æ

UYGULAMALI MATEMATİK ENSTİTÜSÜ, Kriptografi Bölümü
ODTÜ, TÜRKİYE

KRİPTOLOJİYE GİRİŞ DERS NOTLARI

Kriptolojiye Giriş Ders Notları,

Prof. Dr. Ersan Akyıldız
Doç. Dr. Ali Doğanaksoy
Yrd. Doç. Ebru Keyman
Dr. Muhiddin Uğuz
gözetimi altında aşağıda adı geçen kişiler tarafından hazırlanmıştır:

Kadir Altan
Kerem Kaşkaloğlu
Nihal Kındap
Çiğdem Özakın
Zülfükar Saygı
Elif Yıldırım
Murat Yıldırım
Senay Yıldız

Derleyenler: Ebru Keyman - Murat Yıldırım

İÇİNDEKİLER

BÖLÜM 1

GİRİŞ

Şifre sistemleri *açık anahtarlı* ve *gizli anahtarlı (simetrik)* olmak üzere ikiye ayrılır. Açık anahtarlı sistemlerde her kişi, biri açık diğeri gizli olmak üzere bir çift anahtar edinir. Açık anahtar diğer kullanıcıların erişimine açıkken; gizli anahtar sadece sahibinin erişebileceği şekilde saklanmalıdır. Açık anahtarı kullanarak herhangi bir kişi şifreli mesaj gönderebilir, ancak gönderilen şifreli mesajı sadece kullanılan açık anahtarın eşi olan gizli anahtar açabilir. Açık anahtarlı şifre sistemleri sadece şifreli mesaj göndermek amacıyla değil, kimlik denetimi yani sayısal imza ve daha birçok teknik için kullanılır. Açık anahtarlı sistemlerde, her zaman gizli anahtarın açık anahtarla matematiksel bir bağıntısı vardır. Bu anahtarları oluşturmak için çözülememiş matematik problemleri kullanıldığından, açık anahtarı kullananarak gizli anahtarı elde etme işlemi de imkansız kabul edilir.

Örnek 1.0.1 A, B kullanıcılar, K_A, K_B kullanıcıların açık anahtarları ve K'_A, K'_B kullanıcıların gizli anahtarları olsun. Her bir kullanıcı diğerlerinin açık anahtarını bilir. B kullanıcısı A kullanıcısına bir mesajı göndermek için mesajı K_A ile şifreleyip gönderir, A kullanıcısı şifrelenmiş mesajı K'_A ile deşifre eder.

Açık anahtarlı sistemleri ayrıntılı olarak ilerki konularda öğreneceğiz. Öncelikle gizli anahtarlı yani simetrik sistemlerden bahsedelim. Simetrik sistemlerde tek bir anahtar, hem şifreleme hem de deşifre amacıyla kullanılır. Güvenli bir şekilde iletişim kur-

madan önce gönderici ile alıcının gizli anahtar olarak adlandırılan bir anahtar üzerinde uzlaşmaları gerekir.

Simetrik sistemlerde temel problem, göndericinin ve alıcının üçüncü bir kişinin eline geçmesini engelleyerek ortak bir anahtar üzerinde anlaşmalarıdır. Ancak simetrik sistemlerin avantajı da, açık anahtarlı sistemlere göre daha hızlı olmalarıdır.

Bir sistemin güvenliği anahtarda yatar. Şifre çözmeye yönelik ataklar anahtarı bulmaya yöneliktir. Kriptanalist sahip olduğu ön bilgiye göre farklı saldırı çeşitleri uygular. Bunlar:

Sadece Şifreli Metin Saldırısı : Kriptanalist, aynı şifreleme algoritması kullanılarak şifrelenmiş çeşitli açık metinlerin şifreli metinlerine sahiptir. Kriptanalist mümkün olduğunca çok sayıdaki şifreli metinin açık metnini bulmaya çalışır. Asıl önemli olan ise açık metinleri şifrelemek için kullanılan anahtarı ya da anahtarları, aynı anahtarla şifrelenen başka mesajları çözmek için bulmaktır.

Bilinen Açık Metin Saldırısı : Kriptanalist sadece çeşitli açık metinlerin şifrelenmiş haline değil, bu mesajların açık metinlerine de erişebilmektedir.

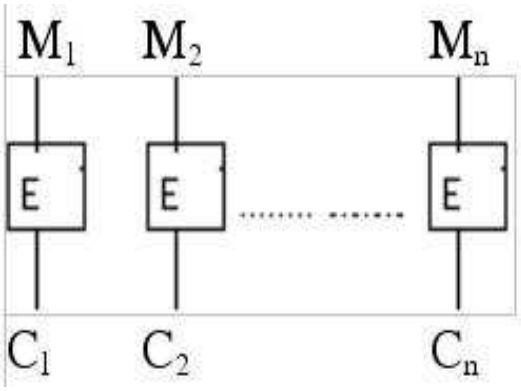
Seçilmiş Açık Metin Saldırısı : Kriptanalist sadece çeşitli açık metinlerin şifreli metinlerine ve bunlarla ilişkilendirilmiş açık metinlere erişmekle kalmayıp, aynı zamanda da şifrelenmiş açık metinleri seçebilmektedir. Bu atak bilinen açık metin atağından daha güçlü bir ataktır. Çünkü kriptanalist şifrelemek için açık metinin belirli bloklarını yani anahtar hakkında daha fazla bilgi sağlayabilecek olanları seçebilmektedir.

Simetrik sistemler *Blok Şifre Sistemleri* ve *Akan Şifre Sistemleri* olarak ikiye ayrılır.

BÖLÜM 2

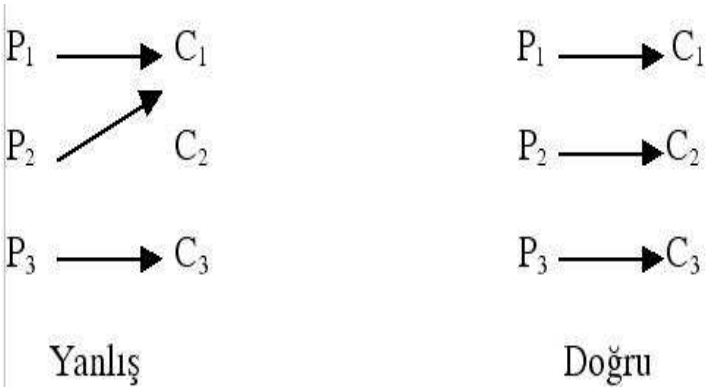
BLOK ŞİFRELER

Blok şifrelemenin en basit tanımı, açık metni bitişik bloklara bölme, her bloğu şifreleyerek şifreli metin bloklarına dönüştürme, bu şifreli blokları şifreli metin çıkışı olarak gruplamaktır. Blok şifre sistemini şekille göstermek istersek, M_1, $M_2, \ldots, M_n$ açık metnin blokları, yani her biri k bitten oluşan ardışık parçaları, C_1, $C_2, \ldots, C_n$ bu bloklara karşılık gelen şifrelenmiş metinler ve E şifreleme işlemi olmak üzere, blok şifre sistemlerini aşağıdaki şemayla gösterebiliriz.



Blok şifre Sistemlerinde şifreleme

Çoğu blok şifre sistemlerinde blok uzunluğu 64 bittir. İşlemcilerin hızı arttıkça blok uzunluğu da artabilmektedir. Son yıllarda üretilen sistemler 128 bit blok uzunluğu kullanılmaya başlanmıştır.



Bir blok şifre sisteminde, şifreli metin bloklarından birinin kaybolması, diğer blokların deşifre işleminde bir yanlışlığa neden olmaz. Bu blok şifre sistemlerinin en büyük avantajıdır. Blok şifre sistemlerinin en büyük dezavantajı ise şifreli metindeki birbirinin aynısı olan blokların, açık metinde de birbirinin aynı olmasıdır.

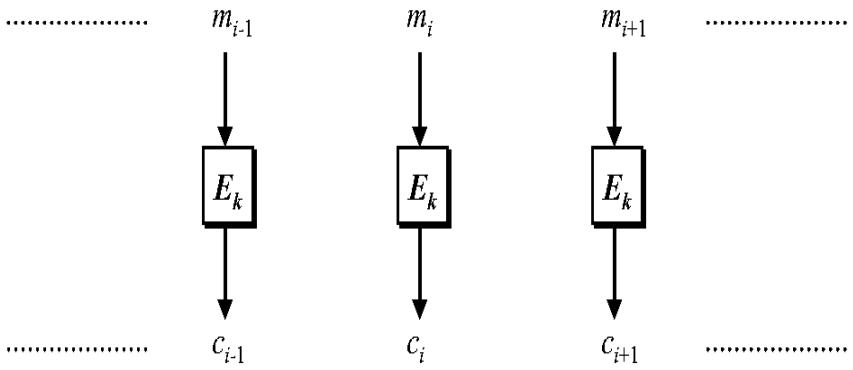
Örnek 2.0.2 "Senay'a kitabı sen ver" cümlesini, blok uzunluğu 3 olacak şekilde bölüp şifrelersek,

Açık Metin: sen-aya-kit-abı-sen-ver
Şifreli Metin: axk-bcg-xkt-ase-axk-hyt

birinci ve beşinci blokların aynı şekilde şifrelendiğini görüyoruz.

Böyle bir sorunun üstesinden gelmek için şifreleme işlemini değişik modellerle yapabiliriz. M_{i-1}, M_i, M_{i+1} açık metnin ardışık 3 bloğu, E şifreleme işlemi, C_{i-1}, C_i, C_{i+1} M_{i-1}, M_i, M_{i+1} ardışık bloklarının şifreli halleri olsun.

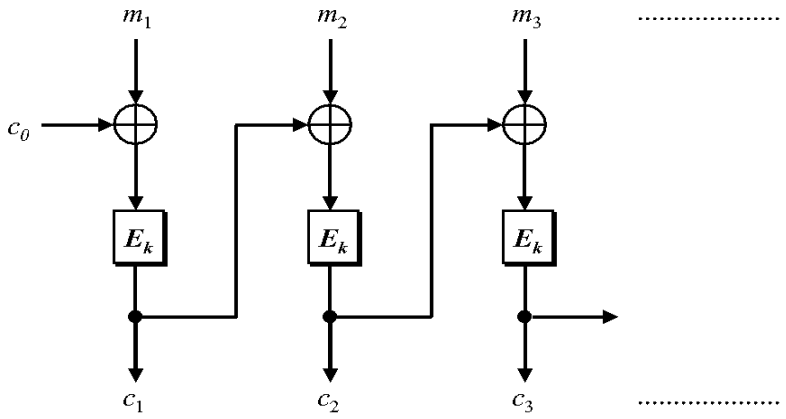
1. *Elektronik Kod Modeli* (Electronic Code Mode)



Elektronik Kod Modeli

Örnekteki gibi işler.

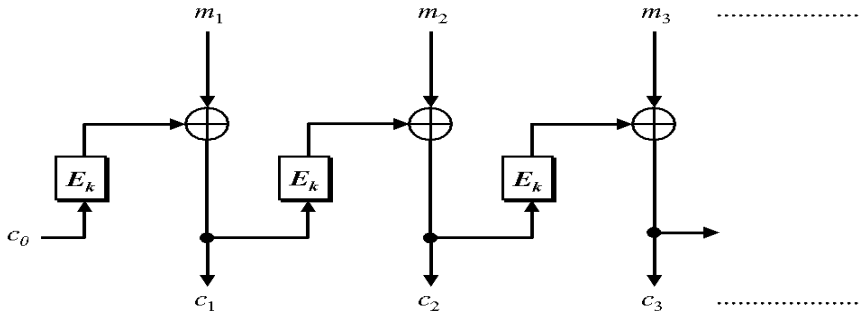
2. *Kapalı Metin Zincirleme Modeli* (Cipher Block Chaining Mode)



Kapalı Metin Zincirleme Modeli

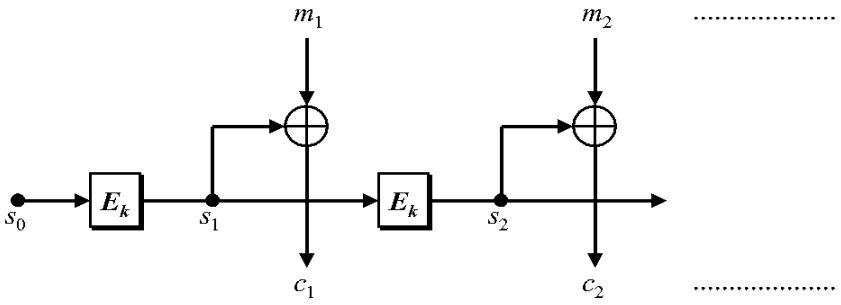
Aslında bu modelde yaptığımız işlemleri, büyük bloklar üzerinde akan şifre sistemini uygulamak olarak görebiliriz.

3. *Çıktıyı Geribesleme Modeli* (Output Feedback Mode)



Çıktıyı Geribesleme Modeli

4. *Girdiyi Geribesleme Modeli* (Input Feedback Mode)



Girdiyi Geribesleme Modeli

Bir blok şifre sisteminin matematiksel olarak şöyle tanımlayabiliriz. $\mathbb{Z}_2 = \{0\,1\}$, $\mathbb{Z}_2^n = \mathbb{Z}_2 \times \ldots \times \mathbb{Z}_2 = \{(x_{n-1}, \ldots, x_0) : \ x_i \in \mathbb{Z}_2\}$ ve K ise anahtar uzayı olsun. $E : \mathbb{Z}_2^n \times K \to \mathbb{Z}_2^n$

ve her $k \in K$ için $E(p,k)$ tersi alınabilir bir fonksiyondur. Bu fonksiyona şifreleme fonksiyonu denir. Blok şifre sistemi ile şifrelenen bir mesajı deşifre ederken aynı sistemi şifreli mesaja aynı anahtar ile uygulamak gerekir. Bunun için şifreleme işleminin tersinin olması gerekir. Şifreleme fonksiyonunun tersine de deşifreleme fonksiyonu denir ve $D(c,k)$ ile gösterilir.

2.1 Blok Şifre Sistemlerinin Parametreleri

2.1.1 Blok Uzunluğu

Bir blok şifre sisteminin güvenli olabilmesi için, blok uzunluğunun bazı blokların diğerlerinden daha fazla görünmeyeceği şekilde uzun olması gerekir. Örneğin bir blok şifreleme sistemi olan DES'teki 64 bit uzunluk, 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.

2.1.2 Anahtar ve Gerçek Anahtar Uzunluğu

Bir blok şifre sisteminin anahtarı deneme-yanılma (exhaustive key search) ile bulunamamalıdır. Bunun için de anahtar uzun olmalıdır. Diğer taraftan da anahtar uzunluğu üretim, dağıtım ve saklama için uygun ve güvenilir olmalıdır. Örneğin DES her zaman anahtar uzunluğunun kısa olmasından dolayı eleştirilmiştir. Diffie ve Hellman

DES'in anahtar deneme-yanılma yolu ile 20 milyon dolara mal olacak bir sistemle 12 saatte 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.

2.2 Blok Şifre Sistemlerinin Tasarım Ölçütleri

Güvenli bir blok şifre sisteminin kırılması zor ama uygulaması kolay olmalıdır. Şifreleme ve deşifreleme fonksiyonlarının kolay uygulanabilir olması gerekirken, $C = E(P, k)$ ve $P = D(C, k)$ eşitliklerinden k yı bulmanın zor olması gerekir. ilk defa Claude Shannon tarafından önerilen tasarım ölçütleri *yayılma* (confusion) ve *nüfuz etme* (diffusion)dir.

2.2.1 Yayılma

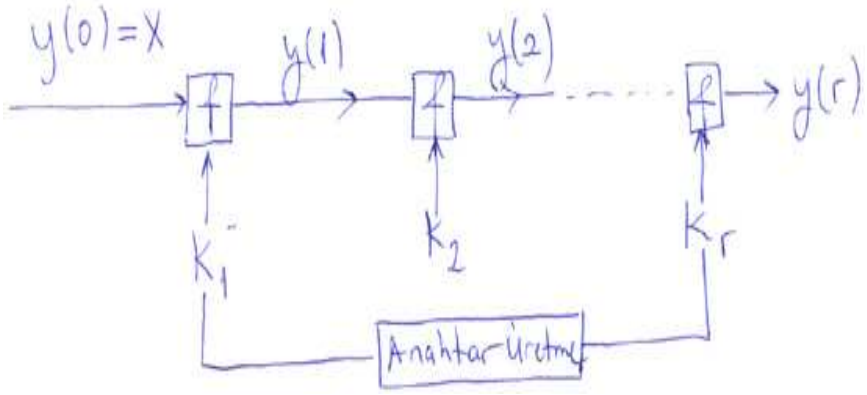
Bir blok şifre sistemini ya da genel olarak bir şifreleme sistemini yayılma ölçütüne göre tasarlamak demek, şifreli metinle anahtar arasındaki ilişkiyi mümkün olduğunca karışık yapmaktır. Daha açık bir tanım verirsek, yayılma, anahtarın açık ve şifreli metne bağlılığının kriptanaliz için faydalı olmayacak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

2.2.2 Nüfuz Etme

Bu ölçüte göre her anahtar için şifreleme fonksiyonu öyle olmalı ki, açık metin ve şifreli metin arasındaki yapılar arasında istatiksel bağlılık olmamalıdır. Bu ölçütün olabilmesi için anahtarın ve açık metinin her bitinin şifreli metini etkilemesi gerekir.

2.3 Döngülü (Iterated) Blok Şifre Sistemleri

Aynı fonksiyonu belli döngüler içinde uygulayan sistemlere döngülü blok şifre sistemleri denir.



Döngülü (Iterated) Blok Şifre Sistemleri

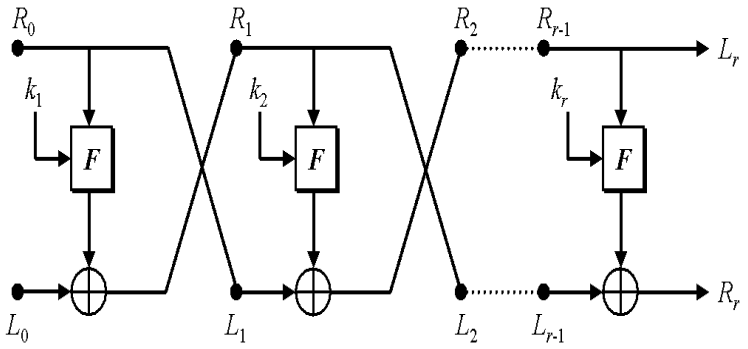
Fonksiyonun ilk kullanım hariç girdisi; bir önceki döngünün çıktısıdı ve anahtar üreten algoritmadan elde edilen döngü anahtarıdır. Örneğin DES'te 16 döngü vardır. Algoritmada

kullanılan f fonsiyonu basit bir fonksiyon olursa uygulamada bize hız yönünden kolaylık sağlar. Döngü sayısı uygun şekilde seçilirse, sistemde gereken yayılma ve nüfuz etme sa?lanır. Bu tür sistemlerde döngü sayısı, sistem tasarlandıktan sonra belli saldırılara karşı dayanıklılığı hesaplanarak belirlenmektedir.

2.4 Feistel Yapılar

Horst Feistel tarafından ilk defa tasarlanan sistem günümüzde birçok modern sistemde kullanılmaktadır.

$k_1, k_2, \ldots, k_n$: Döngü Anahtarları olmak koşuluyla , Feistel yapılarını aşağıdaki gibi gösterebiliriz.



Feistel Yapısı

Şifreleme yapılırken L bloğu üzerine, deşifreleme yapılırken R bloğu üzerine işlem yapılmaktadır. Ancak deşifreleme işleminde döngü anahtarları, ters sırada kullanılmaktadır.

BÖLÜM 3

DES

3.1 DES Algoritması

DES (Data Encryption Standard) algoritması, 1970 yılında IBM tarafından geliştirilen Lucifer algoritmasının biraz daha geliştirilmiş halidir. 1974'te IBM'in NSA ile birlikte geliştirdiği algoritma olan DES'in yayınlanmasından itibaren DES algoritması üzerinde geniş ölçüde çalışmalar yapılmıştır.

İlk tasarladığında donanım uygulamalarında kullanılması amaçlanmıştır. İletişim amaçlı kullanımda hem gönderen, hem de alıcı şifreleme ve deşifrelemede kullanılan aynı gizli anahtar üzerinde anlaşmış olmalıdır. Gizli anahtarın güvenli bir biçimde dağıtımı için açık anahtarlı sistem kullanılabilir. DES aynı zamanda sabit diskte veri saklamak gibi tek kullanıcılı şifreleme amaçlı da kullanılabilir. DES'in en büyük zayıflığı 56 bitlik anahtarıdır. Geliştirildiği zamanlarda çok iyi bir şifreleme algoritması olmasına rağmen modern bilgisayarlar tarafından yapılan anahtar saldırılarına karşı yetersiz kalmıştır. DES'in diğer bir zayıflığı da yavaş olmasıdır.

DES algoritması Feistel yapısındadır. DES'i 16 döngüden oluşan bir döngüye benzetebiliriz. İlk döngüye girmeden önce başlangıç permütasyonu ve son döngüden sonra da başlangıç permütasyonunun tersi uygulanır. DES algoritmasını aşağıdaki şekilde gösterilebilir.

