09 6E 7C 1F 70 A9 0F 93**
	SR:	$\begin{bmatrix} 0 & 7 & 7 & 0 \\ C & 0 & F & 9 \\ A & 9 & 6 & 1 \\ 3 & E & F & 9 \end{bmatrix}$

Tablo 5.9. (Devam) Örnek şifreleme adımları

	MC:	$\begin{bmatrix} 9 & F & 3 & 4 \\ 1 & B & C & 4 \\ 6 & 1 & D & 4 \\ F & 0 & 1 & A \end{bmatrix}$
	ARK:	**91 6F FB 10 3C D1 44 4A ⊕ 94 4D 4D B6 F0 E8 62 46 = 05 22 B6 A6 CC 39 26 0C**
4	**SL:**	**C3 DD 8F EF 00 51 DF C0**
	SR:	$\begin{bmatrix} C & 8 & 0 & D \\ F & 0 & F & 3 \\ 5 & C & D & E \\ 0 & D & F & 1 \end{bmatrix}$
	MC:	$\begin{bmatrix} F & 1 & 1 & 6 \\ 4 & 0 & 1 & 2 \\ 8 & D & 1 & A \\ 2 & 0 & B & 9 \end{bmatrix}$
	ARK:	**F4 82 10 D0 11 1B 62 A9 ⊕ 41 AE C1 D6 60 3D C9 68 = B5 2C D1 06 71 26 AB C1**
5	**SL:**	**83 D0 29 CF 79 DF E8 09**
	SR:	$\begin{bmatrix} 8 & 2 & 7 & E \\ 9 & 9 & 8 & 3 \\ D & 0 & D & C \\ 9 & 0 & F & F \end{bmatrix}$
	MC:	$\begin{bmatrix} 7 & 0 & 9 & 8 \\ 9 & B & 1 & 7 \\ 0 & 6 & B & 4 \\ 7 & 9 & A & F \end{bmatrix}$
	ARK:	**79 07 0B 69 91 BA 87 4F ⊕ 08 3F 9F 38 9E 69 46 23 = 71 38 94 51 0F D3 C1 6C**
6	**SL:**	**79 5B 14 39 C6 25 09 F0**
	SR:	$\begin{bmatrix} 7 & 1 & C & 0 \\ 4 & 6 & 9 & 9 \\ 2 & F & 5 & 3 \\ 0 & B & 9 & 5 \end{bmatrix}$
	MC:	$\begin{bmatrix} 0 & 0 & 0 & A \\ B & 5 & B & 0 \\ 3 & E & C & D \\ 3 & E & 7 & 2 \end{bmatrix}$
	ARK:	**0B 33 05 EE 0B C7 A0 D2 ⊕ 20 C3 04 DC 4A 36 BC B8 = 2B F0 01 32 41 F1 1C 6A**

Tablo 5.9. (Devam) Örnek şifreleme adımları

7	**SL:**	**D8 6C C9 5D 49 69 90 FE**
	SR:	$\begin{bmatrix} D & C & 4 & 9 \\ 9 & 9 & 0 & 8 \\ 6 & F & 6 & 5 \\ E & C & D & 9 \end{bmatrix}$
	MC:	$\begin{bmatrix} A & 9 & A & 1 \\ 1 & 0 & 2 & F \\ 0 & C & B & 1 \\ 2 & 5 & 2 & 2 \end{bmatrix}$
	ARK:	**A1 02 90 C5 A2 B2 1F 12 $\oplus$ 6E 08 31 FA 75 A1 58 CF = CF 0A A1 3F D7 13 47 DD**
8	**SL:**	**06 CE E9 56 27 95 47 22**
	SR:	$\begin{bmatrix} 0 & E & 2 & 4 \\ 9 & 7 & 7 & 6 \\ 9 & 2 & C & 5 \\ 2 & E & 6 & 5 \end{bmatrix}$
	MC:	$\begin{bmatrix} 0 & C & 3 & B \\ 9 & F & 6 & 0 \\ 9 & E & F & E \\ 2 & 9 & B & 0 \end{bmatrix}$
	ARK:	**09 92 CF E9 36 FB B0 E0 $\oplus$ 67 C2 CE 6D CB 42 3F 26 = 6E 50 01 84 FD B9 8F C6**
9	**SL:**	**FA 3C C9 B4 62 81 B6 0F**
	SR:	$\begin{bmatrix} F & C & 6 & B \\ 9 & 2 & 6 & A \\ 8 & 0 & 3 & B \\ F & C & 4 & 1 \end{bmatrix}$
	MC:	$\begin{bmatrix} F & E & D & 8 \\ F & 2 & 8 & A \\ E & 4 & 8 & D \\ 4 & D & 5 & 9 \end{bmatrix}$
	ARK:	**FF E4 E2 4D D8 85 8A D9 $\oplus$ 9B A8 E1 64 28 16 4F 0E = 64 4C 03 29 F0 93 C5 D7**
10	**SL:**	**F4 40 C5 D1 6C 15 03 27**
	SR:	$\begin{bmatrix} F & C & 6 & 0 \\ 5 & C & 3 & 4 \\ 1 & 2 & 4 & D \\ 7 & 0 & 1 & 5 \end{bmatrix}$

Tablo 5.9. (Devam) Örnek şifreleme adımları

	MC:	$\begin{bmatrix} 7 & D & A & 4 \\ 9 & 8 & 7 & 0 \\ E & 4 & B & 1 \\ 6 & A & 4 & 4 \end{bmatrix}$
	ARK:	**79 E6 D8 4A A7 B4 40 14 ⊕ A3 37 1B E3 A4 8C DC 2E = DA D1 C3 A9 03 38 9C 3A**
11	**SL:**	**2E 29 05 E1 C5 5B 10 5E**
	SR:	$\begin{bmatrix} 2 & 0 & C & 1 \\ 5 & 5 & 0 & E \\ 5 & 5 & 2 & E \\ E & 9 & 1 & B \end{bmatrix}$
	MC:	$\begin{bmatrix} 6 & 9 & 0 & 4 \\ E & B & 5 & 9 \\ 6 & A & C & 3 \\ 6 & 9 & 0 & F \end{bmatrix}$
	ARK:	**6E 66 9B A9 05 C0 49 3F ⊕ 61 A6 9D DE 3C 47 58 5C = 0F C0 06 77 39 87 11 63**
12	**SL:**	**C6 0C CF 77 51 B7 99 F5**
	SR:	$\begin{bmatrix} C & C & 5 & 9 \\ F & 1 & 9 & 6 \\ B & F & 0 & 7 \\ 5 & C & 7 & 7 \end{bmatrix}$
	ARK:	**CF B5 C1 FC 59 07 96 77 ⊕ 47 FF DC 9C C5 F4 62 B0 = 88 4A 1D 60 9C F3 F4 C7**
	Şifreli Metin:	**88 4A 1D 60 9C F3 F4 C7**

Tablo 5.10. Örnek şifre çözme adımları

Döngü	**Adımlar**	
	88 4A 1D 60 9C F3 F4 C7 ⊕ 47 FF DC 9C C5 F4 62 B0 = CF B5 C1 FC 59 07 96 77	
1	**invSR:**	$\begin{bmatrix} C & C & 5 & 9 \\ 6 & F & 1 & 9 \\ 0 & 7 & B & F \\ C & 7 & 7 & 5 \end{bmatrix}$
	invSL:	**0F C0 06 77 39 87 11 63**
	ARK:	**0F C0 06 77 39 87 11 63 ⊕ 61 A6 9D DE 3C 47 58 5C = 6E 66 9B A9 05 C0 49 3F**

Tablo 5.10. (Devam) Örnek şifre çözme adımları

	invMC:	$\begin{bmatrix} 2 & 0 & C & 1 \\ 5 & 5 & 0 & E \\ 5 & 5 & 2 & E \\ E & 9 & 1 & B \end{bmatrix}$
2	**invSR:**	$\begin{bmatrix} 2 & 0 & C & 1 \\ E & 5 & 5 & 0 \\ 2 & E & 5 & 5 \\ 9 & 1 & B & E \end{bmatrix}$
	invSL:	**DA D1 C3 A9 03 38 9C 3A**
	ARK:	**DA D1 C3 A9 03 38 9C 3A $\oplus$ A3 37 1B E3 A4 8C DC 2E = 79 E6 D8 4A A7 B4 40 14**
	invMC:	$\begin{bmatrix} F & C & 6 & 0 \\ 5 & C & 3 & 4 \\ 1 & 2 & 4 & D \\ 7 & 0 & 1 & 5 \end{bmatrix}$
3	**invSR:**	$\begin{bmatrix} F & C & 6 & 0 \\ 4 & 5 & C & 3 \\ 4 & D & 1 & 2 \\ 0 & 1 & 5 & 7 \end{bmatrix}$
	invSL:	**64 4C 03 29 F0 93 C5 D7**
	ARK:	**64 4C 03 29 F0 93 C5 D7 $\oplus$ 9B A8 E1 64 28 16 4F 0E = FF E4 E2 4D D8 85 8A D9**
	invMC:	$\begin{bmatrix} F & C & 6 & B \\ 9 & 2 & 6 & A \\ 8 & 0 & 3 & B \\ F & C & 4 & 1 \end{bmatrix}$
4	**invSR:**	$\begin{bmatrix} F & C & 6 & B \\ A & 9 & 2 & 6 \\ 3 & B & 8 & 0 \\ C & 4 & 1 & F \end{bmatrix}$
	invSL:	**6E 50 01 84 FD B9 8F C6**
	ARK:	**6E 50 01 84 FD B9 8F C6 $\oplus$ 67 C2 CE 6D CB 42 3F 26 = 09 92 CF E9 36 FB B0 E0**
	invMC:	$\begin{bmatrix} 0 & E & 2 & 4 \\ 9 & 7 & 7 & 6 \\ 9 & 2 & C & 5 \\ 2 & E & 6 & 5 \end{bmatrix}$

Tablo 5.10. (Devam) Örnek şifre çözme adımları

5	**invSR:**	$\begin{bmatrix} 0 & E & 2 & 4 \\ 6 & 9 & 7 & 7 \\ C & 5 & 9 & 2 \\ E & 6 & 5 & 2 \end{bmatrix}$
	invSL:	**CF 0A A1 3F D7 13 47 DD**
	ARK:	**CF 0A A1 3F D7 13 47 DD ⊕ 6E 08 31 FA 75 A1 58 CF = A1 02 90 C5 A2 B2 1F 12**
	invMC:	$\begin{bmatrix} D & C & 4 & 9 \\ 9 & 9 & 0 & 8 \\ 6 & F & 6 & 5 \\ E & C & D & 9 \end{bmatrix}$
6	**invSR:**	$\begin{bmatrix} D & C & 4 & 9 \\ 8 & 9 & 9 & 0 \\ 6 & 5 & 6 & F \\ C & D & 9 & E \end{bmatrix}$
	invSL:	**2B F0 01 32 41 F1 1C 6A**
	ARK:	**2B F0 01 32 41 F1 1C 6A ⊕ 20 C3 04 DC 4A 36 BC B8 = 0B 33 05 EE 0B C7 A0 D2**
	invMC:	$\begin{bmatrix} 7 & 1 & C & 0 \\ 4 & 6 & 9 & 9 \\ 2 & F & 5 & 3 \\ 0 & B & 9 & 5 \end{bmatrix}$
7	**invSR:**	$\begin{bmatrix} 7 & 1 & C & 0 \\ 9 & 4 & 6 & 9 \\ 5 & 3 & 2 & F \\ B & 9 & 5 & 0 \end{bmatrix}$
	invSL:	**71 38 94 51 0F D3 C1 6C**
	ARK:	**71 38 94 51 0F D3 C1 6C ⊕ 08 3F 9F 38 9E 69 46 23 = 79 07 0B 69 91 BA 87 4F**
	invMC:	$\begin{bmatrix} 8 & 2 & 7 & E \\ 9 & 9 & 8 & 3 \\ D & 0 & D & C \\ 9 & 0 & F & F \end{bmatrix}$
8	**invSR:**	$\begin{bmatrix} 8 & 2 & 7 & E \\ 3 & 9 & 9 & 8 \\ D & C & D & 0 \\ 0 & F & F & 9 \end{bmatrix}$
	invSL:	**B5 2C D1 06 71 26 AB C1**

