T.C.
TRAKYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

İNDİRGENMİŞ SPN (SUBSTITUTION PERMUTATION NETWORK) ALGORİTMASI İÇİN LİNEER KRİPTANALİZ UYGULAMASI

Şenol ŞEN

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

Tez Yöneticisi: Yrd.Doç.Dr. Ercan BULUŞ

EDİRNE – 2006

Yüksek Lisans Tezi
Trakya Üniversitesi Fen Bilimleri Enstitüsü

ÖZET

Modern şifreleme algoritmalarının tasarımında doğrusal kriptanaliz önemli bir yer tutmaktadır. Kriptanaliz, açık metni yada anahtarı elde etme bilimidir.

Çalışmamda, mevcut olan şifreleme ve kriptanaliz metotları incelenerek genel bilgiler verilmiş olup, 16 bit girişe ve 16 bit çıkışa sahip olan, üç ve dört döngülük bir SPN algoritması için doğrusal kriptanaliz uygulaması gerçekleştirilmiştir. Kullanılan SPN algoritmasında, AES S kutusuna benzer olarak tasarlanan 4 bit giriş ve 4 bit çıkışa sahip S kutusu ve permütasyon işleminde de AES'in ShiftRows (Satırları Öteleme) dönüşümü, bit değerlerinin permütasyonu için kullanılmıştır.

Tezin ilk beş bölümünde Şifrelemenin temelleri, tarihçesi, temel şifreleme algoritmaları ve bu algoritmaların gücü hakkında bilgiler verilmiştir.

Altıncı bölümde donanım ve yazılımla şifrelemenin karşılaştırılması yapılarak avantaj ve dezavantajları açıklanmıştır.

Yedinci ve sekizinci bölümde blok şifreleme algoritmalarının en önemlileri olan DES ve AES sırayla ayrıntılı bir şekilde anlatılmış; dokuzuncu bölümde ise AES S kutusuna benzer S kutusu tasarımı hakkında açıklamalara yer verilmiştir.

Onuncu bölümde Kriptanaliz ve Kriptanaliz metodları incelenerek anlatılmış; onbirinci bölümde de üç ve dört döngülük SPN algoritması için doğrusal kriptanaliz uygulaması detaylı bir şekilde açıklanmıştır.

Sonuçta, ortaya konulan algoritmanın son döngüsünde kullanılan 16 bit anahtarın sırayla 4, 8 ve 16 biti başarı ile elde edilmiştir. Ayrıca anahtarın elde edilmesi sırasında çeşitli veri büyüklükleri için deneysel sonuçlar grafiksel olarak gösterilmiştir.

Anahtar Sözcükler: Doğrusal Kriptanaliz, Şifreleme, SPN(Yerdeğiştirme-Permütasyon Ağı)

Yıl: 2006
Sayfa : 112

Master Thesis
Trakya University Graduate School of
Natural and Applied Sciences
Department of Computer Engineering

SUMMARY

Linear cryptanalysis is one of the attacks which are very important while designing block encryption algorithms. Cryptanalysis is the process of obtaining the plaintext or the key which is unknown for the attacker.

In my study, after giving information about available encryption and cryptanalysis techniques, I realized a linear cryptanalysis application for a 16-bit input, 16-bit output and 3 and 4 round SPNs (Substitution Permutation Networks). In used SPNs for linear cryptanalysis, 4-bit input and 4-bit output algebraically designed S-box which is designed similar to AES S-box has been used for necessary S-box and ShiftRows transformation which is used to permute bytes in AES (Advanced Encryption Standard) for permutation of bits for our SPN.

In the first five chapters of my thesis, I give some information about basics of encryption tecniques, history of basic encryption algorithms, basic encryption algorithms and some basic information about the strength of these algorithms.

In the sixth chapter, I compare the encryption tecniques in view of hardware and software. In addition, I give advantageous and disadvantageous of them.

In the seventh and eighth chapters, I examine two important block ciphers : DES, AES in detail. In ninth chapter, I explained how AES S-box like S-box can be designed.

In the tenth chapter, I examine cryptanalysis and cryptanalysis tecniques and in the eleventh chapter, I examine linear cryptanalysis application for a 3-round and 4-round SPN in detail.

As a result, I have realized an SPN application and obtained 4-bit, 8-bit and 16-bit key from the last round of the cipher. In addition, I have showed some experimental results using different amount of data while obtaining used keys in the last round of the cipher.

Keywords: Linear Cryptanalysis, Encryption, SPN (Substitution-Permutation Network)
Year: 2006
Page : 112

TEŞEKKÜR

Tez çalışmam süresince bana yol gösteren değerli Tez Yöneticisi hocam Yrd. Doç. Dr. Ercan BULUŞ'a, bu süreçte bana vermiş olduğu bilgi ve desteği için Arş. Gör. M. Tolga SAKALLI' ya, hayatımın her safhasında olduğu gibi çalışmalarım süresince yanımda olarak bana desteğini veren sevgili eşim ve aileme teşekkürü bir borç bilirim.

Ocak 2006 Şenol ŞEN

ÖZET ..... **II**
SUMMARY ..... **III**
TEŞEKKÜR ..... **IV**
1. GİRİŞ VE AMAÇ **1**
1.1 GIRIŞ 1
1.2 AMAÇ 2
2. ŞİFRELEMENİN TEMELLERİ **3**
3. ŞİFRELEME (KRİPTOGRAFİ) TARİHÇESİ **4**
4. TEMEL ŞİFRELEME ALGORİTMALARI **6**
4.1. SIMETRIK (GIZLI ANAHTAR) ŞIFRELEME ALGORITMALARI 8
4.1.1. Akış şifreler *8*
4.1.2. Blok şifreler *9*
4.1.2.1. Blok şifreleme sistemlerinin parametreleri 10
4.1.2.2. Blok şifreleme sistemlerinin tasarım ölçütleri 11
4.2. ASIMETRIK (AÇIK ANAHTARLI) ŞIFRELEME ALGORITMALARI 11
5. ŞİFRELEME (KRİPTOGRAFİ) ALGORİTMALARININ GÜCÜ **13**
6. DONANIM VE YAZILIMLA ŞİFRELEMENİN KARŞILAŞTIRILMASI **14**
6.1. DONANIMLA ŞIFRELEME YÖNTEMI 14
6.2. YAZILIMLA ŞIFRELEME YÖNTEMI 15
7. DES - VERİ ŞİFRELEME STANDARDI **16**
7.1. DES'IN GELIŞIM SÜRECI 16
7.2. DES ALGORITMASI 18
7.2.1. Başlangıç permütasyonu (BP) *20*
7.2.2. Bir DES döngüsünde gerçekleştirilen işlemler *20*
7.2.3. f fonksiyonu *21*
7.2.4. Ters permütasyon (BP^{-1}) *25*
7.2.5. Anahtarların oluşturulması *25*
7.3. ŞIFRE ÇÖZME 27
8. AES - İLERİ ŞİFRELEME STANDARDI **28**
8.1. AES'IN GELIŞIM SÜRECI 28
8.2. AES ALGORITMASI 28
8.3. ŞIFRE ÇÖZME 32
8.4. GÜVENLIK 32
9. AES S KUTUSUNA BENZER S KUTUSU TASARIMI **33**
9.1. AES S KUTUSU 33
9.2. S KUTUSU TASARIMI 33
10. KRİPTANALİZ **37**
10.1. GIRIŞ 37

10.2. SALDIRI ÇEŞITLERI (SENARYOLARI) 37
10.3. KRIPTANALIZ METODLARI 38
10.4. BLOK ŞIFRELEME ALGORITMALARINA YÖNELIK SALDIRILAR 39
10.4.1. Temel saldırılar *41*
10.4.2. Gelişmiş saldırılar *41*
11. DOĞRUSAL (LİNEER) KRİPTANALİZ 44
11.1. SPN ALGORITMASI 44
11.2. TEMEL SALDIRIYA GENEL BAKIŞ 46
11.3. ŞIFRE BILEŞENLERININ ANALIZI 49
11.4. ÜÇ DÖNGÜLÜK SPN ALGORITMASINDA 4 BIT ANAHTARIN ELDE EDILMESI 51
11.5. DÖRT DÖNGÜLÜK SPN ALGORITMASINDA 8 BIT ANAHTARIN ELDE EDILMESI 55
11.6. 16 BITLIK ANAHTARIN ELDE EDILMESI 66
12. SONUÇLAR 73
KAYNAKLAR 74
ÖZGEÇMİŞ 77
EK – 1 78

1. GİRİŞ VE AMAÇ

1.1 Giriş

İnsanlar kriptografi hakkında konuştuklarında farklı şeylerden bahsederler. Çocuklar oyuncak şifrelerle oynar ve gizli dillerle konuşurlar. Ancak bunların gerçek güvenlik ve güçlü şifreleme ile çok az ilgisi vardır. Güçlü şifreleme gerçekten değerli bilgiyi organize suç örgütlerine, çokuluslu şirketlere ve büyük hükümetlere karşı korumak için kullanılabilecek bir şifreleme türüdür. Güçlü şifreleme eskiden sadece askeri çalışmalarda kullanılırdı; ancak günümüz bilgi toplumunda gizlilik ve kişiselliğin korunması için önemli araçlardan biri haline gelmiştir. Bir bilgi toplumu olmaya doğru ilerledikçe milyonlarca bireyin teknoloji manasında global olarak gözlenmesi/takibi büyük hükümetler için mümkün olmaya başlamıştır. Kriptografi kişisellik, güven, erişim kontrolü, elektronik ödeme, toplu (şirket, ulus, vs.) güvenlik ve pekçok alan için ana araçlardan biri olmuştur.

Günümüz internet çağında ise bireysel iletişimin çok yaygınlaşması ve internet uygulamalarının finans, ticaret ve kamu hizmetleri gibi önemli alanlara yayılmış olması bilgi güvenliğinin sağlanmasının önemini daha da arttırmıştır. Bu ihtiyaç neticesinde şifreleme (kriptografi) alanında yapılan çalışmalarda büyük bir artış olmuş ve çalışmaları tanımlayan yeni kavramlar ortaya çıkmıştır.

Bu kavramlardan kriptografi; güvenilirlik, veri bütünlüğü ve veri kaynağının doğruluğu gibi bilgi güvenliği ile ilgili matematiksel tekniklerin bir araya geldiği bir çalışma alanıdır. Kriptanaliz kriptografik mekanizmaların nasıl çözüleceği üzerine çalışılması, kriptoloji ise kriptografi ve kriptanalizin bir arada olduğu bilim dalıdır (Andiç, 2002).

Şifreleme (encryption), bir verinin uygun bir bilgi olmadan okunabilmesini neredeyse imkansız kılacak bir forma dönüştürülmesidir. Bunun amacı, gizli bilgiyi saklayarak eline geçmesi istenmeyen kişilerden şifrelenmiş veriye ulaşmaya çalışsalar dahi bilginin gizliliğini garanti altına almaktır. Deşifre etme (decryption) ise şifreleme olayının tersidir; şifrelenmiş verinin bilgi alınabilir bir forma yeniden dönüştürülmesidir (Andiç, 2002).

Şifreleme ve deşifreleme işlemi genellikle anahtar (key) olarak bilinen gizli bir bilginin kullanımına ihtiyaç duyar. Bazı mekanizmalarda aynı anahtar hem şifreleme hem de deşifreleme işlemi için kullanılır; diğer mekanizmalarda ise her iki işlem içinde farklı anahtarlar kullanılır. Bugünün kriptografisinin, şifreleme deşifreleme olayından daha öteye gittiği uygulama alanlarına bakıldığında görülmektedir. Doğrulama (authentication) ve kimlik tespiti (identification), elektronik ticaret (e-commerce), belgeleme (certification), anahtar düzeltme (key recovery), güvenli uzaktan ulaşım gibi uygulama alanları kriptografinin günümüzde yaygın olarak kullanıldığı yerlerdir (Andiç, 2002).

1.2 Amaç

Bilginin güvenli bir şekilde bir noktadan diğer bir noktaya iletilmesi için şifreleme yöntemleri geliştirilmiştir. Aynı zamanda şifrelenen bilgileri kırma teknikleri geliştirilmiştir.

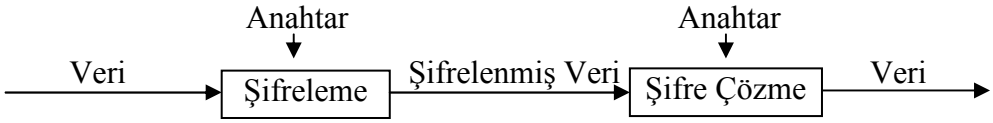
Bilginin üretilmesi ve kullanılması kadar, korunması da aynı öneme sahiptir. Toplumda bir haberleşmenin gizli tutulabilmesi üzerine çalışan bilim olarak düşünülen Şifreleme'nin anlamı çok daha geniştir. Çözülebilmesi çok zor matematik problemleri ve mekanizmaları inceleyen Şifre Bilimini ve bu problemleri ve mekanizmaları çözmeyi hedefleyen ve saldırıları belirleyen Şifre Çözümünü içerir.

Amacımız, mevcut olan şifreleme metotlarını incelemek, bu metotlar doğrultusunda AES şifreleme algoritmasının küçültülmüş bir benzerini ortaya koymak ve saldırı yöntemlerinden biri olan doğrusal kriptanaliz tekniği ile bu ortaya koyduğumuz algoritmanın son döngüsünde kullanılan anahtarı elde etmektir.

Kullandığımız algoritma AES şifreleme algoritmasının küçültülmüş bir benzeri olmakla beraber AES algortimasından farklı olarak ikinci doğrusal dönüşüm-mixcolumns- dönüşümü bizim algoritmamızda kullanılmamıştır. Kabaca bu dönüşümün önemini bu çalışma ile göstermek bir diğer amacımızdır.

2. ŞİFRELEMENİN TEMELLERİ

Şifreleme (Kriptoloji) , mesajları güvenli tutma bilimi ve sanatıdır. Kriptolojinin temel amacı genellikle (Alice ve Bob adı verilen) iki kişinin güvenli olmayan bir kanal üzerinden, ne dediklerinin, muhalifleri (Oscar) tarafından anlaşılmadan haberleşmeye çalışmalarıdır. Bu kanal, bir telefon hattı yada bilgisayar ağı olabilir. Alice'nin Bob'a göndermek istediği mesaja orijinal veri denir. Bu mesaj bir metin dosyası, sayısal veri, sayısal video görüntüsü, bir dizi bit, bitmap ya da sayısallaştırılmış ses olabilir. Alice, orijinal veriyi daha önce belirlenmiş bir anahtar yardımıyla şifreler. Bu işleme, şifreleme (encyption) adı verilir. Sonuçta ortaya çıkan mesaja şifrlenmiş veri adı verilir ve kanal üzerinden gönderilir. Oscar, şifrelenmiş veriyi kanal üzerinden gizlice dinlediğinden orijinal verinin ne olduğunu anlayamaz. Ama Bob, şifreleme anahtarını bildiği için orijinal veriyi tekrar elde eder. Bu işleme çözme (decryption) adı verilir. Bu işlemler dizisi aşağıdaki şekilde görülmektedir (Sönmez, 2002).



Şekil 2.1. Şifreleme ve Şifre Çözme

Şifreleme ve çözme genelde bir anahtar kullanılarak yapılır ve kodlama metodu öyledir ki çözme işlemi ancak doğru anahtarın bilinmesiyle gerçekleştirilebilir.

Kriptografi mesajların gizli tutulması sanatı veya bilimidir. Kriptanaliz şifrelerin kırılması sanatıdır, örneğin doğru anahtarı bilmeden düz-metnin elde edilmesi. Kriptografi ile uğraşanlara *kriptograf* ve kriptanalizin uygulamacılarına *kriptanalist* denir.

Kriptografi gizli mesajlaşma, onaylama, dijital imzalar, elektronik para ve diğer uygulamaların tüm yönleriyle ilgilidir. Kriptoloji, kriptografik metotların matematiksel temelleriyle ilgilenen bir matematik dalıdır.

3. ŞİFRELEME (KRİPTOGRAFİ) TARİHÇESİ

En basit tanımıyla kriptografi, mesajın harflerin değiştirilmesi ile içeriğinin gizlenmesi işlemidir. Harflerin yer değiştirmesinde kullanılan şablona (örneğin a=k, b=l, c=m, d=n, vb.) anahtar ve anahtarın kullanılmasıyla elde edilen okunamayan, anlaşılması zor metine "ciphertext" denilmektedir. Numaralarda harfler gibi yerine koyulan karakterler olarak kullanılabilmektedir.

Yapılan araştırmalar bilgiyi gizleme ve koruma olayının yaklaşık 4000 yıllık bir serüvene sahip olduğunu ortaya koymaktadır. Eski Mısırlıların, M.Ö. 1900 lü yıllarda bir tanrının efsanesini anlatan hiyerogliflerin bazı yerlerinde ilginç sembolleri kullandığı görülmüştür. İncelemeler, yazının bazı yerlerinin bu değişik sembollerle kodlandığını göstermiştir. Mısırlılardan sonra Mezopotamyalılar ve İbraniler de bazı metinleri benzeri şekillerde kodlamışlardır (Andiç, 2002).

İnsanlar kelimeleri yazmaya başlar başlamaz, bu kelimeleri meraklı gözlerden saklamaya da başladılar. Çağlar boyunca icat edilen sayısız kripto sistemler genellikle mürekkep ve kalem tasarımı olmuştur, genellikle zekice olan bu sistemler "kırılamaz" payesini nadir olarak alabilmişlerdir. Karmaşıklıkları sadece kalem ve kağıt kullanarak uygulanacak algoritma için gereken zaman ile kısıtlıydı.

En ilginç erken kripto sistemlerden biri M. Ö. 400 yıllarında Spartalılar tarafından geliştirilmiş olan "scytale" denilen sistemdir (Andiç, 2002). Bu sistemde bir mesajı "encrypt" etmek için uzun bir parşömen yada papirüs silindirik bir sopa etrafına sarılıyordu. Gizlenecek mesajın kelimeleri uzunlamasına sopa üzerine, her bir şerit turunda 1 harf gelecek şekilde yazılıyordu. Daha sonra şerit açılır ve kaldırılırdı böylece anlamsız harflerin oluşturduğu metin ortaya çıkardı. Mesajın "decrypt" (şifreçözme) edilebilmesi için gereken kritik şart "encrypt" işleminde kullanılan silindirle *aynı çapa* sahip silindir kullanılması şartıydı. Farklı çaptaki silindirler anlamsız metinlerin ortaya çıkmasına sebep oluyordu.

En önemli gelişmelerden biri 2000 yıl önce Julius Caesar'ın Galya savaşlarında kendi adıyla anılan şifreleme yöntemini kullanmasıdır.Caesar bu yöntemde her harfi kendinden sonra gelen üçüncü harf ile değiştirmiştir. Bu basit yerine koymalı şifrelemenin örneklerindendir. Daha sonra Caesar'ın kullandığı yöntem geliştirilerek

monoalfabetik yerine koymalı şifre geliştirilmiştir. Monoalfabetik yerine koymalı şifrede alfabedeki her harfin yerine gönderici ve alıcının bildiği bir harf konulur. Caesar şifresinden farkı bir düzene bağlı olmamasıdır. Bu şifreleme yönteminin basitliği ve aksine bir o kadar güçlü olması M.S. ilk bin yılda yaygın olarak kullanılmasına neden olmuştur. Arap bilim adama Al-Kindi'nin 1987 de İstanbul'da bulunan el yazmasından monoalfabetik yerine koymalı şifrenin Araplar tarafından kırıldığını ve onların bu konudaki bilgisini göstermektedir. Şifrelerin çözülmesi yeni şifrelerin geliştirilmesine yol açmıştır. Bunun sonucu olarak her harfin, ortak kelimelerin ve boşlukların yerine birden fazla sembol yerleştirilmeye başlanmıştır. Bu şifrenin kayıtlara geçen örnekleri İngiltere kraliçesi I. Elizabeth'e yapılan suikast planları ve 17. yy Fransa'sında XIV Louis'in Büyük Şifresidir (Great Cipher). Louis'in bu şifresi, 2 yy boyunca kumandan Etienne Bazeries tarafınadan kırılana kadar çözülememiştir (Andiç, 2002).

Bu alanda 20. yüzyıla kadar ciddi bir ilerleme olmadı. 20. yüzyılda askeri olarak yaşanan gelişmeler genellikle güvenli olmayan telefon ve radyo hatlarının kullanımını gerektirdiğinden güçlü kripto sistemler, kod üretimi ve kırma teknolojilerinde yoğun yatırımlar ortaya çıktı.

Çağın ilk zamanlarında otomatik olarak "encrypt" ve "decrypt" işlemini yapan mekanik makineler ortaya çıktı, ayrıca bunlar yüksek askeri daireler dışında bilinmez olarak kaldı. Bu makineler daha uzun, daha güvenli anahtarlar ve daha karmaşık algoritmalara olanak sağladı.

İttifak devletlerinin ünlü Alman "Enigma" ve Japonların "Purple" kodlarını kırmaları II. Dünya Savaşının sonucunu belirleyen faktörler olarak görülür (Andiç, 2002).

Nasıl mekanik "encryption" kriptografi alanında dev bir adım ise, savaştan sonra bilgisayarın keşfi ve geliştirilmesi de kriptografiyi tamamen yeni bir boyuta sürükleyen bir adımdı. Böylece anahtarlar daha uzun, "cipher"lerde daha karmaşık olabildi. Anahtar Yönetimi olarak bilinen anahtar seçimi, dağıtımı ve kurulumu böylece tamamen otomatikleşti. Bunlar güvenliğin gelişmesi ve kullanımının artmasına sebep oldu. Ayrıca önceden ordunun elinde hep gizli kalan kriptografi gizliliğini yitirdi. Bu ayrıca, istenmeyen antikriptografik, kırıcı faaliyetlerin artmasına sebep oldu.

4. TEMEL ŞİFRELEME ALGORİTMALARI

Bazı kriptografik metotlar algoritmanın gizliliğine dayanır; böyle algoritmaların sadece tarihi önemi vardır ve gerçek-dünya ihtiyaçları için yeterli değildir. Tüm modern algoritmalar şifreleme ve çözme işlemlerini kontrol etmek için bir anahtar kullanırlar; bir mesaj sadece kullanılan anahtar şifreleme anahtarıyla uyuştuğunda çözülebilir. Kriptoloji algoritması, şifrelemek ve çözmek için kullanılan matematiksel bir fonksiyondur. Genellikle biri şifrelemek diğeri de çözmek için birbiri ile ilişkili iki fonksiyon kullanılır (Sönmez, 2002). Orijinal veri M (Message) ile, şifrelenmiş veri ise C (cipher) ile gösterilir. Şifreleme fonksiyonu E (Encryption), M üzerinde işlem yapar ve C elde edilir. Matematiksel olarak,

$E(M) = C$ şeklindedir.

Şifreleme fonksiyonunun tersi olan çözme fonksiyonu D (Decryption), C üzerinde işlem yapar ve M elde edilir.

$D(C) = M$

Bir verinin şifrelenmesi ve çözülmesi işleminde en önemli nokta orijinal verinin doğru olarak tekrar elde edilmesini sağlamaktır. Böylece aşağıda verilen özdeşlik sağlanmalıdır.

$D(E(M)) = M$

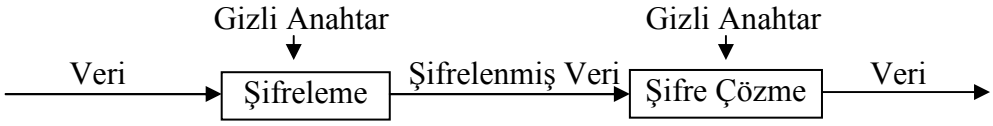
Algoritmanın güvenliği, algoritmanın nasıl işlediğini gizli tutmakla sağlanıyor ise böyle algoritmalara sınırlı (restricted) algoritma adı verilir. Sınırlı algoritmalar geçmişte oldukça kullanılmıştır. Ancak bugünün standartlarını gerçekleştirmekte yetersiz kalmıştır. Çok sayıda kişinin çalıştığı gruplarda, çalışan kişiler değiştikçe gizliliği korumak için algoritmanın da değiştirilmesi gerekmektedir. Şirketler bu algoritmaları yetkisiz kişilerin de satın alabileceğini düşünerek kendi algoritmalarını kendileri yazma yoluna gitmişlerdir. Ancak iyi bir kriptoloji uzmanı tarafından yazılmamışsa algoritmanın güvenli olup olmadığı hiçbir zaman bilinememektedir (Sönmez, 2002).

Modern kriptoloji bu problemi anahtar yardımıyla çözer. Anahtar K (Key) harfi ile gösterilmektedir. Anahtar, büyük sayı değerleri olabilir. Anahtarın alabileceği değer aralığına anahtar uzayı (key space) denir. Hem şifreleme işlemi hem de çözme işlemi

anahtara bağlıdır ve bu değişiklik yukarıda vermiş olduğumuz eşitlikleri aşağıdaki gibi etkiler (Sönmez, 2002). Tek anahtar ile şifreleme yöntemi aşağıda gösterilmektedir.

$E_K(M)=C$, $D_K(C)=M$

Bu fonksiyonlarda $D_K(E_K(M))=M$ eşitliğini sağlamak durumundadır.

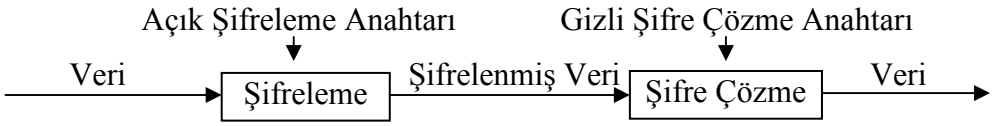


Şekil -4.1. Tek Anahtarla Şifreleme ve Şifre Çözme

Bazı algoritmalar birbirinden farklı şifreleme ve çözme anahtarı kullanır. Bu algoritmalarda K_1 şifreleme anahtarını, K_2 ise çözme anahtarını gösterir. İki farklı anahtar ile şifreleme yöntemi aşağıdaki şekilde verilmiştir.

$E_{K1}(M)=C$, $D_{K2}(C)=M$, $D_{K2}(E_{K1}(M))=M$

Bu algoritmalarda, algoritmanın güvenliği anahtara ya da anahtarlara bağlıdır. Bu algoritmalar herkes tarafından bilinir, analiz edilebilir ve üretilebilir. Kullanılan anahtar bilinmiyor ise algoritmanın yetkisiz kişiler tarafından bilinmesi bilginin güvenliği açısından herhangi bir risk taşımaz.



Şekil -4.2. İki Farklı Anahtarla Şifreleme ve Şifre Çözme

Kullanılan şifreleme anahtarının özelliğine bağlı olarak, iki farklı şifreleme yöntemi bulunmaktadır. Bunlar:

1- Simetrik (Gizli anahtar) şifreleme algoritmaları

- Akış şifreler (stream ciphers)
- Blok şifreler (blok ciphers)

2- Asimetrik (Açık anahtar) şifreleme algoritmalarıdır.

4.1. Simetrik (Gizli Anahtar) Şifreleme Algoritmaları

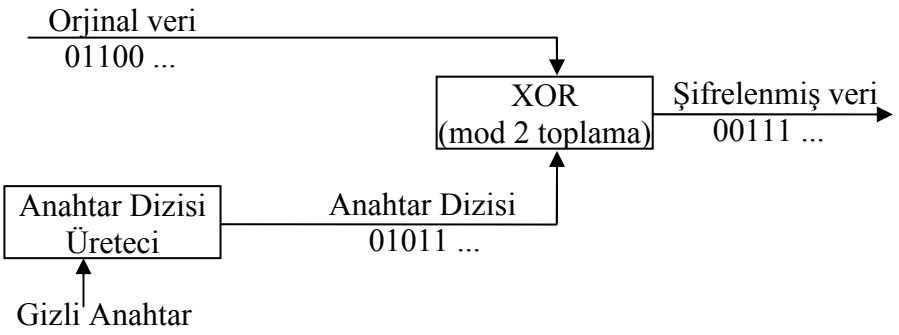
Simetrik algoritmalara bazen geleneksel algortimalar da denir. Şifreleme anahtarının çözümleme anahtarından hesaplanabildiği bir algoritmadır. Çoğu simetrik algortimada şifreleme ve çözümleme anahtarları aynıdır. Bu algoritmalar, aynı zamanda gizli anahtar algoritmaları, tek anahtar algoritmaları veya bir(aynı) anahtar algoritmaları diye de adlandırılır, gönderici ile alıcının güvenle iletişime başlamadan once bir anahtar üzerinde anlaşmalarını gerektirir. Bir simetrik algoritmanın güvenliği anahtara dayanır; anahtarın açığa çıkması herkesin mesajları şifreleyebileceği ve çözebileceği anlamına gelir. İletişimin gizli kalması gerektiği sürece, anahtar gizli kalmalıdır.

Simetrik algortimalar iki sınıfa ayrılabilir. Bazıları belli bir anda bir bitlik açık-metni şifreleyebilir; bunlar *akış(stream) algoritmaları* veya *akış şifreleri* olarak adlandırılır. Diğerleri açık-metini bit gruplarına bölerek işler. Bit grupları *blok* ve algoritmaları *blok algoritmaları* veya *blok şifreleri* diye adlandırılır. Çağdaş bilgisayar algoritmaları için, tipik bir bit bloğunun uzunluğu 64 bittir.

4.1.1. Akış şifreler

Akış şifreleme (stream ciphers) algoritmaları, orijinal veri olarak bit dizilerini almakta çıktı olarak da bit dizileri üretmektedirler.

Akış şifreleme işleminde ilk olarak gizli anahtar ve üreteç yardımı ile bir anahtar dizisi oluşturulmakta daha sonra bu dizi ve girdi mesajın her bir biti ayrı ayrı XOR (mod 2 toplama) işlemine tabi tutulmaktadır. Çözme işlemide aynı şekilde gerçekleştirilmektedir. Akış şifreleme algoritmalarının güvenlikleri anahtar dizisi üreteci tarafından yaratılan dizinin ne kadar rastgele olduğuna bağlıdır. Bu nedenle anahtar dizisi olarak tamamıyla rastgele verinin kullanılması en ideal durumdur. Gerçek uygulamalarda tam anlamı ile rastgele anahtar dizisi yaratılması imkansız olduğundan, anahtar üreteci ve onun girdisi olan gizli anahtar yardımı ile anahtar dizileri yaratılmaktadır. Diğer bir deyişle, bir akış şifreleme algoritmasının en önemli bileşeni kullandığı anahtar dizisi üretecidir. Yaratılan anahtar dizisinin kendini tekrarlamaması ve sonraki anahtar bitlerinin öncekiler yardımı ile elde edilememesi anahtar üreteçlerinin sağlaması gereken önemli özelliklerindendir (Sönmez, 2002).
Aşağıdaki şekil akış şifreleme algoritmasını göstermektedir.



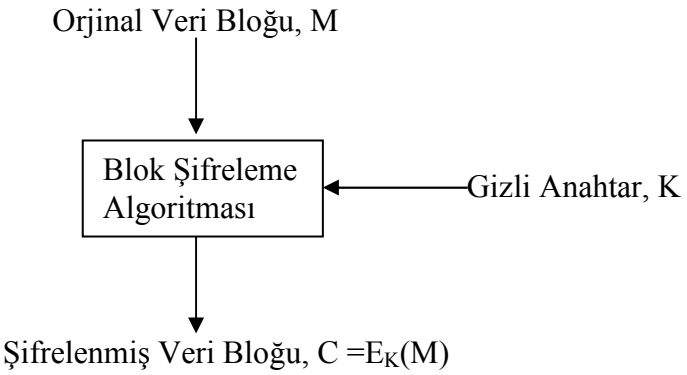
Şekil -4.3. Akış Şifreleme Algoritması

Akış şifreleme algoritmaları, hem hızlı olmaları hem de basit bir yapıya sahip olmaları nedeni ile bir çok uygulama alanına sahiptir. 256 byte uzunluğuna kadar gizli anahtarlarla çalışabilen RC4, akış şifreleme algoritmaları içinde en yaygın olarak kullanılanıdır (Sönmez, 2002).

4.1.2. Blok şifreler

Blok şifreleme algoritmaları, orijinal veri olarak bit gruplarını alır. Bu bit gruplarına blok adı verilirken, kullanılan algoritmalara da blok şifreleri (blok ciphers) denir.

Modern bilgisayar algoritmalarında genel olarak tipik blok boyutu, üzerinde analiz yapılmasını engellemeyecek kadar büyük ve çalışma yapılabilecek kadar küçük olmasını sağlamak amacı ile genel olarak 32, 64 veya 128 bit olarak seçilmiştir. Bununla birlikte sözlük ataklarını önleyebilmek amacı ile blok boyunun 64 bit ve üzerinde seçilmesi önerilmektedir (Sönmez, 2002).



Şekil-4.4 Blok Şifreleme Algoritması

Yukarıdaki şekil, bir mesaj bloğunun blok şifreleme algoritması yardımı ile şifrelenmesi işlemini göstermektedir.