DES Algoritması

Her döngü bir önceki döngüden gelen mesajı ikiye ayırır: L_i ve $R_i, i = 1, 2, \ldots, 16$. İşlemler R_i üzerinde yapılır. Her döngü için anahtardan döngü anahtarları üretilir. Deşifreleme işleminde de aynı algoritma kullanılır. Ancak anahtarların kullanım sırası tersten olur. Şimdi algoritmanın bileşenlerinin tek tek inceleyelim.

3.1.1 Başlangıç Permütasyonu

Başlangıç permütasyonu DES'e hiçbir kuvvet katmamaktadır.Başlangıç permütasyonunu aşağıda verilmiştir.

58 50 42 34 26 18 10 02 60 52 44 36 28 20 12 04
62 54 46 38 30 22 14 06 64 56 48 40 32 24 16 08
57 49 41 33 25 17 09 01 59 51 43 35 27 19 11 03
61 53 45 37 29 21 13 05 63 55 47 39 31 23 15 07

Görüldüğü üzere, başlangıç permütasyonunda 58. bit 1. bit yerine, 50.bit 2. bit yerine,... gelmektedir.

3.1.2 Başlangıç Permütasyonun Tersi

Başlangıç permütasyonunun tersi olan permütasyon son rounddan sonra uygulanır.Bu permütasyon aşağıda verilmiştir.

40 08 48 16 56 24 64 32 39 07 47 15 55 23 63 31

38 06 46 14 54 22 62 30 37 05 45 13 53 21 61 29

36 04 44 12 52 20 60 28 35 03 43 11 51 19 59 27

34 02 42 10 50 18 58 26 33 01 41 09 49 17 57 25

3.1.3 Anahtar Permütasyonu ve Döngü Anahtarının Üretilmesi

Anahtar üzerine ilk işlem 64 bitten 56 bite indirgemektir.Bunun için her 8. bit doğruluk kontrolü(parity check)için atılır. Daha sonra 56 bitlik anahtar aşağıda verilen permütasyona girer.

57 49 41 33 25 17 09

01 58 50 42 34 26 18

10 02 59 51 43 35 27

19 11 03 60 52 44 36

63 55 47 39 31 23 15

07 62 54 46 38 30 22

14 06 61 53 45 37 29

21 13 05 28 20 12 04

Bu permütasyondan sonra 56 bitlik anahtar 28 bitlik sağ ve sol olmak üzere iki parçaya ayrılır. Döndürme(rotation) olarak adlandırdığımız kısımda, 28 bitlik parçalar her döngü için 1 yada 2 bit sola kayar.Bu kaydırma döndürme olarak adlandırılır çünkü kayan bitler sona eklenir.

Döngü	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16
Kayma Miktarı	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Daha sonra anahtarı döngüye göndermeden önce tekrar bir permütasyon gerçekleşir. Bu permütasyon sonucu 56 bit 48 bite iner. Bu permütasyon aşağıda verilmiştir.

14 17 11 24 01 05

03 28 15 06 21 10

23 19 12 04 26 08

16 07 27 20 13 02

41 52 31 37 47 55

30 40 51 45 33 48

44 49 39 56 34 53

46 42 50 36 29 32

3.1.4 f fonksiyonu

Önceden de bahsedildiği üzere her döngüde sağ 32 bitlik kısım (R_i) üzerine işlemler yapılır. Öncelikle bu 32 bitlik kısım aşağıdaki gibi 48 bite genişletilir.

32 01 02 03 04 05 04 05 06 07 08 09

08 09 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 01

48 bitlik bu kısım döngü anahtarı ile x-or işlemine ($\oplus$) gönderilir. Sonra 48 bit, 6 bitlik 8 gruba bölünür ve her bir grup ayrı bir S-kutusuna gönderilir. S-kutularında 6 bitler 4 bite çevrilir. 8 S-kutusu aşağıda verilmiştir.

1. S-kutusu

	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	00	15	07	04	14	02	13	01	10	06	12	11	09	05	03	08
2	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

Örnek 3.1.1 *girdi = 101110 satır = 10(ilk ve son bitler) = 2 , sütun = 0111(ortada kalan bitler) = 7 çıktı = 11 (onbir) = 1011*

2. S-kutusu

	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0	15	01	08	14	06	11	03	04	09	07	02	13	12	00	05	10
1	03	13	04	07	15	02	08	14	12	00	01	10	06	09	11	05
2	00	14	07	11	10	04	13	01	05	08	12	06	09	03	02	15
3	13	08	10	01	03	15	04	02	11	06	07	12	00	05	14	09

3. S-kutusu

	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0	10	00	09	14	06	03	15	05	01	13	12	07	11	04	02	08
1	13	07	00	09	03	04	06	10	02	08	05	14	12	11	15	01
2	13	06	04	09	08	15	03	00	11	01	02	12	05	10	14	07
3	01	10	13	00	06	09	08	07	04	15	14	03	11	05	02	12

4. S-kutusu

	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0	07	13	14	03	00	06	09	10	01	02	08	05	11	12	04	15
1	13	08	11	05	06	15	00	03	04	07	02	12	01	10	14	09
2	10	06	09	00	12	11	07	13	15	01	03	14	05	02	08	04
3	03	15	00	06	10	01	13	08	09	04	05	11	12	07	02	14

5. S-kutusu

	00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15
0	02 12 04 01 07 10 11 06 08 05 03 15 13 00 14 09
1	14 11 02 12 04 07 13 01 05 00 15 10 03 09 08 06
2	04 02 01 11 10 13 07 08 15 09 12 05 06 03 00 14
3	11 08 12 07 01 14 02 13 06 15 00 09 10 04 05 03

6. S-kutusu

	00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15
0	12 01 10 15 09 02 06 08 00 13 03 04 14 07 05 11
1	10 15 04 02 07 12 09 05 06 01 13 14 00 11 03 08
2	09 14 15 05 02 08 12 03 07 00 04 10 01 13 11 06
3	04 03 02 12 09 05 15 10 11 14 01 07 06 00 08 13

7. S-kutusu

	00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15
0	04 11 02 14 15 00 08 13 03 12 09 07 05 10 06 01
1	13 00 11 07 04 09 01 10 14 03 05 12 02 15 08 06
2	01 04 11 13 12 03 07 14 10 15 06 08 00 05 09 02
3	06 11 13 08 01 04 10 07 09 05 00 15 14 02 03 12

8. S-kutusu

	00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15
0	13 02 08 04 06 15 11 01 10 09 03 14 05 00 12 07
1	01 15 13 08 10 03 07 04 12 05 06 11 00 14 09 02
2	07 11 04 01 09 12 14 02 00 06 10 13 15 03 05 08
3	02 01 14 07 04 10 08 13 15 12 09 00 03 05 06 11

S-kutuları blok şifrelerin doğrusal olmayan kısımlarıdır. Hiçbir S-kutusu girdinin doğrusal yada afin fonksiyonu değildir.

S-kutularından çıkan 4 bitlik parçalar yanyana gelerek birleşir ve aşağıdaki permütasyona gider.

16 07 20 21

29 12 28 17

01 15 23 26

05 18 31 10

02 08 24 14

32 27 03 09

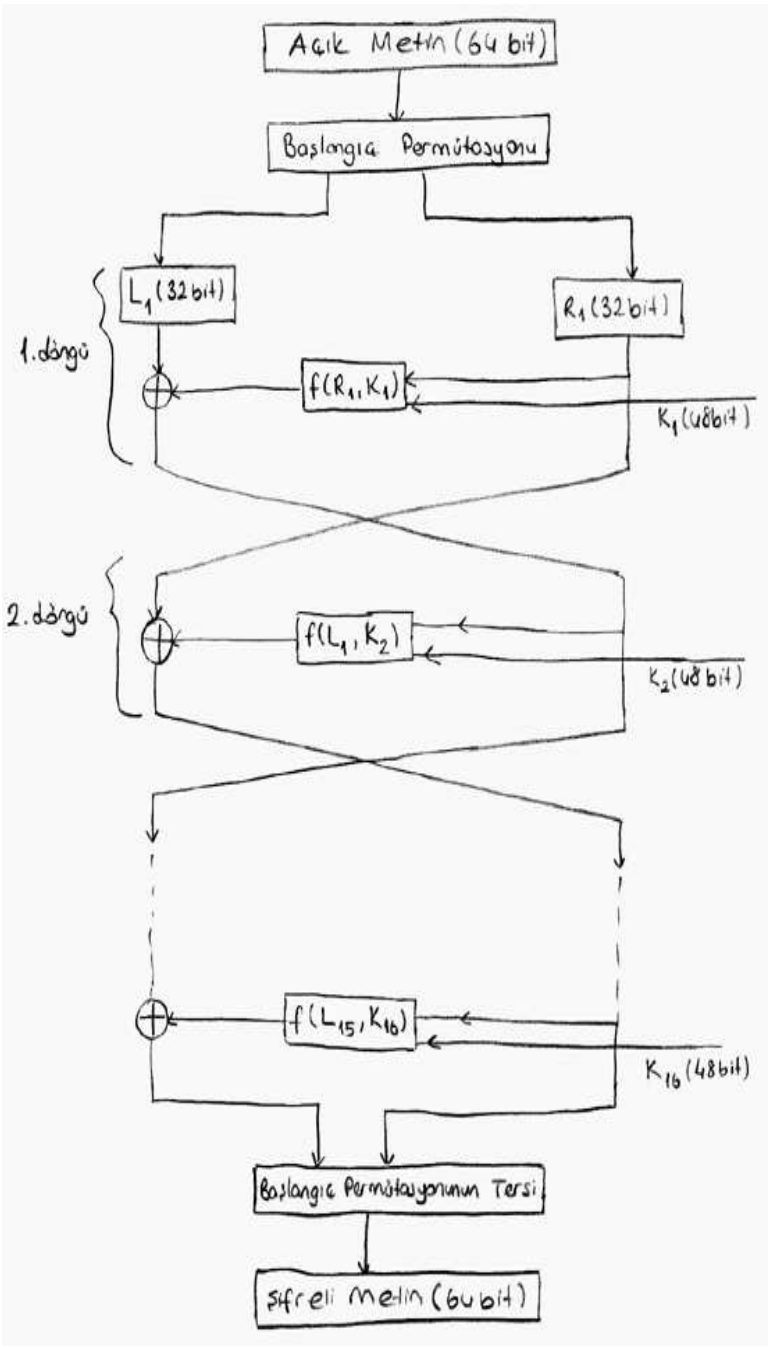
19 13 30 06

22 11 04 25

Permütasyondan çıkan 32 bitlik kısım döngünün başaında ayrılan 32 bitlik kısımla XOR işlemi uygulanır.

3.2 DES'in Tasarım Özellikleri

DES'in en önemli özelliği yayılma(confusion) ve nüfuz etme(diffusion)özellikleridir. DES'te her bloğun her biti diğer bitlere ve anahtarın her bitine bağımlıdır. Bunun iki amacı vardır:Öncelikle, anahtar üzerindeki bilinmezlik(uncertanity)artmaktadır. İkinci olaraksa, açık metindeki veya anahtardaki 1 bitin değişmesi bütün şifreli metinin değişmesine sebep olur ki bu bizim ileride öğreneceğimiz diffrensiyel kriptanalizde işe yaramaktadır.


Açık Metin (64 bit)
Başlangıç Permütasyonu
L1 (32 bit)
R1 (32 bit)
1. döngü
f(R1, K1)
K1 (48 bit)
2. döngü
f(L1, K2)
K2 (48 bit)
f(L15, K16)
K16 (48 bit)
Başlangıç Permütasyonunun Tersi
Şifreli metin (64 bit)

BÖLÜM 4

RIJNDAEL

4.1 Matematiksel Özellikler

8 bitten oluşan bir byte 16'lık tabanda yazılabildiği gibi polinom olarak ifade ediliyor. İki byte'ı toplamak, çarpmak ve bir byte'ın tersini almak polinomlarca ifade edilecektir.Buna göre bir $b = (b_7\, b_6\, b_5\, b_4\, b_3\, b_2\, b_1\, b_0)$ bytının *polinom gösterimi*:

$$p(a) = b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$$

Bir *byte'ın değeri* 10'luk tabanda kendisine karşılık gelen sayıdır.

Örnek 4.1.1 $(53)_{16}$ byte'ının değeri: $(53)_{16} =$ '53' $= 3 + 5.16 = 83$
BİT olarak ifadesi yani ikilik düzende: $83 = (0\,1\,0\,1\,0\,1\,1\,1)$
$p_a(x) = x^6 + x^4 + x^2 + x + 1$

4.1.1 Toplama İşlemi

Byte'ların polinomlarını modulo 2'de toplamaktır. Bu işlem aynı zamanda iki byte'ı XOR'lamaya da denktir.

Örnek 4.1.2

$$\begin{aligned} a = (1\,0\,0\,1\,1\,0\,1\,0) \Rightarrow p_a(x) &= x^7 + x^4 + x^3 + x \\ b = (1\,0\,1\,0\,1\,0\,1\,1) \Rightarrow p_b(x) &= x^7 + x^5 + x^3 + x + 1 \\ p_a(x) + p_b(x) &= x^7 + x^4 + x^3 + x + x^7 + x^5 + x^3 + x + 1 \bmod 2 \\ &= x^5 + x^4 + 1 \quad \Rightarrow \quad (0\,0\,1\,1\,0\,0\,0\,1) \\ a \oplus b &= (0\,0\,1\,1\,0\,0\,0\,1) \end{aligned}$$

4.1.2 Çarpma İşlemi

1. İki Byte'ı Çarpma:

   çarpma işleminde, iki byte polinom olarak ifade edilir. İki polinom çarpılır, çarpma işlemi mod $m(x) = x^8 + x^4 + x^3 + x + 1$ de yapılır. $m(x)$ modulo 2'de çarpanlarına ayrılamayan bir polinomdur. Modulo 2'de çarpanlarına ayrılamamak demek katsayıları 1 veya 0 olan polinomların çarpımı şeklinde yazılamamak demektir. Bir polinomun modulo $m(x)$'deki değeri polinomun $m(x)$'e bölümünden kalana denktir.

   Bir polinomun $m(x)$'e bölümünden kalanı bulmak için polinomda $x^8+x^4+x^3+x+1$ görülen yere 0 koymaktır. Bu aynı zamanda x^8 görülen yere x^4+x^3+x+1 koymakla aynıdır.

   Bir byte'ın polinom gösteriminde en fazla yedinci dereceden bir terim olacağından iki byte'ın çarpımının yine bir byte olabilmesi polinom ifadesinde derecesi sekiz ve sekizden büyük terimlerin yok edilmesi gerekiyor. Bu nedenle çarpma işlemi modulo

2'de çarpanlarına ayrılamamyan bir polinom olan mod $m(x)$ de yapılıyor

Not: Bu işlem için derecesi 8 olan ve modulo 2'de çarpanlarına ayrılamamyan başka bir polinom da seçilebilirdi.

Kısaca:

$$\begin{aligned} a &\rightarrow p_a(x) \\ b &\rightarrow p_b(x) \\ (a).(b) &= p_a(x).p_b(x) \bmod m(x) \end{aligned}$$

Örnek 4.1.3

$$\begin{aligned} a = (0\,1\,0\,1\,0\,1\,0\,1) \Rightarrow p_a(x) &= x^6 + x^4 + x^2 + 1 \\ b = (1\,0\,0\,0\,0\,0\,1\,1) \Rightarrow p_b(x) &= x^7 + x + 1 \end{aligned}$$

$$
\begin{aligned}
(a).(b) &= p_a(x).p_b(x) \mod m(x) = x^8 + x^4 + x^3 + x + 1 \\
&= (x^6 + x^4 + x^2 + 1)(x^7 + x + 1) \\
&= x^{13} + x^7 + x^6 + x^{11} + x^5 + x^4 + x^9 + x^3 + x^2 + x^7 + x + 1 \\
&= x^{13} + x^{11} + x^9 + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1 \\
&= x^5x^8 + x^3x^8 + x + x^8 + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1 \\
&= x^5(x^4 + x^3 + x + 1) + x^3(x^4 + x^3 + x + 1) + x(x^4 + x^3 + x + 1) \\
&\quad +x^6 + x^5 + x^4 + x^3 + x^2 + x + 1 \\
&= x^9 + x^8 + x^6 + x^5 + x^7 + x^6 + x^4 + x^3 + x^5 + x^4 + x^2 + x \\
&\quad +x^6 + x^5 + x^4 + x^3 + x^2 + x + 1 \\
&= x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + 1 \\
&= xx^8 + x^8 + x^7 + x^6 + x^5 + x^4 + 1 \\
&= x(x^4 + x^3 + x + 1) + x^4 + x^3 + x + 1 + x^7 + x^6 + x^5 + x^4 + 1 \\
&= x^5 + x^4 + x^2 + x + x^4 + x^3 + x + 1 + x^7 + x^6 + x^5 + x^4 + 1 \\
&= x^7 + x^6 + x^4 + x^3 + x^2 \\
\Rightarrow\ & (a)(b) = (1\,1\,0\,1\,1\,1\,0\,0)
\end{aligned}
$$

2. 4 Byte'lık Vektörleri Çarpma:

   4 byle'lık bir vektör olan $\vec{a} = (a_3, a_2, a_1, a_0)$ polinom olarak ifade edilir.

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

   Burada a_3, a_2, a_1, a_0'ın byte oldukları unutulmamalıdır.

$\vec{a}$ ve $\vec{b} = (b_3, b_2, b_1, b_0)$ vektörleri için:

$$\vec{a}.\vec{b} = p_{\vec{a}}(x).p_{\vec{b}}(x) \bmod M(x) = x^4 + 1$$

Aynı zamanda

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} . \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Not: 4 byte'lık bir $\vec{b} = (b_3, b_2, b_1, b_0)$vektörünü $\vec{a} = (\text{`00`}, \text{`00`}, \text{`01`}, \text{`00`}) \Rightarrow p_{\vec{a}}(x) = x$ ile çarpmak bir sola kaydırmaya denktir. Yani

$$(b_3b_2b_1b_0).(0010) = (b_2b_1b_0b_3)$$

3. $a = (a_7\ a_6\ a_5\ a_4\ a_3\ a_2\ a_1\ a_0)$ byte'ının çarpmaya göre tersi

$$p_a(x).p_b(x) = 1 \mod m(x) = x^8 + x^4 + x^3 + x + 1$$

eşitliğini sağlayan $p_b(x)$ polinomuna karşılık gelen byte'tır.

$$p_a^{-1}(x) = p_b(x) \mod m(x) = x^8 + x^4 + x^3 + x + 1.$$

Buna göre $a = (0\ 0\ 0\ 0\ 0\ 0\ 0\ 0)$ nın tersi kendisidir.

4. 4 byte'lık $\vec{a} = (a_3, a_2, a_1, a_0)$ vektörünün çarpmaya göre tersi

$$p_{\vec{a}}(x).p_{\vec{b}}(x) = 1 \mod M(x) = x^4 + 1$$

eşitliğini sağlayan $p_{\vec{b}}(x)$ polinomuna karşılık gelen 4 byte'lık $\vec{b}$ vektörüdür.

$$p_{\vec{a}}^{-1}(x) = p_{\vec{b}}(x) \mod M(x) = x^4 + 1.$$

4.2 Algoritma

Rijndael Algoritması

Metin uzunluğu: 128,192,256 bit olabilir.

Anahtar uzunluğu: 128,192,256 bit olabilir.

Döngü (*round*) sayısı: Anahtar uzunluğu ve metin uzunluğuna göre değişiklik göstermektedir. Aşağıdaki tabloda gösterilmiştir. Satırlar metin uzunluklarını, sütünlar anahtar uzunlukarını göstermektedir.

	128	192	256
128	10	12	14
192	12	12	14
256	14	14	14

Her döngüde üç ayrı bölüm vardır.

1. Doğrusal (linear) işlemlerin olduğu bölüm. Bu katmanlarda (*layer*) difüzyon sağlanmaktadır.
2. Doğrusal olmayan bölüm. S kutularından (S-*box*) oluşmaktadır.
3. Anahtarın XOR'landığı katman.

Bit olarak ifade edilen mesajı byte'lara ayrılır.

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

Metin 4 byte'lık sütün vektörleri şeklinde, yani 128 bit için 4×4, 192 bit için 4×6, 256 bit için 4×8'lik matrislerle ifade edilir. 128 bitlik bir metin için aşağıdaki gibidir.

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

4.2.1 Byte Sub

S-kutusunun olduğu katmandır. Matristeki her byte'ın modulo $m(x) = x^8+x^4+x^3+x+1$'e göre tersi bulunur. $a \longrightarrow a^{-1} = b = (b_7\ b_6\ b_5\ b_4\ b_3\ b_2\ b_1\ b_0)$ S-kutusunun çıktısı $y = (y_7\ y_6\ y_5\ y_4\ y_3\ y_2\ y_1\ y_0)$ olmak üzere:

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Burdaki 8×8'lik matrisin özelliği modulo 2'de tersi olan bir matris olmasıdır.