Tablo 5.10. (Devam) Örnek şifre çözme adımları

	ARK:	**B5 2C D1 06 71 26 AB C1** $\oplus$ **41 AE C1 D6 60 3D C9 68 = F4 82 10 D0 11 1B 62 A9**
	invMC:	$\begin{bmatrix} C & 8 & 0 & D \\ F & 0 & F & 3 \\ 5 & C & D & E \\ 0 & D & F & 1 \end{bmatrix}$
9	**invSR:**	$\begin{bmatrix} C & 8 & 0 & D \\ 3 & F & 0 & F \\ D & E & 5 & C \\ D & F & 1 & 0 \end{bmatrix}$
	invSL:	**05 22 B6 A6 CC 39 26 0C**
	ARK:	**05 22 B6 A6 CC 39 26 0C** $\oplus$ **94 4D 4D B6 F0 E8 62 46 = 91 6F FB 10 3C D1 44 4A**
	invMC:	$\begin{bmatrix} 0 & 7 & 7 & 0 \\ C & 0 & F & 9 \\ A & 9 & 6 & 1 \\ 3 & E & F & 9 \end{bmatrix}$
10	**invSR:**	$\begin{bmatrix} 0 & 7 & 7 & 0 \\ 9 & C & 0 & F \\ 6 & 1 & A & 9 \\ E & F & 9 & 3 \end{bmatrix}$
	invSL:	**C1 FA 70 96 7C E1 C6 15**
	ARK:	**C1 FA 70 96 7C E1 C6 15** $\oplus$ **07 CB 5E 65 96 FD 7E A8 = C6 31 2E F3 EA 1C B8 BD**
	invMC:	$\begin{bmatrix} 9 & 9 & 8 & A \\ 7 & 6 & C & 4 \\ 8 & 2 & B & 5 \\ E & 9 & 4 & 1 \end{bmatrix}$
11	**invSR:**	$\begin{bmatrix} 9 & 9 & 8 & A \\ 4 & 7 & 6 & C \\ B & 5 & 8 & 2 \\ 9 & 4 & 1 & E \end{bmatrix}$
	invSL:	**14 81 17 34 BF B9 E0 DA**
	ARK:	**14 81 17 34 BF B9 E0 DA** $\oplus$ **5B 89 26 F1 3F D0 F1 1A = 4F 08 31 C5 80 69 11 C0**

Tablo 5.10. (Devam) Örnek şifre çözme adımları

	invMC:	$\begin{bmatrix} 1 & 2 & 0 & E \\ 3 & 2 & A & 5 \\ 6 & D & F & 5 \\ 6 & 0 & 3 & D \end{bmatrix}$
12	**invSR:**	$\begin{bmatrix} 1 & 2 & 0 & E \\ 5 & 3 & 2 & A \\ F & 5 & 6 & D \\ 0 & 3 & D & 6 \end{bmatrix}$
	invSL:	**93 6C D5 35 CD F2 AE 2F**
	ARK:	**93 6C D5 35 CD F2 AE 2F ⊕ 75 35 B1 CB C7 33 ED 39 = E6 59 64 FE 0A C1 43 16**
	Açık Metin:	**E6 59 64 FE 0A C1 43 16**

80-bit anahtar için, verilen ana anahtar, RCon ve $RCon_R$ değerleri ile her bir döngü için önerilen 80-bit anahtar planlama algoritması kullanılarak döngü alt anahtarları üretilmiş ve bu anahtarlar ile üretilen açık metin şifreli metin çiftleri için test vektörleri Tablo 5.11'de verilmiştir.

Tablo 5.11. 80-bit için test vektörleri

Ana Anahtar		**00 01 02 03 04 05 06 07 08 09**
Döngü	**$RCon_i$ ve $RCon_{Ri}$ Değerleri**	**Üretilen Anahtar ve Döngü Anahtarı**
1	7F CE 54 AC 89 01 24 7B 36 18	1D 4E C8 6A 5C F7 E7 E0 51 A2 **Döngü Anahtarı: 1D 4E C8 6A 5C F7 E7 E0**
	81 63 B7 42 10 98 CA 45 EC F7	
2	9E DD 32 01 76 BA 9C F3 14 57	92 80 61 C3 D0 08 C1 B1 20 25 **Döngü Anahtarı: 51 A2 92 80 61 C3 D0 08**
	75 41 3F C9 AB 67 10 23 DD E9	
3	F2 03 25 0C 8D 7B 6E 34 AB 2F	B8 20 5F FD 00 C9 85 2E 3D E3 **Döngü Anahtarı: C1 B1 20 25 B8 20 5F FD**
	F2 BA 43 E6 B7 D8 C0 52 30 2F	
4	1D 6B 39 2E 41 3E 62 0C 13 57	C7 1A 7D 4E 61 10 5F B2 7A F5 **Döngü Anahtarı: 00 C9 85 2E 3D E3 C7 1A**
	75 31 C0 26 E3 14 E2 93 B6 D1	
5	DD 3E 09 FF 68 3A A5 48 6B 3A	D7 B7 D3 91 F6 D4 9C 04 95 D3 **Döngü Anahtarı: 7D 4E 61 10 5F B2 7A F5**
	A3 B6 84 5A A3 86 FF 90 E3 DD	
6	5B 1F CC 8B 45 6E 2C D9 49 40	46 0C 6C 62 77 93 EC CE FF 58 **Döngü Anahtarı: D7 B7 D3 91 F6 D4 9C 04**
	04 94 9D C2 E6 54 B8 CC F1 B5	
7	8C 2B 76 91 BC 6A A2 D1 20 C9	1B 94 93 67 3F CF 8C 1C 45 8B **Döngü Anahtarı: 95 D3 46 0C 6C 62 77 93**
	9C 02 1D 2A A6 CB 19 67 B2 C8	

Tablo 5.11. (Devam) 80-bit için test vektörleri

8	14 DE B8 9F C7 D0 2A FE 45 72 27 54 EF A2 0D 7C F9 8B ED 41	6D F8 6A B0 CF CC 13 29 15 69 **Döngü Anahtarı:** **EC CE FF 58 1B 94 93 67**
9	CD 18 8A A3 AA B8 F5 49 B3 71 17 3B 94 5F 8B AA 3A A8 81 DC	9E 61 85 C1 6F 47 1E 65 85 37 **Döngü Anahtarı:** **3F CF 8C 1C 45 8B 6D F8**
10	5F 6B 7E 56 43 10 D7 3A 12 C9 9C 21 A3 7D 01 34 65 E7 B6 F5	87 D6 F1 8A 10 EE 5E 94 DD 20 **Döngü Anahtarı:** **6A B0 CF CC 13 29 15 69**
11	5E 6A 4E 21 52 34 E4 5B 7F AD DA F7 B5 4E 43 25 12 E4 A6 E5	42 C4 5C 75 A7 2B 86 1F 48 AD **Döngü Anahtarı:** **9E 61 85 C1 6F 47 1E 65**
12	16 39 D3 41 A3 C2 B2 55 45 71 17 54 55 2B 2C 3A 14 3D 93 61	41 C1 BA F8 2E 30 ED 18 64 E1 **Döngü Anahtarı:** **85 37 87 D6 F1 8A 10 EE**
13	-	**Döngü Anahtarı:** **5E 94 DD 20 42 C4 5C 75**
14	-	**Döngü Anahtarı:** **A7 2B 86 1F 48 AD 41 C1**
Açık Metin: 28 10 B8 A3 59 C4 81 6F		**Şifreli Metin:** 57 3C 7F 20 B5 F6 EF ED
Açık Metin: 9B A3 05 FC D1 99 6A 3C		**Şifreli Metin:** CE 87 A2 F6 F9 82 F5 B6
Açık Metin: AA 57 D2 6C 31 9E 51 3D		**Şifreli Metin:** ED 94 4E 05 6D 55 26 19
Açık Metin: 70 F4 12 7D A1 87 3B 5C		**Şifreli Metin:** 57 DF 60 72 13 B9 51 BB
Açık Metin: CC 63 BA 41 F7 37 C0 19		**Şifreli Metin:** 80 AE 09 5B 34 34 1A 31

128-bit anahtar için, verilen ana anahtar, RCon ve $RCon_R$ değerleri ile her bir döngü için önerilen 128-bit anahtar planlama algoritması kullanılarak döngü alt anahtarları üretilmiş ve bu anahtarlar ile üretilen açık metin şifreli metin çiftleri için test vektörleri Tablo 5.12'de verilmiştir.

Tablo 5.12. 128-bit için test vektörleri

Ana Anahtar		**00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F**
Döngü	**$RCon_i$ ve $RCon_{Ri}$ Değerleri**	**Üretilen Anahtar ve Döngü Anahtarı**
1	7F CE 54 AC 89 01 24 7B 36 18 CD 18 8A A3 AA B8 8B AA 3A A8 81 DC 81 63 B7 42 10 98 CA 45 EC F7	58 11 76 6A 82 25 9F 2C 45 81 3D 6C 15 11 A8 B4 **Döngü Anahtarı:** **58 11 76 6A 82 25 9F 2C**
2	9E DD 32 01 76 BA 9C F3 14 57 F5 49 B3 71 5F 6B B6 F5 17 3B 94 5F 75 41 3F C9 AB 67 10 23 DD E9	0F 22 F9 C9 B1 26 14 2B 9C 13 0E 06 ED 18 FB 9B **Döngü Anahtarı:** **45 81 3D 6C 15 11 A8 B4**
3	F2 03 25 0C 8D 7B 6E 34 AB 2F 7E 56 43 10 D7 3A A3 7D 01 34 65 E7 F2 BA 43 E6 B7 D8 C0 52 30 2F	10 80 38 94 2B 91 92 7D B7 4D 99 58 CA 90 77 2B **Döngü Anahtarı:** **0F 22 F9 C9 B1 26 14 2B**