Feistel ağları ve Yerdeğiştirme-Permütasyon ağları (SPN) olmak üzere iki ana blok şifreleme mimarisi vardır. Her ikisi de yerdeğiştirme ve lineer transformasyonu kullanır. Ayrıca her iki mimari ürün şifrelerinin örneklerindendir. Yani birden fazla şifreleme işleminin birleşmesi ile oluşturulurlar. Tekrarlanan şifreler yine ürün şifreleridir ve aynı şifreleme adımının tekrarlanan uygulamasını içerir ve her şifreleme adımına *döngü* denir. Bir döngü birden fazla şifreleme adımı içerebilir. Genellikle her döngüde farklı anahtar materyali kullanılır. SPN ve Feistel mimarileri en yaygın mimariler olmakla beraber bazı şifreleme algoritmaları bu mimari yapılara uymamaktadır. Bununla beraber çoğu tekrarlanan döngülerden oluşmaktadır.

Blok şifreleme algoritmalarında aynı girdi bloğu aynı anahtarla şifrelendiğinde her zaman aynı çıktı bloğunu oluşturmaktadır. Akış şifreleme algoritmalarında ise bir mesaj içinde farklı pozisyonlarda yer alan aynı girdi blokları farklı çıktılar oluşturmaktadır.

Kullanıcı sayısı arttıkça sistemde tanımlanan gizli anahtar sayısı artmakta, böylece anahtarların güvenilir yollarla iletimi ve saklanması büyük sorunlar oluşturmaktadır. Bu problemlerin çözümü için açık anahtar şifreleme teknikleri kullanılmaktadır.

4.1.2.1. Blok şifreleme sistemlerinin parametreleri

Blok Uzunluğu : Bir blok şifre sisteminin güvenli olabilmesi için, blok uzunluğunun bazı blokların diğerlerinden daha fazla görünmeyeceği şekilde uzun olması gerekir. Örneğin bir blok uygulama sistemi olan DES'teki 64 bit uzuluk, sıklık analizine karşı DES'i güçlü kılmaktadır. Aynı zamanda blok uzunluğu n olan bir blok için, sabit bir anahtarla saldırı yapan kişinin elde edebileceği açık metin – şifreli metin çiftlerinin sayısı büyük olmalıdır (bu sayı 2^n yi geçemez). Blok uzunluğu büyüdükçe sistemin uygulaması da daha karışık hale gelmektedir.

Anahtar ve gerçek anahtar uzunluğu : Bir blok şifre sisteminin anahtarı deneme-yanılma (axhaustive key search) ile bulunamamalıdır. Bunun için de anahtar uzun olmalıdır. Diğer taraftan da anahtar uzunluğu üretim dağıtım ve saklama için uygun ve

güvenilir olmalıdır. Öneğin DES her zaman anahtar uzunluğunun kısa olmasından dolayı eleştirilmiştir. Diffie ve Hellman DES'in anahtarını deneme-yanılma yolu ile 20 milyon dolara mal olacak bir sistemle 12 saatta kırılabileceğini öne sürdüler. Gelen öneriler doğrultusunda, DES'in gerçek anahtar uzunluğu 128 bite çıkarıldı ve üçlü şifreleme ile DES daha güvenli bir şekilde kullanılabilir hale getirildi.

4.1.2.2. Blok şifreleme sistemlerinin tasarım ölçütleri

Güvenli bir blok şifre sisteminin kırılması zor ama uygulaması kolay olmalıdır. Şifreleme ve deşifreleme fonksiyonlarının kolay uygulanabilir olması gerekirken, $C=E_K(M)$ ve $M=D_K(C)$ eşitliklerinden K yı bulmanın zor olması gerekir. İlk defa Claude Shannon tarafından önerilen tasarım ölçütleri yayılma (confusion) ve nüfuz etmedir (diffusion).

Yayılma : Bir blok şifre sistemini yada genel olarak bir şifreleme sistemini yayılma ölçütüne göre tesarlamak demek, şifreli metinle anahtar arasındaki ilişkiyi mümkün olduğunca karışık yapmaktır. Daha açık bir tanım verirsek, yayılma, anahtarın açık ve şifreli metne bağlılığının kriptanaliz için faydalı olmayacak kadar karışık olması demektir. Yani blok şifre sistemini tanımlayan eşitliklerin doğrusal olmaması; karışık olması ve böylece C=E(P,k) denkleminden anahtarı bulmanın imkansız olması gerekir.

Nüfuz etme : Bu ölçüte göre her anahtar için şifreleme fonksiyonu öyle olmalı ki, açık metin ve şifreli metin yapıları arasında istatistiksel bağlılık olmamalıdır. Bu ölçütün olabilmesi için anahtarın ve açık metinin her bitinin şifreli metini etkilemesi gerekir.

4.2. Asimetrik (Açık Anahtarlı) Şifreleme Algoritmaları

Asimetrik şifreler (simetrik olmayan) açık-anahtar şifreleme algoritmaları veya açık-anahtar kriptografisi olarak da adlandırılırlar. Bu algoritma şifreleme için kullanılan anahtarın çözümleme için kullanılacak anahtardan farklı olması için tasarlanmıştır. Daha da ötesi, çözümleme anahtarı şifreleme anahtarından, en azından makul bir zaman zarfında hesaplanamaz. Bu algoritmalar "genel anahtar" diye adlandırılır. Çünkü şifreleme anahtarı kamuya açılabilir. Tamamıyla yabancı bir kişi bir mesajı şifrelemek için şifreleme anahtarını kullanabilir, fakat sadece belirli bir kişi

mesajı karşılık gelen çözümleme anahtarı ile çözebilir. Şifreleme anahtarı açık-anahtar, ve çözme anahtarı da özel veya gizli anahtar olarak da adlandırılır. Modern kriptografik algoritmalar artık kalem-kağıt şifreleyicisi olmaktan çok uzaktır. Güçlü kriptografik algoritmalar bilgisayarla veya özelleştirilmiş cihazlarda çalıştırılmak üzere tasarlanmaktadır. Pek çok uygulamada kriptografi, bilgisayar yazılımlarıyla yapılmaktadır. Genel olarak, simetrik algoritmalar bilgisayarda asimetrik olanlardan çok daha hızlıdırlar. Açık anahtar tabanlı kriptolojik yöntemler, veri gizliliği ve sayısal imzalama konularında kendilerine uygulama alanı bulmaktadır. Veri gizliliği, bilgilerin gizliliği kavramının hayata geçirilmesine yardımcı olurken sayısal imzalama ile iki kişi ya da uygulama arasında akan verinin bütünlüğü ve veriyi yaratanın kimliği garanti altına alınmaktadır.

Uygulamada bunlar sık sık beraber kullanılırlar öyleki açık-anahtar algoritmalar rasgele üretilmiş bir şifreleme anahtarını şifrelemek için kullanılır ve rasgele anahtar hakiki mesajı simetrik bir algoritma kullanarak şifrelemek için kullanılır. Bu bazen, hibrid (melez) şifreleme olarak adlandırılır. En çok çalışılan ve muhtemelen en yaygın simetrik cipher DES'tir; yeni geliştirilen AES en yaygın şifreleme algoritması olarak DES in yerini alabilir. RSA muhtemelen en iyi bilinen asimetrik şifreleme algoritmasıdır.

Günümüzde, dayandıkları matematiksel problemlere göre 3 çeşit farklı açık anahtar kriptoloji tekniği bulunmaktadır. Bunlar:

-*Büyük sayıları çarpanlarına ayırma ilkeli tabanlı sistemler:* RSA

-*Ayrık logaritma sistemleri:* Diffie Hellman Açık Anahtar Kriptoloji Algoritması ve El- Gamal

- *Eliptik eğri ayrık logaritma sistemleri:* Eliptik Eğri Kriptoloji Teknikleri

5. ŞİFRELEME (KRİPTOGRAFİ) ALGORİTMALARININ GÜCÜ

İyi kriptografik sistemler öyle tasarlanmalıdırlar ki bunları kırmak mümkün olduğu kadar zor olsun. Uygulamada kırılamayacak sistemler yapmak olasıdır (ancak genelde bu ispatlanamaz). Bir sistem tasarımcısı için sistemi kırılabilir bırakmanın hiçbir özrü yoktur. Güvenliği aşmak için kullanılabilecek her mekanizma açığa çıkarılmalı, belgelenmeli ve son kullanıcının dikkatine sunulmalıdır. Teoride, herhangi bir anahtarlı kriptografik metot olası tüm anahtarların denenmesi ile kırılabilir. Eğer tüm anahtarların denendiği kaba kuvvet (brute force) kullanımı tek yolsa, gerekli hesaplama gücü anahtarın uzunluğu ile üstel olarak artar. 32 bitlik bir anahtar (4 294 967 296), 10^6 şifre/sn hızında bir işlem gücü ile ortalama 36dk'da çözülmektedir. Bu herhangi bir ev bilgisayarı ile yapılabilecek bir şeydir. 40 bitlik anahtarlar yine 10^6 şifre/sn hızında bir işlem gücü ile ortalama 6 günde çözülebilmektedir. Bu tür bir hesaplama (kullanılan algoritmanın etkinliğine bağlı olarak) modern bir ev bilgisayarında bir hafta gibi bir zaman gerektirir. 56 bit anahtarlı bir sistem (DES gibi) esaslı bir zahmet gerektirir (çok sayıda ev bilgisayarının güç paylaşımı ile bunu kırmak birkaç ay alır), ama özel donanımlarla kolayca kırılabilir. Özel donanımların maliyetleri de doğal olarak yüksektir, ama organize suç örgütleri, büyük hükümet ve şirketler bunları alabilirler. 64 bitli anahtarlarda şimdiden kırılabilir durumdadırlar. 80 bitli anahtarlar bir kaç yıl sonra kırılabilecekken ve 128 bitli anahtarlar kaba kuvvet ile muhtemelen kırılamayacaklardır. Ancak anahtar uzunluğu tek önemli konu değildir. Pek çok şifre olası tüm anahtarlar denenmeden de kırılabilir. Çoğunlukla, algoritmanın gizliliğine dayanan algoritmalar güvenli değillerdir.

Açık-anahtarlı kriptografide kullanılan anahtarların uzunluğu simetrik şifrelerde kullanılanlardan daha uzundur. Bunun nedeni, kriptanalistler için kullanılan ekstra yapıdır. Burada problem doğru anahtarın tahmin edilmesi değil, gizli anahtarın açık-anahtardan türetilmesidir. RSA da bu işlem iki asal çarpanı olan bir tamsayının üretilmesi ile yapılmaktadır. RSA kripto-sisteminde, 256 bitlik bir anahtar evde kolayca ve 512 bitlik anahtarlar üniversitedeki araştırma grupları tarafından birkaç ay içinde kırılabilir. 768 bitlik anahtarlar muhtemelen uzun vadede güvende sayılmazlar. 1024 ve daha büyük bitli anahtarlar RSA ya karşı büyük kriptografik ilerlemeler kaydedilmedikçe güvende sayılırlar.

6. DONANIM VE YAZILIMLA ŞİFRELEMENİN KARŞILAŞTIRILMASI

6.1. Donanımla Şifreleme Yöntemi

Uzun yıllardır şifreleme ürünleri donanım olarak gerçekleştirilmiştir. Donanım uygulamaları hala askeri ve ciddi ticari firmalarının seçimidir. Son yıllarda yazılım uygulamaları daha fazla kullanılmaya başlanmıştır.

Örneğin, Ulusal Güvenlik Kurumu (NSA) sadece donanım uygulamalarını onaylamaktadır. Bunun ilk nedeni hızdır. Şifreleme algoritmalarında birçok bit değiştirme işlemleri yapılmaktadır. Yapılan işlemler bilgisayar yapısına uygun değildir. Genelde kullanılan şifreleme algoritmalarından DES ve RSA, genel amaçlı işlemcilerde iyi sonuç vermemektedir. Kriptografi uzmanları birçok yazılım uygulamaları gerçekleştirmiştir. Fakat, herzaman donanım uygulamaları hız yarışını kazanmıştır.

Diğer bir neden güvenliktir. Bilgisayar ortamında çalışan şifreleme algoritması fiziksel olarak korunamaz. Yetkili olmayan kişiler, algoritmayı kimsenin haberi olmadan değiştirebilirler. Donanım şifreleme aygıtları değiştirilemez. Özel amaçlı VLSI yongaları kimyasal maddeler içine konarak herhangi bir girişimde yonganın mantığının bozulması sağlanmaktadır. Aynı zamanda yongalar şifrelenmemiş anahtarın okunmasını engelleyecek bir şekilde tasarlanırlar (Sönmez, 2002).

Son neden ise donanımı yükleme kolaylığıdır. Birçok şifreleme uygulaması, genel amaçlı bilgisayar yardımı ile uygulanmamaktadır. Kişiler telefon konuşmalarını veya verilerini şifrelemek isteyebilirler. Telefona, fotokopi makinasına ya da modeme bir şifreleme aygıtının konması karmaşık bir şifreleme yazılımının ve mikroişlemcilerin kullanılmasından daha ucuza mal olmaktadır.

Şifrelenmiş veri, bilgisayar ortamından gelse de belirlenmiş bir donanım şifreleme aygıtını yüklemek bilgisayarın yazılımını değiştirmekten daha kolaydır. Şifreleme işlemi görünmez olmalıdır ve kullanıcıya zorluk çıkarmamalıdır. Bunu yazılım uygulamalarında yapmanın tek yolu şifreleme algoritmasının işletim sisteminin ya da uygulama programlarının içine yerleştirilmesidir. Fakat bu işlemi gerçekleştirmek pek kolay değildir (Sönmez, 2002).

6.2. Yazılımla Şifreleme Yöntemi

Doğal olarak bütün şifreleme algoritmaları yazılım uygulaması olarak gerçekleştirilebilir.

Dezavantajları,

- Hızı
- Maliyeti
- Kolay değiştirilebilir olmasıdır.

Avantajları,

- Esneklik
- Taşınabilirlik
- Kolaylıkla kullanılabilmesi
- Yenilenebilir olması
- Ucuz bir şekilde kopyalanabilir
- Birçok makinaya yüklenebilir
- Daha geniş uygulamaların içine yerleştirilebilir olmasıdır.

Şifreleme yazılım programları birçok işletim sistemi için bulunmaktadırlar ve popülerdirler. Kullanıcı kimi zaman kendi eli ile şifreleme ve çözme işlemlerini gerçekleştirmektedir. Fakat anahtar yönetimi güvenli olmak zorundadır. Anahtar ve şifrelenmemiş dosyalar şifreleme işleminden sonra silinmelidir. Kesinlikle disk üzerinde saklanmamalıdır.

Güvenlik yazılım uygulamaları, yaşamımızın çoğu yerinde karşımıza çıkmaktadırlar. Bu yazılım uygulamalarının birçoğu esnekliği nedeni ile yine bilgisayar programlarının içinde yer almaktadırlar. Esnekliği olmayan ve müdahaleye daha az imkan veren kimi hassas uygulamalarda donanım daha uygun bir çözüm olmaktadır (Sönmez, 2002).

7. DES - VERİ ŞİFRELEME STANDARDI

DES (Fips 46-3, 1999), Data Encryption Standart (Veri Şifreleme Standartı) kısaltmasıdır. IBM tarafından Lucifer adıyla 1970'lerin başlarında geliştirilmiştir. NSA (Ulusal Güvenlik Ajansı) ve NIST (Uluslararası Standartlar Enstitüsü ve Teknolojisi) tarafından desteklenmiştir.

Veri şifreleme standardı (DES), günümüze kadar gelebilen ve ABD tarafından onaylanan ilk ve tek halka açık kriptolojik algoritmadır. Bir dünya standardı olan Veri Şifreleme Standardı, ANSI tarafından Veri Şifreleme Algoritması (DEA), ISO tarafından da DEA-1 olarak bilinir. DES senelerin kriptanaliz uzmanlarına karşı 25 yıldır güvenli bir algoritma özelliğini devam ettirmektedir (Sönmez, 2002).

DES, 64-bit bloklarda veriyi şifreleyen bir blok şifredir. 64-bit açık metni, 56-bit anahtar altında 64-bit şifreli metine çevirir. DES bir simetrik algoritmadır. Aynı algoritma ve anahtar hem şifrelemede hem de deşifrelemede(şifre çözme) kullanılır.

7.1. DES'in Gelişim Süreci

1970'li yılların başlarında askeri olmayan kriptografik araştırmalar gelişigüzel devam etmiştir. Bu konu üzerinde hiç denecek kadar az araştırma makaleleri yayınlanmıştır. Askerlerin haberleşmek için özel bir kodlama aracı kullandığı bilinmekte, ancak kriptografi bilimini çok az insan bilmektedir.

NSA, bu konuda birçok bilgiye sahip olmuştur. Fakat kamuya bu konuda detaylı bilgi vermemiştir. Birçok küçük şirket kendi kriptografik araçlarını tasarlamıştır. Bu kriptografik araçlar hem birbirlerinden farklı olmuş hemde birbirleri ile uyum içerisinde çalışamamışlardır. Hiç kimse bu araçların güvenli olduğu hakkında bir bilgiye sahip olamamıştır.

1972'de Ulusal Standart ve Teknoloji Enstitüsü (NIST) olarak bilinen Ulusal Standartlar Bürosu (NBS), bilgisayar ve veri haberleşmesini korumak için bir program geliştirmiştir. Bu program kapsamında standart bir kriptografik algoritma gerçekleştirilecektir. Bu algoritma test edilebilecek, doğrulanabilecek ve farklı kriptografik araçlar birbirleri ile uyum içinde çalışabileceklerdir.

1973'te NBS, tasarım kriterlerini de belirttiği standart bir kriptografik algoritma için halka açık bir teklifte bulunmuştur. Bu kriterler:

- Algoritma yüksek düzeyde güvenlik sağlamalıdır.
- Algoritma tamamen tanımlanmalı ve anlaşılması kolay olmalıdır.
- Algoritmanın güvenliği anahtara dayanmalıdır; güvenlik algoritmanın gizliliğine bağlı olmamalıdır.
- Algoritma tüm kullanıcılar tarafından elde edilebilir olmalıdır.
- Algoritma çeşitli uygulamaların kullanımına uyarlanabilmelidir.
- Algoritma elektronik aygıtların ekonomik olarak gerçekleştirilmesini sağlamalıdır.
- Algoritma verimli olarak kullanılabilmelidir.
- Algoritma onaylanabilmeli ve test edilebilmelidir.
- Algoritma ihraç edilebilir olmalıdır.

Fakat hiçbir teklif istenen şartları sağlayamamıştır. NBS, 1974 Ağustos'unda ikinci bir istekte bulunmuştur. Sonunda IBM tarafından 1970'li yılların başlarında geliştirilen Lucifer algoritması umut vaat etmiştir. Karmaşık olduğu halde oldukça düz olan bu algoritma küçük bit grupları arasında basit mantıksal işlemler kullanılarak donanımsal olarak gerçekleştirilebilmiştir.

NBS, algoritmanın federal bir standart olmasını sağlamak için güvenli ve uygun olduğu konusunda NSA'nın yardımını istemiştir. IBM patent için başvurmuştur. Fakat algoritmanın tüm özelliklerinin herkes tarafından bilinmesini, imal edilmesini ve kullanılmasını istemiştir. NBS, IBM'e gerçekleştirdiği algoritmanın kullanımını ve donanımının satılması için açık olmayan lisansları herkese vermiştir.

1975'te sonunda NBS, algoritmanın detaylarını ve herkese açık olan lisansları yayımlamıştır. Halktan ve kurumlardan algoritma hakkındaki görüşlerini talep etmiştir. Bu konudaki raporlar ve atölye çalışmalarından sonra 23 Kasım 1976'da Veri Şifreleme Standardı federal bir standart olarak kabul edilmiş ve hükümetin gizli olmayan haberleşmesinde yetkili kılınmıştır. Standardın resmi tarifi Federal Bilgi İşleme Yayımları FIPS PUB 46, 1977 yılında "Veri Şifreleme Standardı" olarak yayımlanmıştır. 1980 yılında DES yöntemlerinin anlatıldığı FIPS PUB 81 yayımlanmıştır (Sönmez, 2002).

Bu standartlar, NSA'nın halka açıkladığı ilk standartlardır. Bunun nedeni NSA ve NBS arasındaki yanlış anlaşılmalardır. NSA, DES'i sadece donanım olarak

düşünmüştür. Fakat NBS algoritmayı oldukça detaylı anlatmıştır. Böylece DES'in yazılımları yapılabilmiştir (Sönmez, 2002).

7.2. DES Algoritması

Algoritma, mesajı hem şifrelemek hem de çözmek için kullanılmaktadır. İki tane 64 bitlik giriş veri bloklarını alır. Biri orijinal veri (şifrelenmiş veri) diğeri ise şifreleme anahtarı bloğudur. Çıkıştan 64 bitlik şifrelenmiş veri bloğu alınır. Algoritma çalışma moduna göre orijinal veriyi, şifrelenmiş veriye ya da şifrelenmiş veriyi, orijinal veriye dönüştürür. 64 bitlik şifreleme anahtar bloğunun sadece 56 biti direkt olarak algoritmaya girer. Federal standartlara göre kalan 8 bit tek eşlenik kontrolü (odd parity check) için kullanılır. Bu bitler her byte'ın LSB (en az önemli) bitleridir. 56 bitlik anahtar istendiği zaman değiştirilebilir. Çünkü algoritmanın güvenliği anahtarın güvenliğine bağlıdır. Algoritma permütasyon, yerine koyma (substitution) ve mod 2 (XOR) işlemlerine dayanmaktadır. DES'te 3 çeşit permütasyon bulunmaktadır. Bunlar:

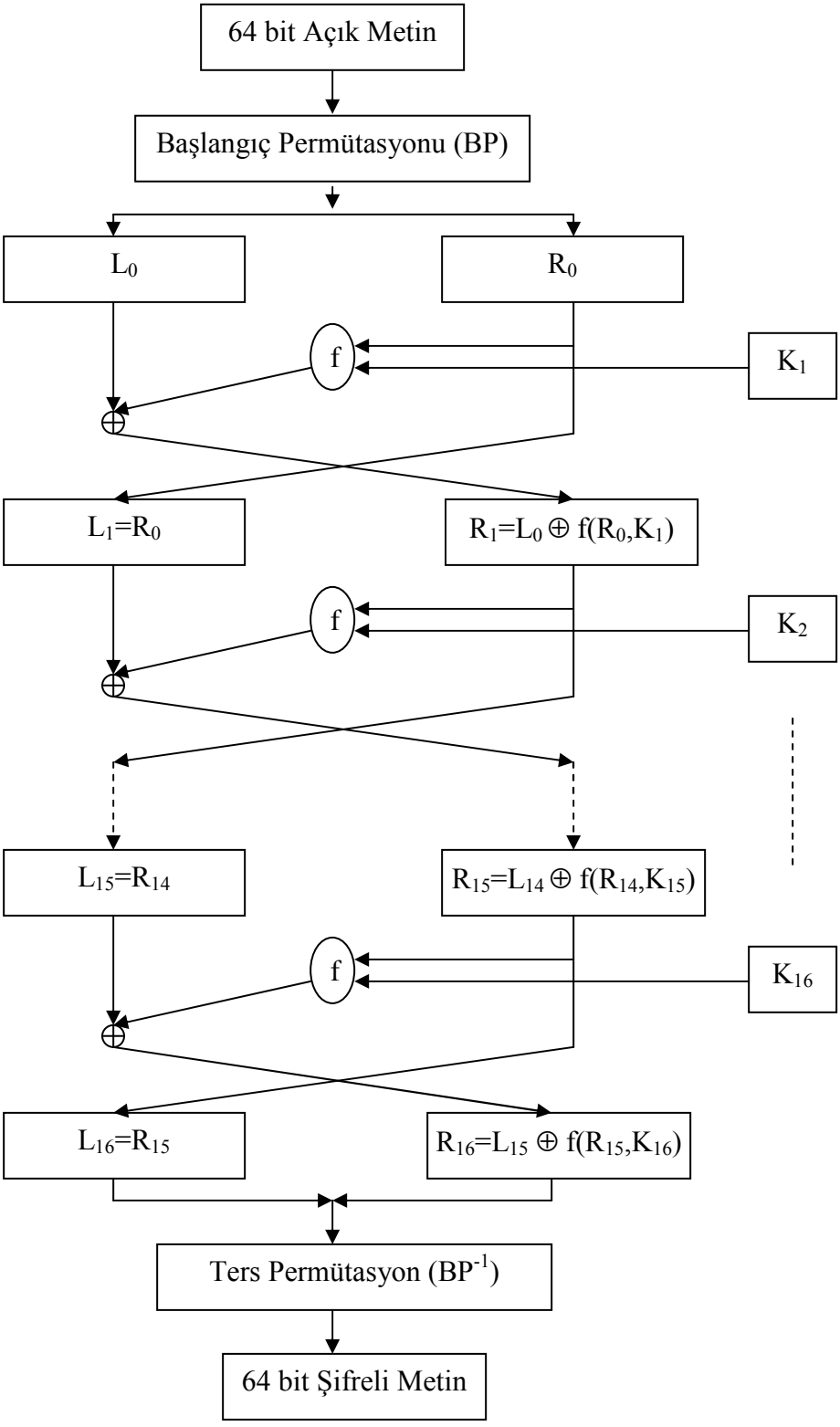
- Düz permütasyon
- Genişletme permütasyonu
- Permütasyonlu seçenekler (permuted choices)dir.

Düz permütasyonda, bitler basit bir şekilde yeniden düzenlenir. Genişletme permütasyonunda, bazı bitler ikişer kere kullanılır ve yeniden düzenlenir. Permütasyonlu seçeneklerde ise bitler ihmal edilir ve kalanlar tekrar düzenlenir.

DES'te yerine koyma işlemleri S-kutuları adı verilen birbirinden farklı tablodan oluşur. DES'te S-kutularının girişleri 6 bit, çıkışları ise 4 bit'tir. Algoritmada 16 döngü bulunması nedeni ile tekrarlanabilir yapıda olması algoritmanın donanım olarak gerçekleştirilmesini kolaylaştırmaktadır.

Anahtar uzunluğu 56 bittir. Algoritmada şifrelemenin iki temel tekniği vardır. Bu teknikler yerine koyma ve permütasyon diye adlandırılır.

DES 64-bit blok üzerinde açık metinleri işler. DES 64 bitlik veri bloğunu alır ve başlangıç permütasyonu (initial permütasyon) işleminden sonra 32 bitlik sağ ve sol yarılara ayırır. Sonra verinin anahtar ve f fonksiyonu ile birleştirildiği 16 döngülük işlemler gerçekleştirilir. 16. döngüden sonra sağ ve sol yarı tekrar biraraya getirilir. Başlangıç permütasyon işleminin tersi olan son permütasyon diğer bir deyişle ters permütasyon işlemi gerçekleştirilir (Sönmez, 2002).



Şekil 7.1. DES Algoritması

7.2.1. Başlangıç permütasyonu (BP)

64 bitlik veri bloğu alındıktan sonra Başlangıç Permütasyonu işlemi uygulanır. Bu permütasyonda Tablo 7.1're gore bitler sırayla karıştırılır. Tablo soldan sağa, yukarıdan aşağıya doğru okunur. Tabloya gore 58. bit çıkışta 1.bit, 50.bit'te çıkışta 2.bit vb. olarak çıkmaktadır.

Tablo 7.1. Başlangıç Permütasyonu (BP)

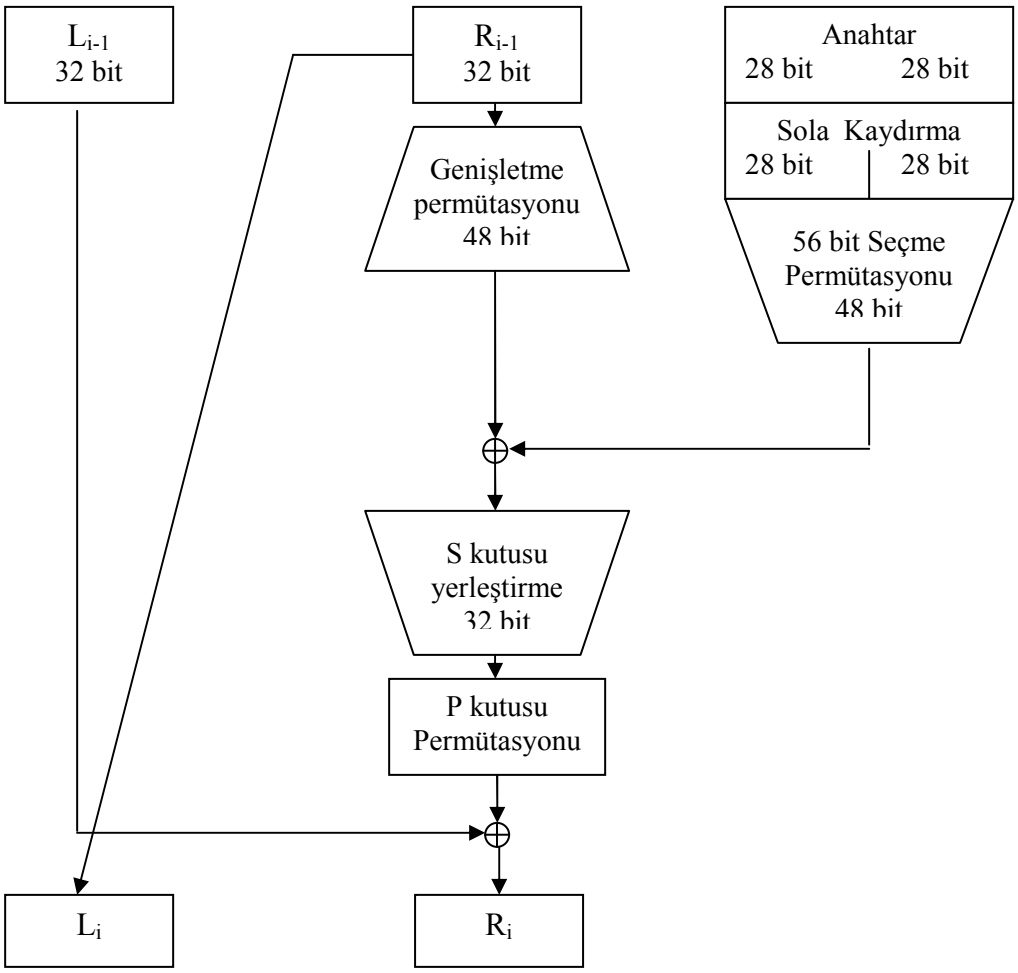
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

7.2.2. Bir DES döngüsünde gerçekleştirilen işlemler

Her döngüde anahtar (şifreleme ise sola, çözme ise sağa) kaydırılır ve 56 bitin 48'i seçilir. Sağ yarıdan gelen 32 bit, genişlemiş permütasyon yardımı ile 48 bite dönüştürülür ve 48 bitlik anahtar ile XOR'lanır. Sonuç 8 tane S-kutusuna gönderilir. Çıkışta 32 bit üretilir ve çıkış bitlerine P permütasyonu uygulanır. Bu dört işlem f fonksiyonunu oluşturur. f fonksiyonunun çıkışı soldan gelen 32 bitlik veri ile XOR'lanır. Bu işlemin sonucu yeni sağ yarıyı, eski sağ yarı da yeni sol yarıyı oluşturur. Bu işlemler 16 kez tekrarlanarak DES'in 16 döngüsü gerçekleştirilmiş olur.

Her bir döngüdeki işlemler aşağıdaki gibidir.

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$$



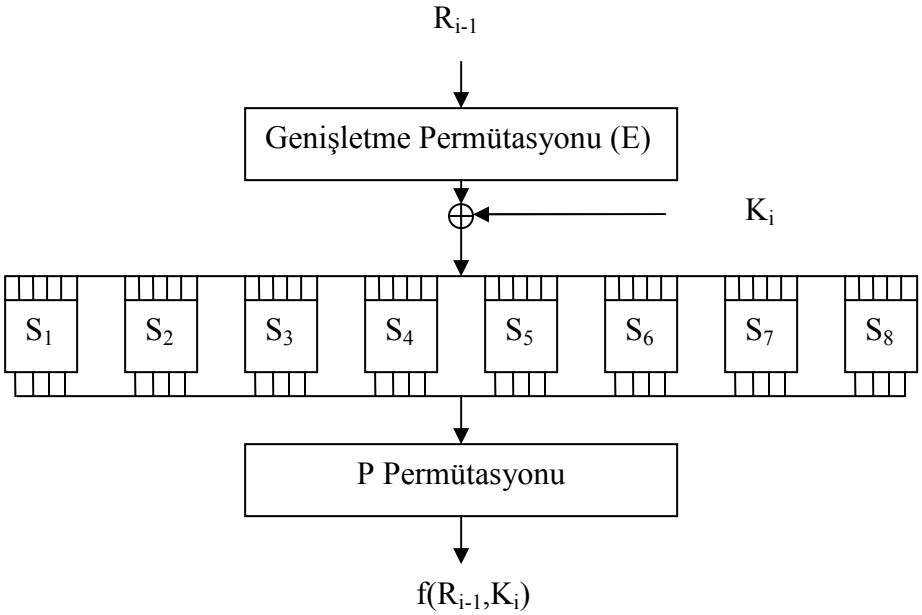
Şekil 7.2. Bir DES Döngüsünde Gerçekleştirilen İşlemler

7.2.3. f fonksiyonu

f Fonksiyonunun detaylı şeması Şekil 7.3'te görülmektedir.