4.2.2 Shift Row

128 bit için:

$$\begin{array}{cccc} 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \\ 4 & 8 & 12 & 16 \end{array} \longrightarrow \begin{array}{cccc} 1 & 5 & 9 & 13 \\ 6 & 10 & 14 & 2 \\ 11 & 15 & 3 & 7 \\ 16 & 4 & 8 & 12 \end{array}$$

Bu permütasyona göre 6. pozisyondaki byte 2. pozisyona, 3. pozisyondaki byte 11. pozisyona geçmiş.

192 bit için:

$$\begin{array}{cccccc} 1 & 5 & 9 & 13 & 17 & 21 \\ 2 & 6 & 10 & 14 & 18 & 22 \\ 3 & 7 & 11 & 15 & 19 & 23 \\ 4 & 8 & 12 & 16 & 20 & 24 \end{array} \longrightarrow \begin{array}{cccccc} 1 & 5 & 9 & 13 & 17 & 21 \\ 6 & 10 & 14 & 18 & 22 & 2 \\ 11 & 15 & 19 & 23 & 3 & 7 \\ 16 & 20 & 24 & 4 & 8 & 12 \end{array}$$

256 bit için:

$$\begin{array}{cccccccc} 1 & 5 & 9 & 13 & 17 & 21 & 25 & 29 \\ 2 & 6 & 10 & 14 & 18 & 22 & 26 & 30 \\ 3 & 7 & 11 & 15 & 19 & 23 & 27 & 31 \\ 4 & 8 & 12 & 16 & 20 & 24 & 28 & 32 \end{array} \longrightarrow \begin{array}{cccccccc} 1 & 5 & 9 & 13 & 17 & 21 & 25 & 29 \\ 6 & 10 & 14 & 18 & 22 & 26 & 30 & 2 \\ 15 & 19 & 23 & 27 & 31 & 3 & 7 & 11 \\ 20 & 24 & 28 & 32 & 4 & 8 & 12 & 16 \end{array}$$

4.2.3 Mix Column

Byte Sub ve Shift Row işlemlerinden çıkan, her sütünü 4 bytle'lık vektör olan matris, bu katmanda modulo $M(x) = x^4 + 1$'de $c(x) = \text{`}03\text{`}x^3 + \text{`}01\text{`}x^2 + \text{`}01\text{`}x + \text{`}02\text{`}$ polinomuyla

çarpılır. Buna göre:

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

4.2.4 Anahtarla XOR'lama

Anahtarlar da aynı şekilde matrislerle ifade edilir, anahtarla metnin karşılıklı byte'ları XOR'lanır.

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

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

4.3 Anahtar Algoritması

Anahtar olarak üretilen toplam bit sayısı (blok uzunluğu) × (Döngü sayısı+1) kadardır. Eğer 4 bytle'lık vektörlerin her birine kelime (1 kelime=32 bit) dersek, kelime olarak üretilen anahtar sayısı (bloktaki kelime sayısı) × (döngü sayısı+1) dir.

$N_b = 1$ bloktaki kelime sayısı

$N_k =$ Anahtardaki kelime sayısı

$N_r =$ Döngü sayısı

Bu anahtar algoritmasından çıkan her kelimeye $\begin{bmatrix} w_{0i} \\ w_{1i} \\ w_{2i} \\ w_{3i} \end{bmatrix}$ dersek,

$0 \leq i < N_k$ için

$$\begin{bmatrix} w_{0i} \\ w_{1i} \\ w_{2i} \\ w_{3i} \end{bmatrix} = \begin{bmatrix} k_{0i} \\ k_{1i} \\ k_{2i} \\ k_{3i} \end{bmatrix}$$

$N_k \leq i < N_b(N_r + 1)$ için

$N_k \mid i$ ise

$$w_i = w_{i-N_k} \oplus Subbyte(RotByte(w_{i-1})) \oplus RoundConstant[i/N_k]$$

$N_k \nmid i$ ise

$$w_i = w_{i-N_k} \oplus w_{i-1}$$

Subbyte: Byte sub katmanındaki işlemlerden oluşur.

Rotbyte: Rotbyte((a,b,c,d))=(b,c,d,a).

Round Constant:

RC[1]='01'

RC[j]=x.RC[j-1]

$$RoundConstant[\text{j}] = \begin{bmatrix} \text{RC}[j] \\ \text{'00'} \\ \text{'00'} \\ \text{'00'} \end{bmatrix}$$

Anahtar algoritmasında ilk alınan anahtar 32 bitlik kelimeler halinde hiç değişikliğe uğramadan kullanılır. Eğer algoritmadan çıkan anahtar kelimeler (32 bit uzunluğunda 4 byte'lık vektörler) $w_0\, w_1\, w_2\, w_3\, w_4\, w_5 \ldots$ ise 128 bitlik blok uzunluğu için:

$$\begin{aligned} w_0\, w_1\, w_2\, w_3 &\rightarrow \textit{Round}\ 0 \text{ Round'a girmeden} \\ w_4\, w_5\, w_6\, w_7 &\rightarrow \textit{Round}\ 1 \\ w_8\, w_9\, w_{10}\, w_{11} &\rightarrow \textit{Round}\ 2 \\ &\vdots \end{aligned}$$

192 bitlik blok uzunluğu için:

$$\begin{aligned} w_0\, w_1\, w_2\, w_3\, w_4\, w_5 &\rightarrow \textit{Round}\ 0 \text{ Round'a girmeden} \\ w_6\, w_7\, w_8\, w_9\, w_{10}\, w_{11} &\rightarrow \textit{Round}\ 1 \\ w_{12}\, w_{13}\, w_{14}\, w_{15}\, w_{16}\, w_{17} &\rightarrow \textit{Round}\ 2 \\ &\vdots \end{aligned}$$

256 bitlik blok uzunluğu için:

$$\begin{aligned} w_0\, w_1\, w_2\, w_3\, w_4\, w_5\, w_6\, w_7 &\rightarrow \textit{Round}\ 0 \text{ Round'a girmeden} \\ w_8\, w_9\, w_{10}\, w_{11}\, w_{12}\, w_{13}\, w_{14}\, w_{15} &\rightarrow \textit{Round}\ 1 \\ w_{16}\, w_{17}\, w_{18}\, w_{19}\, w_{20}\, w_{21}\, w_{22}\, w_{23} &\rightarrow \textit{Round}\ 2 \\ &\vdots \end{aligned}$$

4.4 ALGORİTMANIN TERSİ

Algoritmaın Tersi

Algoritmanın tersinde katmanların tersi uygulanır.

4.4.1 Mix Column

Mix Column'da kullanılan polinomun tersi kullanılır. $c^{-1}(x) = d(x) = \text{`}0B\text{`}x^3 + \text{`}0D\text{`}x^2 + \text{`}09\text{`}x + \text{`}0E\text{`}$ çarpma işlemi yine modulo $M(x) = x^4 + 1$'de yapılır.

4.4.2 Byte Sub

Byte Sub katmanında yapılan işlemlerin tersi yapılır.

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

Bulunan $x = (x_7\, x_6\, x_5\, x_4\, x_3\, x_2\, x_1\, x_0)$'in modulo $m(x) = x^8 + x^4 + x^3 + x + 1$'de tersi alınır.

BÖLÜM 5

AKAN ŞİFRELER

Akan şifre sistemleri, mesajın her karakterini (bitini) ayrı ayrı şifreler. Şifreleme işlemi mesaj uzunluğunda bir anahtar kullanılarak yapılır. Anahtarın her biti mesajın her bitiyle mod 2'de karşılıklı toplanır. Bu işleme XOR işlemi denir ve $\oplus$ ile gösterilir.

Şifreleme :

$$\begin{aligned} \text{Açık metin } m &= m_1\, m_2 \ldots m_n \\ \text{Anahtar } k &= k_1\, k_2 \ldots k_n \\ \text{Kapalı metin } c &= c_1\, c_2 \ldots c_n \\ \text{Burada her } i \text{ için } c_i &= m_i \oplus k_i \text{ dir.} \end{aligned}$$

Şifreleme işleminde kullanılan iki tip anahtar dizisi vardır.

1. Tam Rastgele (true random) Dizi : Dizideki her bit birbirinden bağımsız olarak üretilir. Buna bir örnek olarak yazı tura atışı verilebilir.

2. Pseudo Rastgele (pseudo random) Dizi : Dizinin her biti kendinden önce gelen bitlere bağlıdır. Aynı zamanda her bit kendinden sonra gelen biti etkiler.

5.1 One Time Pad Sistemi

Şifrelenecek mesajın uzunluğunda tam rastgele bir anahtar dizisi seçilir. Mesaj ve anahtara XOR işlemi uygulanır.

Örnek 5.1.1

$$\begin{aligned} m &= 11010001011010110 \\ k &= 01110100101101000 \\ m \oplus k &= 10100101110111110 \end{aligned}$$

Mesajı açmak için aynı anahtara ve kapalı metine tekrar XOR işlemi uygulanır.

5.1.1 Sistemin Avantajları

Uzunluğu n bit olan bir mesaj için n bitlik bir anahtar dizisi seçilir. Mesaj şifrelenir ve gönderilir. Mesajı ele geçiren birisi olası bütün anahtarları (2^n tane) denese bile mesajı bulamaz. Çünkü bu işlemin sonunda n bitlik bütün kelimeleri bulur. Elinde birden fazla anlamlı mesaj olacağıiçin bu mesajların içinden gerçek mesajı tahmin etmek imkansızdır. Bu açıdan *koşulsuz güvenli* bir sistemdir.

5.1.2 Sistemin Dezavantajları

Uzun bir mesaj şifrelemek için uzun bir anahtar üretmek gerekir. Bu sistem tam rastgele bir anahtar dizisi kullandığından, uzun bir anahtar üretmek, bu anahtarı güvenli bir

şekilde karşı tarafa iletmek ve saklamak zor olur. Ayrıca kullanılan anahtar tekrar kullanılamayacağı için, her seferinde başka bir anahtar üretilmesi gerekir. Bu nedenlerden dolayı sistemin kullanımı zordur.

5.2 Dizi Üreticiler

Gerçekten rastgele dizilerin üretilmesi ve iletilmesi gibi zordur. Bu nedeniyle pseudo rastgele diziler kullanılır. Bu diziler kısa bir anahtar kullanılarak bir dizi üretici tarafından üretilir.

Dizi üreticiler açısından akan şifreleri ikiye ayırabiliriz.

1. Senkronize (Synchronous) Akan Şifre :

   f_s: Faz fonksiyonu, bir sonraki fazı belirleyen fonksiyon.

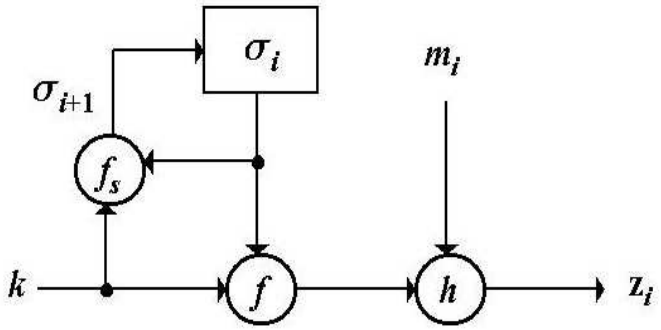
   f : Üreticinin fazına göre bit üreten fonksiyon.

   h : Şifreleme algoritması (XOR işlemi)

   σ_i: Üreticinin i zamandaki fazı

   σ_0: Üreticinin çalışmaya başlaması için belirlenen ilk faz. Gizlidir, genellikle anahtar ilk fazın ne olacağını belirler.

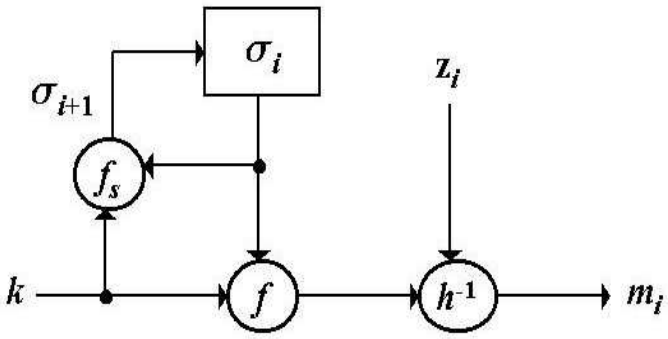
   Şifreleme :



Resim 1: Senkronize sistemlerde şifreleme

$$\begin{aligned} f(k,\sigma_i) &= z_i \\ f_s(k,\sigma_i) &= \sigma_{i+1} \\ h(m_i,z_i) &= c_i \end{aligned}$$

Deşifreleme :



Resim 2: Senkronize sistemlerde deşifreleme

$$\begin{aligned} f(k,\sigma_i) &= z_i \\ f_s(k,\sigma_i) &= \sigma_{i+1} \\ h^{-1}(c_i,z_i) &= m_i \end{aligned}$$

2. Oto-Senkronize (Self Synchronous) Akan Şifre :

f_s: Faz fonksiyonu, bir sonraki fazı belirleyen fonksiyon.

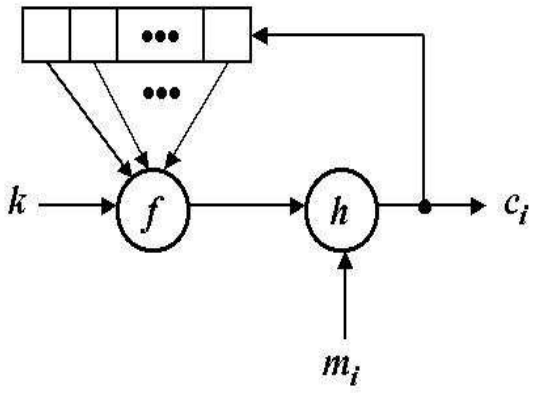
f : Üreticinin fazına göre bit üreten fonksiyon.

h : Şifreleme algoritması

σ_i: $(c_{i-t}, c_{i-t+1}, \ldots, c_{i-1})$

σ_0: $(c_{-t}, c_{-t+1}, \ldots, c_{-1})$

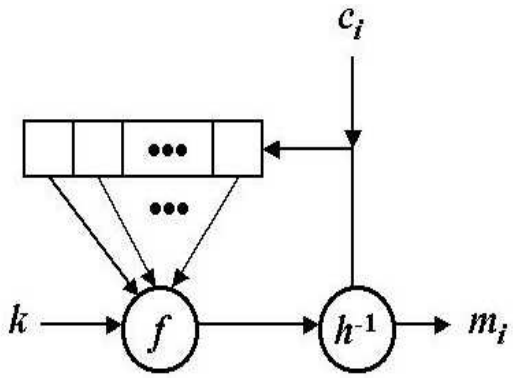
Şifreleme :



Resim 3: Oto-Senkronize sistemlerde şifreleme

$$\begin{aligned} f(k,\sigma_i) &= z_i \\ h(m_i,z_i) &= c_i \end{aligned}$$

Deşifreleme :

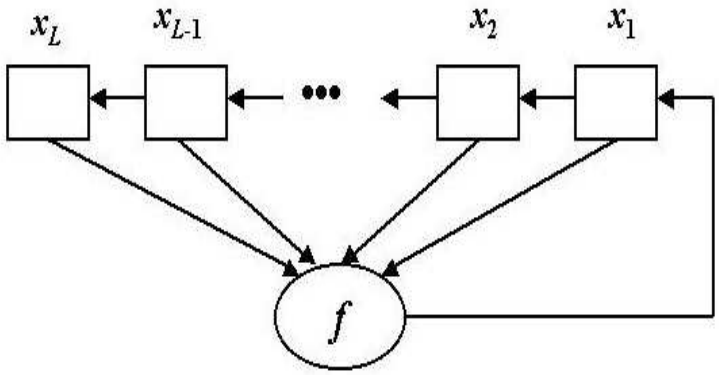


Resim 4: Oto-Senkronize sistemlerde deşifreleme

$$f(k, \sigma_i) = z_i$$
$$h^{-1}(c_i, z_i) = m_i$$

5.3 Geri Beslemeli Kaydırmalı Yazdırgaç (Feedback Shift Register)

Kısaca FSR olarak adlandırılan bir geri beslemeli kaydırmalı yazdırgaçın nasıl çalıŞtığı aŞağıdaki Şekilde gösterilmektedir.



Resim 5: Geri Beslemeli Kaydırmalı Yazdırgaç (FSR)

$f : \mathbb{Z}_2^n \to \mathbb{Z}_2$

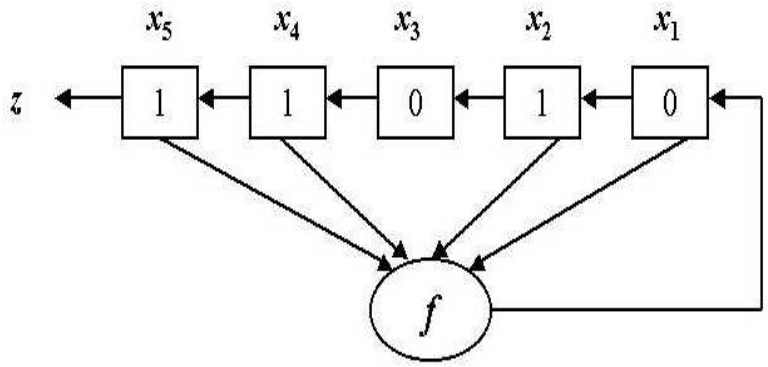
$\mathbb{Z}_2 = \{0, 1\}$ $\mathbb{Z}_2^n = \{(x_1, x_2, \ldots, x_n) \mid x_i \in \mathbb{Z}_2\}$

L: Yazdırgacın boyu

Bir sonraki fazda her bit sağa kayar. f fonksiyonu yeni bir bit üretir, üretilen bit en sağdaki göze yazılır.

$z = z_0 z_1 z_2 z_3 \ldots$ dizisinde her $i \in \mathbb{N}$ için $z_i = z_{i+p}$ eşitliğini sağlayan en küçük p sayısı dizinin periyodudur. Yani bu diziyi üreten FSR p adım sonra başlangıç fazına geri döner.

Örnek 5.3.1 $f(x_1, x_2, x_3, x_4, x_5) = x_1 \oplus x_2 x_3 \oplus x_4 x_5$



Resim 6: Fonksiyonu $f(x_1, x_2, x_3, x_4, x_5) = x_1 \oplus x_2x_3 \oplus x_4x_5$ olan FSR

$$\begin{aligned}
\sigma_0 &= 0\ 1\ 0\ 1\ 1 \qquad f(\sigma_0) = 1 \oplus 1.0 \oplus 1.0 = 1 \\
\sigma_1 &= 1\ 0\ 1\ 1\ 1 \qquad f(\sigma_1) = 1 \oplus 1.1 \oplus 0.1 = 0 \\
\sigma_2 &= 0\ 1\ 1\ 1\ 0 \qquad f(\sigma_2) = 0 \oplus 1.1 \oplus 1.0 = 1 \\
\sigma_3 &= 1\ 1\ 1\ 0\ 1 \qquad f(\sigma_3) = 1 \oplus 0.1 \oplus 1.1 = 0 \\
\sigma_4 &= 1\ 1\ 0\ 1\ 0 \qquad f(\sigma_4) = 0 \oplus 1.0 \oplus 1.1 = 1 \\
\sigma_5 &= 1\ 0\ 1\ 0\ 1 \qquad f(\sigma_5) = 1 \oplus 0.1 \oplus 0.1 = 1 \\
\sigma_6 = \sigma_0 &= 0\ 1\ 0\ 1\ 1
\end{aligned}$$

Böylece periyodu 6 olan $z = (010111)^\infty$ dizisi üretilir.

Örnek 5.3.2 Aynı fonksiyon kullanılarak periyodik olmayan bir dizi üretebiliriz. Eğer başlangıç fazını $\sigma_0 = (01010)$ seçersek:

	x_5	x_4	x_3	x_2	x_1
σ_0	0	1	0	1	0
σ_1	1	0	1	0	0
σ_2	0	1	0	0	0
σ_3	1	0	0	0	0
σ_4	0	0	0	0	0
σ_5	0	0	0	0	0

üretilen dizi $z = 0101000\ldots$, periyodik değildir.

Örnek 5.3.3 $f(x_1, x_2, x_3) = x_1 \oplus x_3$

	x_3	x_2	x_1
σ_0	1	0	1
σ_1	0	1	0
σ_2	1	0	0
σ_3	0	0	1
σ_4	0	1	1
σ_5	1	1	1
σ_6	1	1	0
$\sigma_0 = \sigma_7$	1	0	1

Periyodu 7 olan $z = (1010011)^\infty$ dizisi üretildi.

Bir f fonksiyonu, $a_i \in \{0, 1\}$ olmak üzere, $f(x_1, x_2, \ldots, x_L) = a_1x_1 \oplus a_2x_2 \oplus \ldots \oplus a_Lx_L$ şeklinde yazılabiliyorsa, bu fonksiyona doğrusal denir. FSR'nin kullandığı fonksiyon doğrusalsa bu üretece *doğrusal geri beslemeli kaydırmalı yazdırgaç* veya LFSR (linear feedback shift register) denir.