Tablo 5.12. (Devam) 128-bit için test vektörleri

4	1D 6B 39 2E 41 3E 62 0C 13 57 5E 6A 4E 21 52 34 43 25 12 E4 A6 E5 75 31 C0 26 E3 14 E2 93 B6 D1	A3 0F A3 D3 76 E0 C4 4F 5E 03 68 35 06 AA 45 CD **Döngü Anahtarı:** **9C 13 0E 06 ED 18 FB 9B**
5	DD 3E 09 FF 68 3A A5 48 6B 3A 16 39 D3 41 A3 C2 2C 3A 14 3D 93 61 A3 B6 84 5A A3 86 FF 90 E3 DD	46 A6 BA 8C D8 AE 39 58 EC C9 A9 3B C6 EF B9 FF **Döngü Anahtarı:** **10 80 38 94 2B 91 92 7D**
6	5B 1F CC 8B 45 6E 2C D9 49 40 B2 55 45 71 80 72 27 08 17 54 55 2B 04 94 9D C2 E6 54 B8 CC F1 B5	85 6B 14 1D 24 17 A2 7F 73 D4 23 D9 49 DA F0 E8 **Döngü Anahtarı:** **B7 4D 99 58 CA 90 77 2B**
7	8C 2B 76 91 BC 6A A2 D1 20 C9 20 BA 37 84 67 12 21 76 48 73 AB 02 9C 02 1D 2A A6 CB 19 67 B2 C8	D7 CB 12 B0 43 2D DD F7 9B 42 83 C2 06 F4 34 19 **Döngü Anahtarı:** **A3 0F A3 D3 76 E0 C4 4F**
8	14 DE B8 9F C7 D0 2A FE 45 72 2F 6B 18 44 8C 39 93 C8 44 81 B6 F2 27 54 EF A2 0D 7C F9 8B ED 41	**AE DC E3 7E 03 FA EF 1B 60 32 92 EF 5B 74 1E 30** **Döngü Anahtarı:** **5E 03 68 35 06 AA 45 CD**
9	-	**Döngü Anahtarı:** **46 A6 BA 8C D8 AE 39 58**
10	-	**Döngü Anahtarı:** **EC C9 A9 3B C6 EF B9 FF**
11	-	**Döngü Anahtarı:** **85 6B 14 1D 24 17 A2 7F**
12	-	**Döngü Anahtarı:** **73 D4 23 D9 49 DA F0 E8**
13	-	**Döngü Anahtarı:** **D7 CB 12 B0 43 2D DD F7**
14	-	**Döngü Anahtarı:** **9B 42 83 C2 06 F4 34 19**
15	-	**Döngü Anahtarı:** **AE DC E3 7E 03 FA EF 1B**
16	-	**Döngü Anahtarı:** **60 32 92 EF 5B 74 1E 30**
Açık Metin: 28 10 B8 A3 59 C4 81 6F		**Şifreli Metin:** 02 2C 62 8E CF 6F AA 55
Açık Metin: 9B A3 05 FC D1 99 6A 3C		**Şifreli Metin:** A0 1D 94 F3 84 31 CC 28
Açık Metin: AA 57 D2 6C 31 9E 51 3D		**Şifreli Metin:** 0E C0 3A E9 22 BB A6 4F
Açık Metin: 70 F4 12 7D A1 87 3B 5C		**Şifreli Metin:** B4 B6 91 FF C6 68 9B B6
Açık Metin: CC 63 BA 41 F7 37 C0 19		**Şifreli Metin:** B1 5F 33 9A 5D FA 0F 05

5.1.6. Önerilen hafif sıklet blok şifreleme algoritmasının 8-bit mikrodenetleyici üzerinde yazılım performansı analizi

Atmel Studio 7.0 platformu kullanılarak AVR ATmega128 8-bit mikrodenetleyici üzerinde, geliştirilen hafif sıklet şifreleme algoritmasının yazılım performansı test edilmiştir. Bu mikrodenetleyici düşük-maliyetli ekonomik cihazlardan biridir ve ultra

kaynak kısıtlı mikrodenetleyicilere oranla biraz daha iyi performans gösterir bu nedenle özellikle hafif sıklet uygulamalar için tercih edilen 8-bit mikrodenetleyicilerden biridir [58].

Önerilen hafif sıklet şifreleme algoritmasının dışında, literatürde yer alan farklı mimarilerde tasarlanmış hafif sıklet blok şifreleme algoritmaları ile blok şifreleme algoritmalarının da ATmega128 üzerinde yazılım performansları ölçülmüştür. Karşılaştırmalar, Intel Core i7-2670QM 2.5 GHz işlemci ve Windows 7 Home Premium x64 ortamında gerçekleştirilmiştir. Tüm algoritmalar için saat frekansı (clock frequency) 16 MHz'dir. mCrypton-64 [106], HIGHT-128, PRESENT-80 (alan optimize edilmiş), PRESENT-80 (hız optimize edilmiş), TWINE-80 [107], LED-64, LBlock [108], KLEIN-80, SIMON 64/96, SPECK 64/96 hafif sıklet blok şifreleri ile NOEKEON [109] ve AES-128 blok şifreleri seçilmiştir. Performans karşılaştırmaları, tek bir açık metin bloğunu şifrelemek için gereken saat çevrimi (gecikme) ve byte başına düşen saat çevirimi sayısı metrikleri kullanılarak yapılmıştır. Gecikme cycle, byte başına düşen saat çevrimi cycle/byte olarak ifade edilir. Şifreleme, şifre çözme ve anahtar planlama yapıları için karşılaştırma sonuçları Tablo 5.13'de verilmiştir.

Tablo 5.13. AVR ATmega128 mikrodenetleyici üzerinde yazılım performans karşılaştırmaları

Şifre	**Mimari**	**Blok/ Anahtar**	**Şifreleme**		**Şifre Çözme**		**Anahtar Planlama (cycle)**
			Gecikme (cycle)	**Byte Başına Düşen Saat Çevrimi (cycle/byte)**	**Gecikme (cycle)**	**Byte Başına Düşen Saat Çevrimi (cycle/byte)**	
AES-128	SPN	128/128	25237	1577	28160	1760	15227
NOEKEON	SPN	128/128	51885	3243	52175	3261	-
mCrypton-64	SPN	64/64	35752	4469	35738	4467	2242
HIGHT-128	GFN	64/128	9178	1147	8600	1075	2527
PRESENT-80 (Alan Opt.)	SPN	80/64	171123	21390	154455	19307	10168
PRESENT-80 (Hız Opt.)	SPN	80/64	157064	19633	162027	20253	9083
TWINE-80	GFN	80/64	26343	3293	26341	3293	4802
LED-64	SPN	64/64	336861	42108	337550	42194	19149
LBlock	Feistel	64/80	7222	903	7309	914	6645
KLEIN-80	SPN	64/80	11348	1419	13246	1656	7878

Tablo 5.13. (Devam) AVR ATmega128 mikrodenetleyici üzerinde yazılım performans karşılaştırmaları

SIMON-96	Feistel	64/96	12976	1622	13115	1639	11432
SPECK-96	ARX	64/96	7925	991	7933	992	7729
Önerilen Algoritma-64	SPN	64/64	34285	4286	45046	5631	17928
Önerilen Algoritma-80	SPN	64/80	44476	5560	56484	7061	18744
Önerilen Algoritma-128	SPN	64/128	57342	7168	72413	9052	20984

Tablo 5.13'de kullanılan şifreleme algoritmaları tasarım hedeflerine göre sınıflandırıldığında SIMON 64/96, HIGHT-128, TWINE-80, PRESENT-80, mCrypton-64 şifreleme algoritmaları donanım, AES-128, SPECK 64/96 yazılım, KLEIN-80, LED-64, LBlock, NOEKEON şifreleme algoritmaları hem yazılım hem donanım hedefli şifrelerdir. Kullanılan şifreleme algoritmaları şifreleme ve şifre çözme işlemleri için gecikme metriği kullanılarak karşılaştırılmıştır. Gecikmeden kasıt tek bir açık metin bloğunu şifrelemek/şifre çözmek için gereken çevrim sayısıdır ve cycle olarak ifade edilir. Ayrıca bu işlemler için byte başına düşen çevrim sayısı da tabloda hesaplanmıştır. Aynı şifreleme algoritmalarına ait anahtar planlama yapıları için gereken çevrim sayıları cycle olarak son sütunda verilmiştir. Tablo 5.13'den de görüleceği gibi önerilen hafif sıklet blok şifreleme algoritması için, şifreleme ve şifre çözme işlemlerinde gecikme metriği dikkate alındığında, önerilen algoritma 64-bit anahtar için mCrypton, NOEKEON, PRESENT ve LED şifreleme algoritmalarından daha iyi performans göstermektedir. Özellikle PRESENT şifreleme algoritmasının ISO/IEC standardına sahip bir algoritma olduğu düşünülürse önerilen şifreleme algoritmasının başarısı hafif sıklet şifreler için kabul edilebilir optimum yazılım performansına sahiptir. NOEKEON şifreleme algoritması her döngüde aynı döngü anahtarını kullandığından, bu şifre anahtar planlama algoritması kullanmaz bu nedenle tabloda bu algoritma için ilgili alan boş bırakılmıştır. Anahtar planlama algoritmaları için gereken saat çevrim kıyaslandığında, önerilen hafif sıklet şifreleme algoritması 64-bit için LED ve KLEIN şifreleme algoritmalarından daha iyi performans gösterirken, AES şifreleme algoritması ile yakın çevrim sayısına sahiptir. Önerilen anahtar planlama algoritmasının çığ ve SAC testleri başarısı göz önünde bulundurulduğunda AES şifreleme algoritması ile arasındaki 2701 saat çevrimi farkı ihmal edilebilir seviyededir. PRESENT şifreleme algoritması anahtar planlama yapısı için daha az

saat çevrimi gerektirse de Tablo 5.6'da da verildiği gibi bu algortima güçlü bit yayılımına sahip değildir. Bu bakımdan önerilen hafif sıklet şifreleme algoritmasının anahtar planlama yapısının da 8-bit ATmega128 mikrodenetleyici üzerinde kabul edilebilir yazılım performansına sahip olduğu açıktır.

5.1.7. Önerilen hafif sıklet blok şifreleme algoritmasının WeMos D1 ESP8266 tabanlı WiFi Modül kartı üzerinde performans analizi

WeMos D1 ESP8266 tabanlı Arduino WiFi Modül kartı günümüzde özellikle IoT uygulamaları için kullanılan düşük maliyetli kartlardan biridir. Tez kapsamında bu kartın seçilmesinin en önemli nedenlerinden biri düşük maliyetli, 4 MB bellek kapasitesine sahip olmasıdır. Bu nedenle 128, 256, 512 ve 1024 byte boyutunda açık metinler için performans analizi yapılmıştır. Analizler, Intel Core i7-2670QM 2.5 GHz işlemci ve Windows 7 Home Premium x64 ortamında gerçekleştirilmiştir.

Literatürde yer alan çalışmalar incelendiğinde hafif sıklet blok/blok şifrelerin performans analizlerinin bazı senaryolar altında gerçekleştirildiği görülür [72, 110]. Bu tez kapsamında iki adet WeMos D1 ESP8266 tabanlı IoT kartları üzerinde kimlik doğrulaması sağlayan bir senaryo geliştirilmiştir. Senaryoda hassas (gizli) veriler hafif sıklet blok/blok şifre kullanılarak istemci olarak belirlenen IoT kart üzerinde JAMBU (v2.1) modu ile şifrelenir ve sunucu olarak belirlenen IoT karta WiFi üzerinden gönderilir. Sunucu ilgili şifreleme algoritmasını kullanarak eğer etiket bilgisi doğru ise şifrelenmiş verileri çözerek açık veriye ulaşır. Şifreleme işlemi için AES-128 blok şifresi, HIGHT-128, PRESENT-80, SIMON 64/96 ve SPECK 64/96 hafif sıklet blok şifreleri kullanılmıştır. AES hafif sıklet kullanım için tasarlanmamıştır ancak AES tabanlı kimlik doğrulamaya ihtiyaç duyulduğunda AES-JAMBU diğer AE şifreleme modlarına oranla daha az durum artışı sağladığından bu tür uygulamalar için kullanımı uygundur [77]. Her bir şifreleme algoritması için döngü anahtarları anahtar planlama yapısı ile hesaplanır ve şifre çözme işleminde kullanılmak üzere belleğe kaydedilir yani sunucunun da ana anahtarı bildiği varsayılır ve bu nedenle anahtar paylaşımı için ek mekanizma geliştirilmemiştir. Her bir algoritmada kullanılan parametreler sırasıyla:

- Önerilen Algoritma-JAMBU 64: 64-bit anahtar, 32-bit IV, 32-bit etiket, 96-bit durum biti,
- Önerilen Algoritma-JAMBU 80: 80-bit anahtar, 32-bit IV, 32-bit etiket, 96-bit durum biti,
- Önerilen Algoritma-JAMBU 128: 128-bit anahtar, 32-bit IV, 32-bit etiket, 96-bit durum biti,
- AES-JAMBU 128: 128-bit anahtar, 64-bit IV, 64-bit etiket, 192-bit durum biti,
- HIGHT-JAMBU 128: 128-bit anahtar, 32-bit IV, 32-bit etiket, 96-bit durum biti,
- PRESENT-JAMBU 80: 80-bit anahtar, 32-bit IV, 32-bit etiket, 96-bit durum biti,
- SIMON-JAMBU 64/96: 96-bit anahtar, 32-bit IV, 32-bit etiket, 96-bit durum biti,
- SPECK -JAMBU 64/96: 96-bit anahtar, 32-bit IV, 32-bit etiket, 96-bit durum biti

Her bir algoritma için kullanılan AD (1024 byte) aynıdır ve algoritma kodunun içinde sabit olarak tanımlanmıştır. Tüm algoritmalar için AD, algoritmanın blok uzunluğunun yarısı boyutunda bloklara bölünerek işlenir. Ayrıca her algoritma için şifrelenecek 128, 256, 512 ve 1024 byte uzunluğundaki açık metinler aynı olup, algoritmaların kodlarının içinde sabit olarak tanımlanmıştır. Anahtar planlama yapısında döngü sabiti kullanan algoritmalar için döngü sabitleri de algoritmaların kodları içinde sabit olarak tanımlanmıştır (örn. AES, önerilen algoritmadaki $RCon_i$ ve $RCon_{Ri}$).