Genişletme Permütasyonu

Her sağ yarı genişletme permütasyonu yardımı ile 32 bitten 48 bite genişletilir. Genişletme permütasyonu bitlerin sırasını değiştirmek ile birlikte belirli bazı bitleri tekrar eder.



Şekil 7.3. DES'in f Fonksiyonu detayı

Genişletme Permütasyonu Tablo 7.2'de gösterilmektedir. Bu tablo girişteki bitlerin çıkışta hangi sırayla çıktığını gösteriri Örneğin 32. bit 1. sırada, 3. bit dördüncü sırada vb.

Tablo 7.2. Genişletme Permütasyonu (GP)

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

S-kutuları

Yerleştirme işlemi 8 tane S- kutusu kullanılarak gerçekleştirilir. S- kutusu 6 bitlik verinin 4 bit olarak çıkmasını sağlayan bir tablodur. Genişletme permütasyonundan çıkan 48 bitlik veri, 48 bitlik anahtar ile XOR'landıktan sonra $S_1S_2.....S_8$ olmak üzere 8 tane 6

bitlik bloklara ayrılır. Her S-kutusunda 4 satır ve 16 sütun bulunmaktadır. 6 bitlik giriş verisinin birinci ve dördüncü bitlerinin oluşturduğu 2 bitlik sayı, satır numarasını verirken arada kalan 4 bitlik sayı sütun numarasını verir. Satır ve sütun numarasının kesişme noktasındaki veri, çıkışta görülecek olan 4 bitlik veridir. Örneğin S_7 bloğuna girecek olan veri 010011 ise satır numarası 01, yani 1'dir. Sütun numarası ise 1001, yani 9'dur. Birinci satır ve dokuzuncu sütundaki veri 3'tür. Dolayısı ile çıkışta 0011 verisi görülür. Tablo 7.4'te S- kutuları detaylı olarak gösterilmektedir.

S- kutusu yerleştirmesi, DES'teki en kritik basamaktır. Nedeni ise algoritmaya güvenlik sağlayan en önemli kısmı olmasıdır. DES'teki diğer işlemler kolay ve doğrusaldır. Fakat S- kutusu doğrusal olmayan bir yapı sergilemektedir.

Bu yerleştirme işlemlerinin sonucunda ortaya çıkan 4 bitlik veriler tekrar biraraya getirilerek 32 bitlik blok oluşturulur. Bu blok bir sonraki basamakta P- kutusu permütasyonunda kullanılır.

P kutusu permütasyonu

S- kutusundan çıkan veriler P adı verilen permütasyona tabi tutulur. Bu tabloya göre 16. bit 1. bit olarak çıkarken, 7. bit 2. bit vb olarak çıkar.

Tablo 7.3. P kutusu Permütasyonu (P)

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	31	10
2	8	24	14
19	13	30	6

Tablo 7.4. S Kutuları

S_1															
14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2															
15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3															
10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4															
7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5															
2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6															
12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7															
4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8															
13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

7.2.4. Ters permütasyon (BP^{-1})

Yukarıda anlatılan işlemler 16 kez gerçekleştirildikten sonra başlangıç permütasyonu işlemini tersi olan ters (son) permütasyonu aşağıda verilen şekile göre gerçekleştirilir. Bu tabloya göre 40. bit 1. bit, 8. bit 2. bit vb. olarak çıkar.

Tablo 7.5. Ters Permütasyon (BP^{-1})

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

7.2.5. Anahtarların oluşturulması

64 bitlik anahtar bloğu Permütasyonlu Seçenek 1 (PC1)'e tabi tutulur. Elde edilen veri bloğu başlangıç permütasyonunda olduğu gibi 56 bittir. PC1'in yapısı aşağıdaki tabloda verilmiştir. 56 bitlik blok, 2 bitlik iki bloğa ayrılır. Sonra C ve D yazmaçlarına yerleştirilir. Bu tabloya göre C yazmacının 1. biti 57.bit, D yazmacının 1. biti ise 63. bittir.

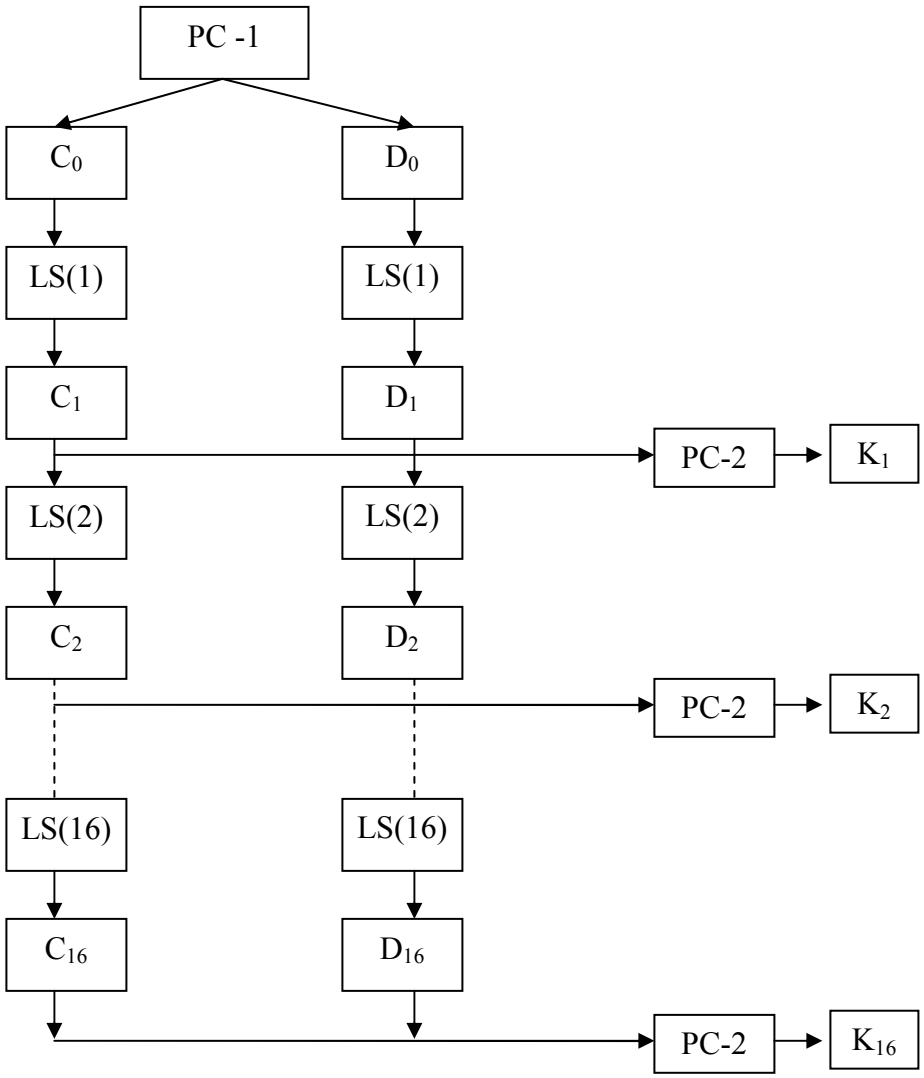
Tablo 7.6. Permütasyonlu Seçenek 1 (PC1)

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36

→ C Yazmacı

63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

→ D Yazmacı



Şekil 7.5. Anahtarların Oluşturulması

Aşağıdaki tabloda belirtildiği gibi şifreleme alt anahtarlarını oluşturmak için 28 bitlik bloklar bir yada iki kere sola kaydırılırken, çözme alt-anahtarlarını oluşturmak için ilk döngü hariç sağa bir yada iki kere kaydırılır. Şekil 7.5, DES alt anahtarlarının oluşturulmasını göstermektedir.

C ve D yazmaçlarındaki gerekli sola kaydırma işlemleri gerçekleştirildikten sonra bir araya getirilen C ve D blokları Permütasyonlu Seçenek 2 (PC2)'ye tabi tutulur ve bit büyüklüğü 48'e indirilir. Her döngüde gerçekleştirilen PC2 permütasyonunun sonucu ana döngüde kullanılacak olan her bir "alt anahtarı" verir.

Tablo 7.7. Alt Anahtarları üretilirken gerçekleştirilen sola veya sağa kaydırma sayısı

Döngü	*Anahtar (K)*	*PC2'den Önce Sola Kaydırma Sayısı (Şifrelemede)*	*PC2'den Önce Sağa Kaydırma Sayısı (Şifre çözmede)*
1	K_1	1	0
2	K_2	1	1
3	K_3	2	2
4	K_4	2	2
5	K_5	2	2
6	K_6	2	2
7	K_7	2	2
8	K_8	2	2
9	K_9	1	1
10	K_{10}	2	2
11	K_{11}	2	2
12	K_{12}	2	2
13	K_{13}	2	2
14	K_{14}	2	2
15	K_{15}	2	2
16	K_{16}	1	1

PC2 Permütasyonunda gerçekleştirilen işlemler aşağıdaki tabloda gösterilmiştir. Bu tabloya göre 14. bit 1. bit, 17. bit 2. bit vb. olarak çıkar.

Tablo 7.8. Permütasyonlu Seçenek 2 (PC2)

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

7.3. Şifre Çözme

DES algoritmasında, çözme işlemi için kullanılacak algoritma, şifrelemek için kullanılan algoritmanın aynısıdır. Tek fark, çözme işleminde kullanılacak olan alt-anahtarların aynı zamanda şifreleme anahtarlarının ters sıradaki değerleridir. Yani K_1, şifreleme alt anahtarının birincisiyse çözme anahtarlarının on altıncısıdır.

8. AES - İLERİ ŞİFRELEME STANDARDI

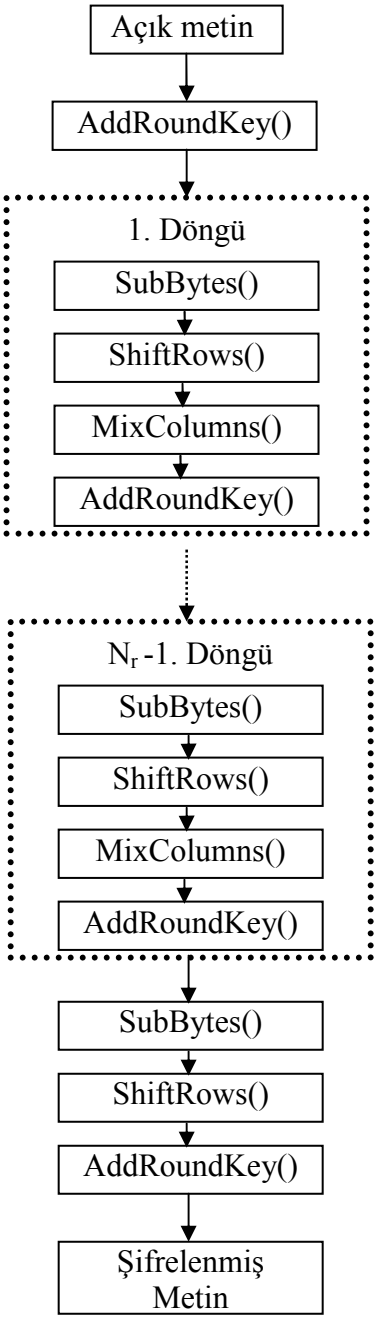
8.1. AES'in Gelişim Süreci

DES blok şifreleme algoritması 1970'li yılların sonuna doğru çıkmış bir algoritmadır. Anahtar uzunluğu 56, blok uzunluğu 64 bittir. Günümüz teknolojisi ile 2^{56} (tüm olası anahtarlar) adet anahtarın denenmesi çok kısa bir zaman almaktadır. Bu nedenden dolayı, DES algoritmasının arka arkaya çalıştırılması anlamına gelen 3DES algoritması kullanılmaktadır. Bu yöntem ile algoritmanın anahtar uzunluğu 112 bite çıkmakla beraber blok uzunluğu 64 bitte kalmaktadır. Anahtar uzayının iyileştirilmesine karşın 3DES algoritması çok yavaş bir algoritmadır (Birinci, 2002). Bu nedenlerden dolayı NIST (Ulusal Teknoloji ve Standartlar Enstitüsü) tarafından anahtar ve blok uzunluğu günümüz şartlarına uygun, değişken bir algoritma oluşturmak için 2 Ocak 1997'de 3 yıllık bir maraton olacak yarışmanın startını vermiştir. Bu yarışmaya dünyadaki kriptografi topluluğu üyelerinden 15 algoritma kabul edilmiş ve 1998 Ağustos ayında yapılan AES1 konferansında duyurulmuştur. 1999 yılında belirlenen, ana temaları güvenlik, maliyet, algoritma ve uygulama karakteristikleri olan kriterlere göre rastgelelik ve ansi c uyarlamalarının etkinlik testleri ve analizleri yapılmıştır. Analizlerin sonuçları AES2 konferansında değerlendirilerek 5 algoritma (MARS, RC6, Rijndael, Serpent, Twofish) finalist olarak seçilmiştir. Bu 5 algoritma üzerinde yine belirlenen kriterlere bağlı olarak testler yapılmış, bunların sonuçları 2000 Nisan ayında yapılan AES3 konferansında tartışılmıştır. NIST tüm bu değerlendirmeler ve sonuçlar üzerinde çalışarak 2 Kasım 2000 tarihinde Rijndael'in AES için seçildiğini duyurmuştur. 26 Aralık 2001 tarihinde FIBS PUB 197 numarasıyla Rijndael algoritmasını temel alan İleri Şifreleme Standardı (Advanced Encryption Standard) yayınlanmıştır (Fibs 197, 2001, Güncan, 2002).

8.2. AES Algoritması

AES (Rijndael) algoritması değişik blok boyunu ve değişik anahtar büyüklüğünü destekleyen bir simetrik blok şifreleme algoritmasıdır. AES 128, 192 ve 256 bit uzunlukta anahtar boyutunu ve blok uzunluğunu desteklemektedir. Ancak kriptanaliz çalışmalarının bir alanda yoğunlaşmasını sağlayabilmek amacı ile 128 bit dışındaki blok boyları AES

standardına dahil edilmemiştir (Güncan, 2002). Standart 128 bit blok boyunu ve 128, 192, 256 bit anahtar uzunluğunu içermektedir.



Şekil 8.1. AES algoritması ana akış şeması

AES'te döngü sayısı anahtar uzunluğuna göre değişmektedir. 128 bit anahtar için 10 döngüde şifreleme yaparken 192 ve 256 bit anahtar için sırasıyla 12 ve 14 döngüde şifreleme yapmaktadır. AES algoritmasında her döngü dört katmandan oluşur.

Bu katmanlar sırasıyla Subbytes(), Shiftrows(), MixColumns() ve AddRoundKey() işlemleridir. Her döngünün çıkışı bir sonrakinin giriş değeridir.

Şekil 8.1'deki N_r değeri döngü sayısını ifade eder. Buda 10,12,14 (128,192,256 bit anahtar için) değerlerinden biridir.

AES en küçük işlem birimi olarak baytları kabul eder. Şifrelenecek metin, şifrelenmiş metin ve anahtar bilgileri bayt dizileri olarak kabul edilirler. Bu diziler 4 satır ve N_b adet sütündan oluşur ve her bir hücre baytlık bilgi tutar. Metin 4 baytlık sütün vektörleri şeklinde, yani 128 bit için 4x4 (N_b= 4), 192 bit için 4x6 (N_b= 6) ve 256 bit içinde 4x8'lik (N_b= 8) matrislerle ifade edilir. 128 bitlik bir metin için aşağıdaki gibidir.

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Subbytes Fonksiyonu (Baytların yerdeğiştirmesi)

S kutusunun olduğu katmandır. Giriş matris bilgisini alıp, her bir baytı tanımlanmış bir S kutusundan geçirerek, bu baytın karşılığı olan S kutusu değeri ile değiştirir. Baytların yerdeğiştirilmesinde 16 bayt değerinin her biri 8 bit girişli ve 8 bit çıkışlı S kutusuna sokulur. S kutusu değerleri, Galois alanı'nda (Galoıs Field - GF) GF(2^8), 8 bitlik polinom ($m(x) = x^8 + x^4 + x^3 + x + 1$) için ters alındıktan sonra lineer bir dönüşüme sokularak elde edilmiştir. Matristeki her baytın tersi bulunur.

ShiftRows Fonksiyonu (Satırların ötelenmesi)

Bu fonksiyon matrisi alır ve son üç satırını belli değerlere göre dairesel olarak sola öteler. $N_b = 4$ olmak üzere öteleme miktarı 2. satır için1, 3. satır için 2, 4. satır için 3'tür.

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

a.

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

b.

Şekil 8.2. ShiftRows örneği a. öteleme öncesi, b. öteleme sonrası

Tablo 8.1. AES S Kutusu (Hexadecimal notasyonda xy byte için)

		y															
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**a**	**b**	**c**	**d**	**e**	**f**
x	**0**	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

MixColumns Fonksiyonu (Sütunların karıştırılması)

Bu fonksiyon matris üzerinde sütün bazında çalışır. Her sütun $GF(2^8)$'de 4 terimli bir polinom olarak kabul edilerek sabit bir $a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$ polinomu ile çarpılır ve elde edilen polinom başlangıç sütununun yerine geçer. b(x) matristen bir sütun olmak üzere, $b'(x) = a(x) \oplus b(x)$ eşitliğinden b'(x) hesaplanarak b(x) sütununun yerini alır.

AddRoundKey Fonksiyonu (Anahtar ekleme)

Bu fonksiyonda, bir döngü anahtarı matrise xor işlemiyle eklenir. Her döngü anahtarı N_b kelimeden oluşmaktadır. $N_b = 4$ için döngü anahtarının boyu 128 bittir.

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \oplus \begin{bmatrix} k_{00} & k_{01} & k_{02} & k_{03} \\ k_{10} & k_{11} & k_{12} & k_{13} \\ k_{20} & k_{21} & k_{22} & k_{23} \\ k_{30} & k_{31} & k_{32} & k_{33} \end{bmatrix}$$

$$= \begin{bmatrix} a_{00} \oplus k_{00} & a_{01} \oplus k_{01} & a_{02} \oplus k_{02} & a_{03} \oplus k_{03} \\ a_{10} \oplus k_{10} & a_{11} \oplus k_{11} & a_{12} \oplus k_{12} & a_{13} \oplus k_{13} \\ a_{20} \oplus k_{20} & a_{21} \oplus k_{21} & a_{22} \oplus k_{22} & a_{23} \oplus k_{23} \\ a_{30} \oplus k_{30} & a_{31} \oplus k_{31} & a_{32} \oplus k_{32} & a_{33} \oplus k_{33} \end{bmatrix}$$

8.3. Şifre Çözme

Algoritmayı oluşturan fonksiyonların tersleri alınarak ve ters sırada işleme sokularak şifre çözme işlemi gerçekleşebilir. Detaylar (Fibs 197, 2001)'de bulunabilir (Güncan, 2002).

8.4. Güvenlik

AES şifreleme algoritması, bilinen tüm saldırı tiplerine karşı güvenlidir. Tasarımında kullanılan S kutusu sonlu alanda ters alma işleminin kullanılması ile gerçekleştirilmiştir. Bu algoritmayı lineer ve diferansiyel saldırılara karşı güvenli kılarken, algoritmanın tasarımında kullanılan, MixColums (sütünları karıştırma) ikinci lineer dönüşüm, lineer ve diferansiyel saldırılarda az sayıda aktif S kutusunun işin içine girmesine mani olmaktadır (Sakallı vd., 2005).

Algoritmada herhangi bir zayıf anahtar şimdiye kadar tespit edilememiştir. Bu yüzden anahtar seçimine dönük herhangi bir kısıtlama standartta tanımlanmamıştır. AES (Rijndael) algoritması hem donanım hemde yazılım uygulamalarında iyi performans vermiştir. Anahtar hazırlama süresi yeterince kısadır. Düşük bellek gereksinimi sayesinde, kısıtlı belleğe sahip ortamlarda da uygulanabilir (Güncan, 2002).

9. AES S KUTUSUNA BENZER S KUTUSU TASARIMI

Çalışmamızda, 4x4 (4-bit giriş, 4-bit çıkış) AES S kutularına benzer olarak geliştirilen S kutusunu kullandık. S kutularının tasarımında sonlu alan GF(2^4)'te işlemler (toplama ve ters alma) kullanılmıştır (Sakallı vd., 2005).

9.1. AES S kutusu

AES'in S kutusu (yerdeğiştirme tablosu) iki dönüşüm kullanılarak elde edilmiştir.

1. GF(2^8)'de ters alma işlemi indirgenemez polinom "$x^8 + x^4 + x^3 + x + 1$" kullanılarak gerçekleştirilir. İkili (binary) temsilde "00000000" kendisine eşleştirilir.
2. Affine dönüşüm, GF(2) üzerinde uygulanır. Dönüşüm aşağıda gösterilmiştir.

$$\begin{bmatrix} Y_0 \\ Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \\ Y_5 \\ Y_6 \\ Y_7 \end{bmatrix} = \begin{bmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{bmatrix} \cdot \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

9.2. S Kutusu tasarımı

4-bit giriş ve 4-bit çıkışa sahip S kutuları tasarımında, GF(2^4)'te indirgenemez polinomlar kullanılır. Bu polinomların sonu 1 ile bitmelidir. Çünkü sonu 1 ile bitmeyen polinom x ile bölünebileceğinden indirgenebilir polinom olarak karşımıza çıkar. GF(2^4)'te olası tüm polinomlar içerisinden sadece 3 tanesi indirgenemez polinomdur. Bu polinomlar $x^4 + x + 1$, $x^4 + x^3 + 1$ ve $x^4 + x^3 + x^2 + x + 1$'dir. Geri kalan polinomlar ise indirgenebilir polinomlardır. Bir polinomun indirgenemez olması için çarpanlarına ayrılamaması gerekir (Sakallı vd., 2005).

Örneğin $x^4 + 1$ polinomu indirgenebilir bir polinomdur. Çünkü çarpanlarına ayrılabilmektedir.

$x^4 + 1 = (x^2 + 1)\,(x^2 + 1) = x^4 + x^2 + x^2 + 1 = x^4 + 1$

GF(2^4) Toplama

Sonlu alan GF(2^4)'te elemanlar, {0,1} katsayılarına sahip polinomlar olarak düşünülebilir. Dolayısıyla iki polinomun toplama işlemi katsayılarının basitçe modulo 2 aritmetiğine göre toplamıdır.

Örneğin n=0101 ve m=0111 olsun. Toplamları n+m = 0010 olur. Polinom olarak notasyonunda ise $n+m = (x^2 + 1) + (x^2 + x + 1) = x$ olacaktır (Sakallı vd., 2005).

GF(2^4) Çarpma

Sonlu alan GF(2^4)'te iki elemanın çarpımı iki polinomun çarpımından ibarettir. Bununla beraber sonuç polinom 3.dereceden büyük bir polinom olabilir. Dolayısıyla elde edilen sonuç polinomun sonlu alan GF(2^4)'ün bir elemanı olabilmesi için indirgenemez bir polinom kullanılarak indirgenmesi gerekmektedir (Sakallı vd., 2005).

Örneğin n=0101, m=0110 ve indirgenemez polinom $x^4 + x + 1$ seçilsin. O zaman

$n.m = (x^2 + 1).(x^2 + x) = x^4 + x^3 + x^2 + x$ olarak yazılabilir ve $x^4 = x + 1$ yazılabileceğinden

$n.m = x^3 + x^2 + 1 = 1101$ olarak elde edilir.

GF(2^4) Ters alma işlemi

4-bit iki polinomun çarpımının kalanı seçilen indirgenemez polinoma göre 1 ise o zaman iki polinom birbirinin, o indirgenemez polinoma göre tersidir denir (Sakallı vd., 2005).

Örneğin n=0101 4-bit değerinin indirgenemez polinom $x^4 + x + 1$'e göre tersi m=1011'dir. Bu polinomların ikisinin çarpımının kalanı $x^4 + x + 1$ indirgenemez polinomuna göre 1 vermelidir.

$n.m = (x^2 + 1).(x^3 + x + 1) = x^5 + x^3 + x^2 + x^3 + x + 1 = x^5 + x^2 + x + 1$ bulunur. Burada

$x^5 = x^4 . x = (x + 1) . x = x^2 + x$ olduğundan formülizasyonda x^5 yerine $x^2 + x$ yazılırsa

$n.m = x^2 + x + x^2 + x + 1 = 1$ bulunur.

Tablo 9.1. GF(2^4)'te 4-bit değerlerin indigenemez polinoma göre ters alma işlemlerinin sonucu

GF(2^4)'te 4-bit değer	İndirgenemez polinom $x^4 + x + 1$'e göre ters alma işleminin sonucu	
	İkili	Hex.
0001	0001	1
0010	1001	9
0011	1110	E
0100	1101	D
0101	1011	B
0110	0111	7
0111	0110	6
1000	1111	F
1001	0010	2
1010	1100	C
1011	0101	5
1100	1010	A
1101	0100	4
1110	0011	3
1111	1000	8

S kutusu tasarımını tamamlamak için Tablo 9.1'de gösterilen 4-bit değerlerin indirgenemez polinomlara göre ters işlemlerinin sonuçlarını affine dönüşüme sokmamız gerekmektedir. Çalışmada $x^4 + x + 1$ indirgenemez polinomuna göre aşağıdaki affine dönüşümü kullanılmış ve Tablo 9.2'deki S kutusu elde edilmiştir.

$$\begin{bmatrix} Y_0 \\ Y_1 \\ Y_2 \\ Y_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Tablo 9.2. Tablo 9.1'deki değerlere ve seçilen affine dönüşümüne göre elde edilen S kutusu

$GF(2^4)$'te 4-bit değer	S Kutusu	
	İkili	Hex.
0000	0011	3
0001	1101	D
0010	1010	A
0011	0010	2
0100	0001	1
0101	0111	7
0110	1011	B
0111	0101	5
1000	1100	C
1001	1110	E
1010	1111	F
1011	0110	6
1100	1001	9
1101	1000	8
1110	0000	0
1111	0100	4

Elde edilen bu S kutusu, bir sonraki bölümde (Doğrusal (Lineer) Kriptanalizin gerçekleştirildiği) AES şifreleme algoritmasının küçültülmüş bir benzeri olan SPN algoritmasında kullanılmıştır.

10. KRİPTANALİZ

10.1. Giriş

Kriptanaliz, açık metni yada anahtarı elde etme bilimidir. Başarılı kriptanaliz sonucunda açıkmetin veya anahtar elde edilebilir. Bir kriptanaliz girişimine saldırı denir.

Kripto-analizci şifreleme algoritmasının bütün detaylarına ulaşma gücüne sahiptir ve sistemde sadece anahtar gizlidir. Bu prensibe göre tasarlanmış ve hala güvenli olduğu kabul edilen birçok algoritma günümüzde mevcuttur. Analizcinin amacı herhangi bir algoritma kullanılarak kapatılmış metinlerin açık halini elde edebilmektir. Genellikle bu amaca algoritmada kullanılan gizli anahtarın tamamı veya belli bir kısmı elde edilerek ulaşılır.

Analiz yönteminin ne kadar kuvvetli ve efektif olduğu analiz için gerekli olan önbilgi ve yapılacak iş miktarı ile ölçülür (bilinmesi gerekenler açık- kapalı metin çiftlerinin sayısı, harcanan zaman, atağın başarı oranı dır).

Modern şifreleme yöntemlerinin gücü söz konusu olduğunda iki ana öğeden bahsetmek mümkündür. Bunlar şifreleme yöntemlerinin kriptanaliz saldırılarına karşın dayanıklılığının ve şifreleme algoritmasının zayıf istatistiksel sonuçlar verip vermediğinin incelenmesidir. Son öğe şifreleme algoritmasının çıkışlarının rastlantısal bir sonuç verip vermemesiyle ilgilidir.

10.2. Saldırı Çeşitleri (Senaryoları)

Saldırgan yada düşman sahip olduğu verilere göre, kriptosisteme saldırısını gerçekleştirir. En yaygın kullanılan saldırı senaryoları aşağıda verilmiştir.

Sadece Şifeli Metin Saldırısı (Ciphertext Only) : En güçlü kripto analitik ataktır. Sadece haberleşme pasif olarak dinlenip (müdahale edilmeden) yeterince kapalı metin elde edilerek yapılabilir.

Bilinen Açık Metin Saldırısı (Known Plaintext) : Bir miktar açık/şifeli metin çifti bilinerek yapılan saldırıdır. Mesajların bir kısmı tahmin edilebilir veya açık gönderilen mesajlar toplanarak atak uygulanabilir.

Seçilmiş Açık Metin Saldırısı (Chosen Plaintext) : Analizcinin seçtiği metni şifeleme imkanına sahip olduğu kabul edilen saldırı tipidir. Analizci şifreleme algoritmasının güvenli olarak yerleştirildiği mekanizmayı elde edebilir. Analizci aktif olarak haberleşme sisteminde rol oynar.

Seçilmiş Şifreli Metin Saldırısı (Chosen Ciphertext) : Deşifreleme makinasına veya algoritmasına ulaşılarak yapılan saldırı tipidir. Düşman bir şifreli metin dizisi seçebilir ve bunların açık metinlerini oluşturabilir.

Seçilmiş Açık veya Şifeli Metin Saldırısı (Adaptive Chosen Plaintext or Ciphertext) : Bu saldırı tipinde analizcinin istediği mesajı deşifreleme veya şifreleme konusunda sınırsız kapasiteye sahip olduğu kabul edilir. Önceki iki senaryonun teorik olarak daha güçlendirildiği saldırı tipidir.

10.3. Kriptanaliz Metodları

Araştırmacı bakış açısından bir şifre sisteminin kırıldığının açıklanması, bu sistemin dizaynı değiştirmeye yol açacak bir zayıflığının bulunması demektir. Bu değişiklik, ek döngülerin eklenmesi veya döngü anahtarlarını oluşturan algoritmanın ve bazı iç yapıların değişmesi anlamına gelebilir. Şifre sisteminin tamamen kırılması ise bu tür sistemi tamir etmek yerine baştan tasarlamanın daha anlamlı veya kolay olduğu durumlardır. Tipik kriptanaliz metodlarını şöyle sıralayabiliriz.

Etraflı Arama (Exhaustive Search) : Etraflı Arama, şifre sistemleri üzerine doğrudan uygulanabilen açık bir yöntemdir. Tüm olası gizli anahtarları bilinen açık/şifreli metin örnekleri üzerinde dener. Doğru gizli anahtar bilinen açık metinden şifreli metnin elde edilmesini sağlar. Günümüz hesaplama imkanlarına göre modern blok şifreleme sistemlerinin anahtar uzunlukları (128 bit ve yukarısı) bu tip saldırıyı imkansız kılacak şekilde seçilmektedir. DES'in en önemli zayıflığı 56 bit olan kısa anahtar uzunluğudur ve günümüz koşulları düşünüldüğünde bu anahtar uzunluğu etraflı aramayı mümkün kılmaktadır.

Sözlük Atakları (Dictionary Attacks) : Basit fakat blok şifre sistemleri için önemli bir saldırı çeşididir. Bu yöntemde şifrelenen metinlerin uzunlukları kısa ise saldırgan birçok metni toplar ve farklı metinlerin tekrarlama analizini yapar.

Eş Tanımlama (Equivalent Description) : Bazen şifeleme sistemi tasarlayanlar sistemin veya parçalarının basit eşdeğer tanımlarını gözden kaçırmaları, bu saldıyı başarılı kılar.

Değişmezler için Devirlilik veya Arama (Periodicity or Search for Invariants) : Değişmezler, şifreleme işlemi boyunca değişmeyen özellikler olarak düşünülebilir ve şifre sisteminin istenilmeyen bir özelliğidir. Eğer kriptanalist sistemin herhangi bir değişmezini yada yakınsamasını bulmayı başarırsa, gerekli saldırı için malzeme edinmiş olur. Her çeşit devirli davranış veya şifrelemeler arasındaki korelasyon mutlaka engellenmelidir.

Ortada Buluşma Saldırısı (Meet in the Middle Attack) : Bu yöntemde şifreleme algoritması alt ve üst olmak üzere ikiyi bölünür. Sonra kısmi tahmin yoluyla yukarıdan ortaya ve alttan ortaya kısmi deşifreleme yapılır. Sonuç karşılaştırılır ve eğer uyumlu ise aday anahtar saklanır.

İstatistiksel Yaklaşımlar (Statistic Approaches) : Bu metodlar şifreli ve açık metin arası ilişki veren istatistiksel örnekleri arar. Bir tür ayıran saldırıdır ve diğer saldırılar için ilk adımdır. İstatistiksel yaklaşımlar hem oluşması yüksek olasılıkta olayları hem de gerçekleşmesi imkansız olayları bulmaya yöneliktir.