5.4 Üretecinin Sahip Olması Gereken Özellikler

1. Ürettiği dizi iyi istatistikler özellikler göstermelidir.
2. Periyodu büyük olan bir dizi üretmelidir.
3. Ürettiği dizinin doğrusal karmaşıklığı(linear complexity) büyük olmalıdır.

5.4.1 İstatistiksel Özellikler

Anahtar olarak kullanılacak dizinin tesadüfi ve kuralsız olması tercih edilir. Belirgin özellikleri olan bir dizi genellikle anahtar olarak kullanılmaz. Örneğin, $z = (10100100010000100000\ldots)$ dizisi kuralsız gözükmemektedir.

Dizinin kuralsız olmasını veya kuralsız bir diziye yakın olup olmadığını ölçen bir çok test vardır. AŞağıdaki üç testi LFSR'lar sağlar.

1. Dizinin bir tam periyodunda 1 ve 0 ların sayısı eşit olmalı, ya da aralarındaki fark 1 olmalıdır. Yani

$$0 \leq |(1\text{ 'lerin sayısı}) - (0\text{ 'ların sayısı})| \leq 1.$$

Örnek 5.4.1 $n = 21$ bit uzunluğunda $z = 000110101110010001101$ dizisinde 10 tane 1 ve 11 tane 0 vardır. Dolayısıyla aranan koşul sağlanır.

Örnek 5.4.2 $n = 14$ bit uzunluğunda ki $z = 10101010101010$ dizisinde
0 ve 1 yedi kere gözükürler ama bu dizi tesadüfi bir dizi değildir.

2. Tek çeşit karakterden oluşan bloklara *run* denir.

n bitlik bir dizide toplam $\frac{n+1}{2}$ tane run olması beklenir. Bunlardan

uzunluğu 1 olanların sayısının $\frac{n+1}{2^2}$,

uzunluğu 2 olanların sayısının $\frac{n+1}{2^3}$,

uzunluğu 3 olanların sayısının $\frac{n+1}{2^4}$,

$\vdots$

uzunluğu k olanların sayısının $\frac{n+1}{2^{k+1}}$ olması beklenir.

Örnek 5.4.3 $z = 00110001101$, n=11

$$\text{Beklenen run sayısı} \quad \frac{11+1}{2} = 6$$
$$\text{Uzunluğu 1 olan run sayısı} \quad \frac{11+1}{2^2} = 3$$

Dizinin run sayısı beklendiği gibi 6 dır, ancak uzunluğu 1 olan runların sayısı beklediği gibi 3 değil, 2 dir. Dolayısıyla, bu dizi run sayısı testini geçemez.

Örnek 5.4.4 $z = 10000101110110001111100110100 10$, n=31

Beklenen run sayısı	$\frac{31+1}{2} = 16$	✓
Uzunluğu 1 olan run sayısı	$\frac{31+1}{2^2} = 8$	✓
Uzunluğu 2 olan run sayısı	$\frac{31+1}{2^3} = 4$	✓
Uzunluğu 3 olan run sayısı	$\frac{31+1}{2^4} = 2$	✓
Uzunluğu 4 olan run sayısı	$\frac{31+1}{2^5} = 1$	✓
Uzunluğu 5 olan run sayısı	$\frac{31+1}{2^6} = 0.5 \cong 1$	✓

Bu dizi beklenen bütün değerleri sağlar ve run sayısı testinden geçer.

3. Periyodu p olan bir dizinin otokorelasyon fonksiyonu $C(\tau)$, dizinin kendisinin τ kadar kaydırılmasıyla oluŞan diziye ne kadar uyduğunu gösterir. Periyodu p olan $\{a_i\}$ dizisinin otokorelasyon fonksiyonunu

$$C(\tau) = \frac{1}{p}\sum_{i=1}^{p}(-1)^{a_i}(-1)^{a_{i+\tau}}$$

şeklinde ifade edilir. Her dizi için

$$C(0) = \frac{1}{p}\sum_{i=1}^{p}(-1)^{a_i}(-1)^{a_i} = \frac{1}{p}\sum_{i=1}^{p}(-1)^{a_i \oplus a_i} = \frac{1}{p}\sum_{i=1}^{p}1 = 1 \text{ dir.}$$

Bir dizinin rastgele olup olmadığına karar vermekte aranılan bir başka koşul da $C(\tau)$ fonksiyonunun

$C(1) = C(2) = \ldots = C(p-1)$ eşitliğini sağlamasıdır.

Örnek 5.4.5

$$\begin{aligned} a_i &= 0\,0\,0\,1\,0\,1\,1 \qquad C(0)=1,\ p=7 \\ (-1)^{a_i} &= 1\,1\,1\,-1\,1\,-1\,-1 \end{aligned}$$

$$\begin{array}{rcccccccc}
(-1)^{a_{i+1}} &=& 1 & 1 & 1 & 1 & -1 & -1 & 1 \\
(-1)^{a_{i+2}} &=& 1 & -1 & 1 & -1 & -1 & 1 & 1 \\
(-1)^{a_{i+3}} &=& -1 & 1 & -1 & -1 & 1 & 1 & 1 \\
(-1)^{a_{i+4}} &=& 1 & -1 & -1 & 1 & 1 & 1 & -1 \\
(-1)^{a_{i+5}} &=& -1 & 1 & 1 & 1 & 1 & -1 & 1 \\
(-1)^{a_{i+6}} &=& -1 & 1 & 1 & 1 & -1 & 1 & -1
\end{array}$$

$C(1)=\frac{1}{7}(1+1-1-1-1+1-1)=\frac{-1}{7}$

$C(2)=\frac{1}{7}(1-1-1-1+1-1+1)=\frac{-1}{7}$

$C(3)=\frac{1}{7}(-1+1-1+1+1-1-1)=\frac{-1}{7}$

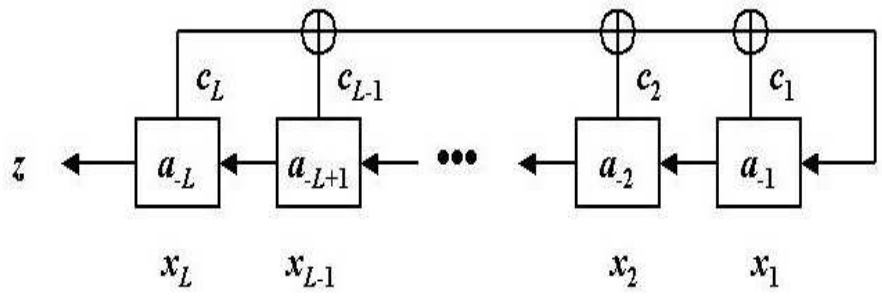
$C(4)=\frac{1}{7}(1-1-1-1+1-1+1)=\frac{-1}{7}$

$C(5)=\frac{1}{7}(1+1-1-1-1+1-1)=\frac{-1}{7}$

$C(6)=\frac{1}{7}(1+1-1-1-1+1-1)=\frac{-1}{7}$

$C(1)=C(2)=C(3)=C(4)=C(5)=C(6)=\frac{-1}{7}$ olduğu için bu dizi testten geçer.

5.5 Doğrusal Geri Beslemeli Kaydırmalı Yazdırgaç (LFSR)



Resim 7: LFSR

L: LFSR'ın boyu

Başlangıç fazı $\sigma_0 :\ a_{-L}, a_{-L+1}, \ldots, a_{-2}, a_{-1}$

Geri besleme katsayıları (feedback coefficients): $c_1, c_2, \ldots, c_{L-1}, c_L \in \mathbb{Z}_2 = \{0, 1\}$

LFSR'ın doğrusal fonksiyonu:

$$f(x_1, x_2 \ldots x_L) = c_1 x_1 \oplus c_2 x_2 \oplus \ldots \oplus c_L x_L$$

Buna göre:

$$a_0 = c_L a_{-L} \oplus c_{L-1} a_{-L+1} \oplus \ldots c_2 a_{-2} \oplus c_1 a_{-1}$$

$$\Rightarrow \sigma_1 :\ a_{-L+1}, a_{-L+2}, \ldots, a_{-1}, a_0$$

$$a_1 = c_L a_{-L+1} \oplus c_{L-1} a_{-L+2} \oplus \ldots c_2 a_{-1} \oplus c_1 a_0$$

Genel olarak:

$$a_n = c_L a_{n-L} \oplus c_{L-1} a_{n-L+1} \oplus \ldots c_2 a_{n-2} \oplus c_1 a_{n-1}.$$

Bu recursive bağıntının karakteristik polinomu aynı zamanda LFSR'ın karakteristik polinomudur. Buna göre karakteristik polinom:

$$m(x) = x^L + c_1 x^{L-1} + c_2 x^{L-2} \ldots + c_{L-1} x + c_L.$$

LFSR'ın bağlayıcı polinomu (connection polynomial):

$$C(D) = 1 + c_1 D + c_2 D^2 + \ldots + c_{L-1} D^{L-1} + c_L D^L.$$

Bağlayıcı polinom ile karakteristik polinom arasındaki bağıntı aşağıdaki gibidir:

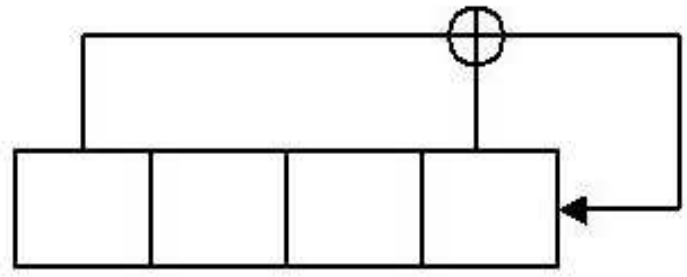
$$m(x) = x^L C\left(\frac{1}{x}\right).$$

Bir LFSR, boyu L ve bağlayıcı polinomu $C(D)$ ile belirlenir:

$$LFSR = \langle L, C(D) \rangle.$$

Örnek 5.5.1 LFSR=$\langle 4, C(D) = 1 + D + D^4 \rangle$

$c_1 = 1, \quad c_2 = c_3 = 0, \quad c_4 = 1 \;\Rightarrow f(x_1, x_2, x_3, x_4) = x_1 \oplus x_4$



Resim 8: $\langle 4, C(D) = 1 + D + D^4 \rangle$

Bu LFSR'ı çalıştırmak için başlangıç fazı olarak $\sigma_0 = (0\ 0\ 1\ 1)$ alınırsa, periyodu 5 olan $z = (001111010110010)^\infty$ dizisi üretilir.

x_1	x_2	x_3	x_4	$x_1 \oplus x_4$
0	0	1	1	1
0	1	1	1	1
1	1	1	1	0
1	1	1	0	1
1	1	0	1	0
1	0	1	0	1
0	1	0	1	1
1	0	1	1	0
0	1	1	0	0
1	1	0	0	1
1	0	0	1	0
0	0	1	0	0
0	1	0	0	0
1	0	0	0	1
0	0	0	1	1
0	0	1	1	$= \sigma_0$

LFSR'ın fonksiyonu doğrusal olduğundan $f(0, 0, \ldots, 0) = 0$ dır. Dolayısıyla başlangıç fazını 0 vektörü alınırsa 0 geri beslenir. 0 vektöründen başka bir faz görülmez. 0'dan farklı bir vektörle başlanırsa 0 vektörü faz olarak hiç görülmez.

5.5.1 Dizinin Periyodu

Boyu L olan bir LFRS'ın ürettiği dizinin periyodu en fazla $2^L - 1$ olabilir çünkü LFSR'da faz olarak L uzunluğunda vektörler gözükür. L uzunluğunda $2^L - 1$ tane vektör vardır. LFSR da 0 vektörünü görülmezse en fazla $2^L - 1$ tane değişik vektör görülebilir. Yani LFSR en fazla $2^L - 1$ adım sonra başlangıç noktasına geri döner.

$\langle L, C(D)\rangle$ LFSR'ının ürettiği dizinin periyodu $C(D)$ polinomunun çarpanlarına ayrılabilir olup olmamasıyla ve başlangıç fazıyla iliŞkilidir.

- Eğer $C(D)$ çarpanlarına ayrılıyorsa üretilen dizinin periyodu başlangıç fazına göre değişir.

  Örnek 5.5.2 $\langle L, C(D) = 1 + D^2 + D^4\rangle$

  $\Rightarrow \quad c_1 = 0, c_2 = 1, c_3 = 0, c_4 = 1 \quad \Rightarrow \quad f(x_1, x_2, x_3, x_4) = x_2 \oplus x_4$ ve $C(D) = 1 + D^2 + D^4 = (1 + D + D^2)^2$

	x_1	x_2	x_3	x_4
σ_0	1	0	0	0
	0	0	0	1
	0	0	1	0
	0	1	0	1
	1	0	1	0
	0	1	0	0
σ_0	1	0	0	0

Periyod = 6

	x_1	x_2	x_3	x_4
σ_0	1	1	1	1
	1	1	1	0
	1	1	0	0
	1	0	0	1
	0	0	1	1
	0	1	1	1
σ_0	1	1	1	1

Periyod = 6

	x_1	x_2	x_3	x_4
σ_0	1	0	1	1
	0	1	1	0
	1	1	0	1
σ_0	1	0	1	1

Periyod = 3

- Maksimum periyodda dizi üretmek için $C(D)$ polinomunun çarpanlarına ayrılamaz olması gerekir. Eğer $C(D)$ çarpanlarına ayrılamayan bir polinomsa dizinin periyodu başlangıç fazına bağlı değildir ve $C(D)$ polinomunun böldüğü $1+D^p$ polinomlarından en küçük dereceli olanın derecesi, üretilen dizinin periyoduna eşittir. Buradaki p sayısı $2^L - 1$ sayısının bir bölenidir. Örneğin $C(D) \mid 1 + D^5$ ve $C(D)$, $1 + D^k$, $k = 1, 2, 3, 4$, polinomlarını bölmüyorsa üretilen dizinin periyodu 5 tir.

- Dizinin periyodunun maksimum yani $2^L - 1$ olması için $C(D)$ polinomunun böldüğü

en küçük dereceli polinom $1 + D^{2^L-1}$ olmalıdır. Bunu sağlayan $C(D)$ polinomuna *ilkel polinom*(primitive polynomial) denir.

Maksimum periyodda dizi üreten bir LFSR'a *maksimum uzunlukta LFSR* denir.

5.6 Doğrusal Karmaşıklık (Linear Complexity)

Bir dizinin *doğrusal karmaşıklığı* (L.C.) onu üretebilecek LFSR'lardan en kısa olanın boyuna eşittir.

$z = 0000\ldots0\ldots$ dizisinin doğrusal karmaşıklığı 0 'dır.

$z = 1111\ldots1\ldots$ dizisinin doğrusal karmaşıklığı 1 'dir.

$z = \underbrace{0000\ldots1}_{n}$ dizisinin doğrusal karmaşıklığı n 'dir.

z	L.C	$C(D)$
0	0	0
1	1	$1+D$
01	2	$1+D^2$
001	3	$1+D^3$
011	2	$1+D+D^2$
100	1	1
101	2	$1+D^2$
110	2	$1+D+D^2$
111	1	$1+D$

Doğrusal karmaşıklık ile ilgili bazı özellikler:

- $z = z_0z_1z_2z_3\ldots$ dizisi için, bu diziden alınan her n bitlik z^n dizisinin doğrusal karmaşıklığı en fazla n olabilir. Yani $0 \leq \text{L.C.}(z^n) \leq n$ dir. Bu nedenle bir dizinin

doğrusal karmaşıklığı en fazla dizinin boyu kadardır.

- Eğer dizinin periyodu N ise L.C.$(z) \leq N$ dir.
- s ve t birer dizi olmak üzere L.C.$(s \oplus t) \leq$ L.C.(s) + L.C.(t) dir.

5.6.1 Doğrusal Karmaşıklık Profili (Linear Complexity Profile)

$s = s_0 s_1 s_2 \ldots$ bir dizi olsun. $N \geq 1$ için s^N sonlu dizisi $s^N = s_0 s_1 s_2 \ldots s_{N-1}$ şeklinde tanımlanır. O zaman her $N \geq 1$ için $L_N =$ L.C.(s^N) şeklinde tanımlanan $L_1, L_2, \ldots, L_N$ dizisine s^N dizisinin *doğrusal karmaşıklık profili* denir. Bu dizi aşağıdaki özellikleri gösterir:

- Eğer $j \geq i$ ise $L_j \geq L_i$ dir.
- $L_{N+1} > L_N$ olması için $L_N \leq N/2$ olması gerekir.
- Eğer $L_{N+1} > L_N$ ise $L_{N+1} + L_N = N + 1$ dir.

Profilde önemli noktalar $L_N \leq N/2$ olduğu yerlerdir. Bu noktalarda L_{N+1} artabilir. Boyu L_N olan hiç bir LFSR s^{N+1} dizisini üretmezse $L_{N+1} > L_N$ dir, dolayısıyla $L_{N+1} = L_N - N - 1$ dir.

Kriptografik anlamda iyi bir LFSR'ın ürettiği bir dizinin, doğrusal karmaşıklık profilinin grafiği $y = N/2$ doğrusuna yakın olmalıdır. Bu doğrudan fazla sapmamalıdır.

5.6.2 Berleckamp Massey Algoritması

Berleckamp Massey Algoritması sonlu bir dizinin doğrusal karmaşıklığını ve onu üretebilecek en kısa LFSR'ın bağlayıcı polinomu $C(D)$ yi bulmak için kullanılır.
$s^{N+1} = s_0 s_1 \dots s_{N-1} s_N$ dizisi verilsin. $\langle K, C(D) \rangle$ LFSR'ı $s^N = s_0 s_1 \dots s_{N-1}$ dizisini üretsin. s^{N+1} dizisinin ve $\langle K, C(D) \rangle$ LFSR'ının ürettiği dizinin $(N+1)$. terimi arasındaki farka *uymazlık sayısı* (next discrepancy) denir. Bu sayı

$$d_N = s_N + \sum_{i=1}^{L} c_i s_{N-i} \pmod 2$$

şeklinde hesaplanır. $d_N = 0$ olması için bu LFSR'ın s^{N+1} dizisinin $(N+1)$. terimini üretmesi gerekir.

Berleckamp Massey Algoritmasının işleyişi aşağıdaki gibidir:
Girdi olarak $s^n = s_0 s_1 s_2 \dots s_{n-1}$ dizisini alır.

1. BaŞlangıç konumu: $C(D) = 1,\ L = 0,\ m = -1,\ B(D) = 1,\ N = 0$
2. $N < n$ durumunda
   (a) $d = s_N + \sum_{i=1}^{L} c_i s_{N-i} \pmod 2$.
   (b) $d = 1$ ise
   $T(D) \longleftarrow C(D)$
   $C(D) \longleftarrow C(D) + B(D) D^{N-m}$
   Eğer $L \leq N/2$ ise $L \longleftarrow N + 1 - L$

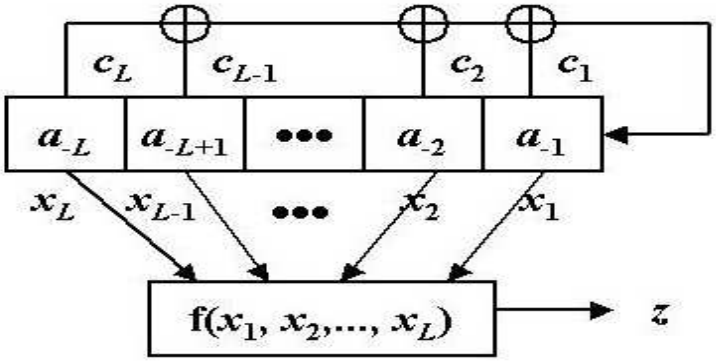
$$m \longleftarrow N$$
$$B(D) \longleftarrow T(D)$$

(c) $N \longleftarrow N + 1$

5.7 LFSR Kullanılarak Yapılan Akan Şifre Sistemleri

Bir dizinin doğrusal karmaşıklığı en fazla onu üreten LFSR'ın boyu kadar olabilir. Maksimum uzunlukta bir LFSR, doğrusal karmaşıklığı en fazla kendi boyuna eşit bir dizi üretebilir. Bu da doğrusal karmaşıklık için küçük bir sayıdır. Dizinin doğrusal karmaşıklığını arttırmak için çeşitli yollar vardır:

- LFSR'a doğrusal olmayan bir filtre bağlanır. Bu filtre doğrusal olmayan yani derecesi en az iki olan bir fonksiyondur.