Belirlenen senaryo altında, farklı boyutlardaki açık metinler üzerinde altı farklı şifreleme algoritması için istemci olarak belirlenen IoT kartında şifreleme işlemi gerçekleştirilmiştir. Tablo 5.14'de, elde edilen sonuçlar JAMBU modunda kullanılan ilklendirme, AD'nin işlenmesi, şifreleme ve etiket üretimi adımları için gereken çalışma süreleri (ms) karşılaştırılmıştır.

Tablo 5.14. İstemci olarak belirlenen WeMos D1 ESP8266 IoT kartı üzerinde JAMBU modunda farklı şifreleme algoritmalarının şifreleme işlemi için gereken çalışma süreleri

Şifre	Şifrelenecek Veri Boyutu (byte)	İlklendirme Adımı (ms)	AD'nin İşlenmesi Adımı (ms)	Şifreleme Adımı (ms)	Etiket Üretimi Adımı (ms)
Önerilen Algoritma-JAMBU 64	128	0,464	8,807	8,838	0,606
Önerilen Algoritma-JAMBU 80		0,506	10,117	10,141	0,688

Tablo 5.14. (Devam) İstemci olarak belirlenen WeMos D1 ESP8266 IoT kartı üzerinde JAMBU modunda farklı şifreleme algoritmalarının şifreleme işlemi için gereken çalışma süreleri

Önerilen Algoritma-JAMBU 128		0,546	11,417	11,449	0,707
AES-JAMBU 128		0,38	4,92	7,893	0,42
HIGHT-JAMBU 128		0,116	1,823	1,824	0,125
PRESENT-JAMBU 80		1,556	46,523	46,507	2,921
SIMON-JAMBU 64/96		0,034	0,348	0,348	0,33
SPECK -JAMBU 64/96		0,022	0,213	0,215	0,021
Önerilen Algoritma-JAMBU 64	256	0,465	8,810	17,617	0,606
Önerilen Algoritma-JAMBU 80		0,506	10,117	20,23	0,688
Önerilen Algoritma-JAMBU 128		0,547	11,417	22,844	0,770
AES-JAMBU 128		0,386	4,923	9,012	0,42
HIGHT-JAMBU 128		0,113	1,823	3,635	0,125
PRESENT-JAMBU 80		1,556	46.549	93.010	2.933
SIMON-JAMBU 64/96		0,035	0,348	0,684	0,33
SPECK -JAMBU 64/96		0,022	0,214	0,416	0,021
Önerilen Algoritma-JAMBU 64	512	0.465	8.811	35.218	0.607
Önerilen Algoritma-JAMBU 80		0.506	10.118	40.433	0.688
Önerilen Algoritma-JAMBU 128		0.546	11.423	45.667	0.77
AES-JAMBU 128		0,388	4,93	12,043	0,423
HIGHT-JAMBU 128		0.113	1.823	7.261	0.125
PRESENT-JAMBU 80		1.556	46.523	186.071	2.909
SIMON-JAMBU 64/96		0.035	0.348	1.355	0.33
SPECK -JAMBU 64/96		0.023	0.214	0.819	0.02
Önerilen Algoritma-JAMBU 64	1024	0.465	8.811	70.365	0.607
Önerilen Algoritma-JAMBU 80		0.506	10.118	80.827	0.688
Önerilen Algoritma-JAMBU 128		0.546	11.422	91.276	0.771
AES-JAMBU 128		0,39	4,939	16,993	0,427
HIGHT-JAMBU 128		0.116	1.823	14.505	0.125
PRESENT-JAMBU 80		1.556	46.523	372.037	2.912
SIMON-JAMBU 64/96		0.035	0.348	2.697	0.33
SPECK -JAMBU 64/96		0.019	0.213	1.626	0.02

Tablo 5.15'de sunucu IoT kartı üzerinde istemci IoT kartından gelen şifreli metinler için JAMBU modu kullanılarak, ilklendirme, AD'nin işlenmesi, şifre çözme ve etiket doğrulama adımları için gereken çalışma süreleri (ms) karşılaştırılmıştır.

Tablo 5.15. Sunucu olarak belirlenen WeMos D1 ESP8266 IoT kartı üzerinde JAMBU modunda farklı şifreleme algoritmalarının şifreleme işlemi için gereken çalışma süreleri

Şifre	Şifre Çözülecek Veri Boyutu (byte)	İlklendirme Adımı (ms)	AD'nin İşlenmesi Adımı (ms)	Şifreleme Adımı (ms)	Etiket Doğrulama Adımı (ms)
Önerilen Algoritma-JAMBU 64	128	0,459	8,806	8,875	0,612
Önerilen Algoritma-JAMBU 80		0,500	10,112	10,181	0,692
Önerilen Algoritma-JAMBU 128		0,541	11,417	11,486	0,774
AES-JAMBU 128		0,381	4,92	7,901	0,42
HIGHT-JAMBU 128		0,113	1,823	1,828	0,126
PRESENT-JAMBU 80		1,556	46,533	46,5	2,926
SIMON-JAMBU 64/96		0,34	0,347	0,353	0,37
SPECK -JAMBU 64/96		0,015	0,209	0,214	0,025
Önerilen Algoritma-JAMBU 64	256	0,459	8,802	17,69	0,611
Önerilen Algoritma-JAMBU 80		0,502	10,137	20,306	0,693
Önerilen Algoritma-JAMBU 128		0,541	11,412	22,929	0,774
AES-JAMBU 128		0,388	4,923	9,113	0,42
HIGHT-JAMBU 128		0,116	1,823	3,64	0,13
PRESENT-JAMBU 80		1,556	46,519	93,038	2,937
SIMON-JAMBU 64/96		0,34	0,347	0,69	0,37
SPECK -JAMBU 64/96		0,019	0,21	0,416	0,03
Önerilen Algoritma-JAMBU 64	512	0,458	8,807	35,347	0,611
Önerilen Algoritma-JAMBU 80		0,5	10,112	40,56	0,693
Önerilen Algoritma-JAMBU 128		0,541	11,416	45,789	0,775
AES-JAMBU 128		0,38	4,93	12,10	0,424
HIGHT-JAMBU 128		0,116	1,823	7,265	0,126
PRESENT-JAMBU 80		1,557	46,519	186,082	2,913
SIMON-JAMBU 64/96		0,34	0,347	1,364	0,37
SPECK -JAMBU 64/96		0,014	0,21	0,819	0,017

Tablo 5.15. (Devam) Sunucu olarak belirlenen WeMos D1 ESP8266 IoT kartı üzerinde JAMBU modunda farklı şifreleme algoritmalarının şifreleme işlemi için gereken çalışma süreleri

Önerilen Algoritma-JAMBU 64	1024	0,459	8,804	70,607	0,61
Önerilen Algoritma-JAMBU 80		0,5	10,115	80,913	0,707
Önerilen Algoritma-JAMBU 128		0,541	11,414	91,306	0,791
AES-JAMBU 128		0,39	4,939	16,94	0,428
HIGHT-JAMBU 128		0,113	1,823	14,512	0,126
PRESENT-JAMBU 80		1,556	46,519	372,007	2,916
SIMON-JAMBU 64/96		0,3	0,347	2,714	0,37
SPECK -JAMBU 64/96		0,015	0,211	1,629	0,025

Tablo 5.14 ve Tablo 5.15'de elde edilen sonuçlar analiz edildiğinde SPECK-JAMBU 64/96 ve SIMON-JAMBU 64/96 hafif sıklet şifreleme algoritmaları en iyi performansa sahiptir. Literatürde yapılan diğer çalışmalar [4, 5, 58, 72, 106] incelendiğinde SPECK ve SIMON şifreleme algoritmalarının yüksek performans gösterdiği açıktır. [77]'de verilen çalışmada AES-JAMBU 128 şifreleme algoritması için tüm döngü anahtarları sabit tanımlanmış bu nedenle algoritmanın yazılım performansı byte başına düşen çevrim sayısı metriği için SIMON-JAMBU 64/96'dan daha iyi performans göstermiştir. Tez kapsamında yapılan karşılaştırmada kullanılan şifreleme algoritmaları için döngü anahtarları, anahtar planlama yapıları kullanılarak hesaplanmıştır, sabit tanımlanmamıştır. Bu nedenle AES-JAMBU 128 şifreleme algoritmasının performansı SIMON-JAMBU 64/96'dan daha düşük çıkmıştır. [77]'de kullanılan AES-JAMBU 128 şifreleme algoritması WeMos D1 ESP8266 tabanlı IoT kartları üzerine tez kapsamında belirlenen senaryo altında istemci ve sunucu yapısında uygulandığında; 128, 256, 512 ve 1024 byte uzunluğundaki açık metinler için şifreleme süreleri sırasıyla 0,18, 0,345, 0,673 ve 1,33 ms'dir. Şifre çözme süreleri ise sırasıyla 0,186, 0,351, 0,682 ve 1,342 ms'dir. Elde edilen bu değerler [77]'de kullanılan AES-JAMBU 128 algoritmasının, SIMON-JAMBU 64/96 şifreleme algoritmasının şifreleme ve şifre çözme sürelerinden (Tablo 5.14 ve Tablo 5.15) daha iyi bir performans gösterdiğini doğrular, bu nedenle elde edilen sonuç [77]'de verilen performans sonucuyla örtüşür. HIGHT-JAMBU 128 SPECK ve SIMON şifreleme algoritmalarından sonra iyi performans gösteren algoritmadır. 128 byte uzunluğundaki açık metin bloklarının şifreleme ve şifre çözme adımları için

Önerilen Algoritma-JAMBU 64 ve AES-JAMBU 128'in birbirlerine yakın performans gösterdiği görülebilir. PRESENT-JAMBU 80 şifreleme algoritmasının diğer algoritmalara kıyasla tüm işlem adımları için çok yüksek işlem süresi gerektirdiği görülür. Standartlaştırılmış bir algoritma olmasına rağmen PRESENT şifreleme algoritmasının belirlenen senaryo için kimlik doğrulamalı şifrelemenin gerçekleştirildiği IoT kart üzerinde de iyi performans göstermediği açıktır.