Özel bir Saldırıya göre Zayıf Anahtarlar (Weak Keys with espect to a Particular Attack) : Bazı durumlarda bir zayıf anahtar kümesinin bir şifreleme sisteminin analizini özel bir saldırı modelini düşünüldüğünde kolaylaştırmsı mümkün olmaktadır. Eğer bir kriptosistemin tüm anahtar uzayına göre yüksek oranda zayıf anahtarlara sahip ise tekrar dizayn edilmesi bile söz konusu olabilir.

Örnek olarak DES'te dört tane zayıf ve oniki tane yarı zayıf anahtar bulunmaktadır. Gizli amahtar K olmak üzere DES'i E_k olarak tanımlarsak, dört tane zayıf anahtar için $E_k(E_k(m)) = m$ ve oniki tane yarı zayıf anahtardan ikitanesi için $E_{k1}(E_{k2}(m)) = m$ sağlanmaktadır. IDEA blok şifre sistemi için 2^{128} anahtar uzayı üzerinden 2^{63} elemana sahip zayıf anahtar kümesi bulunmaktadır (Uygulamalı Matematik Enstitüsü, 2004).

10.4. Blok Şifreleme Algoritmalarına Yönelik Saldırılar

Blok şifrelerine birçok saldırı yöntemi geliştirilmiştir. Genelde bir saldırının amacı herhangi bir şifreli metni deşifreleyebilme yeteneğini kazanmaktır. Bu genellikle anahtarı

elde etme, yada bir algoritma tasarlayarak anahtarı bilmeden şifrelimetinleri deşifreleme ile ilişkilidir. Genel varsayımda düşmanın kullanılan kriptosistemi bildiği kabul edilir (*Kerchoffs'un prensibi*). Tabiki, eğer düşman kullanılan kriptosistemi bilmiyorsa, bu onun işini zorlaştıracaktır. Düşmanın bir kriptosisteme saldırabilmesi için sahip olması gereken veriler vardır. Bu sahip olduğu verilere göre saldırı çeşitlerinden (senaryolarından) birini seçebilir.

Kriptanaliz yöntemlerinin başarılı sayılabilmesi için *brute-force* saldırısı yani tüm olası anahtarların şifrelenmiş mesaj üzerinde denenerek anlamlı bir mesaj elde etme işleminden daha az maliyete sahip olması gerekmektedir. DES algoritması 56 bit anahtara sahiptir. 2^{56} deşifreleme işlemi onun kırılmasını sağlayacaktır (1 olasılıkla) (Sakallı vd., 2004).

Günümüzde en çok bilinen blok şifreleme algoritmalarına örnek olarak DES (Data Encryption standard) (Fips 46-3, 1999) ve AES (Advanced Encryption Standard) (Fips 197, 2001) verilebilir.

DES, 1974 yılında tasarlandığından beri iki saldırı yöntemi başarıyla gerçekleştirilmiştir. Bunlar doğrusal kriptanaliz (Matsui, 1993) ve diferansiyel kriptanalizdir (Biham ve Shamir, 1993). Bu saldırılardan doğrusal kriptanaliz, bilinen açık metin saldırısı, için DES'in kırılmasında 2^{43} açık metin/şifreli metin gerekmektedir. Anahtarı elde etmek için 2^{43} şifreleme işlemi yaparak saldırı gerçeklenir. Fakat bu saldırı için 256 terabyte veri biriktirmek gerekmektedir (Sakallı vd., 2004).

Tarihsel olarak, blok şifrelerinin kriptanalizi yada blok şifrelerine saldırı yöntemleri ile ilgili akademik alanda çalışmalar 1981 yılında ilk CRYPTO konferansı ve DES'in istenilmeyen özelliklerinin incelemeleri ile başladı. Pratikte çoğu kriptanalitik teknikler 1990'lı yıllarda geliştirildi. Bunların bir çoğu bu alandaki iki kesin gelişmenin varyasyonlarıdır. Diferansiyel kriptanaliz Biham ve Shamir tarafında 1990 yılında bulundu ve CRYPTO 90 konferansında sunuldu. Doğrusal Kriptanaliz Matsui tarafından geliştirildi ve EUROCRYPT 93 konferansında sunuldu. Bu iki teknik, blok şifreler için güvenlik tayinininde kullanılan kriterlerin gelişmesine olanak sağladı. Son zamanlarda geliştirilen AES (*Advanced Encryption Standard*) - Rijndael bu iki saldırıya ve onların ilerlemelerine karşın ispatlanabilir güvenlik fikri temelli geliştirilmiştir (Sakallı vd., 2004).

Blok Şifreleme Algortimalarına yönelik saldırıları iki gruba ayırabiliriz. Bunlardan ilki Temel Saldırılar, ikincisi de Gelişmiş Saldırılardır.

10.4.1. Temel saldırılar

Blok uzunluğu n bit olan ve k bit anahtar uzunluğuna sahip bir blok şifresi için en temel saldırılardan biri *sözlük saldırısıdır*. (*Dictionary attack*) Bu saldırıda k bitlik anahtarı kullanan saldırgan bir açık metni mümkün 2^k anahtarla şifreler ve şifreli metinleri sıralı bir sözlükte tutar. Daha sonra gizli anahtarla şifrelenmiş seçilmiş bir açık metni elde eder ve uygun bir eşleşmeyi sözlükten kontrol eder. Sözlükte arama ihmal edilebilir fakat saldırı için 2^k tane n-bit bellek word'ü gerekmektedir. Bu yüzden bu saldırı pahalı bir saldırı olarak nitelenebilir (Şahin vd., 2005).

Diğer bir saldırı türü olan *kodkitabı saldırısında* (C*odebook attack*) saldırgan 2^n mümkün açık metnin gizli bir anahtar ile şifrelenmiş şifreli metinlerini elde eder ve bunları bir tabloya (kodkitabı) depolar. Bir şifreli metin için bekler ve elde ettiği gibi bu şifreli metni tablodakiler ile karşılaştırarak açık metni elde eder. Saldırı 2^n açık metin ve 2^n bellek word'ü gerektirir. Bu saldırı türü de pahalı bir saldırı türüdür (Şahin vd., 2005).

Geniş anahtar arama saldırısında (*Exhaustive key search*) ise 2^k olası anahtar denenerek şifreli metinden anlamlı bir açık metin elde edilince saldırı tamamlanır. 2^k olası deneme her ne kadar mümkün görünmese de zaman karmaşıklığı veri ve alan karmaşıklığından daha ucuz olarak düşünülmektedir. Eğer bir kriptanaliz saldırısı geniş anahtar aramadan daha az efor ile blok şifreyi kırarsa bu saldırı başarılı bir saldırı olarak görülebilir (Şahin vd., 2005).

10.4.2. Gelişmiş saldırılar

Bu tür saldırılar matematiksel bir fikrin kullanılması ile geliştirilmişlerdir. Bu saldırılara örnek olarak Doğrusal kriptanaliz, Diferansiyel kriptanaliz, İmkansız Diferansiyel kriptanaliz ve Çokluset saldırıları-Square saldırısı verilebilir.

Doğrusal (*Lineer) Kriptanaliz :* 1993 yılında Matsui (Matsui., 1993) tarafından teorik bir saldırı olarak keşfedilmiş ve sonradan DES algoritmasına karşı başarı ile uygulanmıştır. Doğrusal Kriptanaliz, şifreli metin bitleri ile açık metin bitleri arasındaki yüksek olasılıkta doğrusal ifadelerin meydana gelme avantajını kullanır. Yerdeğiştirme kutularının (S kutusu) doğrusal ifadelere dönüştürülmesi ve doğrusal ifadeleri birleştirerek bilinmeyen anahtar bitlerini elde etme prensibine dayanır. Bu işlem için son döngüde yerine getirilen yerdeğiştirmelerden önceki durum bitleri ile açık metin bitleri arasında doğrusal bir ilişki bulunması gerekir. Bu doğrusal ifade olası tüm anahtar bitleri ile test edilir ve anahtar bitlerinin sapması teorik olarak elde edilen sapmayla karşılaştırılır. En yüksek mutlak sapma değerine sahip anahtar bizim aradığımız hedef anahtardır.

Diferansiyel Kriptanaliz : 1991 yılında Biham (Biham ve Shamir, 1993) tarafından keşfedilmiştir. Doğrusal kriptanalize benzemektedir. Farkı, seçilmiş açık metin saldırısı olmasıdır. Saldırı şöyle açıklanabilir. Elimizde açık metin/şifreli metin çiftleri (x_1, x_2, y_1, y_2) olsun ve $\Delta x = x_1 \oplus x_2$ sabit şekilde seçilmiş olsun. Saldırgan bir çok sayıda bu açık metin/şifreli metin çiftlerini üretir ve Δx farkına karşılık algoritmadan yüksek olasılıkta meydana gelen Δu (son döngüdeki S kutusundan önceki durum bitleri) farkını bulduktan sonra her açık/şifreli metin çiftleri için olası anahtar değerlerini dener ve Δu farkını tutması durumunda sayaç değerini o anahtar değeri için 1 arttırır. Yüksek olasılığı yakalayan anahtar değeri aranan anahtar olmuş olur. S kutularının diferansiyel kriptanalize karşı güçlü olması, algoritmanın gücü için gerekli bir şarttır. Bu saldırılardan doğan kesik diferansiyel kriptanaliz (Knudsen, 1995), imkansız diferansiyel kriptanaliz (Raphael ve Phan, 2004), doğrusal kriptanalizde çoklu yaklaşımlar (Robshaw ve Kaliski, 1994) ve doğrusal kriptanalizde nonlineer yaklaşımların kullanılması (Knudsen ve Robshaw, 1996) gibi düzenlenen saldırılarda vardır (Şahin vd., 2005).

İmkansız Diferansiyel Kriptanaliz : Kesik diferansiyel kriptanalizin (Knudsen, 1995) bir çeşididir. Önceden bir diferansiyel öngörülür ve buna göre bazı özel farklar asla meydana gelmeyecektir. Kriptanalizde imkansız durumların kullanılabileceği gerçeği eski bir fikirdir. Ortada ıskalama salsırısı (Miss in the middle attacks) ya da imkansız diferansiyel saldırısı olarak isimlendirilen bu saldırılar bir blok şifrede imkansız bir

davranışın nasıl belirleneceği ve bunun nasıl anahtarı elde etmek için kullanılacağı ile ilişkili sistematik analizdir (Sakallı vd., 2005).

Çokluset saldırıları-Square Saldırısı : Çokluset saldırıları ilk defa J. Daemen, V. Rijmen, ve L. Knudsen, *Square* algoritmasını (Daemen vd., 1997) ortaya koyduklarında öne sürülmüştür. Dolayısıyla diğer ismi Square (Kare) saldırısı olarak bilinir. O zamandan beri diğer bir çok algoritmaya uygulanmıştır. (Twofish, IDEA, Camellia, Skipjack gibi) Seçilmiş açıkmetin saldırısıdır ve iyi seçilmiş açıkmetin setleri ile şifrenin ileri doğru incelenmesiyle gerçekleştirilir. Bu saldırı tipinde doğrusal ve diferansiyel kriptanalizden farklı olarak açıkmetin grubunun tümü düşünülerek şifre hakkında bilgi toplanır (Sakallı vd., 2005).

11. DOĞRUSAL (LİNEER) KRİPTANALİZ

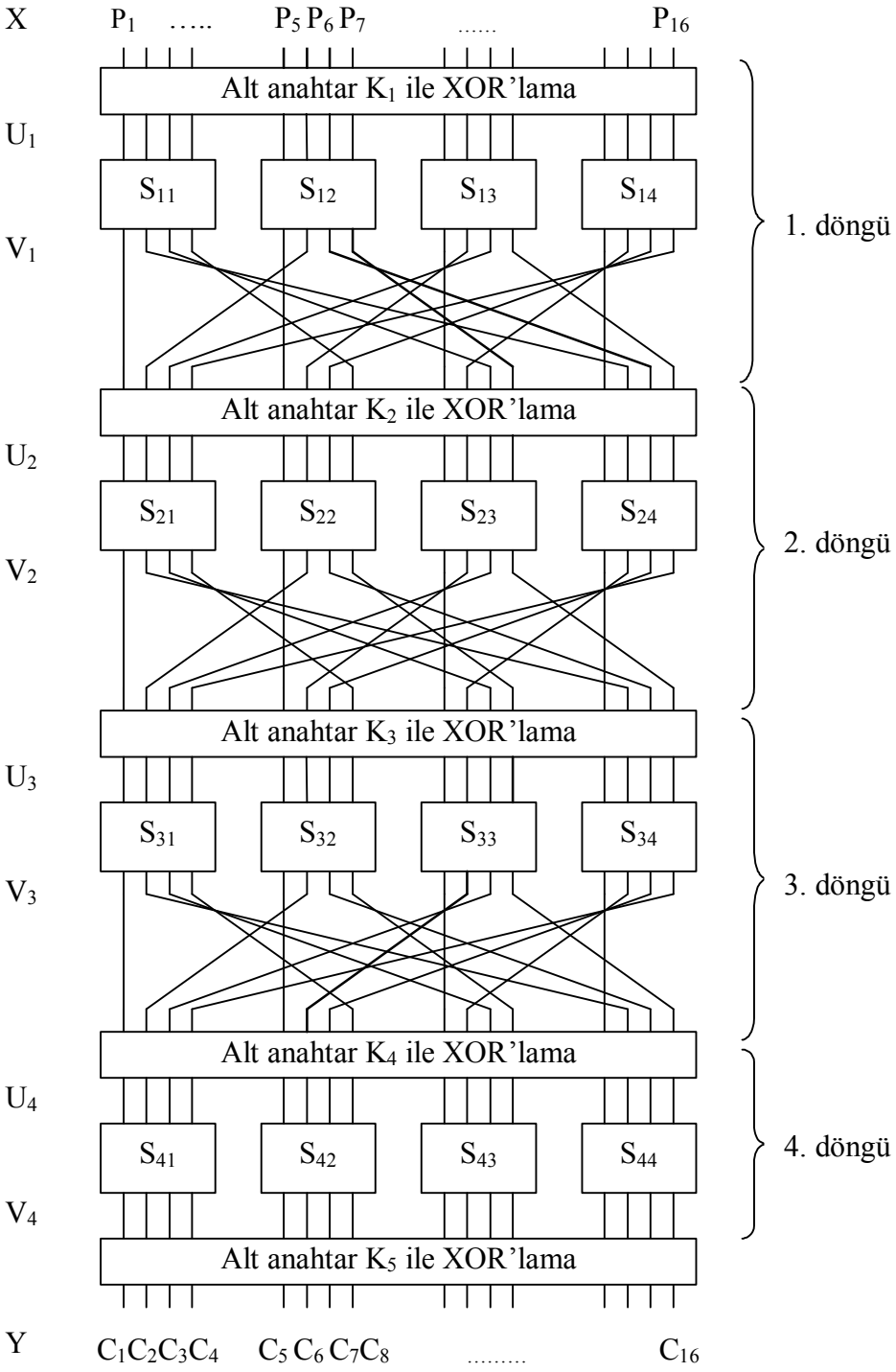
Doğrusal kriptanaliz, 1993 yılında Matsui (Matsui, 1993) tarafından DES algoritmasına kriptanalitik bir saldırı tipi olarak keşfedilmiştir. Modern şifreleme algoritmalarının tasarımında dikkate alınması gereken önemli bir unsurdur.

Doğrusal kriptanaliz, yerdeğişirme kutularının (S kutusu) doğrusal ifadelere dönüştürülmesi ve doğrusal ifadeleri birleştirerek bilinmeyen anahtar bitlerini elde etme prensibine dayanır. Bu işlem için son döngüde yerine getirilen yerdeğiştirmelerden önceki durum bitleri ile açık metin bitleri arasında doğrusal bir ilişki bulunması gereklidir. Bu doğrusal ifade olası tüm anahtar bitleri ile test edilir ve anahtar bitlerinin sapması teorik olarak elde edilen sapma ile karşılaştırılır. En yüksek sapma (1/2 den + yada -) değerine sahip anahtar bizim aradığımız hedef anahtardır. Aradığımız hedef anahtar yanlış ise sapma 0 değerine yakın olacaktır (Sakallı vd., 2004).

Tezde, 16 bit girişe ve 16 bit çıkışa sahip olan, üç ve dört döngülük bir SPN algoritması için doğrusal kriptanaliz uygulaması gerçekleştirilmiştir. Kullanılan SPN algoritması aşağıda anlatılmıştır.

11.1. SPN Algoritması

R döngüden oluşan bir SPN(Substitution-Permutation Network) algoritması (R+1) tane N bit anahtar gerektirir. Her döngü üç katmana sahiptir. Anahtar karıştırma safhasında N bit döngü girişi alt anahtar ile XOR işlemine tabi tutulur. Yerdeğiştirme safhasında anahtar safhasının çıkışı n genişliğinde M alt bloğa bölünür (N=M.n) ve her alt blok $n \times n$ yani n bit girişe n bit çıkışa sahip bir S kutusuna giriş olur. Doğrusal transformasyon (permütasyon) safhasında yerdeğiştirme safhasının çıkışı, tersine çevrilebilir N bit doğrusal transformasyon yolu ile işlenir. Bu aşamada bit pozisyonlarının yerleri değiştirilir. Son döngüde doğrusal transformasyon işlemi gözardı edilir. Kriptanaliz uygulamasında kullanacağımız SPN algoritması Şekil 11.1'de gösterilmektedir. Tablo 11.1'de ise bu algortimada kullanılan S kutusu ve permütasyonun özellikleri görülmektedir (Sakallı vd., 2004).



Şekil 11.1. SPN Algoritması (N =16, M = n = 4, R = 4)

Çalışmamızda 4 döngülük SPN algoritması için $P_1,P_2,....,P_{16}$ açık metin bitlerini C_1, $C_2,....,C_{16}$ şifreli metin bitlerini temsil etmektedir. Ayrıca K, anahtarı temsil etmekte ve $K_{1,1}$ anahtar biti için ilk indis döngü numarasını ikincisi ise anahtarın bit pozisyonunu vermektedir. U, S kutusuna giriş bitlerini temsil etmekte ve $U_{1,1}$ giriş biti için ilk indis döngü numarasını ikincisi ise giriş bitinin pozisyonunu vermektedir. V, S kutusuna ait çıkış bitlerini temsil etmekte ve $V_{1,1}$ çıkış biti için ilk indis döngü numarasını ikincisi ise çıkış bitinin pozisyonunu vermektedir. S_{11}, S kutusu için ilk indis döngü numarasını ikinci indis ise o S kutusunun döngüdeki yerini gösterir.

Tablo 11.1. a.) S-kutusu Tablosu
b.) PermütasyonTablosu

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

a

Giriş Değeri	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Çıkış Değeri	1	14	11	8	5	2	15	12	9	6	3	16	13	10	7	4

b

Çalışmamızda kullandığımız SPN algoritmasında AES S kutusuna benzer olarak tasarlanan 4 bit giriş ve 4 bit çıkışa sahip S kutusunu (Sakallı vd., 2005) ve Permütasyon (Sakallı vd., 2005) işleminde de AES'in ShiftRows (Satırları Öteleme) dönüşümünü kullandık. AES şifreleme algoritmasında ShiftRows dönüşümü bayt değerlerinin permütasyonunda kullanılırken, bizim çalışmamızda bitlerin permütasyonunda kullanıldı.

11.2. Temel Saldırıya Genel Bakış

Doğrusal Kriptanalizin amacı, açık metin bitlerini, şifreli metin bitlerini ve alt anahtar bitlerini içine alan doğrusal bir ifadenin elde edilmesidir. Temel fikir, doğrusal ifadesiyle şifrenin bir kısmının çalışma tarzını tahmin etmektir. Bu lineer olarak 2 modluk

bit işlemi için işaret edilir. XOR ile ifade edilir ve $\oplus$ ile gösterilir. Formun ifadesi şu şekildedir:

$$X_{i1} \oplus X_{i2} \oplus \ldots \oplus X_{iu} \oplus Y_{j1} \oplus Y_{j2} \oplus \ldots \oplus Y_{jv} = 0$$

Burada X_i, $X=[X_1,X_2,....]$ girişin i'ninci bitini anlatır ve Y_j ise $Y=[Y_1,Y_2,....]$ çıkışın j'ninci bitini anlatır. Bu denklem "u" giriş bitlerinin ve "v" çıkış bitlerinin XOR toplamını göstermektedir.

Kriptanalist, mümkün tüm yaklaşımları ve bu yaklaşımların tuttuğu olasılıkları araştıracaktır. S kutuları 2^n mümkün girişe sahip ise ve eğer x, doğrusal yaklaşımın tuttuğu sayı ise sonuçlanan olasılık $p= x/2^n$ ve bunu takip eden sapma (bias) $\varepsilon = p-1/2$ olacaktır.

S kutularının doğrusal yaklaşımları bulunduktan sonra, problem şifrenin açıkmetin bitleri, şifrelimetin bitleri ve anahtar bitlerini ilgilendiren son yaklaşımı oluşturabilmek için bir yol bulmaktır. Böyle bir oluşum, şifre boyunca bir çok S kutusunun, dolayısıyla bu S kutularına ait doğrusal yaklaşımlarının hesaplamanın içine girmesine neden olacaktır. Takip eden *önerme (Lemma)* bir çok S kutusu boyunca doğrusal yaklaşımın elde edilmesine izin verir.

Saldırı R döngülük bir şifreleme algoritması için şifrenin (R-1) döngülük doğrusal yaklaşımını gerektirir. Dolayısıyla son döngüden bir önceki döngüye doğru kısmi deşifreleme işlemi yapılır ve (R-1). döngüdeki durum bitleri ve giriş bitleri arasında doğrusal bir ifade bulunur. Bu kısmi deşifreleme işlemi sırasında tüm olası anahtar değerleri ve tüm açık metin/şifreli metin çiftleri işlenerek her olası anahtar değerine karşı bir olasılık değeri dolayısıyla bir sapma değeri bulunur. Eğer sapma 0 yada bu değere yakın bir değer ise anahtar doğru anahtar değildir. Mutlak sapması en büyük olan anahtar değeri uygulamada kullanılan anahtar olacaktır.

Matsui, kriptanalizi gerçekleştirmek için gerekli olan açık metin/şifreli metin çiftlerinin sayısının ε^{-2} ile orantılı olduğunu göstermiştir. Saldırının başarı olasılığı, düşünülen açıkmetinlerin sayısı (N) ile artacaktır. Pratikte, ε^{-2}'nin birkaç katının düşünülmesi saldırının uygulanabilirliği açısından uygun olacaktır. Düşünülen açık metinlerin sayısı $N = c / \varepsilon^{-2}$ formülünden hesaplanır. Uygulamada c = 5, 10, 15 olarak alınabilir.

En Temelden Yukarıya Kuralı

Doğrusal ifadesinin kuruluşu göz önüne alınmadan önce, burada örnek şifre için bazı temel araçlara ihtiyacımız olacaktır. Farzedelim X_1 ve X_2 iki adet tesadüfi binary(ikili) değer olsun. Biz en basit bağıntı notasyonu ile başlayalım. $X_1 \oplus X_2 = 0$ bir doğrusal ifadedir ve $X_1=X_2$ 'ye eşittir. $X_1 \oplus X_2 = 1$ bir affine(ilgin)düzensiz ifadedir ve $X_1 \neq X_2$ dir.

Şimdi, kabul edelim ki aşağıdaki olasılık dağılımları verilmiş olsun.

$$P_r(X_1 = i) = \begin{cases} p_1 & , i=0 \\ 1-p_1 & , i=1 \end{cases}$$

ve

$$P_r(X_2 = i) = \begin{cases} p_2 & , i=0 \\ 1-p_2 & , i=1 \end{cases}$$

Eğer bağımsız olan iki tesadüfi değer varsa,

$$\Pr(X_1 = i, X_2 = j) = \begin{cases} p_1 p_2 & ,i=0, j=0 \\ p_1(1-p_2) & ,i=0, j=1 \\ (1-p_1)p_2 & ,i=1, j=0 \\ (1-p_1)(1-p_2) & ,i=1, j=1 \end{cases}$$

aşağıdaki gibi gösterilebilir.

$$\begin{aligned} \Pr(X_1 \oplus X_2 = 0) &= P_{r_r}(X_1 = X_2) \\ &= P_r(X_1=0, X_2=0) + P_r(X_1=1, X_2=1) \\ &= p_1 p_2 + (1-p_1)(1-p_2) \end{aligned}$$

p_1 yerine $p_1=1/2+\varepsilon_1$ ve p_2 yerine de $p_2=1/2+\varepsilon_2$ değerlerini yazalım. Kullanılan ε_1 *ve* ε_2 ihmal edilen olasılıklardır ve $-1/2 \le \varepsilon_1, \varepsilon_2 \le +1/2$ arasındadır. Buradan,

$\Pr(X_1 \oplus X_2 = 0) = 1/2 + 2\varepsilon_1\varepsilon_2$ olur ve $X_1 \oplus X_2 = 0$ 'ın $\varepsilon_{1,,2}$ aralığı $\varepsilon_{1,,2} = 2\varepsilon_1\varepsilon_2$ dir.

Bu iki tesadüfi ikili(binary) X_1'den X_n'e kadar olan $p_1=1/2+\varepsilon_1$'den $p_n=1/2+\varepsilon_n$'e olasılıkla daha fazla büyütülebilir. $X_1 \oplus \ldots\ldots \oplus X_n = 0$'da tutulan olasılık bütün n adet tesadüfi ikili bağımsız değerleri olduğunu kabul eden önerme Lemma tarafından tanımlanmıştır (Heys, 2002).

Önerme (Lemma) : *N adet bağımsız* tesadüfi ikili değerler $X_1,X_2,...,X_n$ olsun. Bunların toplam sapması;

$$\Pr(X_1 \oplus \ldots\ldots \oplus X_n = 0) = 1/2 + 2^{n-1} \prod_{i=1}^{n} \varepsilon_i$$

veya eşdeğer olarak,

$\varepsilon_{1,2,\ldots,n} = 2^{n-1} \prod_{i=1}^{n} \varepsilon_i$ formülünden hesaplanır (Heys, 2002).

11.3. Şifre Bileşenlerinin Analizi

İlk olarak nonlineer yapıda olan S-kutusunun doğrusal yaklaşımlarını bulmalıyız.

S kutusuna olan girişleri $X_1X_2X_3X_4$ olarak ve çıkışları da $Y_1Y_2Y_3Y_4$ olarak ifade edelim. Tablo 11.2'deki giriş toplamlarını a = (a_1,a_2,a_3,a_4), çıkış toplamlarını da b = (b_1,b_2,b_3,b_4) olan 4 bitlik ikili vektörlerle ifade edelim ve a=1110 olan giriş toplamı ile b=0001 olan çıkış toplamının ½'den sapmasını hesaplayalım.

Denklemimizin ana formu $a_1.X_1 \oplus a_2.X_2 \oplus a_3X_3 \oplus a_4.X_4 \oplus b_1.Y_1 \oplus b_2.Y_2 \oplus b_3.Y_3 \oplus b_4Y_4 = 0$ şeklindedir. Örneğimizde a = (1110) ve b = (0001) olduğundan dolayı denklemimiz $1.X_1 \oplus 1.X_2 \oplus 1.X_3 \oplus 0.X_4 \oplus 0.Y_1 \oplus 0.Y_2 \oplus 0.Y_3 \oplus 1.Y_4 = 0$ şeklinde yazılabilir ve sonuçta $X_1 \oplus X_2 \oplus X_3 \oplus Y_4 = 0$ şeklini alır. $X_1X_2X_3X_4$ giriş değerleri için 16 olası değerin herbiri, S kutusuna sokulur ve $Y_1Y_2Y_3Y_4$ çıkışları bulunarak her bir olası değer için Lineer Yaklaşım Tablosu oluşturulur.

Tablo 11.2. Tablo 11.1.b'deki S Kutusunun Lineer Yaklaşım 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	4	-2	0	-2	-2	0	2	0	2	0	2	4
	2	0	-2	-2	0	0	-2	2	0	-2	-4	-4	2	2	0	0	-2
	3	0	0	2	-2	0	-4	-2	-2	4	0	-2	-2	0	0	-2	2
	4	0	2	-4	-2	-2	0	-2	0	-2	0	-2	0	4	-2	0	2
	5	0	0	0	4	-2	2	-2	-2	0	-4	0	0	-2	-2	-2	2
	6	0	0	2	2	-2	-2	-4	4	0	0	2	2	2	2	0	0
	7	0	2	-2	4	2	0	0	2	2	0	-4	-2	0	2	2	0
	8	0	-4	-2	-2	2	2	-4	0	2	-2	0	0	0	0	+2	-2
	9	0	2	-2	0	2	-4	0	2	0	-2	+2	0	-2	-4	0	-2
	A	0	-2	0	2	-2	0	2	0	4	2	0	2	2	-4	2	0
	B	0	0	0	0	2	2	2	2	2	-2	2	-2	4	0	-4	0
	C	0	2	2	0	4	2	-2	0	0	2	-2	4	0	-2	-2	0
	D	0	0	-2	-2	0	0	2	2	2	-2	0	4	-2	2	0	4
	E	0	-4	0	0	0	0	0	4	-2	2	-2	-2	-2	-2	-2	2
	F	0	-2	-4	2	0	-2	0	-2	0	2	0	2	0	2	-4	-2

$X_1 \oplus X_2 \oplus X_3 \oplus Y_4=0$ denkleminin sonucunda 16 olası değer içerisinden 4 tanesi 0, 12 taneside 1 olarak bulundu (Şekil 11.2.). Denklemimizi sağlayan 0 değerleri olduğu için 0'ların olasılığı 4/16=1/4=0,25 tir. Bu olasılığın ½'den sapmasına hesaplarsak;

Sapmayı $\varepsilon = 4/16 - 1/2 = (4 - 8)/16 = -4/16 = -1/4$ olarak buluruz.

Böylece giriş toplamı $a = (1110)_2 = (E)_{16}$ ve çıkış toplamı $b = (0001)_2 = (1)_{16}$ olan örneğimizin ½'den sapmasını -1/4 olarak hesaplamış olduk.

Şekil 11.2'de örnek uygulamada gösterildiği gibi tüm olası giriş toplamı değerleri ile çıkış toplamı değerlerinin sapmaları hesaplanarak, Doğrusal Kriptanaliz için kullanacağımız Tablo 11.2'deki LAT(Linear Approximation Table - Lineer Yaklaşım Tablosu) elde edilir.

$X_1X_2X_3X_4$	$Y_1Y_2Y_3Y_4$	$X_1 \oplus X_2 \oplus X_3$	$X_1 \oplus X_2 \oplus X_3 \oplus Y_4$
0000	0011	0	1
0001	1101	0	1
0010	1010	1	1
0011	0010	1	1
0100	0001	1	0
0101	0111	1	0
0110	1011	0	1
0111	0101	0	1
1000	1100	1	1
1001	1110	1	1
1010	1111	0	1
1011	0110	0	0
1100	1001	0	1
1101	1000	0	0
1110	0000	1	1
1111	0100	1	1

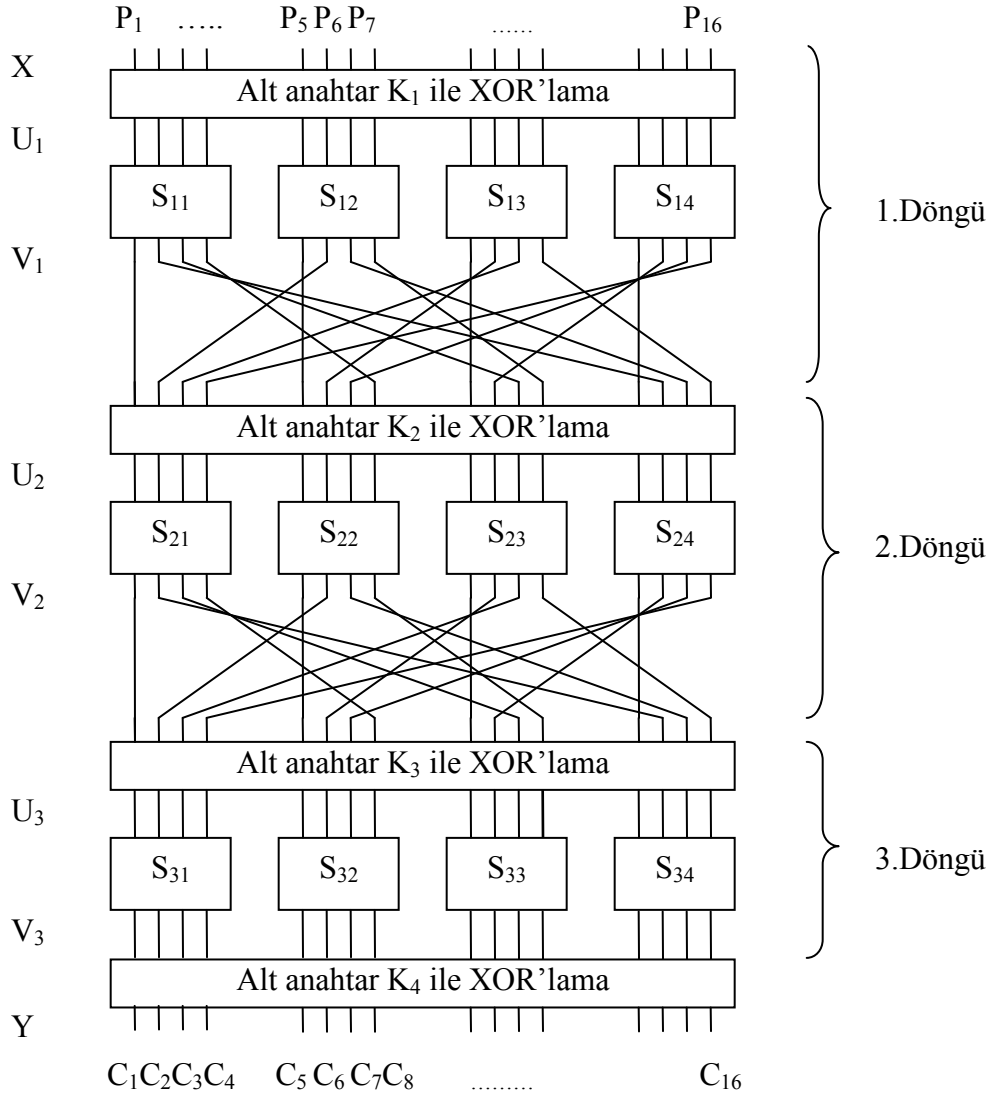
Şekil 11.2. Örnek Uygulama

Farklı döngüleri birleştirirken Matsui'nin DES algoritmasında ortaya koyduğu gibi tüm sapma, $\varepsilon_t = 2^{n-1}\ \varepsilon_1\ \varepsilon_2\ ....\ \varepsilon_n$ ile hesaplanır. Burada n denklem sayısını göstermektedir ve denklemlere ait sapmalar tüm algoritma için sapmanın değerinin hesaplanmasında kullanılmıştır (Sakallı vd., 2004).