Resim 9: Doğrusal olmayan filtre

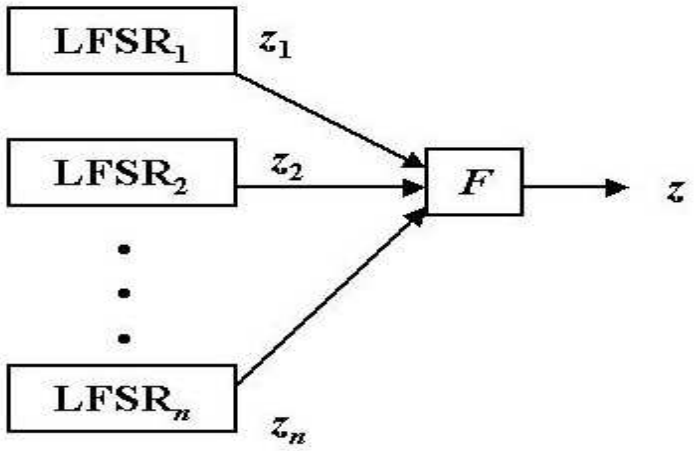
Doğrusal olmayan bir filtre: $f(x_L, x_{L-1}, \ldots, x_2, x_1) : \mathbb{Z}_2^L \to \mathbb{Z}_2$

Örnek 5.7.1 $\langle 3, 1 + D + D^2 \rangle$ ve doğrusal olmayan filtre $f(x_3, x_2, x_1) = x_1 \oplus x_1x_2 \oplus x_2x_3$

	x_3	x_2	x_1	$f(x_3, x_2, x_1)$
σ_0	1	0	1	1
	0	1	0	0
	1	0	0	0
	0	0	1	1
	0	1	1	0
	1	1	1	1
	1	1	0	1
σ_0	1	0	1	

$z = (1001011)^\infty$

- Birden fazla LFSR doğrusal olmayan bir fonksiyonla bağlanabilir.



Resim 10: Birden fazla LFSR'ın doğrusal olmayan bir fonksiyonla bağlanması

$F : (z_1, z_2, \ldots, z_n) : \mathbb{Z}_2^n \to \mathbb{Z}_2$ doğrusal olmayan bir fonksiyon.

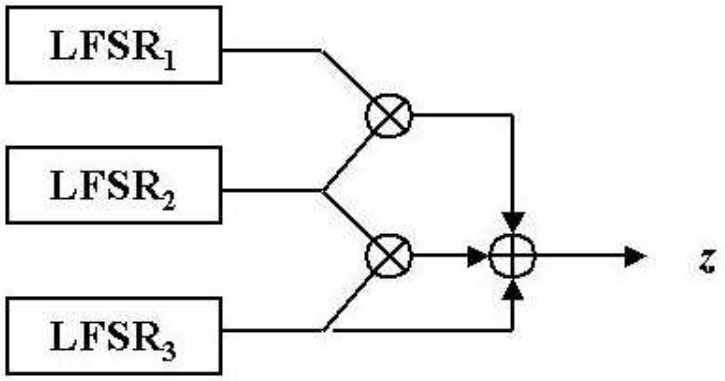
z dizisinin periyodu: $T(z) = \text{okek}(T_1, T_2, \ldots, T_n)$ dir.

Eğer LFSR'lar maksimum uzunlukta ise ve ürettikleri dizilerin periyodu ikiden büyük ve birbirlerinden farklı ise z dizisinin doğrusal karmaşıklığı

$F(L_1, L_2, \ldots, L_n)$ dir.

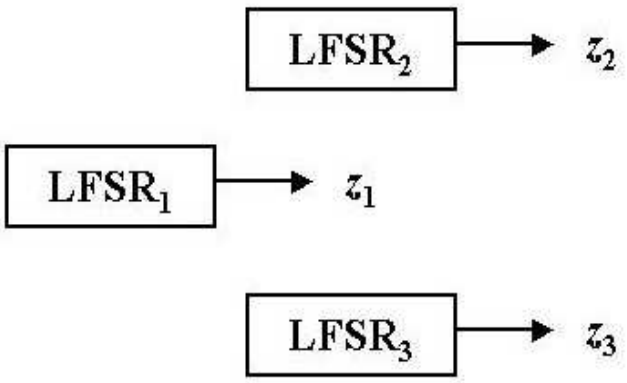
Örnek 5.7.2 *Geffe Üreteci:* Üç tane LFSR kullanır.

$$F(x_1, x_2, x_3) = x_1x_2 \oplus (1 \oplus x_2)x_3.$$



Resim 11: Geffe Üreteci

- Saat Kontrollü Üreteçler (Clock Controlled Generators):
  - Değişen Adımlı Üreteç (*Alternating Step Generator*): Üç tane LFSR kullanılır.



Resim 12: Değişen Adımlı Üreteç

$\mathrm{LFSR}_1 = \langle L_1, C_1(D) \rangle$

$\mathrm{LFSR}_2 = \langle L_2, C_2(D) \rangle$

$\mathrm{LFSR}_3 = \langle L_3, C_3(D) \rangle$

LFSR_1 çalıştırılır;

$x_1 = 1$ ise LFSR_2 çalıştırılır. LFSR_3'ün bir önce ürettiği bit tekrar eder, LFSR_3 daha önce çalışmamışsa 0 alınır.

$x_1 = 0$ ise LFSR_3 çalıştırılır. LFSR_2'nin bir önce ürettiği bit tekrar eder, LFSR_2 daha önce çalışmamışsa 0 alınır.

LFSR_2 ve LFSR_3 XOR işlemine tabi tutulur. Eğer LFSR_1 periyodu 2^{L_1} olan bir dizi üretiyorsa LFSR_2 ve LFSR_3 maksimum periyodda diziler üretiyorsa ve o.b.e.b.(L_1, L_2) ise üretilecek z dizisinin

1. periyodu $2^{L_1}(2^{L_2} - 1)(2^{L_3} - 1)$,
2. doğrusal karmaşıklığı $(L_2 + L_3)2^{L_1 - 1} < \mathrm{L.C.}(z) \leq (L_2 + L_3)2^{L_1}$ dir.

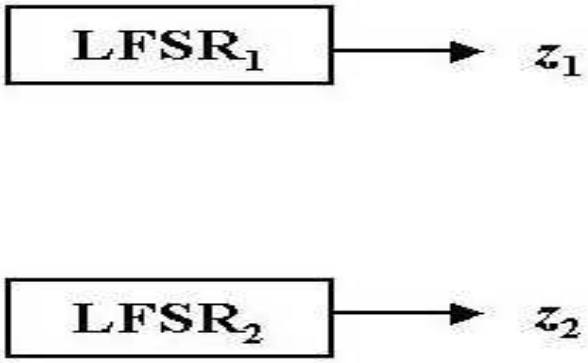
Örnek 5.7.3 $\text{LFSR}_2 \rightarrow 1,1,0,0,1,1,0,1,\ldots$

$\text{LFSR}_1 \rightarrow 0,1,1,1,0,0,1,1,\ldots$

$\text{LFSR}_3 \rightarrow 0,0,1,1,0,1,1,0,\ldots$

x_2	0	1	1	0	0	0	0
x_1	$0\downarrow$	$1\uparrow$	$1\uparrow$	$1\uparrow$	$0\downarrow$	$0\downarrow$	$1\uparrow$
x_3	0	0	0	0	0	1	1
z	0	1	1	0	0	1	1

– Küçülen Üreteç (*Shrinking Generator*): İki tane LFSR kullanılır. İkisi de aynı anda çalışır.



Resim 13: Küçülen Üreteç

$x = 1$ ise y 'den al.

$x = 0$ ise y 'den alma.

$\text{LFSR}_1 = \langle L_1, C_1(D) \rangle \;\Rightarrow\; T(x) = 2^{L_1} - 1$

$\text{LFSR}_2 = \langle L_2, C_2(D) \rangle \;\Rightarrow\; T(y) = 2^{L_2} - 1$

1. L_1 ve L_2 aralarında asal ise oluşan dizinin periyodu
   $(2^{L_1-1})2^{L_2}-1$,
2. doğrusal karmaşıklığı ise $L_2(2^{L_1-2}) < \text{L.C.}(z) \leq L_2(2^{L_1-1})$ dir.

Örnek 5.7.4 $\text{LFSR}_1 \to (101)^\infty$

$\text{LFSR}_2 \to (0101101)^\infty$

x	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1
y	0	1	0	1	1	0	1	0	1	0	1	1	0	1	0	1	0	1	1	0	1
z	0		0	1		0	1		1	0		1	0		0	1		1	1		1

$z = (00101101101111)^\infty$

BÖLÜM 6

SAYILAR TEORİSİ

6.1 Tamsayılar

Tamsayılar kümesi $\{\cdots, -3, -2, -1, 0, 1, 2, 3, \cdots\}$ sayılarından oluşur ve Z sembolü ile gösterilir.

6.1.1 Bölünebilirlik:

a ve b verilen tamsayılar olsun. Eğer $b = a \cdot d$ eşitliğini sağlayan bir d sayısı varsa a b'yi böler(b, a tarafından bölünür ya da da a, b nin bir çarpanı) denir ve $a|b$ şeklinde gösterilir.

Her $b > 1$ tamsayısı en azından iki pozitif bölene sahiptir; bunlar 1 ve b dir.

Örnek 6.1.1 *1.* $-5|15$*, çünkü* $15 = 5 \cdot 3$.

2. $256|0$*, çünkü* $0 = 256 \cdot 0$.

3. $16|48$*, çünkü* $48 = 16 \cdot 3$.

6.1.2 Bölünebilirlik Özellikleri

Bütün $a, b, c \in Z$ için, aşağıdakiler doğrudur.

1. $a|a$.

2. Eğer $a|b$ ve $b|c$ ise $a|c$.

3. Eğer $a|b$ ve $a|c$ ise bütün $x, y \in Z$ için $a|bx + cy$ ifadesi doğrudur.

4. Eğer $a|b$ ve $b|a$ ise $a = \pm b$.

6.1.3 Tamsayılar için Bölüm Algoritması:

Eğer a ve b, $b \geq 1$ olmak koşulu ile, tamsayılar ise a'nın b'ye bölümü q tamsayısı gibi bir bölüm ve r tamsayısı gibi bir kalan verir.

$$a = qb + r,\ 0 \leq r < b.$$

Üstelik q ve r tektir. Bu bölümün kalanı $a \, mod \, b$ olarak gösterilir.

Örnek 6.1.2 *Eğer* $a = 73$, $b = 17$ *ise* $q = 4$ *ve* $r = 5$ *tir. Böylece* $73 \, mod \, 17 \equiv 5$ *tir.*

6.1.4 En Büyük Ortak Bölen (Greatest Common Divisor)

a ve b her ikisi birlikte 0 olmamak koşulu ile iki tamsayı olsun. a ve b nin en büyük ortak böleni, a ve b yi bölen en büyük d tamsayısıdır. a ve b nin en büyük ortak böleni $gcd(a, b)$ ya da kısaca (a, b) ile gösterilir.

Örnek 6.1.3 *1.* $gcd(7, 11) = 1$, *çünkü* $7 = 7 \cdot 1$, $11 = 11 \cdot 1$

2. $gcd(48, 40) = 8$*, çünkü* $48 = 2^4 \cdot 3$*,* $40 = 2^3 \cdot 5$

6.1.5 En Küçük Ortak Kat (Least common Multiple)

a ve b her ikisi birlikte 0 olmamak koşulu ile iki tamsayı olsun. a ve b nin en küçük ortak katı a ve b nin her ikisinin de böldüğü en küçük tamsayıdır ve $lcm(a, b)$ ile gösterilir.

Örnek 6.1.4 $lcm(8, 12) = 24$ *çünkü* $8 = 2^3$ *ve* $12 = 2^2 \cdot 3$ *tür.*

Teorem 6.1.5 *a ve b her ikiside birlikte* 0 *olmayacak şekilde tamsayılar olsun.* $gcd(a, b) = ax + by$ *eşitliğini sağlayan* x *ve* y *tamsayılar her zaman vardır.*

6.1.6 Asal Sayı

1 den büyük, 1 ve kendisinden başka böleni olmayan tamsayılara asal sayı denir.Asal olmayan sayılara da *bölünebilir sayı* denir.

Örnek 6.1.6 $2, 3, 5, 7, 11, 13, 17, 19, 23, \cdots$ *sayıları bazı asal sayılara örnektir.*

NOT: Asal sayılarla ilgili bazı özellikler:

- Eğer p sayısı asal sayı ve $p|ab$ ise $p|a$'yi veya $p|b$'yi böler.
- Sonsuz sayıda asal sayı vardır.

6.1.7 Aralarında Asal Sayı

a ve b iki tamsayısı $gcd(a, b) = 1$ koşulunu sağlıyorsa bu sayılara *aralarında asal* denir. $gcd(12, 5) = 1$ olduğu için 2 ve 5 sayıları aralarında asaldır.

6.1.8 Aritmetiğin Esas Teoremi

$n \geq 2$ olan her tamsayı asal sayıların çarpımları şeklinde tek olarak yazılır. Yani,

$$n = p_1^{e_1} \cdot p_2^{e_2} \cdots p_k^{e_k}$$

sayısında p_k lar farklı asal sayıları e_k lar da pozitif tamsayıları göstermektedir.

Örnek 6.1.7 $4200 = 2^3 \cdot 3 \cdot 5^2 \cdot 7$.

6.1.9 Öklid Algoritması(Euclidean Algorithm)

a ve b şeklinde olan iki tamsayının en büyük ortak bölenini aritmetiğin esas teoreminde bahsedildiği gibi çarpanlarına ayırarak ve ortak carpanların en büyüğünü alarak bulabiliriz. Eğer a ve b büyük sayılarsa bunların asal çarpanlarını bulmak zor olur; bunun sonucunda da en büyük ortak böleni bulmak da zorlaşır. Sayılar teorisinin önemli bir araştırma alanı da büyük tamsayıları daha çabuk çarpanlarına ayırma üzerine araştırmadır. Eğer a ve b nin asal çarpanları bilinmiyorsa, $gcd(a, b)$ yi bulmak için çabuk bir yol vardır. O da Öklid algoritmasıdır.

Öklid Algoritması şöyle çalışır.

- $a > b$ olmak üzere, a, b 'ye bölünür. Bölüm q_1, kalan r_1 olsun

$$a = b \cdot q_1 + r_1$$

- İkinci bölme işlemi gerçekleştirilir. b, r_1 'e bölünür ve bölüm q_2, kalan ise r_2 olur.

$$b = q_2 \cdot r_1 + r_2$$

- Üçüncü olarak r_1, r_2 'ye bölünür ve bölüm q_3, kalan ise r_3 olur.

$$r_1 = q_3 \cdot r_2 + r_3$$

$\vdots$

- Son olarak r_{n-1}, r_n 'e bölünür ve bölüm q_{n+1}, kalan ise $r_{n+1} = 0$ olur.

$$r_{n-1} = q_{n+1} \cdot r_n + r_{n+1}$$

- $r_{n+1} = 0$ olduğu için r_n değeri a ve b tamsayılarının en büyük ortak böleni olur. Yani $gcd(a, b) = r_n$ dir.

Bu algoritmadaki işlemler sonsuza kadar gitmez, çünkü 0 ile a tamsayısı arasında sonlu sayıda tamsayı vardır.

Örnek 6.1.8 • *$gcd(24, 138)$ 'in sonucu kaçtır? $gcd(24, 138) = ax + by$ ifadesinde x ve y sayıları kaç olur?*

$$\begin{aligned} 138 &= 5 \cdot 24 + 18 \\ 24 &= 1 \cdot 18 + 6 \quad \textit{ise} \quad gcd(24, 138) = 6 \textit{ dır.} \\ 18 &= 3 \cdot 6 + 0 \end{aligned}$$

x ve y aşağıdaki şekilde bulunur. $gcd(24, 138) = 6$

$$\begin{aligned} 6 &= 24 - 1 \cdot 18 &&= 24 - 1 \cdot (138 - 5 \cdot 24) \\ &= 24 - 1 \cdot 138 + 5 \cdot 24 &&= 6 \cdot 24 - 1 \cdot 138 \end{aligned}$$

Yani,

$6 = 6 \cdot 24 + (-1) \cdot 138$. *Buradan* $x = 6$ *ve* $y = -1$ *bulunur.*

- $gcd(1547, 560)$ *'ın sonucu kaçtır?* $gcd(1547, 560) = ax+by$ *ifadesinde* x *ve* y *sayıları kaç olur?*

$$\begin{aligned} 1547 &= 2 \cdot 560 + 427 \\ 560 &= 1 \cdot 427 + 133 \quad \textit{ise} \quad gcd(1547, 560) = 7 \textit{ dir.} \\ 427 &= 3 \cdot 133 + 28 \\ 133 &= 4 \cdot 28 + 21 \\ 28 &= 1 \cdot 21 + 7 \\ 21 &= 3 \cdot 7 + 0 \end{aligned}$$

x ve y aşağıdaki şekilde bulunur. $gcd(1547, 560) = 7$

$$\begin{aligned} 7 &= 28 - 1 \cdot 21 \\ &= 28 - 1 \cdot (133 - 4 \cdot 28) &&= 28 - 1 \cdot 133 + 4 \cdot 28 \\ &= 5 \cdot 28 - 1 \cdot 133 &&= 5 \cdot (427 - 3 \cdot 133) - 1 \cdot 133 \\ &= 5 \cdot 427 - 15 \cdot 133 - 1 \cdot 133 &&= 5 \cdot 427 - 16 \cdot 133 \\ &= 5 \cdot 427 - 16 \cdot (560 - 1 \cdot 427) &&= 5 \cdot 427 - 16 \cdot 560 + 16 \cdot 427 \\ &= 21 \cdot 427 - 16 \cdot 560 &&= 21 \cdot (1547 - 2 \cdot 560) - 16 \cdot 560 \\ &= 21 \cdot 1547 - 42 \cdot 560 - 16 \cdot 560 &&= 21 \cdot 1547 - 58 \cdot 560 \end{aligned}$$

Yani,

$7 = 21 \cdot 1547 + (-58) \cdot y$. *Buradan* $x = 21$ *ve* $y = -58$ *bulunur.*

6.2 Asal Sayılar

Tanım 6.2.1 1 *den büyük olan,* 1 *ve kendisinden başka böleni olmayan sayılara asal sayı denir.*

Soru: Verilen bir tamsayının asal sayı olup olmadığı nasıl anlaşılır?

6.2.1 Eratosthenes Kalburu(The Sieve of Eratosthenes)

Verilen bir tamsayının asal sayı olup olmadığı Eratosthenes Kalburu metodu ile bulunabilir. Verilen tamsayı kendisinden önce gelen her pozitif tamsayıyla bölünür. Eğer hiç bir sayıya bölünemiyor ise bu sayıya asal sayı denir.

Metod:

$a > 1$ bir tamsayı olsun.Bu sayı eğer bölünebilir bir sayı ise $1 < b < a\,, 1 < c < a$ olmak üzere $a = b \cdot c$ şeklinde yazılabilir.Bütünlüğü bozmadan $b \leq c$ olduğu farzedilsin. O zaman,

$$b^2 \leq b \cdot c = a \Rightarrow b \leq \sqrt{a}$$

Aritmetiğin esas teoremini kullanarak b yi bölen ve $p \leq b \leq \sqrt{a}$ koşulunu sağlayan bir p sayısı bulunur. Öyle ki bu p sayısı b yi böldüğü ve b de a yı böldüğü için p a yı da bölmüş

olur.

Örnek 6.2.2

- $a = 173$. *a asal mıdır?* $13 < \sqrt{173} < 14$. 173 *sayısını bölebilecek asal sayılar* $2, 3, 5, 7, 9, 11, 13$ *olabilir. Bu sayıların* 173 *ü bölüp bölmediği kontrol edilir. Hiç birisi* 173 *sayısını bölmediği için sayı asal sayıdır.*

- 701 *ve*1009 *sayıları asal mıdır?* $26 < 701 < 27$. 701 *sayısını bölebilecek asal sayılar* $2, 3, 5, 7, 9, 11, 13, 17, 19, 23$ *olabilir. Bu sayıların* 173 *ü bölüp bölmediği kontrol edilir. Hiç birisi* 701 *sayısını bölmediği için sayı asal sayıdır.*

  $31 < 1009 < 32$. 1009 *sayısını bölebilecek asal sayılar* $2, 3, 5, 7, 9, 11, 13, 17, 19, 23, 29, 31$ *olabilir. Bu sayıların* 1009 *ü bölüp bölmediği kontrol edilir. Hiç birisi* 1009 *sayısını bölmediği için sayı asal sayıdır.*

6.3 Eratosthenes Metodu (Method of Eratosthenes)

Bu metod, verilen bir tamsayının altında kalan bütün asal sayıları bulmak için kullanılır. Öncelikle 2 den n ye kadar olan tamsayılar sırasıyla yazılır ve $\sqrt{n}$ den küçük ve eşit olan asalların çarpanları $(2p, 3p, \cdots)$ elimine edilir. Listede geri kalan sayılar asal sayıları gösterir.

Örnek 6.3.1 49 *u aşmayan bütün asalları bulunuz.*

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	

$\sqrt{49} = 7$. *Böylelikle cevap:*$2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47$ *asal sayılarıdır.*

6.4 Denklik Teorisi(Theory of Congruence (Modularity))

Tanım 6.4.1 *n sabit pozitif bir tamsayı olsun. Eğer $n|a-b$ ya da bir k tamsayısı için $a-b=nk$ eşitliği sağlanırsa, a b ye mod n e göre denktir denir ve $a \equiv b$ mod n ile gösterilir.*

Örnek 6.4.2

- $1 \equiv 5$ *mod* 4, *çünkü* $1-5=-4$ *ve* $4|-4$.
- $-2 \equiv 9$ *mod* 11, *çünkü* $-2-9=-11$ *ve* $11|-11$.