Elde edilen sonuçlar analiz edildiğinde SPECK-JAMBU 64/96 ve SIMON-JAMBU 64/96 şifreleme algoritmaları yüksek performans sağlarken, SPECK ve SIMON şifreleme algoritmalarının her ikisinin de birkaç döngüden sonra diferansiyel [111] kriptanalize ve SIMON'un doğrusal kriptanalize [112] karşı savunmasız olduğunu gösteren çalışmalar dikkate alındığında, gizliliğin ve kimlik doğrulamanın gerekli olduğu durumlarda bu algoritmaların kullanımı güvenlik açıklarına neden olabilir. HIGHT-JAMBU 128 şifreleme algoritması da yüksek performans sağlar, ancak HIGHT için de farklı kriptanaliz yöntemlerinin uygulandığı ve önemli güvenlik zafiyetlerinin ortaya çıkarıldığı çalışmalar [113-116] mevcuttur. AES şifreleme algoritması için kriptanaliz yöntemleriyle kanıtlanan güvenlik zafiyetletinin yanı sıra hafif sıklet yapılar için uygun olmadığından hem hafif sıklet uygulama hem de kimlik doğrulamanın gerekli olduğu senaryolarda tercih edilmez. PRESENT-JAMBU 80 hafif sıklet şifreleme algoritması belirlenen senaryo altınnda kötü performans göstermiştir bu nedenle bu algoritmanın kullanılmasının uygun olmadığı çok açıktır. İstemci sunucu mimarisi kullanılan iki IoT cihaz üzerinde, kimlik doğrulamanın gerekli olduğu güvenli iletişimin sağlanacağı hafif sıklet uygulamalar için, tez kapsamında önerilen hafif sıklet blok şifreleme algoritmasının kullanılması hem performans hem de güvenlik açısından uygun olacaktır. Ayrıca JAMBU modunda hafif sıklet blok şifreler kullanılarak, istemci sunucu mimarisinde iki IoT cihaz arasında güvenli iletişimin sağlandığı senaryo literatürde daha önce uygulanmamış bir yapı olduğundan çalışma bu yönüyle de özgünlük taşımaktadır.

5.1.8. Önerilen hafif sıklet blok şifreleme algoritmasının doğrusal ve diferansiyel kriptanalize karşı gücünün test edilmesi

Önerilen hafif sıklet şifreleme algoritmasında kullanılan özgün S-kutunun maksimum doğrusal ve diferansiyel olasılıkları 2^{-2} olarak hesaplanmış ayrıntılar

Bölüm 5.1.1'de verilmiştir. Önerilen hafif sıklet şifreleme algoritması AES şifreleme algoritması ile benzer yapıda satırların ötelenmesi dönüşümüne sahip olduğundan, önerilen algoritma için de 4 döngü sonunda 25 aktif S-kutusu elde edilir. Önerilen hafif sıklet şifreleme algoritmasının 4 döngü sonunda maksimum doğrusal ve diferansiyel olasılıkları $\left(2^{-2}\right)^{25} = 2^{-50}$'dir. Ancak bu değer önerilen algoritmanın 64-bit, 80-bit ve 128-bit anahtar uzunlukları için SPN mimarisinde tasarlanmış 2r döngüye sahip bu blok şifre için sağlanması gereken diferansiyel karakteristiklerinin maksimum olasılıkları $p_D^{2r} \le \left(2^{-2}\right)^{\text{2r döngüdeki aktif S–kutusu sayısı}}$ ve doğrusal karakteristiklerinin maksimum olasılıkları $q_L^{2r} \le \left(2^{-2}\right)^{\text{2r döngüdeki aktif S–kutusu sayısı}}$ değerleri $p_D^{2r} \le 2^{-\text{AnahtarUzunluğu}}, q_L^{2r} \le 2^{-\text{AnahtarUzunluğu}}$ sınırında değildir. Örneğin 64-bit anahtar için 4 döngüden oluşan şifre için $2r = 4$ olur buradan $p_D^4 = \left(2^{-2}\right)^{25} = 2^{-50}$ ancak $2^{-50} \le 2^{-64}$ eşitliği sağlanmaz. 64- bit anahtar için 8 döngüde 50 aktif S-kutusu elde edilir buradan $2^{-100} \le 2^{-64}$ olduğundan, 8 döngüde diferansiyel ve doğrusal kriptanalize karşı dayanlıklıdır denir ancak ilişkili, imkansız, Boomerang, kesik gibi önemli diferansiyel kriptanaliz yöntemlerine karşı da direnç sağlamak için 4 döngü daha eklenerek (güven marjı) 64-bit anahtar için önerilen hafif sıklet şifreleme algoritması için döngü sayısı 12 olarak belirlenmiştir. 80-bit anahtar için, 8 döngüde $2^{-100} \le 2^{-80}$ sınırına ulaşılır ancak döngü sayısı belirtilen kriptanaliz yöntemlerine karşı dayanıklılık için 14 döngü olarak belirlenmiştir. 128-bit anahtar için de 12 döngüde $2^{-150} \le 2^{-128}$ sınırına ulaşır, bu yapı için de belirtilen nedenlerden dolayı döngü sayısı 16 olarak belirlenmiştir.

Şifrenin kullanılabilir hale gelmesi için belirtilen saldırılara karşı da analizinin yapılması önemli ve gereklidir.

6. SONUÇLAR VE ÖNERİLER

Tez çalışmasında özellikle hafif sıklet blok şifrelerin yayılım tabakalarında maksimum yayılımın sağlanması ve çığ etkisinin güçlendirilmesi amaçlı kullanılabilecek, minimum XOR sayısıyla gerçeklenebilen (tersi kendisine eşit) MDS matrislerin üretilmesini sağlayan, GHadamard olarak adlandırılan yeni bir matris formu verilmiştir.

Tersi kendisine eşit MDS matrislerin aranması literatürdeki açık problemlerden biridir çünkü bu matrislerin aranacağı uzay çok büyüktür. Tersi kendisine eşit MDS matrislerin üretilmesi için, rastgele üretim yöntemi ve Hadamard matrislerin kullanımı dışında literatürde bilinen çok fazla yöntem yoktur. Ancak hem rastgele üretim yöntemi hem de Hadamard matris formunun (bu formla her ne kadar arama uzayı küçültülse bile) kullanımı arama maliyeti gerektirir çünkü bu yapılarla aranan özellikteki tersi kendisine eşit MDS matrisler doğrudan üretilemez. Literatürdeki çalışmalar incelendiğinde, var olan özel matris formları kullanılarak arama uzayını küçültme amaçlı, bazı arama yöntemlerinin geliştirildiği görülmüştür. Ancak arama maliyetini tamamen ortadan kaldıran diğer bir deyişle doğrudan bu matrislerin üretilebildiği yapıların tasarımını amaçlayan hibrit bir yönteme rastlanmamıştır.

GHadamard formu literatürdeki (tersi kendisine eşit) MDS matrislerin üretilmesi için tasarlanan üretim yöntemlerinden farklı olarak, hibrit bir tasarım yöntemi sunar. Form, merkezine özel matris formlarından (tersi kendisine eşit) Hadamard matrisleri alırken bu matrislerden arama yapmadan GHadamard formda yeni (tersi kendisine eşit) MDS matrisler üretir. Bu yapısıyla GHadamard formu, literatürde (tersi kendisine eşit) MDS matrislerin üretimi için tasarlanmış tek hibrit üretim yöntemidir. Hadamard matris formuyla 4×4 boyutunda GF(2^4) ve GF(2^8) sonlu cisimleri üzerinde sırasıyla 1512 ve 15,937,992 tane tersi kendisine eşit MDS matris üretilebiliyorken, GHadamard matris formuyla 4×4 boyutunda GF(2^4) üzerinde $1512\cdot(2^4-1)^3 \simeq 2^{22,28}$, GF(2^8) üzerinde $15,937,992\cdot(2^8-1)^3 \simeq 2^{47,90}$ tane tersi kendisine eşit MDS matris üretilir. GHadamard formu ile arama yapmadan

üretilebilecek tersi kendisine eşit MDS matris sayısı oldukça yüksektir. GHadamard matris formu ile üretilebilecek sayıda tersi kendisine eşit MDS matris sayısı göz önünde bulundurulduğunda, Hadamard matris formu ile üretilebilecek matris sayısının son derece sınırlı olduğu açıktır. Ayrıca, bu kadar çok sayıda tersi kendisine eşit MDS matrisin, rastgele üretim yöntemi elde edilmesinin mümkün olamayacağı, bu arama yöntemi için günümüz bilgisayarlarının hesaplama gücünün yetersiz kalacağı ortadadır. Bir diğer taraftan üretilen matrislerin donanım özelliklerinin belirlenmesi (gereken XOR sayısı açısından) için gerekli hesaplama gücü de bazı büyük boyutlar ve matrisin tanımlandığı cisim boyutu için yetersiz kalmaktadır. Örneğin, 1 tane 16×16 boyutunda $GF(2^8)$ üzerinde tanımlı Hadamard tersi kendisine eşit MDS matristen GHadamard matris formu ile $\sim 2^{119,91}$ tane tersi kendisine eşit MDS matris üretilir.

Bu tasarım ile literatürde özellikle yayılım tabakalarının tasarımı için genel yapıların eksikliği açık probleminin çözümü için bir yaklaşım sunulmuş, Hadamard matris tanımı genelleştirilmiştir, genelleştirmeden kasıt tek bir Hadamard matris ile çok sayıda matrisin, hatta bu form ile elde edilemeyecek matrislerin arama yapmadan doğrudan üretilmesidir.

GHadamard matris formu ile (tersi kendisine eşit) MDS matrisler için literatürde bilinen en iyi sonuçlar geliştirilmiştir. $GF(2^4)$ üzerinde 4×4 tersi kendisine eşit MDS matrisler için bilinen en iyi XOR sayısı 64 XOR sayısına eşit literatürde yer almayan GHadamard formda yeni matrisler üretilmiştir. $GF(2^8)$ üzerinde 4×4 tersi kendisine eşit MDS matrisler için bilinen en iyi XOR sayısı 160'dan 158 XOR'a düşürülmüştür. $GF(2^4)$ üzerinde 8×8 tersi kendisine eşit MDS matrisler için bilinen en iyi XOR sayısı 512 'den 407 XOR'a düşürülmüştür. $GF(2^4)$ üzerinde 8×8 MDS matrisler için bilinen en iyi XOR sayısı 432'den 380 XOR'a düşürülmüştür. $GF(2^8)$ üzerinde 8×8 MDS matrisler için bilinen en iyi XOR sayısı 680 XOR sayısına eşit literatürde yer almayan GHadamard formda yeni matrisler üretilmiştir. Artan kaynak kısıtlı cihaz kullanımı minimum donanım ve yazılım maliyetli tasarım yöntemlerinin önemini daha da arttıracaktır. Bu açıdan önerilen yeni matris formu ile üretilen

matrislerin özellikle hafif sıklet blok şifreler için maliyetli olan yayılım tabakalarını XOR sayısı metriği açısından minimize etmesi bu yapının önemini vurgulamaktadır.

Tez kapsamında doğrusal ve diferansiyel kriptanalize karşı dirençli tersi kendisine eşit 4-bit S-kutusu verilmiştir. Ayrıca yapılan çalışmalar doğrultusunda AES şifreleme algoritmasının anahtar planlama yapısı iyileştirilerek yeni bir anahtar planlama algoritması önerilmiş, algoritma Shannon'ın önerdiği karıştırma ve yayılma ilkelerini destekleyen SAC testinden geçirilerek % 94 oranında başarı sağlanmıştır. Ayrıca literatürde yer alan bazı hafif sıklet şifreleme algoritmalarının anahtar planlama yapıları ile kıyaslandığında, önerilen algoritmanın başarılı bir yöntem olduğu gösterilmiştir.

Tez çalışmasında geliştirilen özgün yapıları (GHadamard tersi kendisine eşit MDS matrisi kullanan yayılım tabakası, önerilen tersi kendisine eşit 4-bit S-kutusu ve önerilen anahtar planlama algoritması) kullanan yeni bir hafif sıklet blok şifreleme algoritması tasarlanmıştır. Tasarlanan algoritma yazılım performansı açısından 8-bit mikrodenetleyici üzerinde literatürde yer alan bazı hafif sıklet blok şifreleme algoritmalarına oranla daha iyi ve kabul edilebilir performans göstermektedir. Ayrıca geliştirilen algoritma JAMBU modunda kimlik doğrulamalı şifreleme yapısında, iki adet WeMos D1 ESP8266 tabanlı WiFi Modül kartı üzerinde istemci-sunucu mimarisi ile şifreli haberleşme için kullanılmıştır. Senaryoda, IOT kartlardan biri sunucu diğeri istemci olarak belirlenmiştir. İstemci olarak belirlenmiş WiFi modül üzerinde gönderilecek veri önerilen algoritma ile şifrelenir ve WiFi üzerinden sunucu kartına gönderilir. Sunucu kartı üzerinde önerilen algoritmanın şifre çözme algoritması kullanılarak açık veriye ulaşılır.