11.4. Üç Döngülük SPN Algoritmasında 4 Bit Anahtarın Elde Edilmesi

Aşağıda şekildeki SPN algoritmasına (Şekil 11.3.) lineer kriptanaliz saldırısı yapabilmek için açık metin bitleri ile son döngüde S kutusuna yada kutularına giriş bitleri arasında bir lineer yaklaşım bulmalı; ve bu yaklaşımı kullanarak anahtar bitlerine saldırı yapmamız gerekmektedir.

İlk olarak Şekil 11.4'te görüldüğü gibi P_5, P_6, P_7 açık metin bitleri ile $S_{3,2}$ kutusuna giriş olan $U_{3,6}$ biti arasında lineer bir yaklaşım bulmalıyız.



Şekil 11.3. 3 döngülük SPN algoritması

Lineer yaklaşımın elde edilmesindeki aşamalar aşağıda gösterilmiştir.

S_{12} : $X_1 \oplus X_2 \oplus X_3 = Y_4$ $\varepsilon_1 = -1/4$

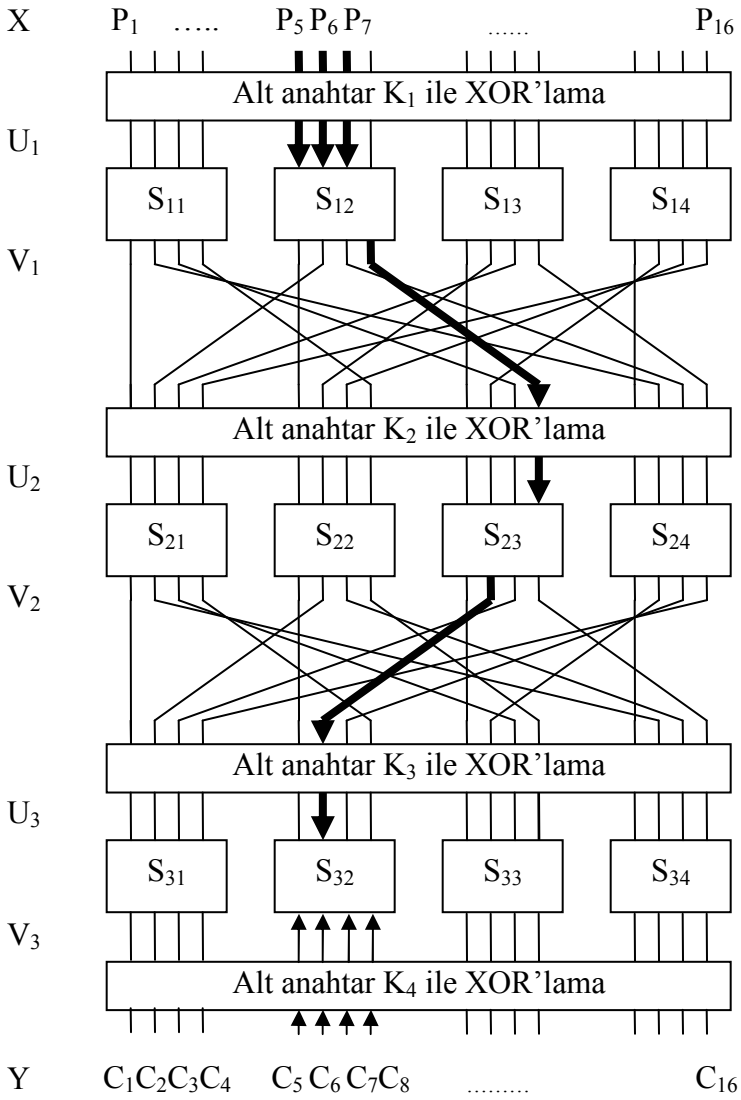
$$P_5 \oplus K_{1,5} \oplus P_6 \oplus K_{1,6} \oplus P_7 \oplus K_{1,7} = V_{1,8} \qquad V_{1,8} \oplus K_{2,12} = U_{2,12}$$

$$P_5 \oplus K_{1,5} \oplus P_6 \oplus K_{1,6} \oplus P_7 \oplus K_{1,7} \oplus K_{2,12} \oplus U_{2,12} = 0 \qquad (11.1)$$

S_{23} : $X_4 = Y_2$ $\varepsilon_2 = 1/4$

$$U_{2,12} = V_{2,10} \qquad V_{2,10} \oplus K_{3,6} = U_{3,6}$$

$$U_{2,12} = V_{2,10} = K_{3,6} \oplus U_{3,6} \qquad (11.2)$$



Şekil 11.4. Doğrusal (Lineer) Kriptanaliz Saldırısı

11.2 denklemini 11.1 denkleminde yerine yazarsak;

$$P_5 \oplus K_{1,5} \oplus P_6 \oplus K_{1,6} \oplus P_7 \oplus K_{1,7} \oplus K_{2,12} \oplus U_{2,12} = 0$$

$$P_5 \oplus K_{1,5} \oplus P_6 \oplus K_{1,6} \oplus P_7 \oplus K_{1,7} \oplus K_{2,12} \oplus K_{3,6} \oplus U_{3,6} = 0$$

$$P_5 \oplus P_6 \oplus P_7 \oplus \underbrace{K_{1,5} \oplus K_{1,6} \oplus K_{1,7} \oplus K_{2,12} \oplus K_{3,6}} \oplus U_{3,6} = 0$$

$\Sigma K = 0$ yada $\Sigma K = 1$ olur.

$P_5 \oplus P_6 \oplus P_7 \oplus U_{3,6} = 0$ olarak bulunur.

Buna göre şifrenin tamamı için Toplam Sapmamızı hesaplarsak;

$\varepsilon_T = 2^{n-1} \varepsilon_1 \varepsilon_2 \ldots\ldots \varepsilon_n$ formülünde n değerimiz 2 olduğundan,

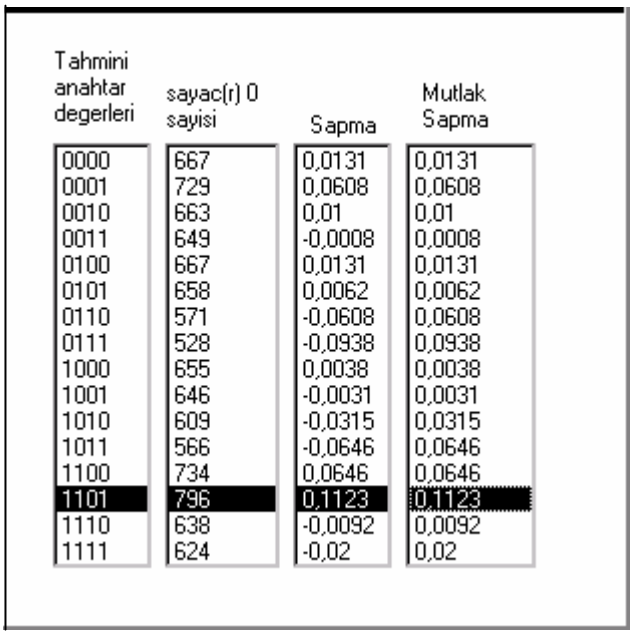
$\varepsilon_T = 2^1 \varepsilon_1 \varepsilon_2$

$\varepsilon_T = 2^1 (-1/4) (1/4) = - 1/8$ sonucunu elde ederiz.

$\Sigma K = 0$ ise sapmamız $\varepsilon_T = - 1/8$, $\Sigma K = 1$ ise sapmamız $\varepsilon_T = 1/8$ olur.

Bu aşamadan sonra kismi deşifreleme işlemi yapmamız gerekmektedir. Matsui saldırıda gerekli bilinen açık metinlerinin sayısının ε^{-2} ile orantılı olduğunu göstermiştir. Eğer L gerekli açık metin/şifreli metin sayısını temsil ederse $L = c.\varepsilon^{-2}$ ifadesi bize kullanmamız gereken açık metin/şifreli metin çiftlerinin sayısını verecektir. c sabit değerini de 20 seçersek sahip olmamız gereken açık/şifreli metin sayısı $L = 20. 8^2 = 1280$ 'dir.

Biz çalışmamızda 1300 adet açık/şifreli metin çifti kullandık. Algoritmadaki anahtar bitlerinin $K_{4,5}$, $K_{4,6}$, $K_{4,7}$, $K_{4,8}$ bulunabilmesi için her açık/şifreli metin çiftinin işlenmesi gerekmektedir. Anahtar bitlerinin bulunabilmesi için tüm olası anahtar değerler denenerek, (çalışmamızda 4 bit anahtar arandığından bu $2^4 = 16$ olası değer demek) $P_5 \oplus P_6 \oplus P_7 \oplus U_{3,6} = 0$ ifadesi bir anahtar değeri tuttuğunda sayacı o anahtar değeri için bir arttırdık ve bunu tüm açık/şifreli metin çiftleri için yaptık. Tablo 11.3'te örnek bir açık metin / şifreli metin üzerinde yapılan işlemler gösterilmektedir. Sonuçta 1/8 değerine en yakın sonucu veren anahtar, uygulamada kullanılan $K_{4,5}$, $K_{4,6}$, $K_{4,7}$, $K_{4,8}$ anahtar bitlerini bize verecektir.



Tahmini anahtar degerleri	sayac(r) 0 sayisi	Sapma	Mutlak Sapma
0000	667	0,0131	0,0131
0001	729	0,0608	0,0608
0010	663	0,01	0,01
0011	649	-0,0008	0,0008
0100	667	0,0131	0,0131
0101	658	0,0062	0,0062
0110	571	-0,0608	0,0608
0111	528	-0,0938	0,0938
1000	655	0,0038	0,0038
1001	646	-0,0031	0,0031
1010	609	-0,0315	0,0315
1011	566	-0,0646	0,0646
1100	734	0,0646	0,0646
1101	796	0,1123	0,1123
1110	638	-0,0092	0,0092
1111	624	-0,02	0,02

Şekil 11.5. 4 bit anahtarı ($K_{4,5}K_{4,6}K_{4,7}K_{4,8}$) bulma işeminde rastgele seçilen 1300 adet açık/şifreli metin çifti için program çıktısı örneği

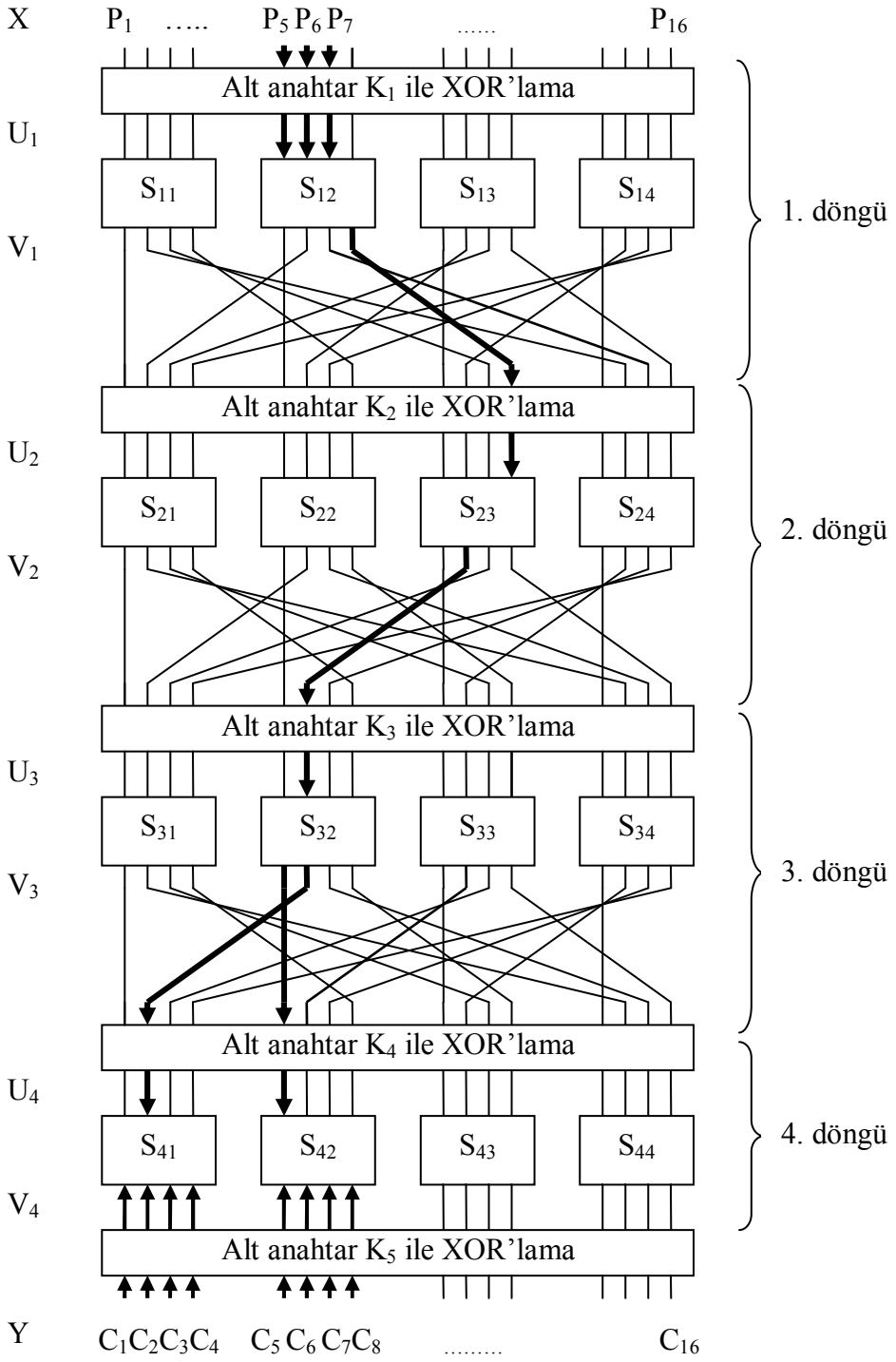
Şekil 11.5'te bu anahtar değerinin İkilik notasyonda "1101" olduğu görülmektedir. "1101" anahtarı (Hexadecimal olarak D) için elde edilen 0,1123 değeri en büyük mutlak sapma ve 1/8'e en yakın olan değerdir. Sapmamız L=1300 olduğundan ε = (Sayaç - 650) / 1300 formülünden hesaplanır. Ancak $\Sigma K = 0$ yada $\Sigma K = 1$ olabileceğinden sapmanın mutlağını düşünerek anahtarı belirleriz.

11.5. Dört Döngülük SPN Algoritmasında 8 Bit Anahtarın Elde Edilmesi

Şekil 11.1'deki SPN algoritmasına doğrusal kriptanaliz saldırısı yapabilmek için açık metin bitleri ile son döngüde S kutusuna yada kutularına giriş bitleri arasında bir doğrusal yaklaşım bulmalı; ve bu yaklaşımı kullanarak anahtar bitlerine saldırı yapmamız gerekmektedir.

Tablo 11.3. Örnek bir açık metin / şifreli metin üzerinde yapılan işlemlerin gösterimi

Açık metin	**Şifreli metin**	**Olası anahtar**				**F=1110**			**Çıkış bitleri=7**				**Çıkış bitleri ile olası anahtarın XOR sonucu**	**Xor sonucu ters S kutusu**				**P_5 P_6 P_7 $U_{3,6}$ xor sonucu**
		$K_{4,5}$	$K_{4,6}$	$K_{4,7}$	$K_{4,8}$	P_5	P_6	P_7	C_5	C_6	C_7	C_8		$U_{3,5}$	$U_{3,6}$	$U_{3,7}$	$U_{3,8}$	
8**F**34	5783	0	0	0	0	**1**	**1**	**1**	0	1	1	1	0111	0	**1**	0	1	**0**
		0	0	0	1	**1**	**1**	**1**	0	1	1	1	0110	1	**0**	1	1	**1**
		0	0	1	0	**1**	**1**	**1**	0	1	1	1	0101	0	**1**	1	1	**0**
		0	0	1	1	**1**	**1**	**1**	0	1	1	1	0100	1	**1**	1	1	**0**
		0	1	0	0	**1**	**1**	**1**	0	1	1	1	0011	0	**0**	0	0	**1**
		0	1	0	1	**1**	**1**	**1**	0	1	1	1	0010	0	**0**	1	1	**1**
		0	1	1	0	**1**	**1**	**1**	0	1	1	1	0001	0	**1**	0	0	**0**
		0	1	1	1	**1**	**1**	**1**	0	1	1	1	0000	1	**1**	1	0	**0**
		1	0	0	0	**1**	**1**	**1**	0	1	1	1	1111	1	**0**	1	0	**1**
		1	0	0	1	**1**	**1**	**1**	0	1	1	1	1110	1	**0**	0	1	**1**
		1	0	1	0	**1**	**1**	**1**	0	1	1	1	1101	0	**0**	0	1	**1**
		1	0	1	1	**1**	**1**	**1**	0	1	1	1	1100	1	**0**	0	0	**1**
		1	1	0	0	**1**	**1**	**1**	0	1	1	1	1011	0	**1**	1	0	**0**
		1	1	0	1	**1**	**1**	**1**	0	1	1	1	1010	0	**0**	1	0	**1**
		1	1	1	0	**1**	**1**	**1**	0	1	1	1	1001	1	**1**	0	0	**0**
		1	1	1	1	**1**	**1**	**1**	0	1	1	1	1000	1	**1**	0	1	**0**



Şekil 11.6. Doğrusal (Lineer) Kriptanaliz Saldırısı

İlk olarak Şekil 11.6'da görüldüğü gibi P_5, P_6, P_7 açık metin bitleri ile $S_{4,1}$ kutusuna giriş olan $U_{4,2}$ biti ve $S_{4,2}$ kutusuna giriş olan $U_{4,5}$ biti arasında doğrusal bir yaklaşım bulmalıyız. Daha sonra son döngüdeki $K_{5,1}$, $K_{5,2}$, $K_{5,3}$, $K_{5,4,}$ $K_{5,5}$, $K_{5,6}$, $K_{5,7}$, $K_{5,8}$

alt anahtar bitlerini elde edebilmek için, elde edilen bu doğrusal yaklaşım kullanılarak, istatistiksel saldırıyı gerçekleştirebiliriz.

Doğrusal yaklaşımın elde edilmesindeki aşamalar aşağıda gösterilmiştir.

$S_{12}: X_1 \oplus X_2 \oplus X_3 = Y_4 \qquad \varepsilon_1 = -1/4$

$P_5 \oplus K_{1,5} \oplus P_6 \oplus K_{1,6} \oplus P_7 \oplus K_{1,7} = V_{1,8} \qquad V_{1,8} \oplus K_{2,12} = U_{2,12}$

$P_5 \oplus K_{1,5} \oplus P_6 \oplus K_{1,6} \oplus P_7 \oplus K_{1,7} \oplus K_{2,12} \oplus U_{2,12} = 0 \qquad (11.3)$

$S_{23}: X_4 = Y_2 \qquad \varepsilon_2 = 1/4$

$U_{2,12} = V_{2,10} \qquad V_{2,10} \oplus K_{3,6} = U_{3,6}$

$U_{2,12} = V_{2,10} = K_{3,6} \oplus U_{3,6} \qquad (11.4)$

$S_{32}: X_2 = Y_1 \oplus Y2 \qquad \varepsilon_3 = -1/4$

$U_{3,6} = V_{3,5} \oplus V_{3,6} \qquad V_{3,5} \oplus K_{4,5} = U_{4,5}$

$V_{3,6} \oplus K_{4,2} = U_{4,2}$

$U_{3,6} = K_{4,5} \oplus U_{4,5} \oplus K_{4,2} \oplus U_{4,2} \qquad (11.5)$ olur.

11.4 ve 11.5 denklemlerimizi 11.3 denklemimizle birleştirirsek;

$P_5 \oplus K_{1,5} \oplus P_6 \oplus K_{1,6} \oplus P_7 \oplus K_{1,7} \oplus K_{2,12} \oplus U_{2,12} = 0$

$P_5 \oplus K_{1,5} \oplus P_6 \oplus K_{1,6} \oplus P_7 \oplus K_{1,7} \oplus K_{2,12} \oplus K_{3,6} \oplus U_{3,6} = 0$

$P_5 \oplus K_{1,5} \oplus P_6 \oplus K_{1,6} \oplus P_7 \oplus K_{1,7} \oplus K_{2,12} \oplus K_{3,6} \oplus K_{4,5} \oplus U_{4,5} \oplus K_{4,2} \oplus U_{4,2} = 0$

$P_5 \oplus P_6 \oplus P_7 \oplus \underbrace{K_{1,5} \oplus K_{1,6} \oplus K_{1,7} \oplus K_{2,12} \oplus K_{3,6} \oplus K_{4,5} \oplus K_{4,2}} \oplus U_{4,2} \oplus U_{4,5} = 0$

$\Sigma K = 0$ yada $\Sigma K = 1$ olur.

$P_5 \oplus P_6 \oplus P_7 \oplus U_{4,2} \oplus U_{4,5} = 0$ olarak bulunur.

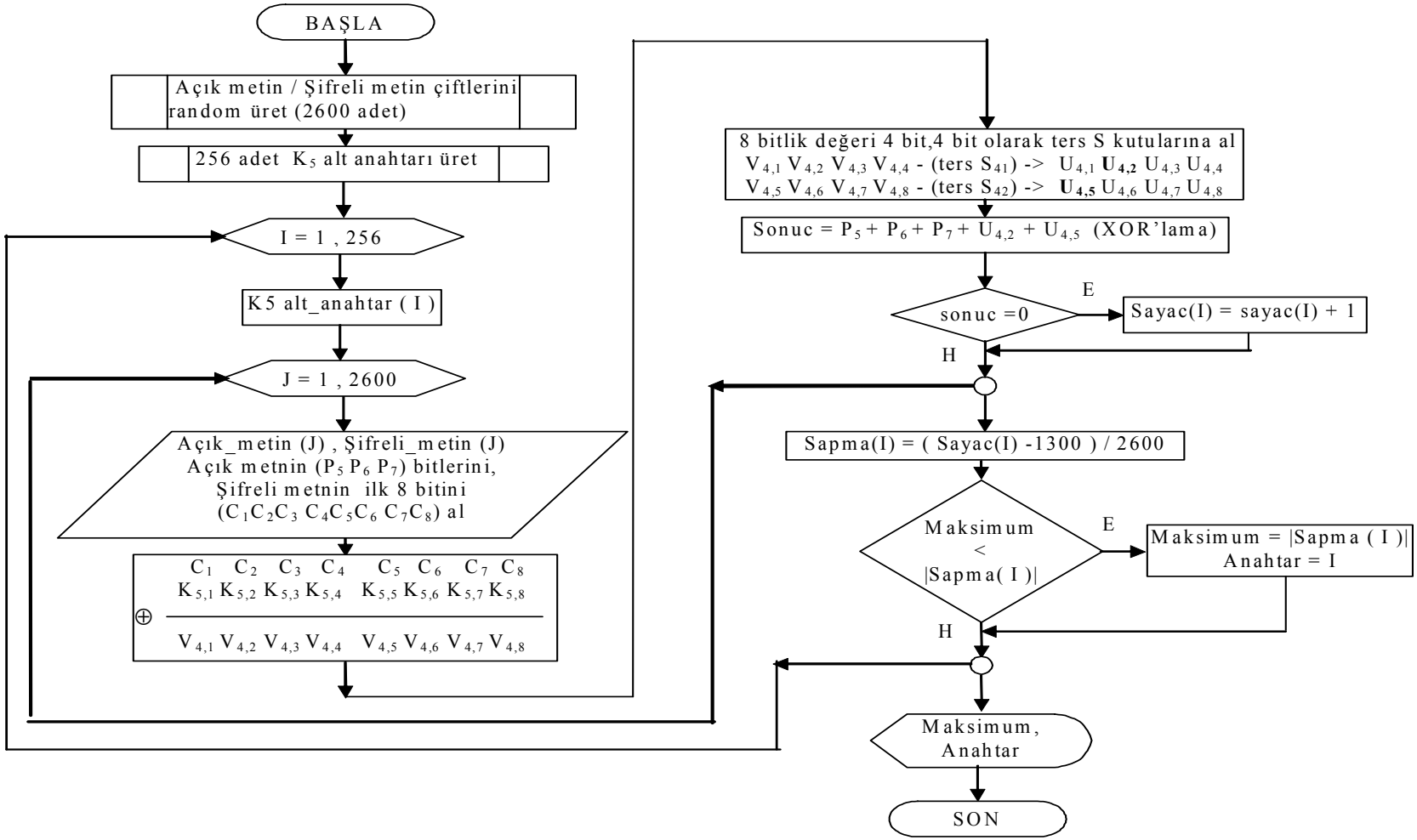
Buna göre şifrenin tamamı için Toplam Sapmamızı hesaplarsak;

$\varepsilon_T = 2^{n-1} \varepsilon_1 \varepsilon_2 \ldots\ldots \varepsilon_n$ formülünde n değerimiz 3 olduğundan,

$\varepsilon_T = 2^2 \varepsilon_1 \varepsilon_2 \varepsilon_3$

$\varepsilon_T = 2^2 (-1/4)(1/4)(-1/4) = 1/16$ sonucunu elde ederiz.

$\Sigma K = 0$ ise sapmamız $\varepsilon_T = 1/16$, $\Sigma K = 1$ ise sapmamız $\varepsilon_T = -1/16$ olur.



Şekil 11.6. 2600 adet açık/şifreli metin çifti için 8 bit anahtarı ($K_{5,1}K_{5,2}K_{5,3}K_{5,4}K_{5,5}K_{5,6}K_{5,7}K_{5,8}$) bulmada kullanılan akış şeması

8 bit anahtarın bulunmasında kullanılan algoritma şu şekildedir:

1. Şifreli metnin ilk 8 biti alınır ve K_5 alt anahtarının olası bütün $K_{5,1}$ $K_{5,2}$ $K_{5,3}$ $K_{5,4}$ $K_{5,5}$ $K_{5,6}$ $K_{5,7}$ $K_{5,8}$ değerleri ile (Toplam 16*16=256 olası değer) ile XOR'lanır.
2. 8 bitlik değer 4 bit, 4 bit olarak (S_{41} ve S_{42} S kutularına) sırasıyla ters s kutularına sokulur.
3. Ters S_{41} kutusunun $U_{4,2}$ değeri ile S_{42} kutusunun $U_{4,5}$ değeri kaydedilir.
4. Açık metnin P_5 P_6 P_7 değerleri ile $U_{4,2}$ ve $U_{4,5}$ değerleri XOR'lanarak sonucun 0 verdiği anahtar değerlerinde, o anahtar değeri için sayaç 1 arttırılır.

Bu aşamalardan sonra kısmi deşifreleme işlemi yapmamız gerekmektedir. Çünkü doğrusal kriptanaliz bilinen açık metin saldırısı olduğundan biz belli sayıda açık metine ve bunların şifreli metinlerine sahip olmamız gerekmektedir. Matsui saldırıda gerekli bilinen açık metinlerinin sayısının ε^{-2} ile orantılı olduğunu göstermiştir. Eğer L gerekli açık metin/şifreli metin sayısını temsil ederse $L= c.\varepsilon^{-2}$ ifadesi bize kullanmamız gereken açık metin/şifreli metin çiftlerinin sayısını verecektir. Burada c küçük bir sabit değeri temsil eder. Çalışmamızda açık metin bitleri ile S kutusuna giriş bitleri arasında +1/16 sapma bulduğumuza göre ve c sabit değerini 10 seçersek sahip olmamız gereken açık metin/şifreli metin sayısı $L = 10.16^2 = 2560$'dır. Çalışmamızda ilk olarak 2600 adet açık metin/şifreli metin çifti kullandık. Yani öncelikle Şekil 11.1. algoritması için rastgele 2600 adet açık metin/şifreli metin ürettik. Algoritmadaki anahtar bitlerinin $K_{5,1}$, $K_{5,2}$, $K_{5,3}$, $K_{5,4,}$ $K_{5,5}$, $K_{5,6}$, $K_{5,7}$, $K_{5,8}$ bulunabilmesi için her açık metin/şifreli metin çiftinin işlenmesi gerekmektedir. Bu işlemde Şekil 11.6'teki yöntem kullanılmıştır. Anahtar bitlerinin bulunabilmesi için tüm olası anahtar değerler denenerek, çalışmamızda 8 bit anahtar arandığından bu 256 olası değer demek,

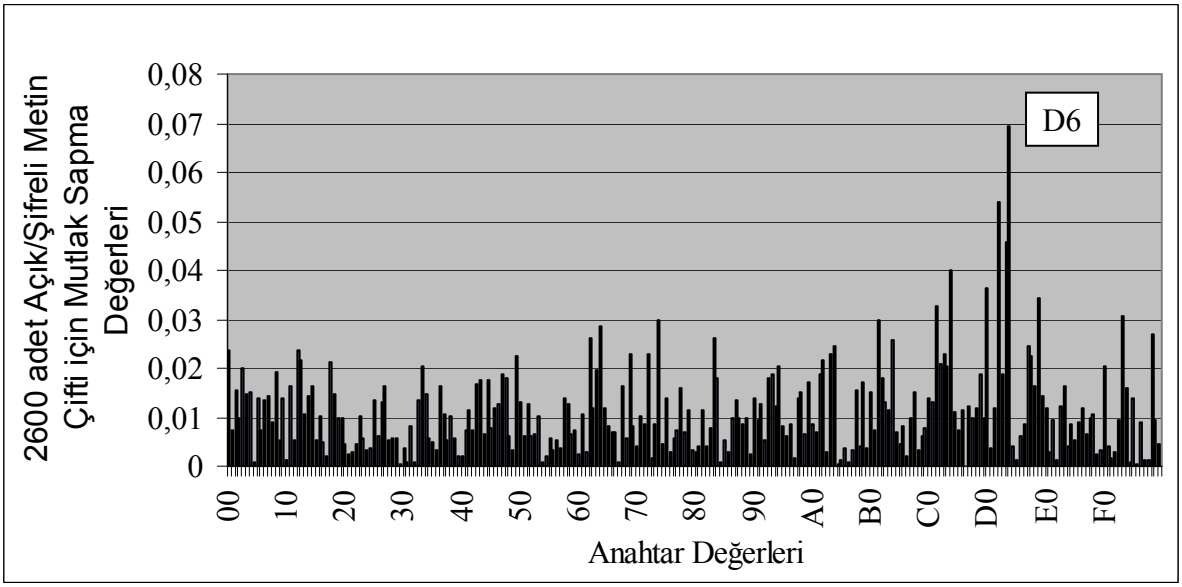
$P_5 \oplus P_6 \oplus P_7 \oplus U_{4,2} \oplus U_{4,5} = 0$ ifadesini bir anahtar değeri tuttuğunda sayacı o anahtar değeri için bir attırdık ve bunu tüm açık metin/şifreli metin çiftleri için yaptık.

Sonuçta 1/16 değerine en yakın sonucu veren yada 1/2 değerinden en büyük mutlak sapmaya uğrayan anahtar uygulamada kullanılan $K_{5,1}$, $K_{5,2}$, $K_{5,3}$, $K_{5,4,}$ $K_{5,5}$, $K_{5,6}$, $K_{5,7}$, $K_{5,8}$ bitlerini bize verecektir. Tablo 11.4'te bu anahtar değerinin Hexadecimal notasyonda "D6" olduğu görülmektedir. "D6" anahtarı için elde edilen 0,0692 değeri en büyük mutlak sapma ve 1/16'ya en yakın olan değerdir. Tablo 11.4'teki sapma değerleri, L= 2600 olduğu için ε = (Sayaç - 1300) / 2600 ifadesinden hesaplanır. Ancak $\Sigma K = 0$ yada $\Sigma K = 1$ olabileceğinden sapmanın mutlağını düşünerek anahtarı belirleriz.