6.4.1 Teoremler:

1. $a \equiv b \bmod n \Leftrightarrow a = bk + n$ esitliğini sağlayan bir k vardır.
2. Her tamsayı mod n ye göre $0, 1, 2, \cdots, n-1$ sayılarından sadece birine denktir.
3. $a \equiv b \bmod n \Leftrightarrow a$ ve b, n ile bölündüğünde aynı kalanı verir.
4. $a \equiv a \bmod n$

5. $a \equiv b \bmod n \Rightarrow b \equiv a \bmod n$

6. $a \equiv b \bmod n$ ve $b \equiv c \bmod n \Rightarrow a \equiv c \bmod n$

7. $a \equiv b \bmod n$ ve $c \equiv d \bmod n \Rightarrow a + c \equiv b + d \bmod n$

8. $a \equiv b \bmod n \Rightarrow a^k \equiv b^k \bmod n$

6.4.2 Aritmetik Tersi

$a \neq 0$ herhangi bir tamsayı olmak üzere,eğer $a \cdot a^* \equiv 1 \bmod n$ denkliğini sağlayan bir a^* tamsayısı var ise bu a^* sayısına a nın mod n ye göre aritmetik tersi denir.

Teorem 6.4.3 *Eğer* $gcd(a, n) = 1 \Rightarrow a$ *nın aritmetik tersi vardır.*

Örnek 6.4.4 $gcd(4, 9) = 1$ *çünkü Öklid algoritmasına göre,* $9 = 4 \cdot 2 + 1$ *böylece* $1 = 4 \cdot 2 - 1 \cdot 9$ *bulunur.*

$1 = 4 \cdot 2 + (-1) \cdot 9 \Rightarrow 4 \cdot 2 \equiv 1$ *mod* 9

Neticede 4 *ün mod* 9 *a göre tersi* 2 *dir.*

NOT: $a \cdot b \equiv 1 \bmod m$ eşitliğinin sağlanması $gcd(a, m) = 1$ olmasıyla mümkündür.

Teorem 6.4.5 *(Fermat's Little Theorem) Eğer* p *asal bir sayı ise ve* $gcd(a, p) = 1$ *koşulunu sağlıyor ise*

$a^{p-1} \equiv 1$ *(mod* p*) denkliği her zaman doğrudur.*

NOT: $a^{p-1} \equiv 1 \pmod{p} \Rightarrow a^p \equiv a \pmod{p}$

Tanım 6.4.6 *Eğer p ve q,* $a^p \equiv a \ (mod\, p)$ *ile* $a^q \equiv a \ (mod\, q)$ *denkliklerini sağlayan farklı asal sayılar ise* $a^{pq} \equiv a \ (mod\, pq)$ *dur.*

Örnek 6.4.7

- $2^{1000000} \equiv ?\ (mod\ 7)$

$$p = 7 \qquad p - 1 = 6$$
$$1000000 = 6 \cdot 166666 + 4 \quad yani \quad 1000000 \equiv 4(mod6)$$

Böylece

$$2^{1000000} = (2^6)^{166666} \cdot 2^4 \equiv 1 \cdot 2^4 = 16 \equiv 2\, mod\, 7$$

- $2^{340} \equiv ?\ (mod\ 341)$

$$2^{11} = 2 \cdot 2^{10} \equiv 2 \cdot 1 \equiv 2\, mod\, 31$$
$$2^{31} = 2 \cdot (2^{10})^3 \equiv 2 \cdot 1^3 \equiv 2\, mod\, 11$$

$$\Rightarrow a = 2,\ p = 11,\ q = 31$$

$$2^{11 \cdot 31} \equiv 2\, mod\, 341$$
$$2^{341} \equiv 2\, mod\, 341 \Rightarrow 2^{340} \equiv 1\, mod\, 341$$

6.5 Euler Fi(ϕ) Fonksiyonu (Euler Phi Function)

Tanım 6.5.1 $n \geq 1$ *bir tamsayı olsun.* $\phi(n)$ *fonksiyonu,* $1 \leq a \leq n$ *ve* $gcd(a, n) = 1$ *koşulunu sağlayan a tamsayılarının sayısını gösterir, yani n ye kadar olan ve n ile*

aralarında asal olan sayıların sayısını verir.

Örnek 6.5.2

$$\Phi(1) = 1$$
$$\Phi(2) = 1 \quad \textit{çünkü} \quad gcd(1,2) = 1$$
$$\Phi(3) = 2 \quad \textit{çünkü} \quad gcd(1,3) = gcd(2,3) = 1$$
$$\Phi(4) = 2 \quad \textit{çünkü} \quad gcd(1,4) = gcd(3,4) = 1$$
$$\Phi(5) = 4 \quad \textit{çünkü} \quad gcd(1,5) = gcd(2,5) = gcd(3,5) = gcd(4,5) = 1$$

NOT: $\phi(n) = n - 1 \Leftrightarrow n$ bir asal sayı olursa.

Teorem 6.5.3 *Eğer p asal bir sayı ise ve $k > 0$ ise $\phi(p^k) = p^k - p^{k-1} = p^k(1 - \frac{1}{p})$ dir.*

Teorem 6.5.4 *(Euler Teoremi) $n > 1$ ve $gcd(a, n) = 1$ ise $a^{\phi(n)} \equiv 1 \, mod \, n$*

Örnek 6.5.5 *$3^{\phi(8)} \equiv 1 \, mod \, 8$ olduğunu gösterin.*

- $\phi(8) = \phi(2^3) = 2^3 - 2^2 = 4$
  $3^{\phi(8)} = 3^4 = 81 \equiv 1 \bmod 8$

BÖLÜM 7

AÇIK ANAHTARLI SİSTEMLER

Farzedin ki siz e-mail yoluyla başka bir kişiyle haberleşmek istiyorsunuz ve mesajlarınızın şifreli olmasını istiyorsunuz. Örnek vermek gerekirse, düşünün ki şifreleme metodunuz üç harf anahtarlı Vigenere olsun. 26 sayı sistemi taban alındığında, bu anahtar 0 ile $26^2 = 676$; 2 lik sistemde 0 ile 1010100100 arasındadır. Sizinle haberleştiğiniz kişi arasındaki bilgisayar ağı, internet ve fiber optik kanallar güvenli olmadığı için, siz anahtarlarınızı e-mail yoluyla değiştirmek istemezsiniz. 2 lik anahtarlarınızı açık kanal üzerinden güvenli bir şekilde değiştirme yolları vardır. Algoritma ve karmaşıklık teorisinde uzmanlar belirli matematik problemlerin çözümü için aşırı zaman gerektiğine inanıyorlar. Açık anahtarlı kriptosistemler de bu mantığa göre geliştirilmiştir ; öyle ki bu kriptosistemlerin birini kırmak bu zor matematik problemleri çözmekle eşdeğerdir.Çok hızlı bilgisayarlar da programlanmış olan çözüm metodları bile haftalar, aylar, yüzyıllar hatta evrenin sonuna kadar olan hayatı bile kapsayabilir.

7.1 MERKLE-HELLMAN KNAPSACK KRİPTOSİSTEM

Knapsack açık anahtarlı şifreleme sistemleri alt küme toplama problemleri temeline dayanır. Buradaki temel düşünce, çözümü kolay olan bir alt küme toplam problemi örneğini seçip, onu çözümü zor olan bir alt küme toplama problemi örneğine çevirerek

gizlemektir. İlk(orjinal) knapsack kümesi gizli anahtarı, dönüştürülmüş(gizlenmiş) knapsack kümesi de kapalı anahtarı oluşturur.

Merkle-Helman açık anahtarlı şifreleme şeması, süperartan alt küme toplama problemi olarak da adlandırılan kolayca çözülebilen bir alt küme toplama problemi örneğini modüler çarpım ve permütasyon yoluyla gizleme girişimidir.

7.1.1 Süperartan dizi (Superincreasing sequence)

$B = (b_1, b_2, \cdots, b_n)$ dizisinde eğer her sayı kendisinden önce gelen sayıların toplamından büyük ise ,yani

$$b_i = \sum_{j=1}^{i-1} b_j \text{ öyleki } 2 \leq i \leq n,$$

bu diziye süperartan dizi denir.

7.1.2 Süperartan Altküme Toplama Problemini çözme Algoritması

GİRDİ: $(b_1, b_2, \cdots, b_n)$ şeklinde olan süperartan bir dizi ve $b_i$lerin altkümesinin toplamını ifade eden s tamsayısı algoritmamızın girdileri olsun.

ÇIKTI: Elemanları $x_i \in \{0, 1\}$ olan ve $\sum_{i=n}^{n} x_i b_i = s$ koşulunu sağlayan $(x_1, x_2, \cdots, x_n)$ dizi aşağıdaki şekilde hesaplanır:

1. $i \leftarrow n$.
2. $i \geq 1$ ise aşağıdakiler yapılır:

- Eğer $s \geq b_i$, $x_i = 1$ yazılır ve $s \leftarrow s - b_i$ uygulanır. Aksi taktirde $x_i = 0$ yazılır.
- $i \leftarrow i - 1$ işlemi $i = 1$ olana kadar sürer.

3. Bulunan x_i ler $(x_1, x_2, \cdots, x_n)$ dizisini oluşturur.

7.1.3 Merkle-HellmanKnapsack Şifrelemesinde Anahtar Oluşturma Algoritması

Bu kriptosistemde her kişi kendi açık anahtarını ve buna bağlı gizli anahtarını şu şekilde oluşturur:

1. Sistem parametresi olarak sabit bir n tamsayısı alınır.
2. Her A kişisi aşağıdaki $3-7$. adımları uygular.
3. Bir tane süperartan $(b_1, b_2, \cdots, b_n)$ dizisi ve $M > b_1 + b_2 + \cdots + b_n$ şartını sağlayan bir M mod sayısı seçer.
4. $1 \leq W \leq M - 1$ ve $gcd(W, M) = 1$ koşullarını sağlayan rastgele bir W tamsayısı seçer.
5. $\{1, 2, \cdots, n\}$ tamsayılarıyla ifade edilen rastgele bir π permütasyonu seçer.
6. $i = 1, 2, \cdots, n$ değerleri için $a_i = W b_{\pi i} \, mod \, M$ ifadelerini hesaplar.
7. A'nın açık anahtarı $(a_1, a_2, \cdots, a_n)$; A'nın kapalı anahtarı ise $(\pi, M, W, (b_1, b_2, \cdots, b_n))$ olur.

7.1.4 Basit Merkle-Hellman Knapsack Açık Anahtar Şifreleme Algoritması

B şahsı A için m mesajını şifreliyor olsun.

1. **Şifreleme**: B, şunları yapar:
   - A'nın açık anahtarı $(a_1, a_2, \cdots, a_n)$ i alır.
   - m mesajını n uzunluğundaki 2 lik dizi,$m = m_1 m_2 \cdots m_n$, olarak ifade eder
   - Daha sonra $c = m_1 a_1 + m_2 a_2 + \cdots + m_n a_n$ değerini hesaplar.
   - Oluşan kapalı metni A'ya gönderir.
2. **Deşifreleme**: c kapalı metnine karşılık gelen m açık metnini çözmek için A şunları yapar:
   - Öncelikle $d = W^{-1} c \, mod \, M$ değerini hesaplar.
   - Süperartan altküme toplam problemini çözerek, $d = r_1 b_1 + r_2 b_2 + \cdots + r_n b_n$ eşitliğini sağlayan $r_1, r_2, \cdots, r_n$ $r_i \in \{0, 1\}$ tamsayılarını bulur.
   - Mesaj bitleri $m_i = r_{\pi(i)}$, $i = 1, 2, \cdots, n$ dir.

Örnek 7.1.1 ***Anahtar Oluşturma:*** *$n = 6$ olsun. A şahsı $(12, 17, 33, 74, 157, 316)$ süperartan bir dizi ve $M = 737 > 12 + 17 + 33 + 74 + 157 + 316 = 609$ tamsayısı seçer. Daha sonra $gcd(W = 635, M = 737) = 1$ koşulunu sağlayan bir $W = 635$ sayısını seçer. Son olarakta $\{1, 2, \cdots, 6\}$ sayılarından oluşup $\pi(1) = 3$, $\pi(2) = 6$, $\pi(3) = 1$, $\pi(4) = 2$, $\pi(5) = 5$, $\pi(6) = 4$ leri sağlayan bir π permütasyonunu alır. A açık anahtarını*

$a_i = Wb_{\pi i} \, mod \, M$ *eşitliğini kullanarak şu şekilde oluşturur:*

$$\begin{aligned}
a_1 &= Wb_{\pi(1)} = Wb_3 = 635 \cdot 33 \, mod \, 737 &&\equiv 319 \\
a_2 &= Wb_{\pi(2)} = Wb_6 = 635 \cdot 316 \, mod \, 737 &&\equiv 196 \\
a_3 &= Wb_{\pi(3)} = Wb_1 = 635 \cdot 12 \, mod \, 737 &&\equiv 250 \\
a_4 &= Wb_{\pi(4)} = Wb_2 = 635 \cdot 17 \, mod \, 737 &&\equiv 477 \\
a_5 &= Wb_{\pi(5)} = Wb_5 = 635 \cdot 157 \, mod \, 737 &&\equiv 200 \\
a_6 &= Wb_{\pi(6)} = Wb_4 = 635 \cdot 74 \, mod \, 737 &&\equiv 559
\end{aligned}$$

Böylece A'nın açık anahtarı $(319, 196, 250, 477, 200, 559)$ *knapsack dizisidir. A'nın gizli anahtarı ise* $(\pi, M, W, (12, 17, 33, 74, 157, 316))$ *dır.*

Şifreleme: *B,* $m = 101101$ *mesajını şöyle şifreler:*

$$\begin{aligned}
c &= 1 \cdot 319 + 0 \cdot 196 + 1 \cdot 250 + 1 \cdot 477 + 0 \cdot 200 + 1 \cdot 559 \\
&= 319 + 250 + 477 + 559 = 1605
\end{aligned}$$

ve bunu A'ya gönderir.

Deşifreleme: *Mesajı çözmek için A,* $d = W^{-1}c \, mod \, M$ *değerini hesaplar ve süperartan altküme toplama problemini çözer. Öncelikle* $W^{-1} = 635^{-1} \equiv 513 \, mod \, 737$*, ikinci olarak* $d = W^{-1}c = 513 \cdot 1605 \equiv 136 \, mod \, 737$ *değerlerini bulur.*

$$\begin{aligned}
136 &= 12 \cdot r_1 + 17 \cdot r_2 + 33 \cdot r_3 + 74 \cdot r_4 + 157 \cdot r_5 + 316 \cdot r_6 \\
&= 12 + 17 + 33 + 74
\end{aligned}$$

Böylelikle $r_1 = 1$*,* $r_2 = 1$*,* $r_3 = 1$*,* $r_4 = 1$*,* $r_5 = 0$*,* $r_6 = 0$ *ve* π *nin permütasyonunun uygulanmasıyla mesaj bitleri* $m_1 = r_3 = 1$*,* $m_2 = r_6 = 0$*,* $m_3 = r_1 = 1$*,* $m_4 = r_2 = 1$*,* $m_5 = r_5 = 0$*,* $m_6 = r_4 = 1$ *bulunur.*

7.2 RSA Kriptosistem

RSA kriptosistem, 1978 yılında " Dijital imza elde etme metodu ve açık anahtarlı kriptosistemler" adlı bir makale ile yayınlandı. Adını yaratıcılarının (Ronald Rivest,Adi Shamir, Leonard Adleman) soyadlarının başharflerinden alan RSA kriptosistem, göndericinin bir metodla ve herkesçe bilinen açık bir anahtarla mesajlarını şifrelediği bir şifre sistemi olarak tanımlanır. Daha önceki gizli(simetrik) anahtarlı sistemlerin tersine anahtarı bilmek deşifre anahtarını ortaya çıkarmaz. Bu sistem hem gizlilik hem de dijital imza sağlamak amaçlı kullanılabilir. Bu sistemin güvenliği tamsayılarda çarpanlara ayırma probleminin kolaylıkla olmaması temeline dayanır.

RSA kriptosisteminde kişilere şifreli mesaj gönderilebilmesi için o kişilerin açık anahtarlarına ihtiyaç vardır. Mesajı alan kişinin de mesajı okuyabilmesi için gizli bir anahtarının olması gerekir.Anahtar oluşturma aşğıdaki algoritmada ifade edilmiştir.

Anahtar Oluşturma algoritması: Her A kişisi anahtarını şu şekilde oluşturur:

- İki tane farklı,rasgele ve yaklaşık aynı uzunlukta olan p ve q asal sayıları seçer.
- $n = pq$ ve $\phi = (p-1)(q-1)$ değerlerini hesaplar.
- $1 < e < \phi$ ve $gcd\,(e, \phi) = 1$ olacak şekilde rastgele bir e sayısı seçer.
- Öklid algoritmasını kullanarak, $1 < d < \phi$ ve $ed \equiv 1\,(mod\,\phi)$ koşulunu sağlayan d sayısını hesaplar.
- A'nın açık anahtarı (n, e); A'nın gizli anahtarı ise d olur.

RSA anahtar oluşumunda e ve d tamsayıları sırasıyla şifreleme üssünü ve deşifreleme üssünü ve n ise mod sayısını gösterir. p ve q sayılarının onluk sistemde uzunluklarının 100 ve dolayısıyla da n nin uzunluğunun 200 olması beklenir. Fakat verilecek örneklerde kolaylık olması açısından küçük sayılar seçilecektir.

Şifreleme Algoritması:

1. B şahsı, A'ya bir m mesajı göndermek istiyor. B, m mesajını şifrelemek için aşağıdakileri yapar:
   - Öncelikle A'nın açık anahtarını (n,e) alır.
   - m mesajını $[0, n-1]$ aralığında yazar.
   - Sonra $c \equiv m^e \,(mod\, n)$ değerini hesaplar.
   - Oluşan c şifresini A'ya gönderir.
2. Şifreli c metninden açık metni bulabilmek için A aşağıdaki işlemi uygular:
   - d gizli anahtarını kullanarak ve $m \equiv c^d \,(mod\, n)$ işlemini uygulayarak m açık metine ulaşır.

NOT: Deşifre sisteminin çalışmasına

$ed \equiv 1 \,(mod\, \phi)$ olduğu içini $ed = 1 + k\phi$ eşitliğini sağlayan mutlaka bir k tamsayısı bulunur. Eğer $gcd\,(m, p) = 1$ ise Fermat teoreminden dolayı

$$m^{p-1} \equiv 1 \,(mod\, p).$$

Eğere bu denkliğin her iki tarafının da $k(q-1)$'inci kuvvetlerini alırsak

$$m^{k(p-1)(q-1)} \equiv 1 \,(\, mod\, p).$$

olur ve her iki tarafı da m ile çarptığımızda

$$m^{1+k(p-1)(q-1)} \equiv m \,(\, mod\, p).$$

sonucuna ulaşırız.

Diğer tarfatan, eğer $gcd\,(m,p) = p$ olursa yukarıdaki denklik yine geçerli olur; çünkü farzedelim belli bir k tamsayısı için $m = kp$ olsun

$$m^{p-1} = (kp)^{(p-1)} = k^{(p-1)}p^{(p-1)} \equiv p \,(\, mod\, p).$$

Eğer bu denkliğin her iki tarafının da $k(q-1)$'inci kuvvetlerini alırsak

$$m^{k(p-1)(q-1)} \equiv p^{k(p-1)(q-1)} \equiv p \,(\, mod\, p).$$

olur ve her iki tarafı da m ile çarptığımızda

$$m^{1+k(p-1)(q-1)} \equiv mp = kp^2 \equiv kp = m \,(\, mod\, p).$$

İki durumda da

$$m^{ed} \equiv m \,(\, mod\, p)$$

olduğu görülür. Aynı şekilde,

$$m^{ed} \equiv m \,(\, mod\, q)$$

olur.

Sonuçta p ve q farklı asal sayılar olğu için,

$$m^{ed} \equiv m \,(\, mod\, n)$$

dir.Böylelikle,

$$c^d = m^{e^d} \equiv m \,(\, mod\, n)$$

Örnek 7.2.1 *1.* ***Anahtar oluşturma****: A şahsı* $p = 2357$ *ve* $q = 2551$ *olan iki tane asal sayı seçmiş olsun. Öncelikle A,*

$$n = pq = 6012707$$

ve

$$\phi = (p-1)(q-1) = 6007800$$

değerlerini hesaplar. A bir tane $e = 3674911$ *değeri seçer.Bu e değeri,* $gcd(e = 3674911, \phi = 6007800) = 1$ *ve* $1 < e = 3674911 < \phi = 6007800$ *koşullarını sağlar. Daha sonra Öklid algoritmasını kullanarak*

$$e \cdot d \equiv 1 \,(mod\, \phi)$$

$$3674911 \cdot d \equiv 1 \,(mod\, 6007800)$$

$d = 422191$ *değerini hesaplar. A' nın açık anahtarı* $(n = 6012707, e = 3674911)$*; gizli anahtarı da* $d = 422191$ *olur.*

2. ***Şifreleme****: B,* $m = 5234673$ *mesajını şifrelemek için A'nın açık anahtarını,yani* $(n = 6012707, e = 3674911)$*, alır ve aşağıdaki şekilde olduğu gibi kapalı metin c 'yi*

hesaplar:

$$c \equiv m^e \,(mod\, n) = 5234673^{3674911} \,(mod\, 6012707) \equiv 3650502$$

ve bu değeri A'ya gönderir.