Tez çalışması yapılan tüm bu çalışmalar ile özgün değere sahiptir ve tez kapsamında geliştirilen GHadamard matris formu ile literatüre önemli katkılarda bulunulmuştur, GHadamard matris formu ve tez kapsamında önerilen diğer yapıların ileriki çalışmalarda literatüre katkı sağlayacağı açıktır.

İleriki çalışmalarda özellikle GHadamard matris formu temelli, Hadamard, Circulant ve Toeplitz formlarının kapsayan daha genel bir formun bulunması üzerine çalışmalar yapılacaktır. Ayrıca GF(2^8) üzerinde daha büyük boyutlu matrislerle

minimum XOR maliyetli matrislerin bulunması üzerine çalışılması planlanmaktadır. Genelleştirilmiş yapıların bulunmasının, özellikle hafif sıklet blok şifrelerin tasarım ilkellerinin belirlenmesi açısından önemli bir girdi sağlayacağı açıktır.

KAYNAKLAR

[1] Aslan B., Blok Şifreler İçin Cebirsel İkili Doğrusal Dönüşüm Tasarımı ve Modern Bir Blok Şifreye Uygulanması, Doktora Tezi, Trakya Üniversitesi, Fen Bilimleri Enstitüsü, Edirne, 2013, 307650.

[2] Kölbl S., Design and Analysis of Cryptographic Algorithms, PhD Thesis, Technical University of Denmark, DTU Compute Cyber Security, Denmark, 2016.

[3] FIPS 46-3, Data Encryption Standard, https://csrc.nist.gov/csrc/media/publications/fips/46/3/archive/1999-10-25/documents/fips46-3.pdf (Ziyaret Tarihi: 01 Mayıs 2018).

[4] Dinu D. D., Efficient and Secure Implementations of Lightweight Symmetric Cryptographic Primitives, PhD Thesis, Universite Du Luxembourg, The Faculty of Sciences Technology and Communication, Luxembourg, 2017.

[5] Dinu D. D., Le Corre Y., Khovratovich D., et al., Triathlon of Lightweight Block Ciphers for the Internet of Things, *In: NIST Workshop on Lightweight Cryptography 2015,* Gaithersburg, USA, 20–21 July 2015.

[6] Khoo K., Peyrin T., Poschmann A. Y., et al., FOAM: Searching for Hardware-Optimal SPN Structures and Components with a Fair Comparison, Editors: Batina L., Robshaw M., *Cryptographic Hardware and Embedded Systems – CHES 2014 Proceedings - Lecture Notes in Computer Science*, 8731, Springer, Berlin, Heidelberg, 433-450, 2014.

[7] Avanzi R., A Salad of Block Ciphers, https://eprint.iacr.org/2016/1171.pdf (Ziyaret Tarihi: 20 Nisan 2018).

[8] Youssef A.M., Mister S., Tavares S. E., On the Design of Linear Transformations for Substitution Permutation Encryption Networks, *Fourth Annual Workshop on Selected Areas in Cryptography*, Ottawa Ontario, Canada, 11-12 August 1997.

[9] Li Y., Wang M., On the Construction of Lightweight Circulant Involutory MDS Matrices, Editors: Peyrin T., *Fast Software Encryption – FSE 2016 Proceedings - Lecture Notes in Computer Science,* 9783, Springer, Berlin, Heidelberg, 121-139, 2016.

[10] Liu M., Sim S.M., Lightweight MDS Generalized Circulant Matrices, Editors: Peyrin T., *Fast Software Encryption – FSE 2016 Proceedings - Lecture Notes in Computer Science,* 9783, Springer, Berlin, Heidelberg, 101-120, 2015.

[11] Sim S.M., Khoo K., Oggier F., et al., Lightweight MDS Involution Matrices, Editors: Leander G., *Fast Software Encryption – FSE 2015 Proceedings - Lecture Notes in Computer Science,* 9054, Springer, Berlin, Heidelberg, 471-493, 2015.

[12] Sarkar S., Syed H., Analysis of Toeplitz MDS Matrices, Editors: Pieprzyk J., Suriadi S., *Information Security and Privacy – ACISP 2017 Proceedings - Lecture Notes in Computer Science,* 10343, Springer, Cham, 3-18, 2017.

[13] Technical Report No. ASU/2014/1, On Constructions of MDS Matrices From Circulant-Like Matrices For Lightweight Cryptography, https://www.isical.ac.in/~asu/TR/TechRepASU201401.pdf (Ziyaret Tarihi: 10 Haziran 2018).

[14] Berger T. P., Construction of Recursive MDS Diffusion Layers from Gabidulin Codes, Editors: Paul G., Vaudenay S., *Progress in Cryptology – INDOCRYPT 2013 Proceedings - Lecture Notes in Computer Science,* 8250, Springer, Cham, 274-285, 2013.

[15] Augot D., Finiasz M., Direct Construction of Recursive MDS Diffusion Layers Using Shortened BCH Codes, Editors: Cid C., Rechberger C., *Fast Software Encryption – FSE 2014 Proceedings - Lecture Notes in Computer Science,* 8540, Springer, Berlin, Heidelberg, 3-17, 2015.

[16] Gupta K. C., Ray I. G., On Constructions of Involutory MDS Matrices, Editors: Youssef A., Nitaj A., Hassanien A.E., *Progress in Cryptology – AFRICACRYPT 2013 Proceedings - Lecture Notes in Computer Science,* 7918, Springer, Berlin, Heidelberg, 43-60, 2013.

[17] Cryptology ePrint Archive: Report 2011/609, The PHOTON Family of Lightweight Hash Functions, https://eprint.iacr.org/2011/609.pdf (Ziyaret Tarihi: 24 Mayıs 2018).

[18] Sim S.M., Khoo K., Oggier F., et al., Lightweight MDS Involution Matrices, Editors: Leander G., *Fast Software Encryption - FSE 2015 Proceedings - Lecture Notes in Computer Science,* 9054, Springer, Berlin, Heidelberg, 471-493, 2015.

[19] Kurt Pehlivanoğlu M., Sakallı M. T., Akleylek S., Duru N., Rijmen V., Generalisation of Hadamard Matrix to Generate Involutory MDS Matrices for Lightweight Cryptography, *IET Information Security*, 2018, **12**(4), 348-355.

[20] Sarkar S., Syed H., Lightweight Diffusion Layer: Importance of Toeplitz Matrices, *IACR Transactions on Symmetric Cryptology*, 2016, **2016**(1), 95-113.

[21] Webster A. F., Tavares S. E., On the Design of S-Boxes, Editors: Williams H. C., *Advances in Cryptology - CRYPTO '85 Proceedings - Lecture Notes in Computer Science,* 218, Springer, Berlin, Heidelberg, 523-534, 1986.

[22] FIPS 197, Advanced Encryption Standard, https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.197.pdf (Ziyaret Tarihi: 11 Mayıs 2018).

[23] Kerckhoffs A., La Cryptographie Militaire I-III, *Journal des Sciences Militaires*, 1883, **9**, 5-38.

[24] Kerckhoffs A., La Cryptographie Militaire IV, *Journal des Sciences Militaires*, 1883, **9**, 161-191.

[25] Shannon C. E. , Communication Theory of Secrecy Systems, *The Bell System Technical Journal*, 1949, **28**(4), 656-715.

[26] Schneier B. A self-study course in block-cipher cryptanalysis, *Cryptologia*, 2000, **24**(1), 18-33.

[27] Sakallı M. T., Modern Şifreleme Yöntemlerinin Gücünün İncelenmesi, Doktora Tezi, Trakya Üniversitesi, Fen Bilimleri Enstitüsü, Edirne, 2006, 183288.

[28] Braeken A., Cryptographic Properties of Boolean Functions and S-Boxes, PhD Thesis, Katholieke Universiteit Leuven, Engineer Sciences Faculty, Belgium, 2006.

[29] Bogdanov A., Knudsen L.R., Leander G., Paar C., Poschmann A., Robshaw M.J.B., Seurin Y., Vikkelsoe C., PRESENT: An Ultra-Lightweight Block Cipher, Editors: Paillier P., Verbauwhede I., *Cryptographic Hardware and Embedded Systems - CHES 2007 Proceedings - Lecture Notes in Computer Science*, 4727, Springer, Berlin, Heidelberg, 450-466, 2007.

[30] Aslan B., Sakallı M. T., S-kutularının Kriptografik Özellikleri, *Eleco International Conference on Electrical and Electronics Eng.*, Bursa, Türkiye, 26-30 November 2008.

[31] Courtois N., Security Evaluation of GOST 28147-89 In View Of International Standardisation, *Cryptologia*, 2012, **36**(1), 2-13.

[32] Rivest R.L., The RC5 encryption algorithm, Editors: Preenel B., *Fast Software Encryption - FSE 1994 Proceedings - Lecture Notes in Computer Science*, 1008, Springer, Berlin, Heidelberg, 86-96, 1995.

[33] Wheeler D.J., Needham R.M., TEA, a tiny encryption algorithm, Editors: Preenel B., *Fast Software Encryption - FSE 1994 Proceedings - Lecture Notes in Computer Science*, 1008, Springer, Berlin, Heidelberg, 363-366, 1995.

[34] Schneier B., Kelsey J., Whiting D., Wagner D., Hall C., Ferguson N., Twofish: A 128-Bit Block Cipher, http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=0D7F6B553E8439F8DC758F17593647E1?doi=10.1.1.35.1273&rep=rep1&type=pdf (Ziyaret Tarihi: 30 Nisan 2018).

[35] Standaert F. X., Piret G., Gershenfeld N., Quisquater J. J., SEA: A Scalable Encryption Algorithm for Small Embedded Applications, Editors: Domingo-Ferrer J., Posegga J., Schreckling D., *Smart Card Research and Advanced Applications - CARDIS 2006 Proceedings - Lecture Notes in Computer Science,* 3928, Springer, Berlin, Heidelberg, 222-236, 2006.

[36] Stallings W., *Cryptography and Network Security: Principles and Practice*, 6th ed., Pearson Education, USA, 2014.

[37] Massey J. L., SAFER K-64: A byte-oriented block-ciphering algorithm, Editors: Anderson R., *Fast Software Encryption - FSE 1993 Proceedings - Lecture Notes in Computer Science,* 809, Springer, Berlin, Heidelberg, 1-17, 1994.

[38] Standaert F. X., Piret G, Rouvroy G., Quisquater J. J.,Legat J. D., ICEBERG : An Involutional Cipher Efficient for Block Encryption in Reconfigurable Hardware, Editors: Roy B, Meier W., *Fast Software Encryption - FSE 2004 Proceedings - Lecture Notes in Computer Science,* 3017, Springer, Berlin, Heidelberg, 279-298, 2004.

[39] Schneier B., Description of a new variable-length key, 64-bit block cipher (Blowfish), Editors: Anderson R., *Fast Software Encryption - FSE 1993 Proceedings - Lecture Notes in Computer Science,* 809, Springer, Berlin, Heidelberg, 191-204, 1994.

[40] Cryptology ePrint Archive: Report 2013/404, The SIMON and SPECK Families of Ligthweight Block Ciphers, https://eprint.iacr.org/2013/404.pdf (Ziyaret Tarihi: 01 Mayıs 2018).

[41] Matsuru M., New block encryption algorithm MISTY, Editors: Biham E., *Fast Software Encryption - FSE 1997 Proceedings - Lecture Notes in Computer Science*, 1267, Springer, Berlin, Heidelberg, 54-68, 1997.