Tablo 11.4 Lineer Kriptanaliz Saldırısının Deneysel Sonuçları (8 bit anahtarı bulmada seçilen 2600 adet açık/şifreli metin çifti için)

Anahtar	Mutlak Sapma	Anahtar	Mutlak Sapma	Anahtar	Mutlak Sapma	Anahtar	Mutlak Sapma	Anahtar	Mutlak Sapma	Anahtar	Mutlak Sapma	Anahtar	Mutlak Sapma	Anahtar	Mutlak Sapma
00	0,0235	**20**	0,0046	**40**	0,0019	**60**	0,0023	**80**	0,0027	**A0**	0,0085	**C0**	0,0138	**E0**	0,0119
01	0,0073	**21**	0,0023	**41**	0,0073	**61**	0,0108	**81**	0,0042	**A1**	0,0069	**C1**	0,0131	**E1**	0,0027
02	0,0154	**22**	0,0027	**42**	0,0115	**62**	0,0027	**82**	0,0115	**A2**	0,0188	**C2**	0,0327	**E2**	0,0092
03	0,0096	**23**	0,0046	**43**	0,0073	**63**	0,0262	**83**	0,0042	**A3**	0,0215	**C3**	0,0208	**E3**	0,0012
04	0,02	**24**	0,0104	**44**	0,0169	**64**	0,0119	**84**	0,0077	**A4**	0,0027	**C4**	0,0227	**E4**	0,0123
05	0,0146	**25**	0,0058	**45**	0,0177	**65**	0,0196	**85**	0,0262	**A5**	0,0227	**C5**	0,0204	**E5**	0,0162
06	0,015	**26**	0,0031	**46**	0,0065	**66**	0,0285	**86**	0,0181	**A6**	0,0246	**C6**	0,04	**E6**	0,0042
07	0,0008	**27**	0,0035	**47**	0,0177	**67**	0,0119	**87**	0,0008	**A7**	0,0004	**C7**	0,0112	**E7**	0,0085
08	0,0138	**28**	0,0135	**48**	0,0077	**68**	0,0081	**88**	0,0054	**A8**	0,0012	**C8**	0,0073	**E8**	0,0054
09	0,0073	**29**	0,0062	**49**	0,0119	**69**	0,0069	**89**	0,0027	**A9**	0,0038	**C9**	0,0115	**E9**	0,0088
0A	0,0135	**2A**	0,0131	**4A**	0,0127	**6A**	0,0069	**8A**	0,0096	**AA**	0,0008	**CA**	0	**EA**	0,0119
0B	0,0142	**2B**	0,0162	**4B**	0,0188	**6B**	0,0008	**8B**	0,0135	**AB**	0,0031	**CB**	0,0123	**EB**	0,0065
0C	0,0088	**2C**	0,0054	**4C**	0,0181	**6C**	0,0162	**8C**	0,0096	**AC**	0,0154	**CC**	0,01	**EC**	0,0096
0D	0,0192	**2D**	0,0058	**4D**	0,0062	**6D**	0,0058	**8D**	0,0085	**AD**	0,0042	**CD**	0,0119	**ED**	0,0108
0E	0,0054	**2E**	0,0058	**4E**	0,0031	**6E**	0,0227	**8E**	0,01	**AE**	0,0173	**CE**	0,0188	**EE**	0,0023
0F	0,0138	**2F**	0,0004	**4F**	0,0223	**6F**	0,0081	**8F**	0,0023	**AF**	0,0035	**CF**	0,0096	**EF**	0,0031
10	0,0012	**30**	0,0038	**50**	0,0131	**70**	0,0042	**90**	0,0138	**B0**	0,015	**D0**	0,0362	**F0**	0,0204
11	0,0165	**31**	0,0008	**51**	0,0062	**71**	0,0104	**91**	0,0092	**B1**	0,0073	**D1**	0,0038	**F1**	0,0042
12	0,0054	**32**	0,0081	**52**	0,0127	**72**	0,0085	**92**	0,0127	**B2**	0,03	**D2**	0,0119	**F2**	0,0015
13	0,0235	**33**	0,0008	**53**	0,0062	**73**	0,0227	**93**	0,0054	**B3**	0,0181	**D3**	0,0538	**F3**	0,0027
14	0,0215	**34**	0,0135	**54**	0,0065	**74**	0,0015	**94**	0,0181	**B4**	0,0131	**D4**	0,0188	**F4**	0,0092
15	0,0108	**35**	0,0204	**55**	0,0104	**75**	0,0085	**95**	0,0188	**B5**	0,0115	**D5**	0,0458	**F5**	0,0308
16	0,0142	**36**	0,0146	**56**	0,0008	**76**	0,0296	**96**	0,0123	**B6**	0,0258	**D6**	**0,0692**	**F6**	0,0158
17	0,0162	**37**	0,0058	**57**	0,0019	**77**	0,0046	**97**	0,0204	**B7**	0,0069	**D7**	0,0042	**F7**	0,0008
18	0,0054	**38**	0,005	**58**	0,0058	**78**	0,0138	**98**	0,0081	**B8**	0,0046	**D8**	0,0012	**F8**	0,0138
19	0,0104	**39**	0,0031	**59**	0,0031	**79**	0,0027	**99**	0,0062	**B9**	0,0081	**D9**	0,0062	**F9**	0,0004
1A	0,005	**3A**	0,0162	**5A**	0,0054	**7A**	0,0058	**9A**	0,0085	**BA**	0,0019	**DA**	0,0085	**FA**	0,0088
1B	0,0019	**3B**	0,0108	**5B**	0,0038	**7B**	0,0073	**9B**	0,0015	**BB**	0,0096	**DB**	0,0246	**FB**	0,0012
1C	0,0212	**3C**	0,0054	**5C**	0,0138	**7C**	0,0158	**9C**	0,0138	**BC**	0,015	**DC**	0,0223	**FC**	0,0012
1D	0,0146	**3D**	0,0104	**5D**	0,0127	**7D**	0,0069	**9D**	0,015	**BD**	0,0031	**DD**	0,0165	**FD**	0,0269
1E	0,01	**3E**	0,0058	**5E**	0,0065	**7E**	0,0115	**9E**	0,0065	**BE**	0,0062	**DE**	0,0342	**FE**	0,0092
1F	0,01	**3F**	0,0019	**5F**	0,0073	**7F**	0,0031	**9F**	0,0173	**BF**	0,0077	**DF**	0,0142	**FF**	0,0046

Şekil 11.7'te, Tablo 11.4'te elde edilen değerlerin grafiksel gösterilimi, deneysel sonuçların anlaşılırlığı için yapılmıştır

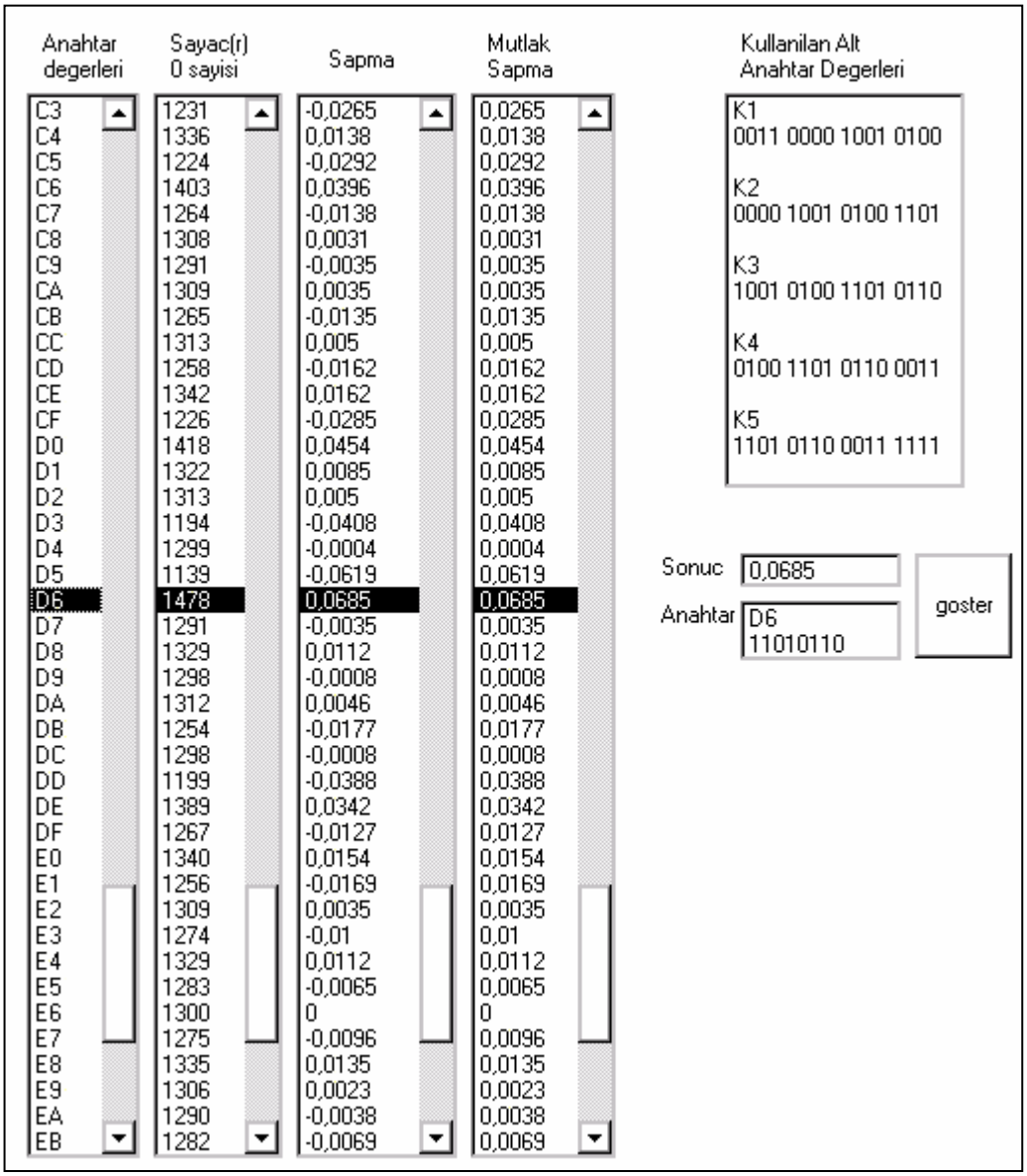


Şekil 11.7. Tablo 11.4'teki Deneysel Sonuçların Grafiksel Gösterimi

Çalışmamızda dört döngülük bir SPN algoritması için doğrusal kriptanaliz uygulaması gerçekleştirdik ve son döngüde kullanılan 16 bit anahtarın 8 bitini ($K_{5,1}K_{5,2}K_{5,3}K_{5,4}$ $K_{5,5}K_{5,6}K_{5,7}K_{5,8}$) başarılı bir şekilde hex notasyonda "D6" olarak elde ettik. Diğer yüksek mutlak sapma değerlerine sahip anahtar değerlerinin sebebini S kutusu özellikleri kaynaklı olarak açıklayabiliriz. Burada kullanılan SPN algoritmasında yer alan S kutusu sonlu alan aritmetiği kullanılarak tasarlanmıştır. Buna ek olarak permütasyon safhası için AES şifreleme algoritmasında kullanılan ShiftRows dönüşümü bit permütasyonu için kullanılmıştır.

Aşağıda, Şekil 11.8'de Rastgele (Random) seçilen farklı 2600 adet açık/şifreli metin çifti için Lineer Kriptanaliz sonucunda yine en yüksek mutlak sapmaya sahip anahtar Hexadacimal notasyonda "D6" olarak bulunmuştur.

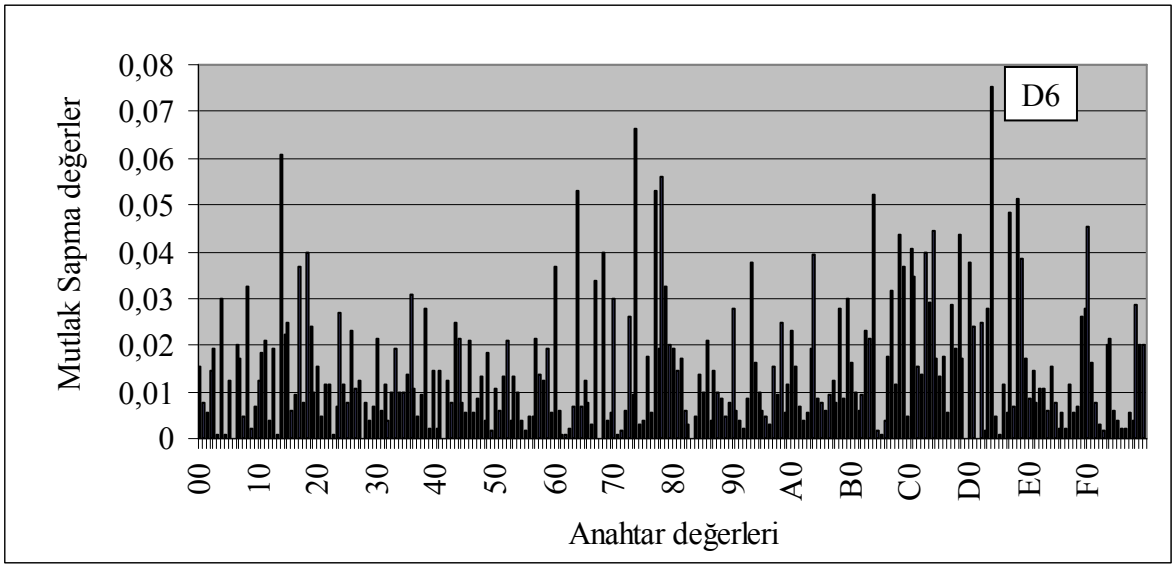
Kullanmamız gereken açık metin/şifreli metin çiftlerinin sayısını $L= c.\varepsilon^{-2}$ ifadesinden hesapladığını yukarıda anlatmiştik. Yine yukarıdaki örnekte, bu förmülde kullanılan c sabitini 10 alıp yaklaşık 2600 adet açık/şifreli metin çifti için doğrusal kriptanalizi gerçekleştirdik.



Şekil 11.8. 8 bit anahtarı bulma işeminde ($K_{5,1}K_{5,2}K_{5,3}K_{5,4}K_{5,5}K_{5,6}K_{5,7}K_{5,8}$) rastgele seçilen 2600 adet açık/şifreli metin çifti için program çıktısı örneği

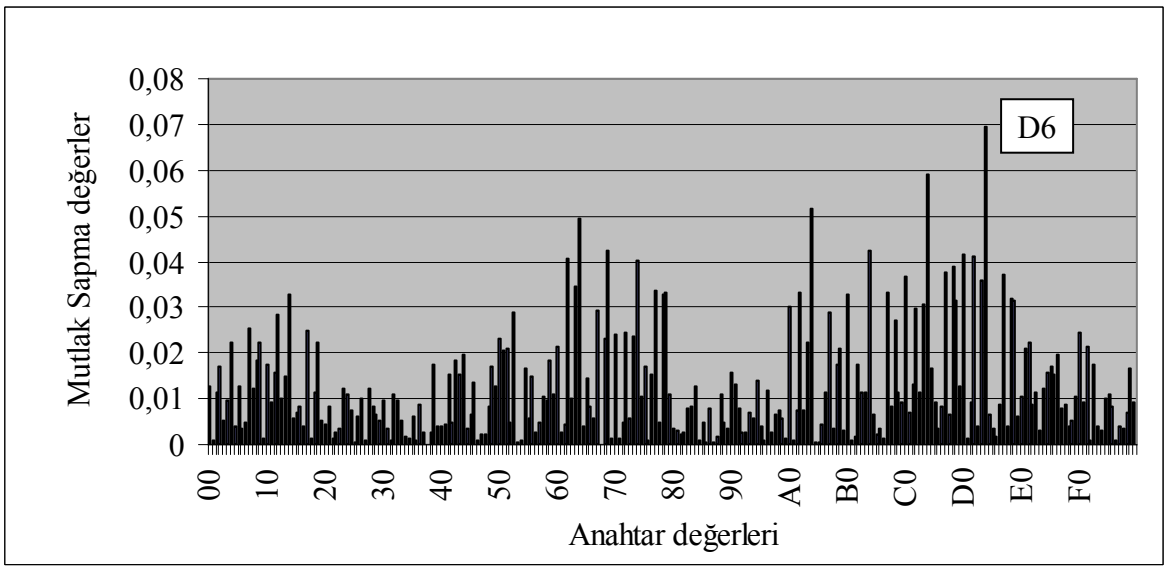
Şimdi de c sabitinin, sırasıyla 5,15 ve 20 değerleri için sahip olmamız gereken açık metin/şifreli metin sayısını bularak her anahtar değeri için mutlak sapmaları hesaplayalım, ve bulduğumuz deneysel sonuçları grafiksel olarak ifade edelim.

c = 5 için $L = c.\varepsilon^{-2} = 5 . 256 = 1280 \sim 1300$ adet açık/şifreli metin çifti için Deneysel sonuçlar Şekil 11.9'de gösterilmektedir.



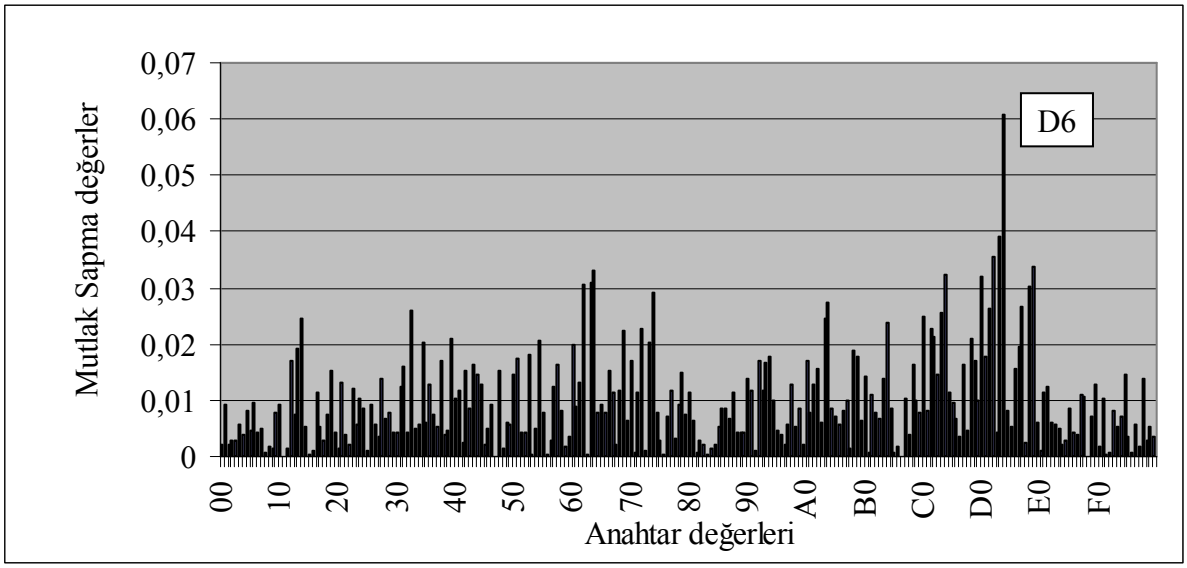
Şekil 11.9. 1300 adet açık/şifreli metin çifti için Mutlak Sapma değerleri

c = 15 için $L = c.\varepsilon^{-2} = 15*256 = 3840 \sim 3900$ adet açık/şifreli metin çifti için Deneysel sonuçlar Şekil 11.10'de gösterilmektedir.



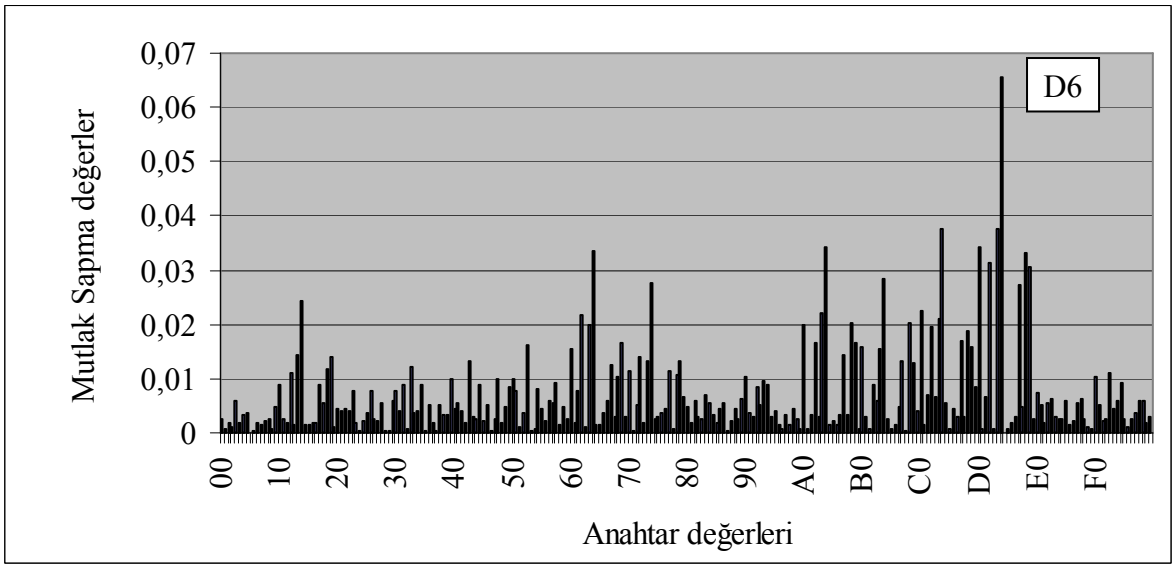
Şekil 11.10. 3900 adet açık/şifreli metin çifti için Mutlak Sapma değerleri

c = 20 için $L = c.\varepsilon^{-2} = 20*256 = 5120 \sim 5200$ adet açık/şifreli metin çifti için Deneysel sonuçlar Şekil 11.11'da gösterilmektedir.



Şekil 11.11. 5200 adet açık/şifreli metin çifti için Mutlak Sapma değerleri

Son olarak tüm açık/şifreli metin çiftleri için ($2^{16} = 65536$ adet) Mutlak Sapma değerlerini bulalım ve grafiksel olarak ifade edelim.



Şekil 11.12. 65536 (tüm) açık/şifreli metin çiftleri için Mutlak Sapma değerleri

Sonuçta grafiklerdende anlaşılacağı gibi denenen açık/şifreli metin sayımız, yani örnek uzayımız arttıkça diğer yüksek mutlak sapma değerlerine sahip anahtar değerlerinde azalmalar olmakta; ve asıl anahtarımız daha belirgin bir farkla ortaya çıkmaktadır.

Diğer yüksek mutlak sapma değerlerine sahip anahtar değerlerinin sebebini S kutusu özellikleri ve rastgele olarak seçilen açık/şifreli metin çiftlerinden kaynaklı olarak açıklayabiliriz.

11.6. 16 Bitlik Anahtarın Elde Edilmesi

Yukarıda yaptığımız çalışma ile son döngüde kullanılan 16 bitlik K_5 alt anahtarının 8 bitini başarıyla elde ettik. Bunun için Şekil 11.13'te görüldüğü gibi P_5, P_6, P_7 açık metin bitleri ile $S_{4,1}$ kutusuna giriş olan $U_{4,2}$ biti ve $S_{4,2}$ kutusuna giriş olan $U_{4,5}$ biti arasında doğrusal bir yaklaşım bulduk. Daha sonra da son döngüdeki $K_{5,1}$, $K_{5,2}$, $K_{5,3}$, $K_{5,4}$, $K_{5,5}$, $K_{5,6}$, $K_{5,7}$, $K_{5,8}$ alt anahtar bitlerini elde edebilmek için, elde edilen bu doğrusal yaklaşım kullanılarak, istatistiksel saldırıyı gerçekleştirdik.

Şimdi de K_5 alt anahtarının, kalan 8 bitini ($K_{5,9}$, $K_{5,10}$, $K_{5,11}$, $K_{5,12}$, $K_{5,13}$, $K_{5,14}$, $K_{5,15}$, $K_{5,16}$) de elde ederek 16 bitlik anahtarın tamamını bulalım. Bunu gerçekleştirmek için Şekil 11.13'te görüldüğü gibi P_{13}, P_{14}, P_{15} açık metin bitleri ile $S_{4,3}$ kutusuna giriş olan $U_{4,10}$ biti ve $S_{4,4}$ kutusuna giriş olan $U_{4,13}$ biti arasında doğrusal bir yaklaşım bularak, istatistiksel saldırıyı gerçekleştirebiliriz.

Lineer yaklaşımın elde edilmesindeki aşamalar şunlardır.

$S_{14}: X_1 \oplus X_2 \oplus X_3 = Y_4 \qquad \varepsilon_1 = -1/4$

$P_{13} \oplus K_{1,13} \oplus P_{14} \oplus K_{1,14} \oplus P_{15} \oplus K_{1,15} = V_{1,16} \qquad V_{1,16} \oplus K_{2,4} = U_{2,4}$

$P_{13} \oplus K_{1,13} \oplus P_{14} \oplus K_{1,14} \oplus P_{15} \oplus K_{1,15} \oplus K_{2,4} \oplus U_{2,4} = 0 \qquad (11.6)$

$S_{21}: X_4 = Y_2 \qquad \varepsilon_2 = 1/4$

$U_{2,4} = V_{2,2} \qquad V_{2,2} \oplus K_{3,14} = U_{3,14}$

$U_{2,4} = V_{2,2} = K_{3,14} \oplus U_{3,14} \qquad (11.7)$

$S_{34}: X_2 = Y_1 \oplus Y2 \qquad \varepsilon_3 = -1/4$

$U_{3,14} = V_{3,13} \oplus V_{3,14} \qquad V_{3,13} \oplus K_{4,13} = U_{4,13}$

$V_{3,14} \oplus K_{4,10} = U_{4,10}$

$U_{3,14} = K_{4,13} \oplus U_{4,13} \oplus K_{4,10} \oplus U_{4,10} \qquad (11.8)$ olur.

11.7 ve 11.8 denklemlerimizi 11.6 denklemimizle birleştirirsek;

$P_{13} \oplus K_{1,13} \oplus P_{14} \oplus K_{1,14} \oplus P_{15} \oplus K_{1,15} \oplus K_{2,4} \oplus U_{2,4} = 0$

$P_{13} \oplus K_{1,13} \oplus P_{14} \oplus K_{1,14} \oplus P_{15} \oplus K_{1,15} \oplus K_{2,4} \oplus K_{3,14} \oplus U_{3,14} = 0$

$P_{13} \oplus K_{1,13} \oplus P_{14} \oplus K_{1,14} \oplus P_{15} \oplus K_{1,15} \oplus K_{2,4} \oplus K_{3,14} \oplus K_{4,13} \oplus U_{4,13} \oplus K_{4,10} \oplus U_{4,10} = 0$

$$P_{13} \oplus P_{14} \oplus P_{15} \oplus \underbrace{K_{1,13} \oplus K_{1,14} \oplus K_{1,15} \oplus K_{2,4} \oplus K_{3,14} \oplus K_{4,13} \oplus K_{4,10}} \oplus U_{4,13} \oplus U_{4,10} = 0$$

$\Sigma K = 0$ yada $\Sigma K = 1$ olur.

$P_{13} \oplus P_{14} \oplus P_{15} \oplus U_{4,13} \oplus U_{4,10} = 0$ olarak bulunur.

Buna göre şifrenin tamamı için Toplam Sapmamızı hesaplarsak;

$\varepsilon_T = 2^{n-1} \varepsilon_1 \varepsilon_2 \ldots\ldots \varepsilon_n$ formülünde n değerimiz 3 olduğundan,

$\varepsilon_T = 2^2 \varepsilon_1 \varepsilon_2 \varepsilon_3$

$\varepsilon_T = 2^2 (-1/4) (1/4) (-1/4) = 1/16$ sonucunu elde ederiz.

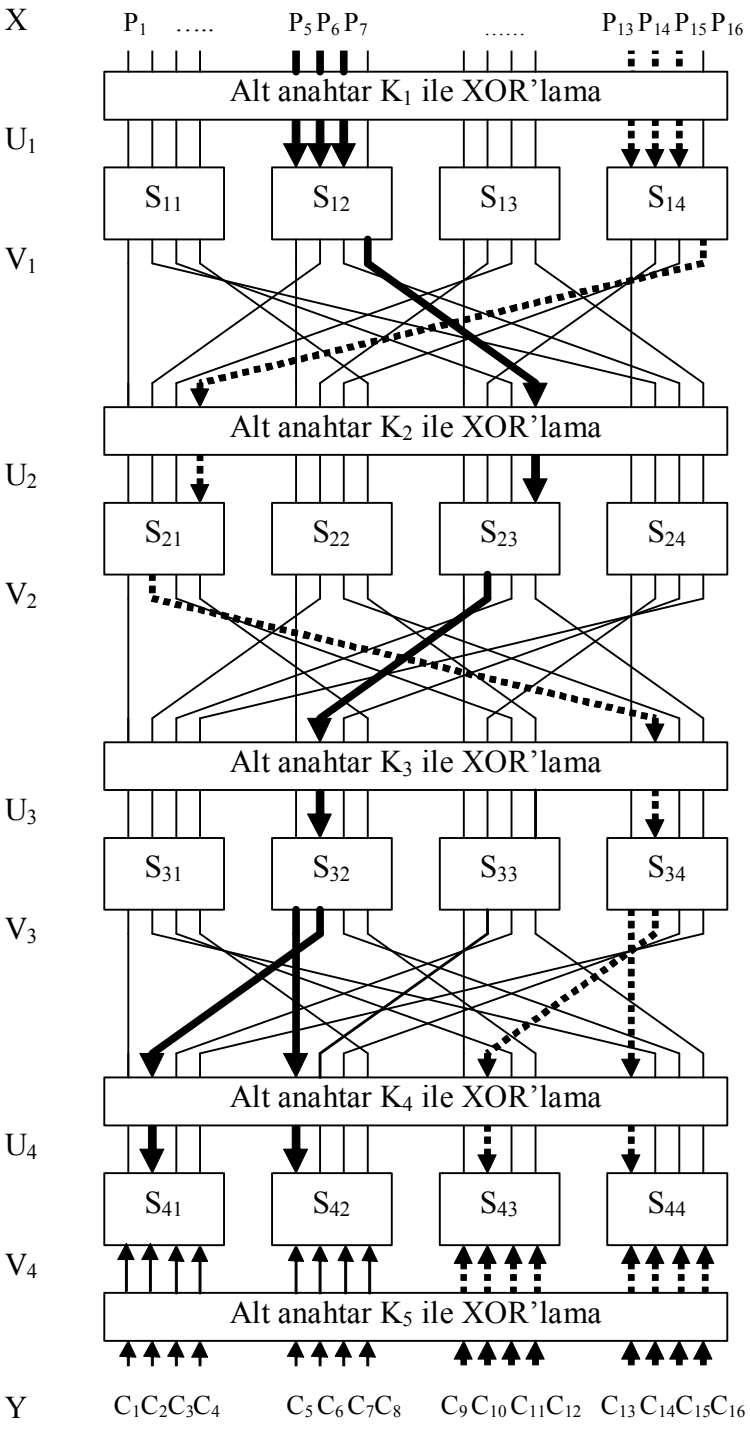
$\Sigma K = 0$ ise sapmamız $\varepsilon_T = 1/16$, $\Sigma K = 1$ ise sapmamız $\varepsilon_T = -1/16$ olacaktır.

Kalan 8 bit anahtarın bulunmasında kullanılan algoritma şu şekildedir:

1. Şifreli metnin son 8 biti alınır ve K_5 alt anahtarının olası bütün $K_{5,9}$, $K_{5,10}$, $K_{5,11}$, $K_{5,12}$, $K_{5,13}$, $K_{5,14}$, $K_{5,15}$, $K_{5,16}$ değerleri ile (Toplam 16*16=256 olası değer) ile XOR'lanır.
2. 8 bitlik değer 4 bit, 4 bit olarak (S_{43} ve S_{44} S kutularına) sırasıyla ters s kutularına sokulur.
3. Ters S_{43} kutusunun $U_{4,10}$ değeri ile S_{44} kutusunun $U_{4,13}$ değeri kaydedilir.
4. Açık metnin P_{13}, P_{14}, P_{15} değerleri ile $U_{4,10}$ ve $U_{4,13}$ değerleri XOR'lanarak sonucun 0 verdiği anahtar değerlerinde, o anahtar değeri için sayaç 1 arttırılır.