3. ***Deşifreleme**: A, gelen c kapalı metninden m açık metni aşağıdaki gibi hesaplar:*

$$m \equiv c^d \,(mod\, n) = 3650502^{422191} \,(mod\, 6012707) \equiv 5234673$$

7.3 RSA İmza Şeması

RSA kriptosistemi dijital imzalar için de kullanılabilir. (n, e) A şahsının açık anahtarı, d sayısı da A'nın gizli deşifreleme üssü olsun.Öncelikle mesajın imzalanabilmesi için m mesajının $\{0, 1, \cdots, n-1\}$ arasında olması istenir, daha sonra hesaplamalar yapılır.

7.3.1 İmzalama

A B'ye imzalı m mesajını göndermek isterse, mesaja kendisinin kapalı anahtarını uygular,yani

$$\sigma = m^d \, mod\, n.$$

Daha sonra (m, σ) imzalı mesajı B'ye gönderir.

7.3.2 İmzayı Doğrulama

B, A'dan aldığı (m, σ) imzalı mesajı doğrulamak için

$$m = \sigma^e \, mod \, n$$

değerini hesaplar. Çıkan sonuç m ise mesaj doğrulanmış olur.

Örnek 7.3.1 ***Anahtar Oluşturma:*** *A kişisi* $p = 7927$ *ve* $q = 6997$ *asal sayılarını seçer ve,* $n = pq = 55465219$ *ve* $\phi = 7926 \cdot 6996 = 55450296$ *değerlerini hesaplar. Daha sonra A,* $ed = 5d \equiv 1 \, (mod \, 55450296)$ *eşitliğinden* $d = 44360237$ *sayısını bulur. A'nın açık anahtarı* $(n = 55465219, e = 5)$*; gizli anahtarı* $d = 44360237$ *olur.*

İmzalama: $m = 31229978$ *mesajını imzalamak için A şunu hesaplar:*

$$\sigma = m^d \, mod \, n = 31229978^{44360237} \, mod \, 55465219 \equiv 30729435$$

ve $(m = 31229978, \sigma = 30729435)$ *'yi B'ye gönderir.*

İmzayı Doğrulama: $(m = 31229978, \sigma = 30729435)$*'yi alan B mesajı doğrulamak için şunu yapar:*

$$m = \sigma^e \, mod \, n = 30729435^5 \, mod \, 55465219 \equiv 31229978$$

Çıkan sayı m *olduğu için imza doğrulanmış olur.*

7.4 Ayrık Logaritma(Discrete Logarithm)

RSA kriptosisteminde, RSA fonksiyonu m olan bir elemanın e. kuvvetini oluşturur. Bu fonksiyon birebir bir fonksiyondur ve etkili bir şekilde hesaplanır.Eğer n nin çarpanlara ayrımı bilinmiyorsa, e. kökü hesaplamak için etkili bir algoritma yoktur. Sayılar teorisinde hesaplaması kolay fakat tersinin hesaplaması zor olan başka fonksiyonlar da vardır. Bunlardan en önemlilerinden biri de sınırlı alanlar da (finite fields) kuvvet almadır. Basit olarak sadece asal alanlar (prime fields) düşünülecektir.

p bir asal sayı ve g de Z_p^* de bir primitif kök olsun. Ayrık kuvvet fonksiyonu (discrete exponential function)

$$\text{Exp} : Z_{p-1} \to Z_p^*,\ x \mapsto g^x,$$

tekrarlı karesini alma algoritması örneğinde olduğu gibi etkili bir şekilde hesaplanabilir. Kuvvetin logaritması fonksiyonunun tersini hesaplamak için etkili bir algoritma bilinmemektedir. Bu tahmine ayrık logaritma tahmini (discrete logarithm assumption) denir.

7.5 El-Gamal Açık Anahtarlı Kriptosistem

ElGamal açık anahtarlı şifre sistemi,anahtar transferi modunda Diffie-Hellman anahtar anlaşması(Diffie-Hellman Key Agreement) olarak görülebilir. Güvenilirliği ayrık logaritma problemi ve Diffie-Helman probleminin kolay çözülememesi temeline dayanır. Temel ElGamal ve genelleştirilmiş ElGamal şifreleme şeması bu bölümde tanımlanmıştır.

7.6 ElGamal Açık Anahtarlı Şifrelemede Anahtar Oluşturma Algoritması

Her kişi kendi açık anahtarını ve buna bağlı gizli anahtarını oluşturur. Bunu oluşturmak için A şahsı şunları uygular:

1. Çok büyük rastgele bir p asal sayısı ve mod p ye göre tamsayıların oluşturduğu çarpım grubu Z_p^* nin bir jeneratörü α yı oluşturur.
2. $1 \leq a \leq p-2$ şeklinde olan bir a tamsayısı seçer ve $\alpha^a \, mod \, p$ değerini hesaplar.
3. A'nın açık anahtarı (p, α, α^a); A'nın gizli anahtarı ise a olur.

7.6.1 ElGamal Açık Anahtarlı Şifreleme Algoritması

B şahsı A için m mesajını şifrelesin.

1. **Şifreleme:** B mesajı şifreleme için şunları yapar:
   - A'nın açık anahtarını (p, α, α^a) alır.
   - mesajı $\{0, 1, \cdots, p-1\}$ aralığında m tamsayısı olarak ifade eder.
   - $1 \leq k \leq p-2$'yi sağlayan rastgele bir k tamsayısı seçer.
   - $\gamma = \alpha^k \, mod \, p$ ve $\delta = m \cdot (\alpha^a)^k \, mod \, p$ değerlerini hesaplar.
   - Son olarak $c = (\gamma, \delta)$ kapalı metnini A'ya gönderir.
2. **Deşifreleme:** c kapalı metninden m açık metine ulaşmak için A şunları yapar:

- a gizli anahtarını kullanarak $\gamma^{-a} \, mod \, p$ değerini hesaplar ($\gamma^{-a} = \alpha^{-ak} \, mod \, p$).
- $\gamma^{-a} \cdot \delta \, mod \, p$ değerini hesaplayarak m'yi bulur.

$$\gamma^{-a} \cdot \delta \equiv \alpha^{-ak} \cdot m\alpha^{ak} \equiv m \, (mod \, p)$$

Örnek 7.6.1 ***Anahtar Oluşturma:*** *A şahsı bir* $p = 2357$ *asal sayısı ve* $\alpha = 2 \in \mathcal{Z}^*_{\in \ni \bigtriangledown}$ *bir jeneratör seçer. Buna ilave olarak bir* $a = 1751$ *gizli anahtarı seçer ve*

$$\alpha^a \, mod \, p = 2^{1751} \, mod \, 2357 \equiv 1185$$

değerini hesaplar. A'nın açık anahtarı $(p = 2357, \alpha = 2, \alpha^a = 1185)$ *tir.*

Şifreleme: $m = 2035$ *mesajını şifrelemek için B şahsı rastgele bir* $k = 1820$ *tamsayısı seçer ve*

$$\gamma = 2^{1520} \, mod \, 2357 \equiv 1430$$

ve

$$\delta = 2035 \cdot 1185^{1520} \, mod \, 2357 \equiv 697$$

değerlerini hesaplar. Son olarak B $(\gamma = 1430, \delta = 697)$ *'yi A'ya gönderir.*

Deşifreleme: *A gelen kapalı metni çözmek için*

$$\gamma^{-a} = 1430^{-1750} \equiv 1430^{605} \, mod \, 2357 \equiv 872$$

bulur ve m mesajına da

$$m = 872 \cdot 697 \, mod \, 2357 \equiv 2035$$

böylece ulaşır.

7.6.2 ElGamal İmzası

ElGamal kriptosisteminde imza RSA 'da olduğu gibi mesajın doğru kişiden geldiğini kontrol etmek için kullanılır. Sadece kapalı metin yerine imzalanmış kapalı metin gönderilerek o kapalı metnin istenen kişiden gelip gelmediği de kontrol edilmiş olur. A şahsının açık anahtarı $(p, \alpha, \alpha^a = y)$ ve gizli anahtarının da a olduğu düşünülsün.

7.6.3 İmza Algoritması

m mesajının Z_p nin bir elemanı olduğu düşünülür.Eğer değilse hash fonksiyonu kullanılarak m mesajının Z_p nin elemanı olması sağlanır. A şahsı m mesajını şu şekilde imzalar:

1. Rastgele bir t tamsayısı seçer öyleki $1 \leq t \leq p-2$ ve $gcd(t, p-1) = 1$ koşulunu sağlamalıdır.
2. $r = \alpha^t$ ve $s = t^{-1}(m - ra) \, mod \, (p-1)$ eşitliklerini kurar.
3. (m, r, s) A'nın imzalı mesajıdır.

7.6.4 Doğrulama

(m, r, s) imzalı mesajı alan B şahsı aldığı mesajın A'dan geldiğini şu şekilde doğrular:

1. Öncelikle $1 \leq r \leq p-1$ olduğunu kontrol eder.Eğer değilse imzayı reddeder.

2. Daha sonra $v = \alpha^m$ ve $w = y^r r^s$ değerlerini hesaplar (Buradaki y sayısı A'nın açık anahtarındaki y sayısıdır.)

3. Eğer $v = w$ eşitliği sağlanıyorsa imza kabul edilir, aksi taktirde reddedilir.

Örnek 7.6.2 ***Anahtar Oluşturma:*** *A şahsı bir* $p = 2357$ *asal sayısı ve* $\alpha = 2 \in \mathcal{Z}^*_{\in \ni \bigtriangledown}$ *bir jeneratör seçer. Buna ilave olarak bir* $a = 1751$ *gizli anahtarı seçer ve*

$$\alpha^a \, mod \, p = 2^{1751} \, mod \, 2357 \equiv 1185$$

değerini hesaplar. A'nın açık anahtarı $(p = 2357, \alpha = 2, \alpha^a = 1185)$ *tir.*

İmza Oluşturma: *Basit olması açısından mesaj* $m = 1463$ *olarak seçilsin (Eğer mesaj* p *asal sayısından büyük olsaydı hash fonksiyonundan geçirilirdi).* $m = 1463$ *mesajını imzalamak için A önce rastgele bir* $t = 1529$ *sayısı seçer, daha sonra*

$$r = \alpha^t \, mod \, p = 2^{1529} \, mod \, 2357 \equiv 1490$$

ve

$$t^{-1} \, mod \, (p-1) = 1529^{-1} \, mod \, (2356) \equiv 245$$

$$s = t^{-1}(m - ra) \, mod \, (p-1) = 245(1463 - 1490 \cdot 1751) \, mod \, 2356 \equiv 1777$$

A'nın imzası $(m = 1463, r = 1490, s = 1777)$

İmzayı Doğrulama: *B aldığı imzalı mesajı doğrulamak için önce*

$$v = \alpha^m \, mod \, p = 2^{1463} \, mod \, 2357 \equiv 1072$$

değerini hesaplar. Daha sonra

$$w = y^r r^s \, mod \, p = 1185^{1490} 1490^{1777} \, mod \, 2357 \equiv 1072$$

değerini hesaplar ve $v = w$ olduğu için imzayı kabul eder.

7.7 Diffie-Hellman Anahtar Anlşması (Diffie-Hellman Key Agreement)

Diffie-Hellman anahtar anlaşması, anahtar dağıtma problemine ilk pratik çözümdür. Üs olarak anahtar değiştirme olarak da bilinen bu sistem daha önce hiç haberleşme sağlamamış iki tarafın açık kanal üzerinden mesajlarını birbirlerine göndererek ortak bir anahtar yaratma temeline dayanır.

p yeteri kadar büyük bir asal sayı olsun öyleki Z_p^* de discrete logaritma problemini çözmek mümkün olmasın. g'de Z_p^* de primitif bir kök (primitive root) olsun. p ve g herkes tarafından bilinsin. A ve B kişileri aşağıdaki yolu izleyerek ortak bir anahtar yaratabilirler:

7.7.1 Diffie-Hellman Anahtar Anlaşması Algoritması:

- A, $0 \leq a \leq p-2$ eşitsizliğini sağlayan ve tesadüfi olan bir a sayısı seçer. $c = g^a$'yı bulur ve bunu B'ye gönderir.
- A, $0 \leq b \leq p-2$ eşitsizliğini sağlayan ve tesadüfi olan bir b sayısı seçer. $d = g^b$'yi bulur ve bunu A'ya gönderir.

- A, ortak anahtar k'yı şu şekilde hesaplar:

$$k = d^a = (g^b)^a$$

- B, ortak anahtar k'yı şu şekilde hesaplar:

$$k = c^b = (g^a)^b$$

Böylelikle A ve B aralarında ortak bir anahtar olan k için anlaşmış olurlar.

BÖLÜM 8

KRİPTANALİZ

Kriptanaliz (Kripto-analiz) bölümünde modern kripto sistemleri içerisinde önemli bir sınıf teşkil eden simetrik anahtarlı sistemler olarak bilinen blok şifrelerin ve akan şifrelerin analizi görülecektir. ***Kerckhoff's prensibi*** : 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.

İkinci Dünya Savaşında Polonyalı ve İngiliz matematikçiler Alman Enigma şifreleme makinasının analizini yaparak algoritmayı kırmışlardır ve Alman kapalı metinlerini kripto-analiz yöntemi ile açmışlardır. Farklı bir algoritma olan ve belli bir süre güvenli olarak kabul edilmiş olan RC4 algoritması "tersine giderek" (reverse engineering) ile kırılmıştır.

- 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).

Atak Çeşitleri (Senaryoları) :

- **Sadece Şifreli Metin Atağı** (Ciphertext-Only) : En güçlü kripto-analitik atakdır. Sadece haberleşme pasif olarak dinlenip (müdahale edilmeden)yeterince kapalı metin elde edilerek yapılabilir.

- **Bilinen Açık Metin Atağı** (Known Plaintext): Bir miktar açık-kapalı metin çifti bilinerek yapılan ataktır. Mesajların bir kısmı tahmin edilebilir veya açık gönderilen mesajlar toplanarak atak uygulanabilir.

- **Seçilmiş Açık Metin Atağı** (Chosen Plaintext): Analizcinin istediği (seçtiği) metni şifreleme imkanına sahip olduğu kabul edilen atak senaryosudur. Analizci şifreleme algoritmasının güvenli olarak yerleştirildiği mekanizmayı elde edebilir. Analizci aktif olarak haberleşme sisteminde rol alır.

- **Seçilmiş Kapalı Metin Atağı** (Chosen Ciphertext): Deşifreleme makinasına ulaşılarak yapılan atak çeşitidir. Bir önceki senaryoya benzemektedir.

- **Seçilmiş Açık veya Kapalı Metin Atağı** (Adaptive Chosen Plaintext or Ciphertext): Bu atak senaryosunda analizcinin istediği mesajı açma veya kapatma konusunda sınırsız kapasiteye sahip olduğu kabul edilir. Önceki iki senaryonun teorik olarak daha güçlendirildiği atak çeşitidir.

8.1 Kriptanalitik Atakların Amaçları

Ayıran Ataklar (Distinguishing Attacks) Ayıran Atakların başarılı olabilmesi için şifre sisteminin çıktısını rastgele bir permutasyonun çıktısından ayırılaması olası olmalıdır.

Kısmi açık metin bilgisi (Partial Knowledge of the Plaintext) Bu atakta kısmi açık metin bilgisine (Şifre sisteminin girdisi için herhangi tahmin) sahip olunur.

Deşifreleme (Decryption) Bu durumda saldıran şifrelenmiş trafiğin bir kısmını deşifre etme yeteneğine sahiptir.

Şifreleme (Encryption(Forgery))

Bu durumda saldıran anlamlı mesajları bilinmeyen gizli anahtar ile şifreleme olanağına sahiptir. Bu gizli anahtar bilgisine sahip olduğu anlamına gelmez. Bu atağa meyilli olan şifre sistemleri gerçekliğini **kanıtlama/kimlik belirtme** işlemlerinde kullanım için uygun değillerdir.

Kısmı Anahtar Edinimi (Partial Key Recovery) Bu atakta gizli anahtarın belli bir kısmı saldıran tarafından ele geçirilir. Belki bu anahtarın geriye kalan kısmı çok büyük olabilir fakat bu arzulanan bir durum değildir. Çünkü tüm anahtarın genellikle ele geçirilmesi için ilk basamaktır.

Tüm Anahtar Edinimi (Total Key Recovery) Bir kriptosistem için en korkunç kriptanalitik atak çeşidir.

8.2 Kriptanaliz Metodları(Methods of Cryptanalysis)

Eğer şifre sistemi temiz ve basit bir yapıya sahip ise hala kalem ve kağıt kriptanalistin elindeki en güçlü silahlardır. Bir çok durumda kağıt analizi bilgisayar analizinden gelen geribildirimlere (özel istatiksel özellikler ve düzensizlikler arama gibi) ihtiyaç duyar. Araştırmacı bakış açısından bir şifre sistemi kırıldığının dile getirilmesi, 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 en açık ve en doğrudan uygulanabilir bir methodur. Tüm olası gizli anahtarları bilinen kısa açık/kapalı metin örnekleri üzerinde dener. Doğru gizli anahtar bilinen açık bir metinden doğru kapalı metinin elde edilmesini sağlar. Günümüz hesaplama imkanlarına göre modern blok şifre sistemlerinin anahtar uzunlukları (128-bit ve yukarısı) bu tip atağı imkansız kılacak şekilde seçilmektedir. DES in en önemli zayıflığı 56-bit olan kısa anahtar uzunluğudur ve günümüz koşulları düşünüldüğünde bu anahatar uzunluğu etraflı aramayı mümkün kılmaktadır.

Sözlük Atakları (Dictionary Attacks)

Bu da basit fakat blok şifre sistemleri için önemli bir atak çeşidir. Eğer şifrelenen metinlerin uzunlukları kısa ise saldıran birçok metin toplar ve farklı metinlerin tekrar-

lama analizini yapar. En uç noktada bu atağa zayıf şifre sistemi basit değiştirmeli şifre sistemidir (simple substitution cipher).

Eş Tanımlama (Equivalent Description)

Bazen şifre sistemi tasarlayanlar sistemin veya parçalarının basit eşdeğer tanımlarını gözden kaçırırmaları, bu atak tarafından sömürülür.

Değişmezler için Devirlilik veya Arama (Periodicity or Search for Invariants) Değişmezler, şifreleme boyunca değişmeyen özellikler olarak düşünelebilir ve şifre sisteminin istenilmeyen bir özelliğidir. Eğer kriptanalist sistemin herhangi değişmezini yada yakınsamasını bulmayı başarırsa bir ayıran atak için malzeme edinmiş olur. Her çeşit devirli davranış veya şifrelenmeler arasındaki korelasyon mutlaka engelllemelidir.

Doğum Günü Paradoks (Birthday Paradox) Blok şifreleme sistemlerinden açık anahtar şifre sistemlerine uzunan sayılamıyacak kadar önemli bir olasılıklı paradoksudur.

Ortada buluşma Atağı (Meet-in-the-Middle Attack) Bu metod şifreleme sistemini alt ve üst olmak üzere böler. Sonra kısmi tahmini ile yukarıdan ortaya ve baştan ortaya kısmi deşifreleme yapar. Sonuç karşılaştırılır ve eğer uyumlu ise aday anahtar saklanır. Aksi takdirde tahmin edilen anahtar yanlış olur.

İstatiksel Yaklaşımlar (Statistic Approaches)

Bu metodlar kapalı ve açık metin arası ilişki veren istatiksel örnekleri arar. Bir tür ayıran ataktır ve diğer ataklar için ilk adımdır. İstatiksel yaklaşımlar hem oluşması yüksek olasılıkta olayları hem de gerçekleşmesi imkansız olayları bulmaya yöneliktir.

Özel bir Atağa göre Zayıf Anahtarlar (Weak Keys with Respect to a Particular Attack)

Bazı durumlarda bir zayıf anahtar kümesinin bir şifre sisteminin analizini özel bir atak modelini düşünüldüğünde kolaylaştırması mümkün olmaktadır. Eğer bir kriptosistemin tüm anahtar uzayına göre yüksek bir oranda zayıf anahtarlara sahip ise tekrar dizayn edilmesi bile söz konusu olabilir.

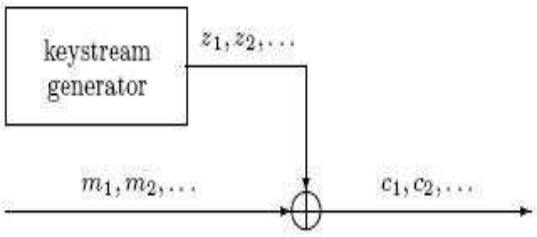
Örnek olarak DES te dört tane zayıf ve oniki tane yarı-zayıf anahtar bulunmaktadır. Gizli anahtar K olmak üzere DES i E_K olarak tanımlarsak dört tane zayıf anahtar için $E_K(E_K(m)) = m$ ve oniki tane yarı-zayıf anahtardan iki tanesi için $E_{K1}(E_{K2}(m)) = m$ sağlanmaktadır. IDEA blok şifre sistemi için 2^{128} anahtar uzayı üzerinde 2^{63} elemana sahip bir zayıf anahtar kümesi bulunmaktadır.