[42] 3rd Generation Partnership Project: Technical Specication Group Services and System Aspects, 3G Security, Specification of the 3GPP Condentiality and Integrity Algorithms, Document 2: KASUMI specification, https://www.garykessler.net/library/crypto/3G_KASUMI.pdf (Ziyaret Tarihi: 01 Mayıs 2018).

[43] Anderson R., Biham E., Two practical and provably secure block ciphers: BEAR and LION, Editors: Gollmann D., *Fast Software Encryption - FSE 1996 Proceedings - Lecture Notes in Computer Science,* 1039, Springer, Berlin, Heidelberg, 113-120, 1996.

[44] Adams C. M., Rani R., The Cast-256 Encryption Algorithm, *First Advanced Encryption Standard Candidate Conference*, National Institute of Standards and Technology (NIST), USA, 20-22 August 1998.

[45] Nyberg K., Differentially Uniform Mappings for Cryptography, Editors: Helleseth T., *Advances in Cryptology — EUROCRYPT 93 Proceedings -*

Lecture Notes in Computer Science, 765, Springer, Berlin, Heidelberg, 55-64, 1994.

[46] Aslan B., Sakallı M. T., Buluş E., Üs Haritalama Tabanlı Cebirsel 8-Bit Giriş 8-Bit Çıkışlı S-Kutularının Sınıflandırılması, *2. Ağ ve Bilgi Güvenliği Ulusal Sempozyumu*, Girne, KKTC, 16-18 Mayıs 2008.

[47] Büyüksaraçoğlu Sakallı F., Akış Şifrelerin Tasarım Teknikleri ve Güç Analizi, Doktora Tezi, Trakya Üniversitesi, Fen Bilimleri Enstitüsü, Edirne, 2011, 300210.

[48] Heys H. M., Tavares S. E., Substitution-Permutation Networks Resistant to Differential and Linear Cryptanalysis, *Journal of Cryptology*, 1996, **9**(1), 1–19.

[49] Daemen J., Rijmen V., *The Design of Rijndael: AES-The Advanced Encryption Standard*, 1st ed., Springer-Verlag Berlin Heidelberg , Germany, 2002.

[50] Menezes A. J., Oorschot P. C., Vanstone S. A., *Handbook Of Applied Cryptography*, 1st ed., CRC Press , USA, 1996.

[51] Ferguson N., Schneier B., *Practical Cryptography*, 1st ed., Wiley Publishing, USA, 2003.

[52] Bellare M., Namprempre C., Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm, Editors: Okamoto T. , *Advances in Cryptology — ASIACRYPT 2000 Proceedings - Lecture Notes in Computer Science*, 1976, Springer, Berlin, Heidelberg, 531-545, 2000.

[53] Cryptology ePrint Archive: Report 2003/069, EAX: A Conventional Authenticated-Encryption Mode, https://eprint.iacr.org/2003/069.pdf (Ziyaret Tarihi: 19 Mayıs 2018).

[54] Švenda P., Basic comparison of Modes for Authenticated-Encryption (IAPM, XCBC, OCB, CCM, EAX, CWC, GCM, PCFB, CS), https://www.fi.muni.cz/~xsvenda/docs/AE_comparison_ipics04.pdf (Ziyaret Tarihi: 19 Mayıs 2018).

[55] Gueron S., Lindell Y., GCM-SIV: Full Nonce Misuse-Resistant Authenticated Encryption at Under One Cycle per Byte, *22nd ACM Conference on Computer and Communications Security*, Colorado, USA, 12-16 October 2015.

[56] Sönmez Turan M., NIST's Lightweight Crypto Project, https://www.nist.gov/sites/default/files/documents/2016/10/17/sonmez-turan-presentation-lwc2016.pdf (Ziyaret Tarihi: 24 Mayıs 2018).

[57] McKay K. A., Bassham L., Sönmez Turan M., Mouha N., Report on Lightweight Cryptography: NISTIR 8114: National Institute of Standards and

Technology U.S. Department of Commerce; 2017, https://doi.org/10.6028/NIST.IR.8114 (Ziyaret Tarihi: 03 Mayıs 2018).

[58] Hatzivasilis G., Fysarakis K., Papaefstathiou I., et al., A review of lightweight block ciphers, *Journal of Cryptographic Engineering*, 2018, **8**(2), 141–184.

[59] Koo B., Hong D., Kwon D., A New Practical Key Recovery Attack on the Stream Cipher RC4 under Related-Key Model, Editors: Rhee KH., Nyang D., *International Conference on Information Security and Cryptology ICISC 2010 Proceedings - Lecture Notes in Computer Science,* 6829, Springer, Berlin, Heidelberg, 49-67, 2010.

[60] Lomné V., Prouff E., Roche T., Behind the Scene of Side Channel Attacks, Editors: Sako K., Sarkar P., *International Conference on the Theory and Application of Cryptology and Information Security ASIACRYPT 2013 Proceedings - Lecture Notes in Computer Science,* 8269, Springer, Berlin, Heidelberg, 506-525, 2013.

[61] Isobe T., Ohigashi T., Watanabe Y., Morii M., Full Plaintext Recovery Attack on Broadcast RC4, Editors: Moriai S., *Fast Software Encryption FSE 13 Proceedings - Lecture Notes in Computer Science,* 8424, Springer, Berlin, Heidelberg, 179-202, 2014.

[62] Chen J., Miyaji A., A New Practical Key Recovery Attack on the Stream Cipher RC4 under Related-Key Model, Editors: Lai X., Yung M., Lin D., *International Conference on Information Security and Cryptology Inscrypt 2010 Proceedings - Lecture Notes in Computer Science,* 6584, Springer, Berlin, Heidelberg, 62-76, 2011.

[63] Chen L., Recommendation for Key Derivation Using Pseudorandom Functions, https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication 800-108.pdf (Ziyaret Tarihi: 23 Mayıs 2018).

[64] Cryptology ePrint Archive: Report 2012/600, The LED Block Cipher, https://eprint.iacr.org/2012/600.pdf (Ziyaret Tarihi: 24 Mayıs 2018).

[65] Chen L., Recommendation for Key Derivation through Extraction-then-Expansion, https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublicatio n800-56c.pdf (Ziyaret Tarihi: 24 Mayıs 2018).

[66] Dang Q., Recommendation for Existing Application-Specific Key Derivation Functions, https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication 800-56c.pdf (Ziyaret Tarihi: 24 Mayıs 2018).

[67] ISO/IEC 29192-2:2012, Information technology-Security techniques-Lightweight cryptography-Part 2: Block ciphers, https://www.iso.org/standard/56552.html (Ziyaret Tarihi: 27 Mayıs 2018).

[68] Gong Z., Nikova S., Law Y.W., KLEIN: A New Family of Lightweight Block Ciphers, Editors: Juels A., Paar C., *International Workshop on Radio Frequency Identification: Security and Privacy Issues RFIDSec 2011*

Proceedings - Lecture Notes in Computer Science, 7055, Springer, Berlin, Heidelberg, 1-18, 2012.

[69] Hong D., Sung J., Hong S., et al. HIGHT: A New Block Cipher Suitable for Low-Resource Device, Editors: Goubin L., Matsui M., *International Workshop on Cryptographic Hardware and Embedded Systems CHES 2006 Proceedings - Lecture Notes in Computer Science,* 4249, Springer, Berlin, Heidelberg, 46-59, 2006.

[70]]ATmega 128 Datasheet, http://www.kjit.bme.hu/images/stories/targyak/jarmufedelzeti_rendszerek/atmel_atmega128_manual.pdf (Ziyaret Tarihi: 08 Haziran 2018).

[71] Cryptology ePrint Archive: Report 2017/511, State of the Art in Lightweight Symmetric Cryptography, https://eprint.iacr.org/2003/069.pdf (Ziyaret Tarihi: 01 Haziran 2018).

[72] Dinu D. D., Biryukov A., Großschädl J., et al., FELICS – Fair Evaluation of Lightweight Cryptographic Systems, *In: NIST Workshop on Lightweight Cryptography 2015*, Gaithersburg, USA, 20–21 July 2015.

[73] Engels D., Saarinen MJ.O., Schweitzer P., Smith E.M., The Hummingbird-2 Lightweight Authenticated Encryption Algorithm, Editors: Juels A., Paar C., *International Workshop on Radio Frequency Identification: Security and Privacy Issues RFIDSec 2011 Proceedings - Lecture Notes in Computer Science,* 7055, Springer, Berlin, Heidelberg, 19-31, 2012.

[74] Bogdanov A., Mendel F., Regazzoni F., et al., ALE: AES-Based Lightweight Authenticated Encryption, Editors: Moriai S., *Fast Software Encryption - FSE 2013 Proceedings - Lecture Notes in Computer Science*, 8424, Springer, Berlin, Heidelberg, 447-466, 2014.

[75] Bilgin B., Bogdanov A., Knežević M., et al., Fides: Lightweight Authenticated Cipher with Side-Channel Resistance for Constrained Hardware, Editors: Bertoni G., Coron JS., *Cryptographic Hardware and Embedded Systems - CHES 2013 - Lecture Notes in Computer Science*, 8086, Springer, Berlin, Heidelberg, 142-158, 2013.

[76] Wu H., Huang T., JAMBU Lightweight Authenticated Encryption Mode and AES-JAMBU v1, *DIAC 2014: Directions in Authenticated Ciphers,* Santa Barbara, USA, 23-24 August 2014.

[77] Wu H., Huang T., The JAMBU Lightweight Authentication Encryption Mode v2.1, https://competitions.cr.yp.to/round3/jambuv21.pdf (Ziyaret Tarihi: 02 Haziran 2018).

[78] Stinson D. R., *Cryptography: Theory and Practice*, 3rd ed., CRC Press, USA, 2003.

[79] Ling S., Xing C., *Coding Theory: A First Course*, 1st ed., Cambridge University Press, UK, 2004.

[80] Lidl R., Niederreiter H., *Introduction to Finite Fields and Their Applications*, 1st ed., Cambridge University Press, Australia, 1986.

[81] Golub G. H., Loan C. F., *Matrix Computations*, 4th ed., The Johns Hopkins University Press, UK, 2013.

[82] Özen M., Şiap İ., Çallıalp F., Non-Hamming (Rosenbloom-Tsfasman) Metriğine Göre Lineer Kodlarin Yapisi, *Anadolu Üniversitesi Bilim Ve Teknoloji Dergisi*, 2004, **5**(2), 253-258.

[83] Barreto, P. S. L. M., Simplício M. J., CURUPIRA-1, a Block Cipher for Constrained Platforms, *25th Brazilian Symposium on Computer Networks and Distributed Systems*, Belém, Brazil, 10 May 2007.

[84] Cryptology ePrint Archive: Report 2016/119, Lightweight Multiplication in $GF(2^n)$ with Applications to MDS Matrices, https://eprint.iacr.org/2016/119.pdf (Ziyaret Tarihi: 05 Haziran 2018).

[85] Biham E., Anderson R., Knudsen L., Serpent: A New Block Cipher Proposal, Editors: Vaudenay S., *Fast Software Encryption - FSE 1998 Proceedings - Lecture Notes in Computer Science*, 1372, Springer, Berlin, Heidelberg, 222-238, 1998.

[86] Knudsen L., Leander G., Poschmann A., et al., PRINTcipher: A Block Cipher for IC-Printing, Editors: Mangard S., Standaert F. X., *Cryptographic Hardware and Embedded Systems – CHES 2010 Proceedings - Lecture Notes in Computer Science*, 6225, Springer, Berlin, Heidelberg, 16-32, 2010.

[87] Aoki K., Ichikawa T., Kanda M., et al., Camellia: A 128-Bit Block Cipher Suitable for Multiple Platforms-Design andAnalysis, Editors: Stinson D.R., Tavares S., *Selected Areas in Cryptography – SAC 2000 Proceedings - Lecture Notes in Computer Science*, 2012, Springer, Berlin, Heidelberg, 39-56, 2001.