Bu aşamalardan sonra kısmi deşifreleme işlemi yapmamız gerekmektedir. Alt Anahtarın ilk 8 bitini bulmada 2600 adet açık/ şifreli metin çifti kullanmıştık. Kalan 8 bitini bulma işleminde de 2600 adet açık/ şifreli metin çifti kullanalım. Algoritmadaki anahtar bitlerinin $K_{5,9}$, $K_{5,10}$, $K_{5,11}$, $K_{5,12}$, $K_{5,13}$, $K_{5,14}$, $K_{5,15}$, $K_{5,16}$ bulunabilmesi için her açık metin/şifreli metin çiftinin işlenmesi gerekmektedir. Bu işlemde Şekil 11.13'teki yöntem kullanılmıştır. Anahtar bitlerinin bulunabilmesi için tüm olası anahtar değerleri denenerek, çalışmamızda 8 bit anahtar arandığından bu 256 olası değer demek, $P_{13} \oplus$

$P_{14} \oplus P_{15} \oplus U_{4,13} \oplus U_{4,10} = 0$ ifadesi bir anahtar değeri tuttuğunda sayacı, o anahtar değeri için bir attırdık ve bunu tüm açık metin/şifreli metin çiftleri için yaptık.

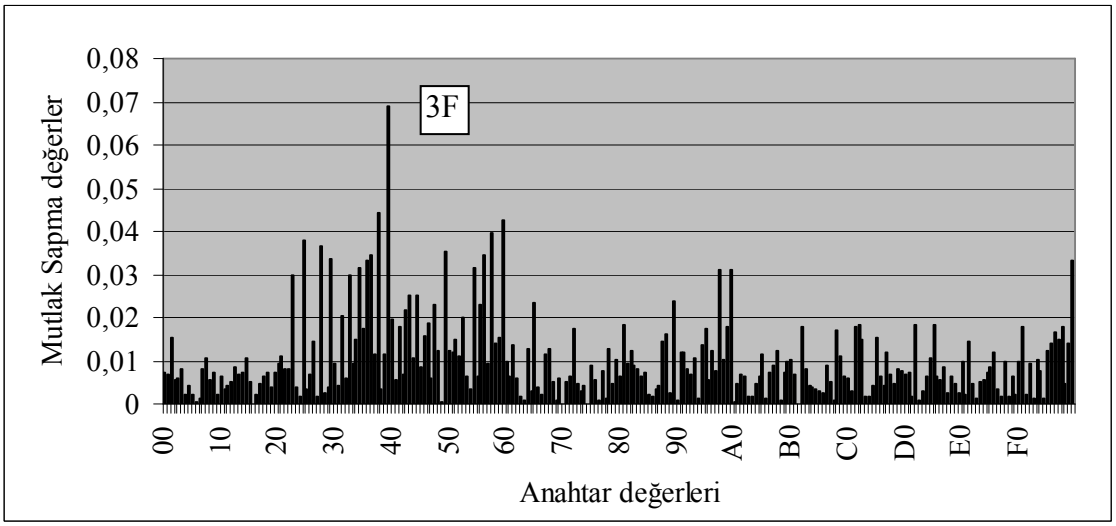


Şekil 11.13. 16 bit alt anahtarın elde edilmesi

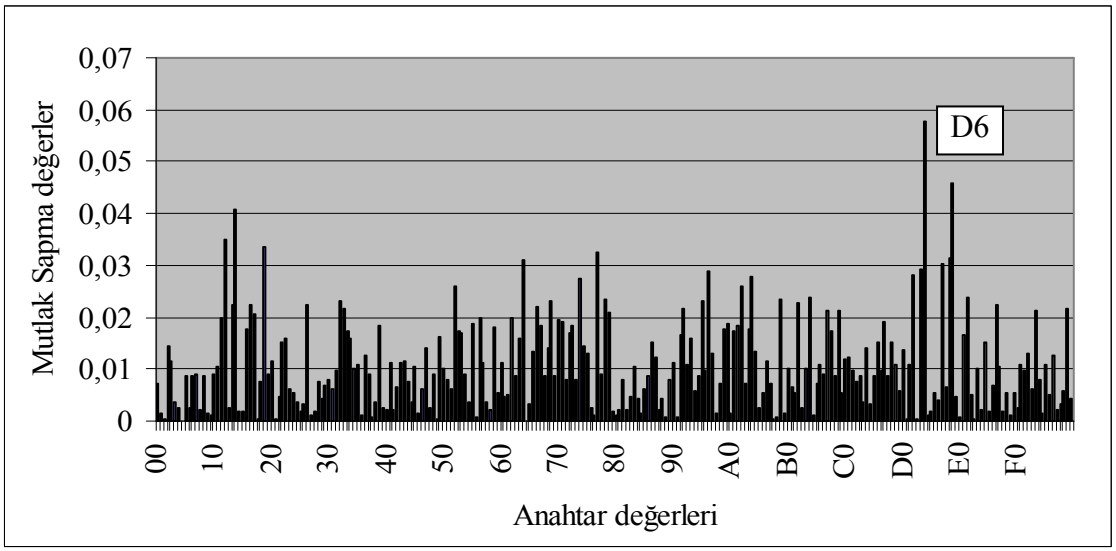
Sonuçta 1/2 değerinden en büyük mutlak sapmaya uğrayan anahtar uygulamada kullanılan $K_{5,9}$, $K_{5,10}$, $K_{5,11}$, $K_{5,12,}$ $K_{5,13}$, $K_{5,14}$, $K_{5,15}$, $K_{5,16}$ bitlerini bize verecektir. Şekil 11.16'da da görüleceği gibi rastgele seçilen 2600 adet açık/şifreli metin çifti üzerinde çalıştırılan programın sonucunda, kullanılan K_5 ("D63F") alt anahtarı 8 bitlik 2 parça olarak ("D6" ve "3F") elde edilmiştir.

Burada D6 anahtarı $K_{5,1}K_{5,2}K_{5,3}K_{5,4}K_{5,5}K_{5,6}K_{5,7}K_{5,8}$ 8 bitlik ilk yarının, 3F anahtarıda $K_{5,9}K_{5,10}K_{5,11}K_{5,12}K_{5,13}K_{5,14}K_{5,15}K_{5,16}$ 8 bitlik 2. yarının sonuçlarıdır.

Aşağıdaki şekillerde (Şekil 11.14 ve Şekil 11.15'te) 16 bit anahtarın (D63F) bulunmasında 2600 adet açık/şifreli metin çifti için, elde edilen deneysel sonuçlar grafiksel olarak gösterilmiştir.



Şekil 11.14. 2600 adet açık/şifreli metin çifti için Mutlak Sapma değerleri (3F için)

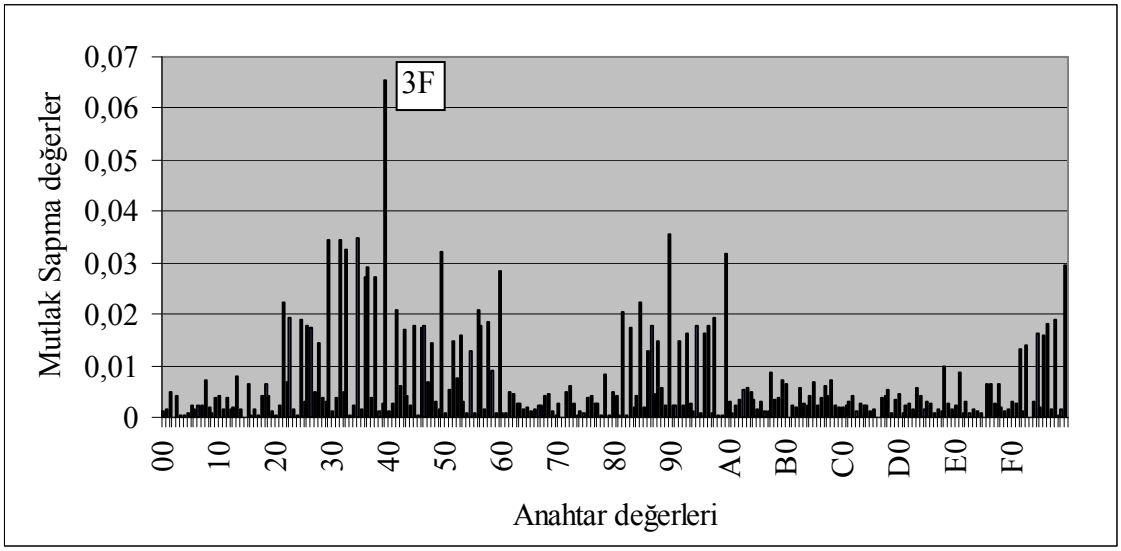


Şekil 11.15. 2600 adet açık/şifreli metin çifti için Mutlak Sapma değerleri (D6 için)

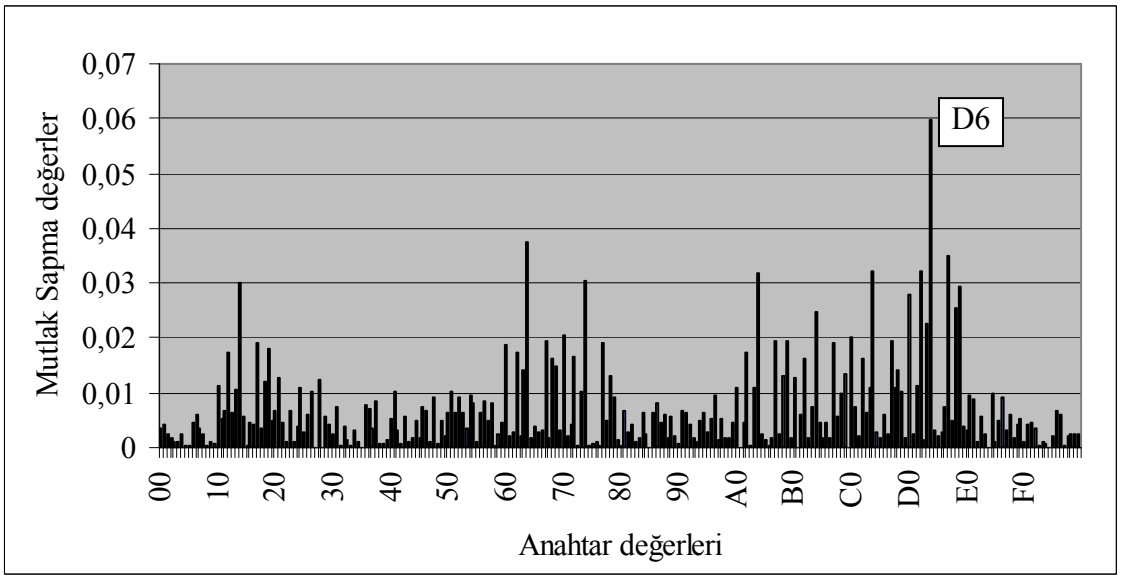
Tablo 11.5. 16 bit anahtarın 8 bitini (3F) bulmada, seçilen 2600 adet açık/şifreli metin çifti için deneysel sonuçlar

Anahtar	Mutlak Sapma	Anahtar	Mutlak Sapma	Anahtar	Mutlak Sapma	Anahtar	Mutlak Sapma	Anahtar	Mutlak Sapma	Anahtar	Mutlak Sapma	Anahtar	Mutlak Sapma	Anahtar	Mutlak Sapma
00	0,0073	20	0,0092	40	0,0196	60	0,01	80	0,0065	A0	0,0004	C0	0,0058	E0	0,0096
01	0,0069	21	0,0112	41	0,0054	61	0,0065	81	0,0185	A1	0,0046	C1	0,0031	E1	0,0023
02	0,0154	22	0,0081	42	0,0177	62	0,0135	82	0,0092	A2	0,0069	C2	0,0177	E2	0,0146
03	0,0054	23	0,0081	43	0,0069	63	0,0058	83	0,0123	A3	0,0062	C3	0,0185	E3	0,0046
04	0,0058	24	0,03	44	0,0219	64	0,0015	84	0,0088	A4	0,0019	C4	0,015	E4	0,0012
05	0,0081	25	0,0038	45	0,025	65	0,0008	85	0,0081	A5	0,0019	C5	0,0019	E5	0,005
06	0,0023	26	0,0019	46	0,0108	66	0,0127	86	0,0062	A6	0,0046	C6	0,0015	E6	0,0054
07	0,0042	27	0,0377	47	0,025	67	0,0031	87	0,0073	A7	0,0065	C7	0,0042	E7	0,0073
08	0,0023	28	0,0035	48	0,0085	68	0,0235	88	0,0023	A8	0,0115	C8	0,0154	E8	0,0085
09	0,0004	29	0,0069	49	0,0158	69	0,0038	89	0,0019	A9	0,0012	C9	0,0065	E9	0,0119
0A	0,0012	2A	0,0146	4A	0,0188	6A	0,0023	8A	0,0035	AA	0,0073	CA	0,0042	EA	0,0035
0B	0,0081	2B	0,0015	4B	0,0058	6B	0,0115	8B	0,0042	AB	0,0088	CB	0,0119	EB	0,0019
0C	0,0108	2C	0,0365	4C	0,0231	6C	0,0127	8C	0,0146	AC	0,0123	CC	0,0069	EC	0,01
0D	0,0054	2D	0,0027	4D	0,0123	6D	0,005	8D	0,0162	AD	0,0008	CD	0,0046	ED	0,0015
0E	0,0073	2E	0,0038	4E	0,0004	6E	0,0008	8E	0,0027	AE	0,0073	CE	0,0081	EE	0,0065
0F	0,0023	2F	0,0335	4F	0,0354	6F	0,0058	8F	0,0238	AF	0,01	CF	0,0077	EF	0,0023
10	0,0062	30	0,0092	50	0,0123	70	0	90	0,0008	B0	0,0104	D0	0,0069	F0	0,0096
11	0,0035	31	0,0042	51	0,0119	71	0,005	91	0,0119	B1	0,0069	D1	0,0073	F1	0,0177
12	0,0042	32	0,0204	52	0,015	72	0,0065	92	0,0119	B2	0	D2	0,0019	F2	0,0023
13	0,005	33	0,0058	53	0,0112	73	0,0173	93	0,0081	B3	0,0177	D3	0,0181	F3	0,0092
14	0,0085	34	0,03	54	0,02	74	0,0046	94	0,0069	B4	0,0081	D4	0,0008	F4	0,0012
15	0,0069	35	0,0092	55	0,0062	75	0,0031	95	0,0108	B5	0,0042	D5	0,0031	F5	0,0104
16	0,0073	36	0,015	56	0,0035	76	0,0042	96	0,0012	B6	0,0038	D6	0,0065	F6	0,0077
17	0,0108	37	0,0315	57	0,0315	77	0	97	0,0138	B7	0,0035	D7	0,0108	F7	0,0012
18	0,005	38	0,0173	58	0,0065	78	0,0088	98	0,0173	B8	0,0031	D8	0,0181	F8	0,0123
19	0	39	0,0331	59	0,0231	79	0,0054	99	0,0054	B9	0,0027	D9	0,0062	F9	0,0142
1A	0,0023	3A	0,0346	5A	0,0346	7A	0,0008	9A	0,0123	BA	0,0088	DA	0,0054	FA	0,0165
1B	0,0046	3B	0,0115	5B	0,0092	7B	0,0077	9B	0,0077	BB	0,005	DB	0,0085	FB	0,015
1C	0,0065	3C	0,0442	5C	0,0396	7C	0,0012	9C	0,0312	BC	0,0008	DC	0,0027	FC	0,0177
1D	0,0073	3D	0,0035	5D	0,0142	7D	0,0127	9D	0,0104	BD	0,0169	DD	0,0065	FD	0,0046
1E	0,0038	3E	0,0115	5E	0,0154	7E	0,0046	9E	0,0177	BE	0,0112	DE	0,0046	FE	0,0142
1F	0,0073	**3F**	**0,0688**	5F	0,0427	7F	0,0104	9F	0,0312	BF	0,0062	DF	0,0027	FF	0,0331

Son olarak, tüm açık/şifreli metin çiftleri için ($2^{16} = 65536$ adet) Mutlak Sapma değerlerini bulalım ve grafiksel olarak ifade edelim.

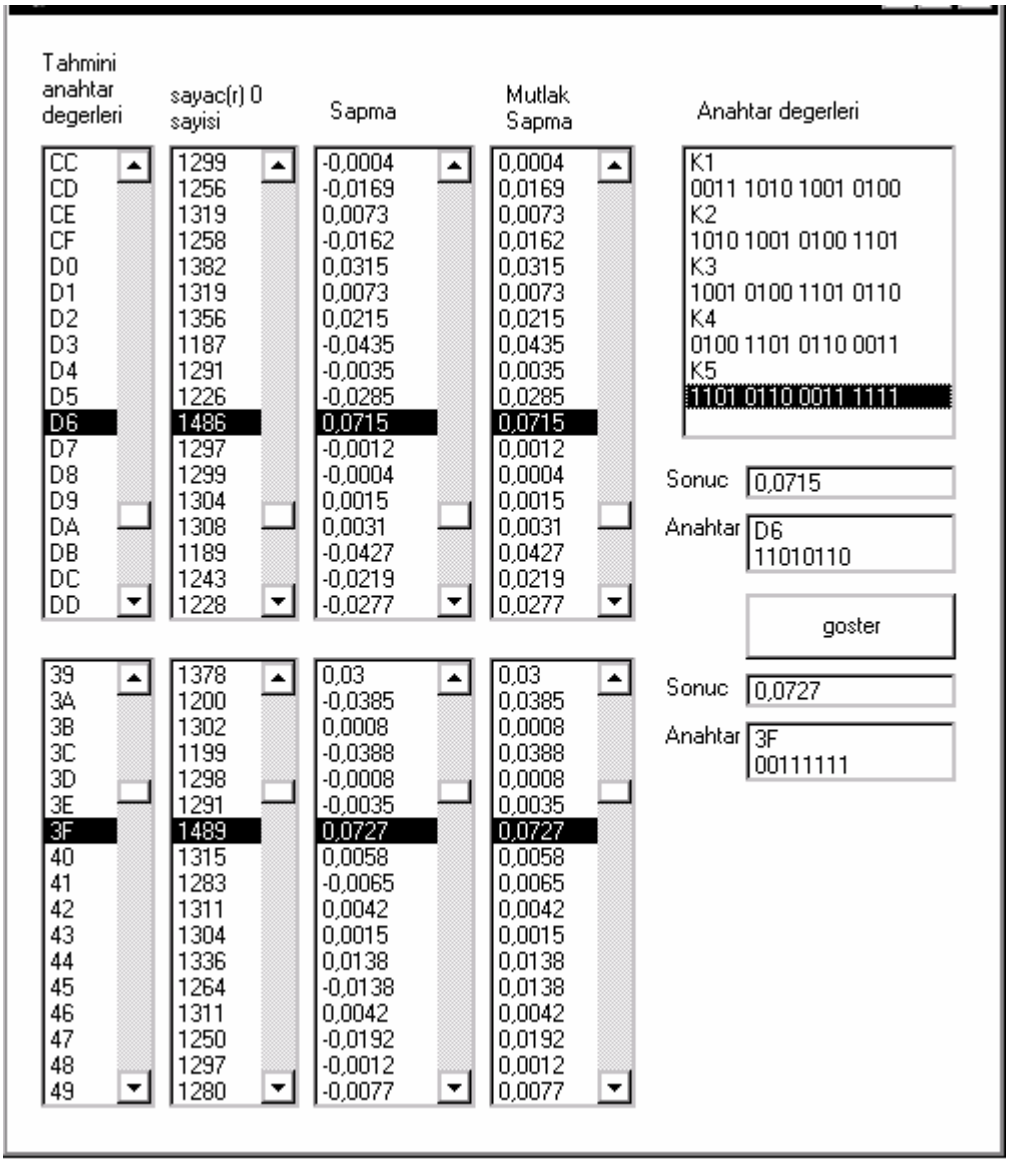


Şekil 11.16. Tüm açık/şifreli metin çiftleri için Mutlak Sapma değerleri (3F anahtarı için)



Şekil 11.17. Tüm açık/şifreli metin çiftleri için Mutlak Sapma değerleri (D6 anahtarı için)

Sonuçta grafiklerdende anlaşılacağı gibi denenen açık/şifreli metin sayımız, yani örnek uzayımız arttıkça asıl anahtarımız daha belirgin bir farkla ortaya çıkmaktadır.



Şekil 11.18. 16 bit anahtarı (D6, 8 bitlik ilk yarının $K_{5,1}K_{5,2}K_{5,3}K_{5,4}K_{5,5}K_{5,6}K_{5,7}K_{5,8}$, 3F'de 8 bitlik 2.yarının $K_{5,9}K_{5,10}K_{5,11}K_{5,12}K_{5,13}K_{5,14}K_{5,15}K_{5,16}$ anahtarlarıdır) elde etmek için rastgele seçilen 2600 adet açık/şifreli metin çifti için program çıktısı örneği

12. SONUÇLAR

Çalışmamızda dört döngülük bir SPN algoritması için doğrusal (lineer) kriptanaliz uygulaması gerçekleştirdik ve son döngüde kullanılan 16 bit anahtarın ilk önce, ilk 8 bitini (D6) sonra da kalan 8 biti (3F) olmak üzere tamamını başarılı bir şekilde hex notasyonda "D63F" olarak elde ettik. Burada "D6" anahtarı $K_{5,1}K_{5,2}K_{5,3}K_{5,4}\,K_{5,5}K_{5,6}K_{5,7}K_{5,8}$ 8 bitlik ilk yarının, "3F" anahtarıda $K_{5,9}K_{5,10}K_{5,11}K_{5,12}\,K_{5,13}K_{5,14}K_{5,15}K_{5,16}$ 8 bitlik 2. yarının sonuçlarıdır.

Diğer yüksek mutlak sapma değerlerine sahip anahtar değerlerinin sebebini S kutusu özellikleri ve rastgele olarak seçilen açık/şifreli metin çiftlerinden kaynaklı olarak açıklayabiliriz. Ayrıca tüm açık metin ve şifreli metin çiftleri için yapılan deneme sonucunun doğrusal ifadeleri birleştirme sonucu elde edilen $\frac{1}{16} = 0{,}0625$ değerinden bir miktar sapma göstermesi bu doğrusal ifadelerin birbirinden tam olarak bağımsız olmadığını da göstermektedir. Yine de bu farklılık saldırı sonucu anahtarı elde etmemize engel olmamaktadır. Burada kullanılan SPN algoritmasında yer alan S kutusu sonlu alan aritmetiği kullanılarak tasarlanmıştır. Buna ek olarak permütasyon safhası için AES şifreleme algoritmasında kullanılan ShiftRows dönüşümü bit permütasyonu için kullanılmıştır. AES'te SPN mimarisine ek olarak kullanılan mixcolums lineer dönüşümü, lineer kriptanaliz için az sayıda aktif S kutusu içeren lineer yaklaşımları ve diferansiyel kriptanaliz için az sayıda aktif S kutusu içeren diferansiyel nitelikleri bulmayı oldukça zorlaştırır. Bu da blok şifreleme yöntemlerinde istenilen bir durumdur.

Sonuçta, 1300, 2600, 3900 ve 5200 adet açık/şifreli metin çiftlerinin deneysel sonuçlarının anlaşılırlığı için çizilen grafiklerden de anlaşılacağı gibi, denenen açık/şifreli metin sayımız, yani örnek uzayımız arttıkça diğer yüksek mutlak sapma değerlerine sahip anahtar değerlerinde azalmalar olmakta; ve asıl anahtarımız daha belirgin bir farkla ortaya çıkmaktadır.

1993 yılında, Matsui (Matsui, 1993) tarafından DES algoritmasına kriptanalitik bir saldırı tipi olarak keşfedilmiş olan Lineer kriptanaliz Modern şifreleme algoritmalarının tasarımında dikkate alınması gereken önemli bir unsurdur.

KAYNAKLAR

FIPS 197, November 26, 2001 Advanced Encryption Standard, Federal Information Processing Standard (FIPS), Publication 197, National Bureau of Standards, U.S. Department of Commerce, Washington D.C.

SÖNMEZ Remziye, 2002, "Veri Şifreleme Standardı (DES) ve Rivest Shamir Adleman (RSA) Güvenlik Algoritmalarının VLSI Tasarımı", Yüksek Lisans Tezi, Hacettepe Üniversitesi.

FIPS 46-3, October 25, 1999, Data Encryption Standard, Federal Information Processing Standart (FIPS), Publication 46-3, National Bureau of Standards, U.S. Department of Commerce, Washington D.C.

BİRİNCİ Fatih, 2002, "Güvenlik Protokollerinin Tasarımı ve Analizi", Yüksek Lisans Tezi, Gebze Yüksek Teknoloji Enstitüsü.

GÜNCAN Murat, 2002, "Kimlik Tabanlı Kriptosistemler ile Güvenli Veri Aktarımı", Yüksek Lisans Tezi, İTÜ.

SAKALLI M.T., BULUŞ E., ŞAHİN A., BÜYÜKSARAÇOĞLU F., 2005, "AES S kutusuna benzer 4 bit girişe ve 4 bit çıkışa sahip S kutularının tasarımı", II. Mühendislik Bilimleri Genç Araştırmacılar Kongresi-MBGAK'2005, İstanbul.

UYGULAMALI MATEMATİK ENSTİTÜSÜ, 2004, "Kriptolojiye Giriş Ders Notları", Kriptografi Bölümü, ODTÜ.

SAKALLI M.T.,BULUŞ E., BÜYÜKSARAÇOĞLU F., 2004, "İki Döngülü bir blok Şifreleme Algoritmasının Lineer Kriptanaliz Uygulaması", ELECO-2004, Bursa.

MATSUI M., 1993, Linear Cryptanalysis Method for DES Cipher. Advances in Cryptology –EUROCRYPT'93, 386-397.

BIHAM E., SHAMIR A., 1993, Differential Cryptanalysis of the full 16-round DES, Advances in Cryptology: Proceedings of CRYPTO'92, Springer-Verlag, Berlin , pp 487-496.

ANDİÇ Ersin, 2002, "Bilgisayar Haberleşmesinde Şifreleme (Kripto) Yazılımıyla Güvenliğin Sağlanması", Yüksek Lisans Tezi, Marmara Üniversitesi.

HEYS H., 2002, A Tutorial on Linear and Differential Cryptanalysis., CRYPTOLOGIA, Vol 26, No 3 pp 189-221.

MATSUI M.,1993, Linear Cryptanalysis Method for DES Cipher. Advances in Cryptology –EUROCRYPT'93, 386-397.

ŞAHİN A., BULUŞ E., SAKALLI M.T., 2005, "Modern Blok Şifreleme Algoritmalarının Gücünün İncelenmesi", II. Mühendislik Bilimleri Genç Araştırmacılar Kongresi-MBGAK'2005, İstanbul.

KNUDSEN L.R., 1995, "Truncated and Higher Order Differentials, Fast Software Encryption", Springer-Verlag, pp.196-211.

RAPHAEL C., PHAN W., 2004, "Impossible Differential Cryptanalysis of 7-round Advanced Encryption Standart (AES)", Information Processing Letter, ELSEVIER, April 9.

KNUDSEN L.R., ROBSHAW M.J.B., 1996, "Nonlinear Approximations in Linear Cryptanalysis", Advances in Cryptology-CRYPTO'96, Springer-Verlag, pp. 224-236.

ROBSHAW M.J.B., KALISKI B.S., 1994, "Linear Cyptanalysis Using Multiple Approximations", Advances in Cryptology-CRYPTO'94, Springer-Verlag, pp. 1-11.

SAKALLI M.T., BULUŞ E., ŞAHİN A., BÜYÜKSARAÇOĞLU F., 2005, "Bir Blok Şifreleme Algoritmasına karşı Square saldırısı", ELECO'2005, Busa.

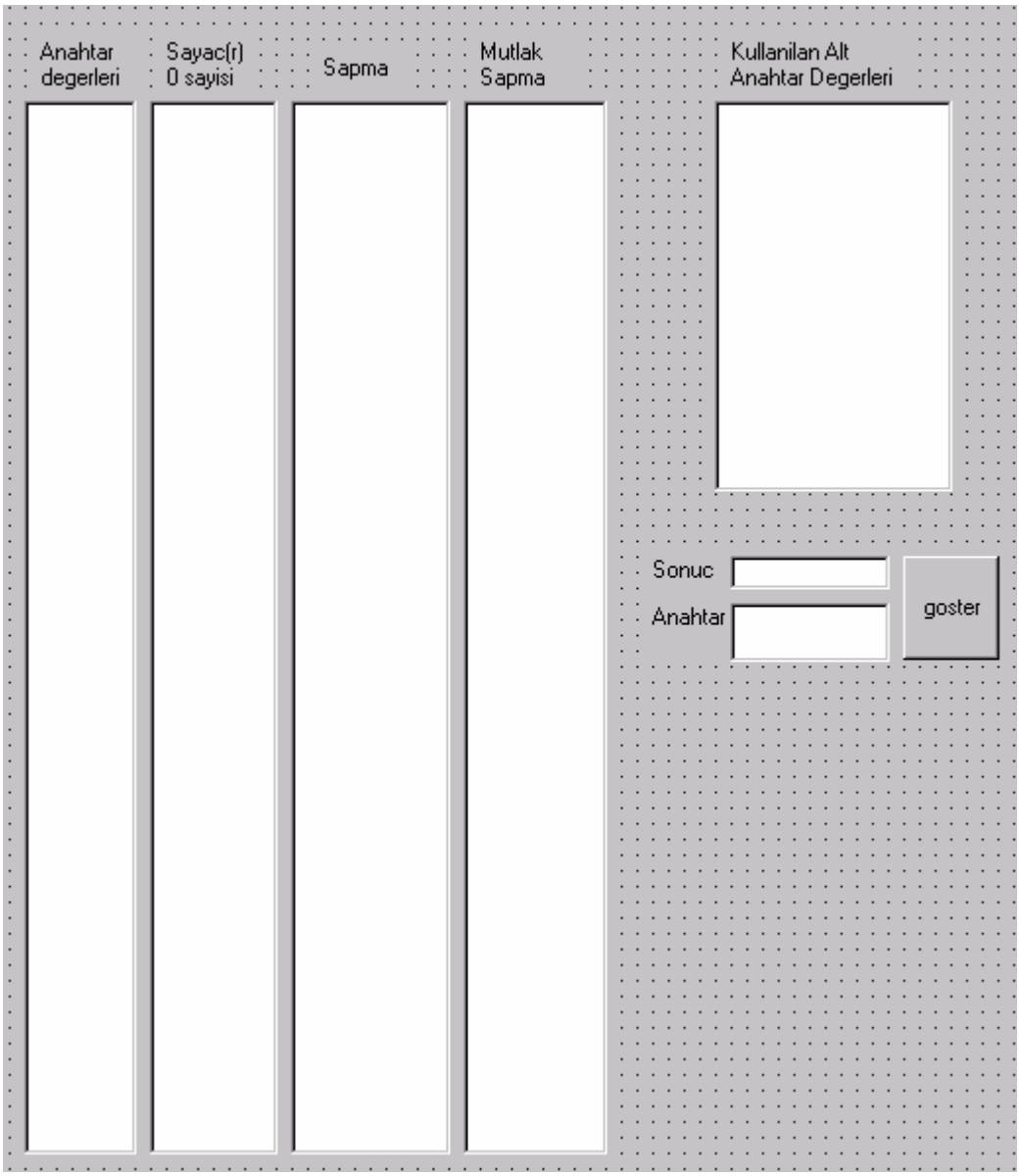
DAEMEN J., KNUDSEN L.R., RIJMEN V., 1997, "The Block cipher Square", Proceedings of Fast Software Encryption, Springer-Verlag, pp. 149-165, New York.

ÖZGEÇMİŞ

Şenol ŞEN 1979 yılında Kırcaali'de doğdu. İlk ve Orta öğrenimini İstanbul'da tamamladıktan sonra 1998 yılında girdiği Trakya Üniversitesi Mühendislik-Mimarlık Fakültesi Bilgisayar Mühendisliği Bölümünden 2002 yılında mezun oldu. 2002 yılında Trakya Üniversitesi Mühendislik Mimarlık Fakültesi Bilgisayar Mühendisliği Bölümünde Araştırma Görevlisi kadrosuna atandı ve aynı yıl Trakya Üniversitesi Fen Bilimleri Enstitüsüne bağlı olarak Yüksek Lisans eğitimine başladı. Şenol ŞEN halen Trakya Üniversitesi Mühendislik Mimarlık Fakültesi Bilgisayar Mühendisliği Bölümünde Araştırma Görevlisi olarak çalışmaktadır.