[88] Kwon D., Kim J., Park S., et al., New Block Cipher: ARIA, Editors: Lim JI., Lee DH., *Information Security and Cryptology – ICISC 2003 Proceedings - Lecture Notes in Computer Science*, 2971, Springer, Berlin, Heidelberg, 432-445, 2004.

[89] Albrecht M.R., Driessen B., Kavun E. B., et al., Block Ciphers – Focus on the Linear Layer (feat. PRIDE), Editors: Garay J.A., Gennaro R., *Advances in Cryptology – CRYPTO 2014 Proceedings - Lecture Notes in Computer Science*, 8616, Springer, Berlin, Heidelberg, 57-76, 2014.

[90] Xu H., Tan L., Lai X., On the Recursive Construction of MDS Matrices for Lightweight Cryptography, Editors: Huang X., Zhou J., *Information Security Practice and Experience – ISPEC 2014 Proceedings - Lecture Notes in Computer Science*, 8434, Springer, Cham, 552-563, 2014.

[91] Barreto, P. S. L. M., Rijmen, V., The Khazad Legacy-Level Block Cipher, *First Open NESSIE Workshop 2000*, Leuven, Belgium, 13–14 November 2000.

[92] Rijmen V., Baretto P. S. L. M., The ANUBIS Block Cipher, https://web.archive.org/web/20160606112246/http://www.larc.usp.br/~pbarreto/AnubisPage.html (Ziyaret Tarihi: 08 Haziran 2018).

[93] Simplício M. J., Barreto, P. S. L. M., Carvalho T. C. M. B., et al., The CURUPIRA-2 Block Cipher for Constrained Platforms: Specification and Benchmarking, *1st International Workshop on Privacy in Location-Based Applications*, Malaga, Spain, 9 October 2008.

[94] Sony Corporation, The 128-bit Blockcipher CLEFIA, http://www.cryptrec.go.jp/english/cryptrec_13_spec_cypherlist_files/PDF/22_00espec.pdf (Ziyaret Tarihi: 08 Haziran 2018).

[95] Shibutani K., Isobe T., Hiwatari H., et al., Piccolo: An Ultra-Lightweight Blockcipher, Editors: Preneel B., Takagi T., *Cryptographic Hardware and Embedded Systems – CHES 2011 Proceedings - Lecture Notes in Computer Science*, 6917, Springer, Berlin, Heidelberg, 342-357, 2011.

[96] Lacan J., Fimes J., Systematic MDS Erasure Codes Based on Vandermonde Matrices, *IEEE Communications Letters* , 2004, **8**(9), 570-572.

[97] Sajadieh M., Dakhilalian M., Mala H., et al., On Construction of Involutory MDS Matrices from Vandermonde Matrices in $GF(2^q)$, *Designs, Codes and Cryptography*, 2012, **64**(3), 287-308.

[98] Gupta K.C., Ray I.G., On Constructions of MDS Matrices from Companion Matrices for Lightweight Cryptography, Editors: Cuzzocrea A., Kittl C., Simos D.E., Weippl E., Xu L., *Security Engineering and Intelligence Informatics – CD-ARES 2013 Proceedings - Lecture Notes in Computer Science*, 8128, Springer, Berlin, Heidelberg, 29-43, 2013.

[99] Gupta K.C., Pandey S. K., Venkateswarlu A., On the Direct Construction of Recursive MDS Matrices, *Designs, Codes and Cryptography*, 2017, **82**(1-2), 77-94.

[100] Bosma W., Cannon J., Playoust C., The MAGMA Algebra System I: The User Language, *Journal of Symbolic Computation*, 1997, **24**(3-4), 235-265.

[101] Junod P., Vaudenay S., Perfect Diffusion Primitives for Block Ciphers, Editors: Handschuh H., Hasan M.A., *Selected Areas in Cryptography – SAC 2004 Proceedings - Lecture Notes in Computer Science*, 3357, Springer, Berlin, Heidelberg, 84-99, 2004.

[102] Gray R. M., Toeplitz and Circulant Matrices: A review, https://ee.stanford.edu/~gray/toeplitz.pdf (Ziyaret Tarihi: 11 Haziran 2018).

[103] Leander G., Poschmann A., On the Classification of 4 Bit S-Boxes, Editors: Carlet C., Sunar B., *Arithmetic of Finite Fields – WAIFI 2007 Proceedings - Lecture Notes in Computer Science*, 4547, Springer, Berlin, Heidelberg, 159-176, 2007.

[104] Borghoff J., Canteaut A., Güneysu T., et al., PRINCE – A Low-Latency Block Cipher for Pervasive Computing Applications, Editors: Wang X., Sako K., *Advances in Cryptology – ASIACRYPT 2012 Proceedings - Lecture Notes in Computer Science*, 7658, Springer, Berlin, Heidelberg, 208-225, 2007.

[105] Büyüksaraçoğlu Sakallı F., Buluş E., Sakallı M. T., et al., Aes Blok Şifresinin Anahtar Genişletme Rutininin Geliştirilmesi Ve Bir Blok Şifreden Bağımsız Anahtar Genişletme Rutininin Tasarımı, *6. Uluslararası Bilgi Güvenliği ve Kriptoloji Konferansı*, Ankara, Türkiye, 20-21 Eylül 2013.

[106] Lim C. H., Korkishko T., mCrypton – A Lightweight Block Cipher for Security of Low-Cost RFID Tags and Sensors, Editors: Song JS., Kwon T., Yung M., *International Workshop on Information Security Applications – WISA 2005 Proceedings - Lecture Notes in Computer Science*, 3786, Springer, Berlin, Heidelberg, 243-258, 2006.

[107] Suzaki T., Minematsu K., Morioka S., et al., TWINE: A Lightweight Block Cipher for Multiple Platforms, Editors: Knudsen L. R., Wu H., *International Conference on Selected Areas in Cryptography – SAC 2012 Proceedings - Lecture Notes in Computer Science*, 7707, Springer, Berlin, Heidelberg, 339-354, 2012.

[108] Wu W., Zhang L., LBlock: A Lightweight Block Cipher, Editors: Lopez J., Tsudik G., *International Conference on Applied Cryptography and Network Security – ACNS 2011 Proceedings - Lecture Notes in Computer Science*, 6715, Springer, Berlin, Heidelberg, 327-344, 2011.

[109] Daemen J., Peeters M., Van Assche G., et al., Nessie Proposal: NOEKEON, http://gro.noekeon.org/Noekeon-spec.pdf (Ziyaret Tarihi: 08 Haziran 2018).

[110] Cazorla M., Marquet K., Minier M., Survey and Benchmark of Lightweight Block Ciphers for Wireless Sensor Networks., *In: 2013 International Conference on Security and Cryptography (SECRYPT)*, Reykjavik, Iceland, 29–31 July 2013.

[111] Biryukov A., Roy A., Velichkov V., *Differential Analysis of Block Ciphers SIMON and SPECK*, London, UK, 03-05 March, 2014.

[112] Cryptology ePrint Archive: Report 2014/681, Improved Linear Cryptanalysis of Reduced-round SIMON, https://eprint.iacr.org/2014/681.pdf (Ziyaret Tarihi: 08 Haziran 2018).

[113] Zhang P., Sun B., Li C., Saturation Attack on the Block Cipher HIGHT, Editors: Garay J.A., Miyaji A., Otsuka A., *International Conference on*

Cryptology and Network Security – CANS 2009 Proceedings - Lecture Notes in Computer Science, 5888, Springer, Berlin, Heidelberg, 76-86, 2009.

[114] Özen O., Varıcı K., Tezcan C., et al., Lightweight Block Ciphers Revisited: Cryptanalysis of Reduced Round PRESENT and HIGHT, Editors: Boyd C., González Nieto Jx., *Australasian Conference on Information Security and Privacy – ACISP 2009 Proceedings - Lecture Notes in Computer Science*, 5594, Springer, Berlin, Heidelberg, 90-107, 2009.

[115] Koo B., Hong D., Kwon D., Related-Key Attack on the Full HIGHT, Editors: Rhee KH., Nyang D., *International Conference on Information Security and Cryptology – ICISC 2010 Proceedings - Lecture Notes in Computer Science*, 6829, Springer, Berlin, Heidelberg, 49-67, 2010.

[116] Hong D., Koo B., Kwon D., Biclique Attack on the Full HIGHT, Editors: Kim H., *International Conference on Information Security and Cryptology – ICISC 2011 Proceedings - Lecture Notes in Computer Science*, 7259, Springer, Berlin, Heidelberg, 365-374, 2010.

KİŞİSEL YAYIN VE ESERLER

[1] **Kurt Pehlivanoğlu M.**, Sakallı M. T., Akleylek S., Duru N., Rijmen V., Generalisation of Hadamard Matrix to Generate Involutory MDS matrices for Lightweight Cryptography, *IET Information Security*, 2018, **12** (4), 348-355.

[2] **Kurt Pehlivanoğlu M.**, Akleylek S., Sakallı M. T., Duru N., On the Design Strategies of Diffusion Layers and Key Schedule in Lightweight Block Ciphers, *2017 International Conference on Computer Science and Engineering (UBMK)*, Antalya, Türkiye, 5-8 October, 2017.

[3] **Kurt Pehlivanoğlu M.**, Sakallı M. T., Duru N., Büyüksaraçoğlu Sakallı F., The New Approach of AES Key Schedule for Lightweight Block Ciphers, *IOSR Journal of Computer Engineering (IOSR-JCE)*, 2017, **19**(3), 21-26.

[4] **Kurt Pehlivanoğlu M.**, Duru N., Sakallı M. T., Sonlu Cisimler Teorisine Dayalı Gri Seviye Görüntü Şifreleme, *Bilecik Şeyh Edebali Üniversitesi Fen Bilimleri Dergisi*, 2016, **3**(2), 10-17.

[5] **Kurt Pehlivanoğlu M.**, Duru N., Encryption of Walsh Hadamard Transform Applied Images with the AES Encryption Algorithm, *24th Signal Processing and Communication Application Conference (SIU)*, Zonguldak, Türkiye, 16-19 May, 2016.

[6] **Kurt Pehlivanoğlu M.**, Duru N., Email Encryption using RC4 Algorithm, *International Journal of Computer Applications*, 2015, **130**(14), 25-29.

[7] **Kurt Pehlivanoğlu M.**, Duru N., Steganography Over Video Files Using Menezes Vanstone Elliptic Curve Cryptography Algorithm, *22th Signal Processing and Communication Application Conference (SIU)*, Trabzon, Türkiye, 23-25 April, 2014.

[8] **Kurt Pehlivanoğlu M.**, Duru N., Encryption with Changing Least Significant Bit on Menezes Vanstone Elliptic Curve Cryptosystem, *International Electronics, Computerand Automation Technology Symposium and Congress (ICEECAT 2014)*, Konya, Türkiye, 09-11 May, 2014.

ÖZGEÇMİŞ

Meltem KURT PEHLİVANOĞLU 1987 yılında Tekirdağ'ın Çorlu ilçesinde doğdu. İlk ve orta öğrenimini Çorlu'da, lise öğrenimini İstanbul Çamlıca Kız Anadolu Lisesi'nde tamamladı. Trakya Üniversitesi Bilgisayar Mühendisliği Bölümü'nden 2010 yılında mezun oldu. Aynı yıl içinde Trakya Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda başladığı Yüksek Lisans eğitimini 2012 yılında tamamladı. 2013 yılından beri Kocaeli Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Bilgisayar Mühendisliği Anabilim Dalı'nda doktora eğitimine devam etmektedir. Aynı zamanda 2012 yılından itibaren Kocaeli Üniversitesi Bilgisayar Mühendisliği bölümünde Araştırma Görevlisi olarak çalışmaktadır.