EK – 1 (Program Kodları)

8 bit Alt Anahtarı Hesaplamada Kullanılan programın kaynak kodları



Formun Kodu

```
Option Explicit
Dim dizz(5) As String
Dim plainhex As String
Dim cipherhex As String
```

```
Dim keyhex As String
Dim tmp As String
Dim sayac2, toplam, say2toplam As Double
Dim a, b, c, d, i As Integer
Dim gecici, maksimum As Currency

Public Sub make_keys(dizi As String)
Dim i
For i = 0 To 4
  keys(i) = Mid(dizi, ((i * 4) + 1), 16)
  List6.AddItem "K" & i + 1
  List6.AddItem text_div(keys(i))
  List6.AddItem " "
Next
End Sub

Private Sub Command1_Click()
List2.SetFocus
List2.ListIndex = maksimum
End Sub

Private Sub Form_Load()
Dim i, k, sonuc, sayici As Integer
Dim plainbin1 As String
Dim cipherbin1, m, v As String
Dim p, r As Integer
Dim f(256), son(256) As Currency

Randomize Timer

uretici(0) = "0"
uretici(1) = "1"
```

uretici(2) = "2"
uretici(3) = "3"
uretici(4) = "4"
uretici(5) = "5"
uretici(6) = "6"
uretici(7) = "7"
uretici(8) = "8"
uretici(9) = "9"
uretici(10) = "A"
uretici(11) = "B"
uretici(12) = "C"
uretici(13) = "D"
uretici(14) = "E"
uretici(15) = "F"

sbox(0) = "3"
sbox(1) = "D"
sbox(2) = "A"
sbox(3) = "2"
sbox(4) = "1"
sbox(5) = "7"
sbox(6) = "B"
sbox(7) = "5"
sbox(8) = "C"
sbox(9) = "E"
sbox(10) = "F"
sbox(11) = "6"
sbox(12) = "9"
sbox(13) = "8"
sbox(14) = "0"
sbox(15) = "4"

tsbox(0) = "E"
tsbox(1) = "4"
tsbox(2) = "3"
tsbox(3) = "0"
tsbox(4) = "F"
tsbox(5) = "7"
tsbox(6) = "B"
tsbox(7) = "5"
tsbox(8) = "D"
tsbox(9) = "C"
tsbox(10) = "2"
tsbox(11) = "6"
tsbox(12) = "8"
tsbox(13) = "1"
tsbox(14) = "9"
tsbox(15) = "A"

per(0) = "1"
per(1) = "14"
per(2) = "11"
per(3) = "8"
per(4) = "5"
per(5) = "2"
per(6) = "15"
per(7) = "12"
per(8) = "9"
per(9) = "6"
per(10) = "3"
per(11) = "16"
per(12) = "13"
per(13) = "10"
per(14) = "7"

```
per(15) = "4"

tahmini_anah(0) = "0000"
tahmini_anah(1) = "0001"
tahmini_anah(2) = "0010"
tahmini_anah(3) = "0011"
tahmini_anah(4) = "0100"
tahmini_anah(5) = "0101"
tahmini_anah(6) = "0110"
tahmini_anah(7) = "0111"
tahmini_anah(8) = "1000"
tahmini_anah(9) = "1001"
tahmini_anah(10) = "1010"
tahmini_anah(11) = "1011"
tahmini_anah(12) = "1100"
tahmini_anah(13) = "1101"
tahmini_anah(14) = "1110"
tahmini_anah(15) = "1111"

keyhex = "3A94D63F"                'sifreleme anahtarimiz
keybin = hextobin(keyhex, 7)
make_keys (keybin)                 'alt anahtarlar üretiliyor(K0,K1,K2,K3,K4)

i = 0
For a = 0 To 15                    'tahmini 2 bit(2A gibi) anahtarlar olusturuluyor
   For b = 0 To 15
      tahmini_anahtar(i) = CStr(uretici(a)) + CStr(uretici(b))
      i = i + 1
   Next
Next

For k = 0 To 2599
```

```
   plainhex = tum_veri(k)
   plainhex = metin_uret
   veriler(k + 1, 1) = plainhex
   plainbin = hextobin(plainhex, 3)
   tmp = plainbin

      For i = 0 To 3                  '4 roundluk spn algoritmasi calistiriliyor
         If i < 3 Then
            dizz(i) = round(tmp, keys(i))
            tmp = dizz(i)
         Else
            dizz(i) = Last_round(tmp, keys(i), keys(i + 1))
            tmp = dizz(i)
         End If
      Next

   cipherbin = tmp
   cipherhex = bintohex(cipherbin, 16)
   veriler(k + 1, 2) = cipherhex
Next
'*****************************************************************
'veriler(1,1) 1. verinin anametni
'veriler(1,2) 1. verinin sifrelimetnini icerir
'*****************************************************************
For r = 0 To 255       ' 256 adet tahmini anahtar degerlerimiz için
   For p = 0 To 2599  ' verilerimizin degiskeni icin 2600 adet deger için
            plainhex = veriler(p + 1, 1)
            plainbin = hextobin(plainhex, 3)
            cipherhex = veriler(p + 1, 2)
            cipherbin = hextobin(cipherhex, 3)
            sonuc = Int(hesapla(plainbin, cipherbin, r))
```

```
            If sonuc = 0 Then
                    sayac(r) = sayac(r) + 1
            End If
        Next
    List1.AddItem sayac(r)
Next

'Open "c:\veriler_65536.txt" For Append As 1  'veriler dosyaya yazdiriliyor

For i = 0 To 255
    son(i) = (sayac(i) - 1300) / 2600
    List4.AddItem son(i)
    f(i) = Abs(son(i))
    If gecici < f(i) Then
        gecici = f(i)
          maksimum = i
    End If
    'List3.AddItem son(i)

'    Print #1, f(i), tahmini_anahtar(i) 'Mutlak Sapma degerlerini dosyaya yazdir

    List3.AddItem f(i)
    List2.AddItem tahmini_anahtar(i)
Next

'Close #1        'dosyayi kapa
List7.AddItem gecici
List5.AddItem tahmini_anahtar(Int(maksimum))
List5.AddItem hextobin(tahmini_anahtar(Int(maksimum)), 1)
End Sub

Private Sub List1_Click()
```

```
List2.ListIndex = List1.ListIndex
End Sub

Private Sub List2_Click()
List1.ListIndex = List2.ListIndex
List3.ListIndex = List2.ListIndex
List4.ListIndex = List2.ListIndex
End Sub

Private Sub List3_Click()
List2.ListIndex = List3.ListIndex
End Sub

Private Sub List4_Click()
List2.ListIndex = List4.ListIndex
End Sub
```

Modülün kodu

```
Option Explicit
Public keys(5) As String
Public sbox(15) As String
Public tsbox(15) As String
Public per(15) As String
Public uretici(15) As String
Public tahmini_anah(15) As String
Public sayac(256) As Double
Public plainbin As String
Public cipherbin As String
Public keybin As String
Public m As String
Public s1 As String
Public s2 As String
```

```
Public veriler(65536, 2) As String
Public tum_veri(65536) As String
Public tahmini_anahtar(256) As String

Public Function text_div(a As String) As String
    text_div = Mid(a, 1, 4) + " " + Mid(a, 5, 4) + " " + Mid(a, 9, 4) + " " + Mid(a, 13, 4)
End Function

Public Function bintohex(str As String, lng As Integer) As String
Dim i, k As Integer
Dim nw_str As String
k = (lng / 4) - 1

For i = 0 To k
Select Case Mid(str, ((i * 4) + 1), 4)
Case "0000":   nw_str = nw_str + "0"
Case "0001":   nw_str = nw_str + "1"
Case "0010":   nw_str = nw_str + "2"
Case "0011":   nw_str = nw_str + "3"
Case "0100":   nw_str = nw_str + "4"
Case "0101":   nw_str = nw_str + "5"
Case "0110":   nw_str = nw_str + "6"
Case "0111":   nw_str = nw_str + "7"
Case "1000":   nw_str = nw_str + "8"
Case "1001":   nw_str = nw_str + "9"
Case "1010":   nw_str = nw_str + "A"
Case "1011":   nw_str = nw_str + "B"
Case "1100":   nw_str = nw_str + "C"
Case "1101":   nw_str = nw_str + "D"
Case "1110":   nw_str = nw_str + "E"
Case "1111":   nw_str = nw_str + "F"
End Select
```

```
Next
bintohex = nw_str
End Function

Public Function hextobin(str As String, k As Integer) As String
Dim j As Integer
Dim bin As String

For j = 0 To k
Select Case Mid(str, (j + 1), 1)
Case 0: bin = bin + "0000"
Case 1: bin = bin + "0001"
Case 2: bin = bin + "0010"
Case 3: bin = bin + "0011"
Case 4: bin = bin + "0100"
Case 5: bin = bin + "0101"
Case 6: bin = bin + "0110"
Case 7: bin = bin + "0111"
Case 8: bin = bin + "1000"
Case 9: bin = bin + "1001"
Case "A": bin = bin + "1010"
Case "B": bin = bin + "1011"
Case "C": bin = bin + "1100"
Case "D": bin = bin + "1101"
Case "E": bin = bin + "1110"
Case "F": bin = bin + "1111"
End Select
Next
hextobin = bin
End Function

Public Function XOR_Sub(txt As String, k As String) As String
```

```
Dim temp(15) As String
Dim yazi As String
Dim i As Integer
For i = 0 To 15
   temp(i) = Mid(txt, (i + 1), 1) Xor Mid(k, (i + 1), 1)
   yazi = yazi + temp(i)
Next
XOR_Sub = yazi
End Function

Public Function Substitution(str As String) As String
Dim nw_str As String
Dim i As Integer
For i = 0 To 3
Select Case Mid(str, ((i * 4) + 1), 4)
Case "0000":   nw_str = nw_str + hextobin(sbox(0), 0)
Case "0001":   nw_str = nw_str + hextobin(sbox(1), 0)
Case "0010":   nw_str = nw_str + hextobin(sbox(2), 0)
Case "0011":   nw_str = nw_str + hextobin(sbox(3), 0)
Case "0100":   nw_str = nw_str + hextobin(sbox(4), 0)
Case "0101":   nw_str = nw_str + hextobin(sbox(5), 0)
Case "0110":   nw_str = nw_str + hextobin(sbox(6), 0)
Case "0111":   nw_str = nw_str + hextobin(sbox(7), 0)
Case "1000":   nw_str = nw_str + hextobin(sbox(8), 0)
Case "1001":   nw_str = nw_str + hextobin(sbox(9), 0)
Case "1010":   nw_str = nw_str + hextobin(sbox(10), 0)
Case "1011":   nw_str = nw_str + hextobin(sbox(11), 0)
Case "1100":   nw_str = nw_str + hextobin(sbox(12), 0)
Case "1101":   nw_str = nw_str + hextobin(sbox(13), 0)
Case "1110":   nw_str = nw_str + hextobin(sbox(14), 0)
Case "1111":   nw_str = nw_str + hextobin(sbox(15), 0)
End Select
```

```
Next
Substitution = nw_str
End Function

Public Function permutation(str2 As String) As String
Dim newstr2 As String
Dim i As Integer
newstr2 = "0000000000000000"
For i = 0 To 15
Mid(newstr2, per(i), 1) = Mid(str2, i + 1, 1)
Next
permutation = newstr2
End Function

Public Function round(txt1 As String, k1 As String) As String
round = permutation(Substitution(XOR_Sub(txt1, k1)))
End Function

Public Function Last_round(txt2 As String, k2 As String, k3 As String) As String
Last_round = XOR_Sub(Substitution(XOR_Sub(txt2, k2)), k3)
End Function

Public Function metin_uret() As String
Dim gecici As String
Dim i As Integer

For i = 0 To 3
  gecici = gecici + uretici(Int(Rnd * 16))
Next

metin_uret = gecici
End Function
```

```
Public Function tSubstitution(str As String) As String
Dim nw_str As String

Select Case str
Case "0000":   nw_str = nw_str + hextobin(tsbox(0), 0)
Case "0001":   nw_str = nw_str + hextobin(tsbox(1), 0)
Case "0010":   nw_str = nw_str + hextobin(tsbox(2), 0)
Case "0011":   nw_str = nw_str + hextobin(tsbox(3), 0)
Case "0100":   nw_str = nw_str + hextobin(tsbox(4), 0)
Case "0101":   nw_str = nw_str + hextobin(tsbox(5), 0)
Case "0110":   nw_str = nw_str + hextobin(tsbox(6), 0)
Case "0111":   nw_str = nw_str + hextobin(tsbox(7), 0)
Case "1000":   nw_str = nw_str + hextobin(tsbox(8), 0)
Case "1001":   nw_str = nw_str + hextobin(tsbox(9), 0)
Case "1010":   nw_str = nw_str + hextobin(tsbox(10), 0)
Case "1011":   nw_str = nw_str + hextobin(tsbox(11), 0)
Case "1100":   nw_str = nw_str + hextobin(tsbox(12), 0)
Case "1101":   nw_str = nw_str + hextobin(tsbox(13), 0)
Case "1110":   nw_str = nw_str + hextobin(tsbox(14), 0)
Case "1111":   nw_str = nw_str + hextobin(tsbox(15), 0)
End Select
tSubstitution = nw_str
End Function

Public Function hesapla(str3 As String, str4 As String, y As Integer) As String
Dim i As Integer
Dim plainbin1 As String
Dim cipherbin1 As String
Dim mbin, v1, v2 As String
Dim xor_s As String
Dim v As String
```

```
plainbin1 = Mid(str3, 5, 1) Xor Mid(str3, 6, 1) Xor Mid(str3, 7, 1)
cipherbin1 = Mid(str4, 1, 8)

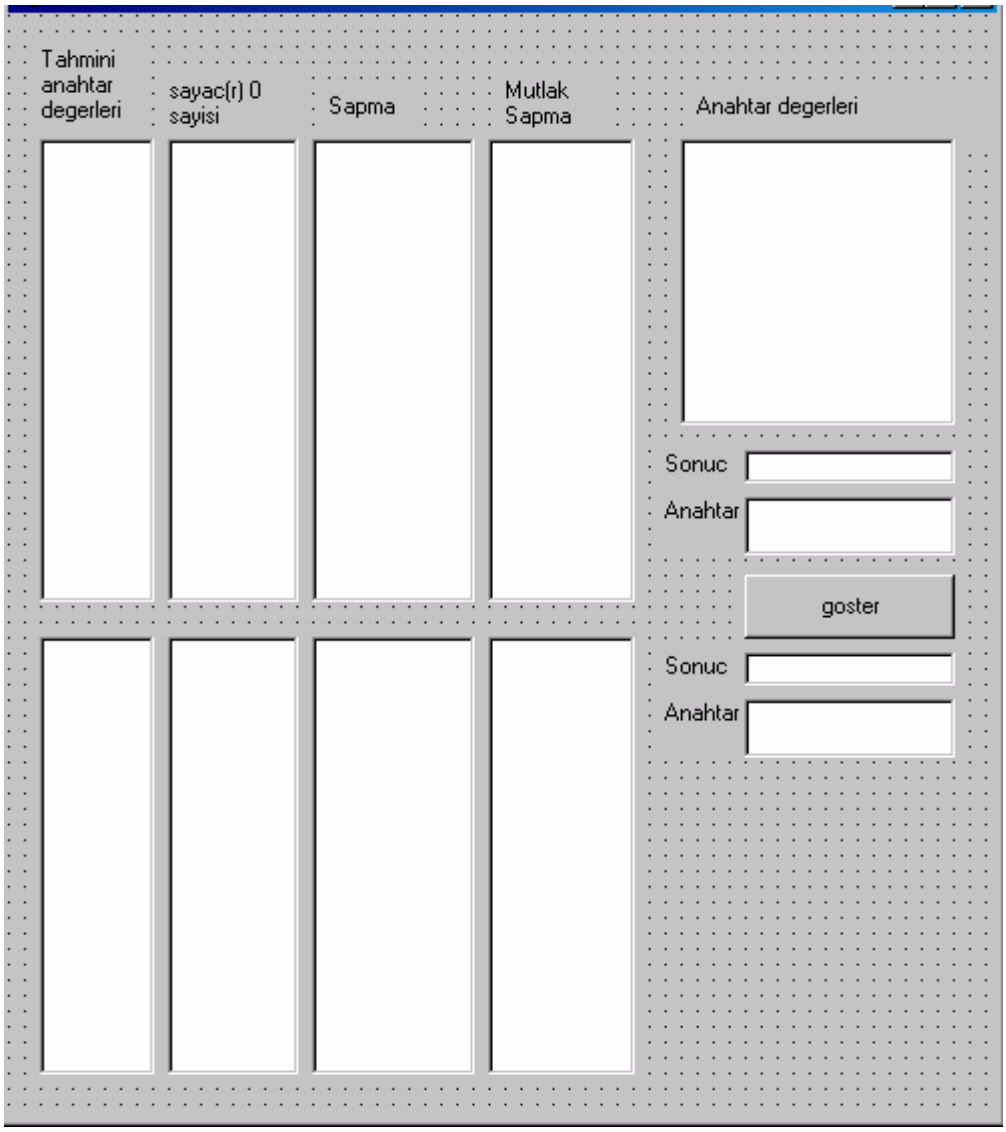
m = tahmini_anahtar(y) 'y. anahtarin degeri
mbin = hextobin(m, 1)

For i = 0 To 7
   v = Mid(cipherbin1, (i + 1), 1) Xor Mid(mbin, (i + 1), 1)
   xor_s = xor_s + v
Next

s1 = Mid(xor_s, 1, 4)
s2 = Mid(xor_s, 5, 4)
v1 = tSubstitution(s1)   ' ters s kutusu sonucu
v2 = tSubstitution(s2)    ' ters s kutusu sonucu

hesapla = Mid(v1, 2, 1) Xor Mid(v2, 1, 1) Xor plainbin1
End Function
```

16 bit Alt Anahtarı Hesaplamada Kullanılan programın kaynak kodları



Formun Kodu

```
Option Explicit
Dim dizz(5) As String
Dim plainhex As String
Dim cipherhex As String
Dim keyhex As String
Dim tmp As String
Dim toplam, say2toplam As Double
Dim a, b, c, d, i As Integer
```

```
Dim gecici, maksimum As Currency
Dim gecici1, maksimum1 As Currency

Public Sub make_keys(dizi As String)
Dim i
For i = 0 To 4
  keys(i) = Mid(dizi, ((i * 4) + 1), 16)
  List6.AddItem "K" & i + 1
  List6.AddItem text_div(keys(i))
Next
End Sub

Private Sub Command1_Click()
List2.SetFocus
List2.ListIndex = maksimum
List8.SetFocus
List8.ListIndex = maksimum1
End Sub

Private Sub Form_Load()
Dim i, k, sonuc, sonuc2, sayici As Integer
Dim plainbin1 As String
Dim cipherbin1, v As String
Dim p, r As Integer
Dim f(256), son(256) As Currency
Dim f1(256), son1(256) As Currency

Randomize Timer

uretici(0) = "0"
uretici(1) = "1"
uretici(2) = "2"
```

uretici(3) = "3"
uretici(4) = "4"
uretici(5) = "5"
uretici(6) = "6"
uretici(7) = "7"
uretici(8) = "8"
uretici(9) = "9"
uretici(10) = "A"
uretici(11) = "B"
uretici(12) = "C"
uretici(13) = "D"
uretici(14) = "E"
uretici(15) = "F"

sbox(0) = "3"
sbox(1) = "D"
sbox(2) = "A"
sbox(3) = "2"
sbox(4) = "1"
sbox(5) = "7"
sbox(6) = "B"
sbox(7) = "5"
sbox(8) = "C"
sbox(9) = "E"
sbox(10) = "F"
sbox(11) = "6"
sbox(12) = "9"
sbox(13) = "8"
sbox(14) = "0"
sbox(15) = "4"

tsbox(0) = "E"

```
tsbox(1) = "4"
tsbox(2) = "3"
tsbox(3) = "0"
tsbox(4) = "F"
tsbox(5) = "7"
tsbox(6) = "B"
tsbox(7) = "5"
tsbox(8) = "D"
tsbox(9) = "C"
tsbox(10) = "2"
tsbox(11) = "6"
tsbox(12) = "8"
tsbox(13) = "1"
tsbox(14) = "9"
tsbox(15) = "A"

per(0) = "1"
per(1) = "14"
per(2) = "11"
per(3) = "8"
per(4) = "5"
per(5) = "2"
per(6) = "15"
per(7) = "12"
per(8) = "9"
per(9) = "6"
per(10) = "3"
per(11) = "16"
per(12) = "13"
per(13) = "10"
per(14) = "7"
per(15) = "4"
```

```
tahmini_anah(0) = "0000"
tahmini_anah(1) = "0001"
tahmini_anah(2) = "0010"
tahmini_anah(3) = "0011"
tahmini_anah(4) = "0100"
tahmini_anah(5) = "0101"
tahmini_anah(6) = "0110"
tahmini_anah(7) = "0111"
tahmini_anah(8) = "1000"
tahmini_anah(9) = "1001"
tahmini_anah(10) = "1010"
tahmini_anah(11) = "1011"
tahmini_anah(12) = "1100"
tahmini_anah(13) = "1101"
tahmini_anah(14) = "1110"
tahmini_anah(15) = "1111"

keyhex = "3A94D63F"              'sifreleme anahtarimiz
keybin = hextobin(keyhex, 7)
make_keys (keybin)               'alt anahtarlar üretiliyor(K0,K1,K2,K3,K4)

For a = 0 To 15                  'tahmini 2 bit(2A gibi) anahtarlar olusturuluyor
   For b = 0 To 15
      tahmini_anahtar(i) = CStr(uretici(a)) + CStr(uretici(b))
      i = i + 1
   Next
Next

For k = 0 To 2599
   plainhex = metin_uret
   veriler(k + 1, 1) = plainhex
```

```
   plainbin = hextobin(plainhex, 3)
   tmp = plainbin

      For i = 0 To 3                  '4 roundluk spn algoritmasi calistiriliyor
         If i < 3 Then
            dizz(i) = round(tmp, keys(i))
            tmp = dizz(i)
         Else
            dizz(i) = Last_round(tmp, keys(i), keys(i + 1))
            tmp = dizz(i)
         End If
      Next

   cipherbin = tmp
   cipherhex = bintohex(cipherbin, 16)
   veriler(k + 1, 2) = cipherhex

Next
'*****************************************************************
'veriler(1,1) 1. verinin anametni
'veriler(1,2) 1. verinin sifrelimetnini icerir
'*****************************************************************
For r = 0 To 255 ' 256 adet tahmini anahtar değerlerimiz için
   For p = 0 To 2599  ' verilerimizin degiskeni icin

   plainhex = veriler(p + 1, 1)
   plainbin = hextobin(plainhex, 3)
   cipherhex = veriler(p + 1, 2)
   cipherbin = hextobin(cipherhex, 3)
   sonuc = Int(hesapla(plainbin, cipherbin, r))
   sonuc2 = Int(hesapla2(plainbin, cipherbin, r))
```

```
    If sonuc = 0 Then
       sayac(r) = sayac(r) + 1
    End If

    If sonuc2 = 0 Then                  '2.  8 bit anahtar icin
       sayac2(r) = sayac2(r) + 1
    End If

    Next
    List1.AddItem sayac(r)
    List9.AddItem sayac2(r)

Next

For i = 0 To 255
    son(i) = (sayac(i) - 1300) / 2600
    son1(i) = (sayac2(i) - 1300) / 2600
    List4.AddItem son(i)
    List10.AddItem son1(i)
    f(i) = Abs(son(i))
    f1(i) = Abs(son1(i))

    If gecici < f(i) Then
       gecici = f(i)
         maksimum = i
    End If
    If gecici1 < f1(i) Then
       gecici1 = f1(i)
         maksimum1 = i
    End If

    List3.AddItem f(i)
```

```
    List11.AddItem f1(i)
    List2.AddItem tahmini_anahtar(i)
    List8.AddItem tahmini_anahtar(i)
Next

List7.AddItem gecici
List14.AddItem gecici1
List5.AddItem tahmini_anahtar(Int(maksimum))
List5.AddItem hextobin(tahmini_anahtar(Int(maksimum)), 1)
List13.AddItem tahmini_anahtar(Int(maksimum1))
List13.AddItem hextobin(tahmini_anahtar(Int(maksimum1)), 1)
End Sub

Private Sub List1_Click()
List2.ListIndex = List1.ListIndex
End Sub

Private Sub List2_Click()
List1.ListIndex = List2.ListIndex
List3.ListIndex = List2.ListIndex
List4.ListIndex = List2.ListIndex
End Sub

Private Sub List3_Click()
List2.ListIndex = List3.ListIndex
End Sub

Private Sub List4_Click()
List2.ListIndex = List4.ListIndex
End Sub

Private Sub List8_Click()
```

```
List9.ListIndex = List8.ListIndex
List10.ListIndex = List8.ListIndex
List11.ListIndex = List8.ListIndex
End Sub
```

Modülün kodu

```
Option Explicit
Public keys(5) As String
Public sbox(15) As String
Public tsbox(15) As String
Public per(15) As String
Public uretici(15) As String
Public tahmini_anah(15) As String
Public sayac(256) As Double
Public sayac2(256) As Double
Public plainbin As String
Public cipherbin As String
Public keybin As String
Public m As String
Public m1 As String
Public s1 As String
Public s2 As String
Public s3 As String
Public s4 As String
Public veriler(65536, 2) As String
'Public tum_veri(65536) As String
Public tahmini_anahtar(256) As String

Public Function text_div(a As String) As String
    text_div = Mid(a, 1, 4) + " " + Mid(a, 5, 4) + " " + Mid(a, 9, 4) + " " + Mid(a, 13, 4)
End Function
```

```
Public Function bintohex(str As String, lng As Integer) As String
Dim i, k As Integer
Dim nw_str As String
k = (lng / 4) - 1

For i = 0 To k
Select Case Mid(str, ((i * 4) + 1), 4)
Case "0000":   nw_str = nw_str + "0"
Case "0001":   nw_str = nw_str + "1"
Case "0010":   nw_str = nw_str + "2"
Case "0011":   nw_str = nw_str + "3"
Case "0100":   nw_str = nw_str + "4"
Case "0101":   nw_str = nw_str + "5"
Case "0110":   nw_str = nw_str + "6"
Case "0111":   nw_str = nw_str + "7"
Case "1000":   nw_str = nw_str + "8"
Case "1001":   nw_str = nw_str + "9"
Case "1010":   nw_str = nw_str + "A"
Case "1011":   nw_str = nw_str + "B"
Case "1100":   nw_str = nw_str + "C"
Case "1101":   nw_str = nw_str + "D"
Case "1110":   nw_str = nw_str + "E"
Case "1111":   nw_str = nw_str + "F"
End Select
Next
bintohex = nw_str
End Function

Public Function hextobin(str As String, k As Integer) As String
Dim j As Integer
Dim bin As String
For j = 0 To k
```

```
Select Case Mid(str, (j + 1), 1)
Case 0: bin = bin + "0000"
Case 1: bin = bin + "0001"
Case 2: bin = bin + "0010"
Case 3: bin = bin + "0011"
Case 4: bin = bin + "0100"
Case 5: bin = bin + "0101"
Case 6: bin = bin + "0110"
Case 7: bin = bin + "0111"
Case 8: bin = bin + "1000"
Case 9: bin = bin + "1001"
Case "A": bin = bin + "1010"
Case "B": bin = bin + "1011"
Case "C": bin = bin + "1100"
Case "D": bin = bin + "1101"
Case "E": bin = bin + "1110"
Case "F": bin = bin + "1111"
End Select
Next
hextobin = bin
End Function

Public Function XOR_Sub(txt As String, k As String) As String
Dim temp(15) As String
Dim yazi As String
Dim i As Integer
For i = 0 To 15
   temp(i) = Mid(txt, (i + 1), 1) Xor Mid(k, (i + 1), 1)
   yazi = yazi + temp(i)
Next
XOR_Sub = yazi
End Function
```

```
Public Function Substitution(str As String) As String
Dim nw_str As String
Dim i As Integer
For i = 0 To 3
Select Case Mid(str, ((i * 4) + 1), 4)
Case "0000":   nw_str = nw_str + hextobin(sbox(0), 0)
Case "0001":   nw_str = nw_str + hextobin(sbox(1), 0)
Case "0010":   nw_str = nw_str + hextobin(sbox(2), 0)
Case "0011":   nw_str = nw_str + hextobin(sbox(3), 0)
Case "0100":   nw_str = nw_str + hextobin(sbox(4), 0)
Case "0101":   nw_str = nw_str + hextobin(sbox(5), 0)
Case "0110":   nw_str = nw_str + hextobin(sbox(6), 0)
Case "0111":   nw_str = nw_str + hextobin(sbox(7), 0)
Case "1000":   nw_str = nw_str + hextobin(sbox(8), 0)
Case "1001":   nw_str = nw_str + hextobin(sbox(9), 0)
Case "1010":   nw_str = nw_str + hextobin(sbox(10), 0)
Case "1011":   nw_str = nw_str + hextobin(sbox(11), 0)
Case "1100":   nw_str = nw_str + hextobin(sbox(12), 0)
Case "1101":   nw_str = nw_str + hextobin(sbox(13), 0)
Case "1110":   nw_str = nw_str + hextobin(sbox(14), 0)
Case "1111":   nw_str = nw_str + hextobin(sbox(15), 0)
End Select
Next
Substitution = nw_str
End Function

Public Function permutation(str2 As String) As String
Dim newstr2 As String
Dim i As Integer
newstr2 = "0000000000000000"
For i = 0 To 15
```

```
Mid(newstr2, per(i), 1) = Mid(str2, i + 1, 1)
Next
permutation = newstr2
End Function

Public Function round(txt1 As String, k1 As String) As String
round = permutation(Substitution(XOR_Sub(txt1, k1)))
End Function

Public Function Last_round(txt2 As String, k2 As String, k3 As String) As String
Last_round = XOR_Sub(Substitution(XOR_Sub(txt2, k2)), k3)
End Function

Public Function metin_uret() As String
Dim gecici As String
Dim i As Integer
For i = 0 To 3
   gecici = gecici + uretici(Int(Rnd * 16))
Next
metin_uret = gecici
End Function

Public Function tSubstitution(str As String) As String
Dim nw_str As String
Select Case str
Case "0000":   nw_str = nw_str + hextobin(tsbox(0), 0)
Case "0001":   nw_str = nw_str + hextobin(tsbox(1), 0)
Case "0010":   nw_str = nw_str + hextobin(tsbox(2), 0)
Case "0011":   nw_str = nw_str + hextobin(tsbox(3), 0)
Case "0100":   nw_str = nw_str + hextobin(tsbox(4), 0)
Case "0101":   nw_str = nw_str + hextobin(tsbox(5), 0)
Case "0110":   nw_str = nw_str + hextobin(tsbox(6), 0)
```

```
Case "0111":   nw_str = nw_str + hextobin(tsbox(7), 0)
Case "1000":   nw_str = nw_str + hextobin(tsbox(8), 0)
Case "1001":   nw_str = nw_str + hextobin(tsbox(9), 0)
Case "1010":   nw_str = nw_str + hextobin(tsbox(10), 0)
Case "1011":   nw_str = nw_str + hextobin(tsbox(11), 0)
Case "1100":   nw_str = nw_str + hextobin(tsbox(12), 0)
Case "1101":   nw_str = nw_str + hextobin(tsbox(13), 0)
Case "1110":   nw_str = nw_str + hextobin(tsbox(14), 0)
Case "1111":   nw_str = nw_str + hextobin(tsbox(15), 0)
End Select

tSubstitution = nw_str
End Function

Public Function hesapla(str3 As String, str4 As String, y As Integer) As String
Dim i As Integer
Dim plainbin1 As String
Dim cipherbin1 As String
Dim mbin, v1, v2 As String
Dim xor_s As String
Dim v As String

plainbin1 = Mid(str3, 5, 1) Xor Mid(str3, 6, 1) Xor Mid(str3, 7, 1)
cipherbin1 = Mid(str4, 1, 8)

m = tahmini_anahtar(y) 'y. anahtarın değeri
mbin = hextobin(m, 1)

For i = 0 To 7
   v = Mid(cipherbin1, (i + 1), 1) Xor Mid(mbin, (i + 1), 1)
   xor_s = xor_s + v
Next
```

```
s1 = Mid(xor_s, 1, 4)
s2 = Mid(xor_s, 5, 4)
v1 = tSubstitution(s1)   ' ters s kutusu sonucu
v2 = tSubstitution(s2)    ' ters s kutusu sonucu

hesapla = Mid(v1, 2, 1) Xor Mid(v2, 1, 1) Xor plainbin1
End Function

Public Function hesapla2(str3 As String, str4 As String, y As Integer) As String
Dim i As Integer
Dim plainbin1 As String
Dim cipherbin1 As String
Dim mbin, v1, v2 As String
Dim xor_s8 As String
Dim v8 As String

plainbin1 = Mid(str3, 13, 1) Xor Mid(str3, 14, 1) Xor Mid(str3, 15, 1)
cipherbin1 = Mid(str4, 9, 8)

m1 = tahmini_anahtar(y) 'y. anahtarin degeri
mbin = hextobin(m1, 1)

For i = 0 To 7
   v8 = CStr(Mid(cipherbin1, (i + 1), 1) Xor Mid(mbin, (i + 1), 1))
   xor_s8 = xor_s8 + v8
Next

s3 = Mid(xor_s8, 1, 4)
s4 = Mid(xor_s8, 5, 4)
v1 = tSubstitution(s3)   ' ters s kutusu sonucu
v2 = tSubstitution(s4)    ' ters s kutusu sonucu
```

```
hesapla2 = Mid(v1, 2, 1) Xor Mid(v2, 1, 1) Xor plainbin1
End Function
```