8.3 Akan Şifrelerin Analizi

İyi bir akan şifre algoritması bilinen açık metin atağa karşı dayanıklı olmalıdır. Genel olarak akan şifrelerin oluşturulmasında temel yapı taşları olarak LFSR'lar kullanılır ve gizli anahtar (secret key) LFSR'ların başlangıç konumları olarak (initial state) seçilir. Akan şifrelerin analizinde korelasyon atağı şu şekildedir.

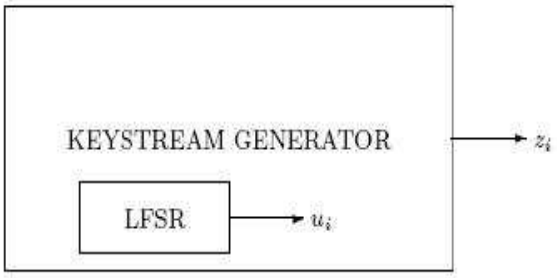
Akan şifrelerde anahtar üreticinin kullanımı

$c_i = m_i \oplus z_i \;\Rightarrow\; z_i = c_i \oplus m_i$



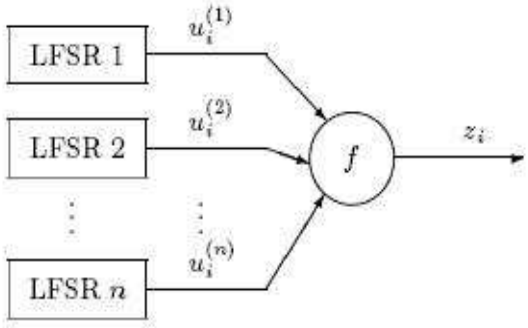
Bilinen açık metin atağı: Belirli açık-kapalı mesaj çiftleri (m_i, c_i), z_i'ler bilinirken gizli anahtarı bulabilmektir.

Korelasyon atağı için gerekli ve yeterli koşul $u_i = z_i$ olma olasılığının 0.5 den farklı olmasıdır. Eğer$P$ olasılığı gösterirse, bunu matematiksel olarak $P(u_i = z_i) \neq 0.5$ şeklinde ifade edebiliriz.

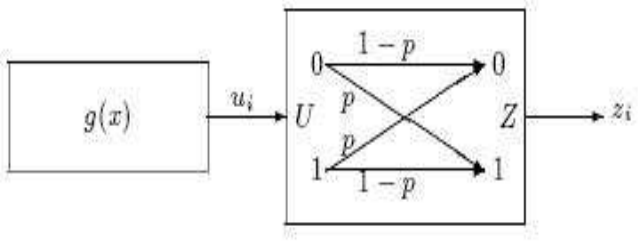


Korelasyon atak için gerek ve yeter şart $P(u_i = z_i) \neq 0.5$

Nonlinear (Doğrusal Olmayan) fonksiyonlarla LFSR'ları birleştirme (prensibi)



Eğer f fonksiyonu $(m-1)$-dayanıklı (m-dayanıklı olmayan) bir fonksiyonsa $P(z_i = u_i^{(a_1)} + u_i^{(a_2)} + \cdots + u_i^{(a_m)}) \neq 0.5$ dir. Bu durumda f'nin korelasyon atağına dayanıklı olması için m değerinin yeterince yüksek olması gereklidir.



Korelasyon atak Modeli

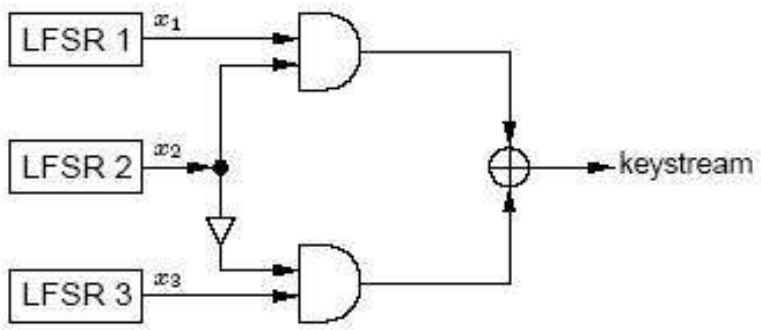
Yukarıdaki sistemde korelasyon ihtimali $1 - p = P(u_i = z_i)$ ve hata ihtimali p'dir.

Korelasyon Atağı: Resim 4 deki sistemde bütün LFSR'ların maksimum periyoda sahip olduğunu kabul edelim ve LFSR'ların uzunluklarını $L_1, L_2, \ldots, L_n$ ile gösterelim. Eğer bu

sistemdeki LFSR'ların bağlantı polinomları ve f fonksiyonu biliniyorsa en fazla $\prod_{i=1}^{n}(2^{L_i}-1)$ adet farklı anahtar üretilebilir. Üretilen anahtar dizisi ile herhangi bir LFSR'ın - buna R_1 diyelim; çıktısı arasındaki korelasyon ihtimali $p > 0.5$ veya $(p < 0.5)$ ise ve anahtar dizisinin yeterince uzun kısmı biliniyorsa R_1'in başlangıç konumu bilinen anahtar dizisi ile R_1'in çıktısının bütün olası kaydırılmış halleri arasındaki çakışmaların sayısı ile bulunabilir. Çakışma sayısının korelasyon ihtimali ile tutması gereklidir. Bu durumda R_1'in ilk durumunu bulmak en fazla $(2^{L_1}-1)$ deneme gerektirir. Eğer diğer LFSR'ların çıktıları ile anahtar dizisi arasında korelasyon varsa aynı yöntem kullanılarak ilk durmları elde edilebilir. Sonuç olarak $\sum_{i=1}^{n}(2^{L_i}-1)$ deneme gerekmektedir. Bu ise $\prod_{i=1}^{n}(2^{L_i}-1)$ göre daha küçük bir sayıdır. Aynı şekilde LFSR'ların belli bir kombinasyonu ile çıktı arasındaki korelasyonda analiz için kullanılabilir.

Örnek:

Geffe üreteci olarak bilinen aşağıdaki sisteme korelasyon atağı uygulayacağız.



Geffe üretici

$z = f(x_1, x_2, x_3) = x_1 \oplus x_2(x_1 \oplus x_3)$

x_1	x_2	x_3	$\Rightarrow$	z
0	0	0		0
0	0	1		0
0	1	0		0
0	1	1		1
1	0	0		1
1	0	1		1
1	1	0		0
1	1	1		1

$P(z = x_1) = \frac{6}{8} = 0.75, \;\; P(z = x_2) = \frac{4}{8} = 0.5, \;\; P(z = x_3) = \frac{6}{8} = 0.75$

Görüldüğü gibi f fonksiyonunun çıktıları 0.75 ihtimalle x_1 ve x_3 ile tutuyor. Dolayısıyla f fonksiyonu yeteri kadar çıktısı elde edilirse 3 LFSR'ın başlangıç konumlarını bulunabilir.

8.4 Blok Şifrelerin Analizi

Blok şifrelerin analizinde en kuvvetli analiz metodları olarak bilinen iki analiz yöntemini inceleyeceğiz. Biham ve Shamir tarafından geliştirilen difransiyel kriptanaliz (differential cryptanalysis) ve Matsui tarafından geliştirilen doğrusal kriptanaliz (linear cryptanalysis).

8.4.1 Difransiyel Kriptanaliz

Difransiyel Kriptanaliz methodu DES, GDES, Lucifer, FEAL, PES, IDEA, LOKI'89, REDOC ve Khafre dahil olmak üzere bir çok sayıda blok şifre sistemine uygulanmış bir seçilmiş açık metin atağıdır. Biham ve Shamir tarafından geliştirilen bu atak, ilk önce DES in indirgenmiş döngü çeşitlerine ve sonra tüm 16-döngü DES e uygulanmıştır.

Günümüzde bilinen en önemli ataklardan birisidir çünkü DES in anahtarları teorik olarak tüm anahtar uzayını denemeyle beklenen masraftan daha azı ile elde edilebilinmektedir. Difransiyel Kriptanaliz, kripto sistemlerin yeniden gözden geçirilmesine, tekrar dizyan edilmesi ve yeni sistemlerinin bu atağa karşı dayanıklı tasarlanmalarına neden olmuştur.

Bu kriptanaliz metodu açık metin ikilileri farkının bunlara karşılık gelen kapalı metin ikilileri üzerindeki etkisini kullanarak analiz yapıar. Bu farklar olası anahtarları ihtimal atamak ve ihtimali en yüksek anahtarları belirlemek için kullanılır. Aynı farka sahip olan bir çok açık metin ikilisini ve karşı gelen kapalı metin ikililerini kullanır.

2n-bit blok şifre sistemleri için Difransiyel Kriptanalizin özet tanımını vereceğiz. İlk olarak eşit uzunluktaki iki bit dizinin X ve $\acute{X}$ arasındaki **farkı** (difference) tanımlayalım:

$$\Delta X = X \otimes \acute{X}^{-1}$$

Burada $\otimes$ bit dizi grupları üzerinde, döngü (round) fonksiyonu içinde anahtar ile metin girdisinin birleştirilmesini sağlayan bir grup operasyonudur ve $\acute{X}^{-1} \otimes$ operasyonuna göre X in tersidir. Yukarıdaki farkı tanımlamada asıl amaç metin girdileri arasındaki farkın anahtar eklenmeden ve eklendikten sonra aynı olması yani farkın anahtardan bağımsız yapılması çabasıdır. Bu bakış açısını anlamak için :

$$\Delta X =(X \otimes K) \otimes (\acute{X} \otimes K)^{-1} = X \otimes K \otimes K^{-1} \otimes \acute{X}^{-1} = X \otimes \acute{X}^{-1}$$

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

Açık metinimiz $P = C_0$ ve C_i de i döngü şifrelemesinden oluşan kapalı metin olsun. α_i ΔC_i nin beklenen değeri ve α_0 seçilen $\Delta P = \Delta C_0$ olmak üzere bir **r-döngü karakteristiği** $(r+1)$ lik $(\alpha_0, \ldots, \alpha_r)$ dır. Burada ΔP açık metin farkı ve ΔC_i de i inci döngüden sonraki kapalı metin farkıdır. Bir karakteristiğin olasılığı verilen $i-1$ döngü şifrelemesinden oluşan $\Delta C_{i-1} = \alpha_{i-1}$ farka göre i döngü şifrelemesinden sonra elde edilen $\Delta C_i = \alpha_i$ farkının edilmesinin koşullu olasılığıdır. Rastgele, hep aynı şekilde seçilmiş döngü anahtarları K_i ler için bir karakteristiğin olasılığı

$$\Pr(\Delta C_{i-1}{=}\alpha_i, \Delta C_{i-1}{=}\alpha_{i-1}, \ldots, \Delta C_1 = \alpha_1 \mid \Delta P = \alpha_0)$$

Bu olasılığı hesaplamak çok zor olabilir. Bununla beraber bazı blok şifre sistemleri için bu olasılık her bir döngünün olasılıkları kullanılarak hesaplanabilir (Markov şifre sistemleri). İstatistiksel işlemleri kolaylaştırmak adına bundan sonra döngü anahtarlarının bağımsız ve hep aynı şekilde rastgele seçildiklerini varsayacağız.

Açık metin ikilisi P ve $\acute{P}$ farkı ΔP, anahtar K ve r-döngü karakteristiğine göre **doğru ikili** olarak adlandırılabilinmesi için P ve $\acute{P}$ şifrelendikten sonra arada yer alan döngülerin kapalı metinlerinden oluşan farklar r-döngü karakteristiği izlemelidir. Eğer anahtar K ve r-döngü karakteristiğine göre P ve $\acute{P}$ doğru ikili değilse **yanlış ikili** olarak adlandırılırlar. p karakteristik olasılığı olmak üzere 2n-bitlik şifre sistemi için yaklaşık $p.2^{2n}$ doğru ikili bulunmaktadır.

Difransiyel Kriptanalizin amacı son döngüde kullanılan K_r anahtarını belirlemektir. Bazı açık metin ikilileri için C_r ve $\acute{C}_r$ kapalı metinler olsun. Seçilmiş açık metin atağında kriptanalist, blok şifre sistemin son döngüsü girdileri C_{r-1} ve $C_{r-1}^{'}$ bilemez fakat seçilen

karakteristiğe göre $r-1$ döngü şifre sonundaki kapalı metinlerin farkı ΔC_{r-1} tamamen veya kısmi olarak p olasılıkla bilir. Ve sonra verilen açık metin P ve $\acute{P}$ ikilisinin farkı ΔP için kriptanalist aşağıdaki denklemi K_r yi çözmeye çalışır:

$$g^{-1}(C_r, K_r) \otimes g^{-1}(\acute{C}_r, K_r)^{-1} = \Delta C_{r-1}$$

Yukarıdaki denklemin çözümü aday döngü anahtarları olarak adlandırabileceğimiz $k_1, k_2, \ldots, k_j$ olsun. Eğer P ve $\acute{P}$ doğru ikili ise $K_r \in \{k_1, k_2, \ldots, k_j\}$. Eğer P ve $\acute{P}$ yanlış ikili ise k_i nin K_r den bağımsız olduğunu kabul edilir. Sonrası eğer çok miktarda P ve $\acute{P}$ ikilileri denenirse, aday anahtarların tekrarı kaydedilir ve doğru döngü anahtarı K_r diğer adaylara göre daha fazla sayılmasını bekleriz. Difransiyel Kriptanaliz methodu aşağıdaki basamaklarla özetlenebilir:

- Tamamen veya kısmı olarak yüksek olasılıkta ΔC_{r-1} i veren bir $(\Delta P, \Delta C_1, \Delta C_2, \ldots, \Delta C_{r-1})$ r-döngü karakteristiği bulunması.
- Doğru ikili olduğunu varsaydığımız hep aynı şekilde P ve $\acute{P}$ açık metin ikilisi (farkları ΔP) yardımıyla aday döngü anahtarları $k_1, k_2, \ldots, k_j$, herbiri k_i gözlemlenen çıktı farkını verenler olmak üzere seçilir. Her aday döngü anahtarı k_i için sayaç bir arttırılır.
- Üsteki iki basamak bir aday anahtar k_i diğerlerine göre çok sayıda sayılana kadar tekrar edilir. En çok sayılan k_i gerçek r-döngü anahtarı K_r olarak kabul edilir.

Difransiyel Kriptanalizin karmaşıklığını tanımlamak için anahtar veya döngü anahtarını belirlemek için seçilen farka göre şifrelenen açık metin ikililerin sayısı kullanılabilir.

Sınırlandırılmış DES sürümleri üzerinde Biham ve Shamir (DES kitabi referans olacak) atağın karmaşıklığını yaklaşık olarak c/p bulmuşlardır (p kullanılan karakteristiğin olasılığı ve c sabit sayı ve $2 \prec c \prec 8$ olmak üzere).

Difransiyel Kriptanaliz seçilmiş açık metin atağı olmasına karşın kullanılan metin ikilileri arttırılarak bilinen metin atağında da çalışması sağlanabilir.

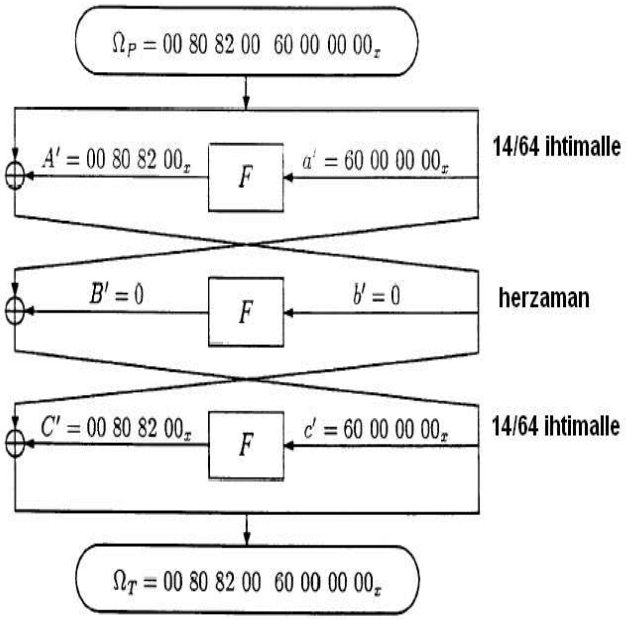
DES şifre sistemini ele aldığımızda yukarıda bahsedilen metodun uygulanması için farklar kullanılarak aşağıdaki gibi bir **XOR** tablosu oluşturulur.

Input XOR	Output XOR															
	0_x	1_x	2_x	3_x	4_x	5_x	6_x	7_x	8_x	9_x	A_x	B_x	C_x	D_x	E_x	F_x
0_x	64	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1_x	0	0	0	6	0	2	4	4	0	10	12	4	10	6	2	4
2_x	0	0	0	8	0	4	4	4	0	6	8	6	12	6	4	2
3_x	14	4	2	2	10	6	4	2	6	4	4	0	2	2	2	0
4_x	0	0	0	6	0	10	10	6	0	4	6	4	2	8	6	2
5_x	4	8	6	2	2	4	4	2	0	4	4	0	12	2	4	6
6_x	0	4	2	4	8	2	6	2	8	4	4	2	4	2	0	12
7_x	2	4	10	4	0	4	8	4	2	4	8	2	2	2	4	4
8_x	0	0	0	12	0	8	8	4	0	6	2	8	8	2	2	4
9_x	10	2	4	0	2	4	6	0	2	2	8	0	10	0	2	12
A_x	0	8	6	2	2	8	6	0	6	4	8	0	4	0	2	10
B_x	2	4	0	10	2	2	4	0	2	6	2	6	6	4	2	12
C_x	0	0	0	8	0	6	6	0	0	6	6	4	6	6	14	2
D_x	6	6	4	8	4	8	2	6	0	6	4	6	0	2	0	2
E_x	0	4	8	8	6	6	4	0	6	6	4	0	0	4	0	8
F_x	2	0	2	4	4	6	4	2	4	8	2	2	2	6	8	8
								⋮								
30_x	0	4	6	0	12	6	2	2	8	2	4	4	6	2	2	4
31_x	4	8	2	10	2	2	2	2	6	0	0	2	2	4	10	8
32_x	4	2	6	4	4	2	2	4	6	6	4	8	2	2	8	0
33_x	4	4	6	2	10	8	4	2	4	0	2	2	4	6	2	4
34_x	0	8	16	6	2	0	0	12	6	0	0	0	0	8	0	6
35_x	2	2	4	0	8	0	0	0	14	4	6	8	0	2	14	0
36_x	2	6	2	2	8	0	2	2	4	2	6	8	0	4	10	0
37_x	2	2	12	4	2	4	4	10	4	4	2	6	0	2	2	4
38_x	0	6	2	2	2	0	2	2	4	6	4	4	4	6	10	10
39_x	6	2	2	4	12	6	4	8	4	0	2	4	2	4	4	0
$3A_x$	6	4	6	4	6	8	0	6	2	2	6	2	2	6	4	0
$3B_x$	2	6	4	0	0	2	4	6	4	6	8	6	4	4	6	2
$3C_x$	0	10	4	0	12	0	4	2	6	0	4	12	4	4	2	0
$3D_x$	0	8	6	2	2	6	0	8	4	4	0	4	0	12	4	4
$3E_x$	4	8	2	2	2	4	4	14	4	2	0	2	0	8	4	4
$3F_x$	4	8	4	2	4	0	2	4	4	2	4	8	8	6	2	2

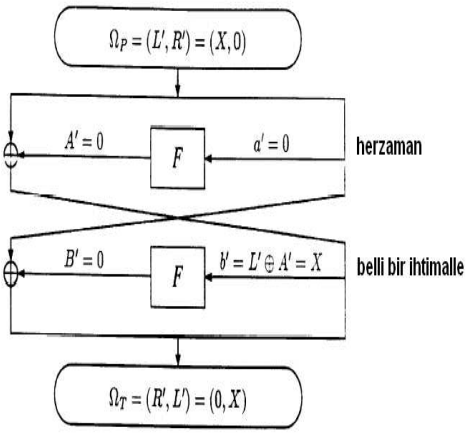
XOR-tablosu (1. S-kutusunun)

DES'in çeşitleri döngü saıyılarına göre atağın başarı durumları aşağıdaki tabloda verilmiştir.

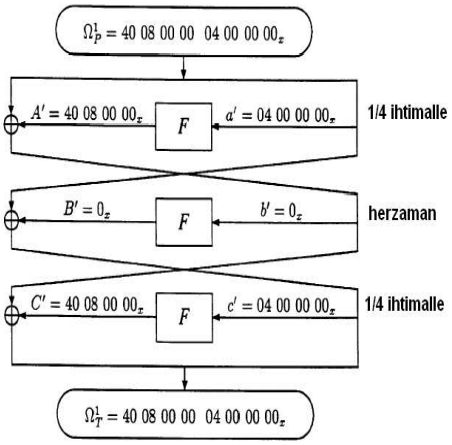
Rounds	Complexity
4	2^4
6	2^8
8	2^{16}
9	2^{26}
10	2^{35}
11	2^{36}
12	2^{43}
13	2^{44}
14	2^{51}
15	2^{52}
16	2^{58}



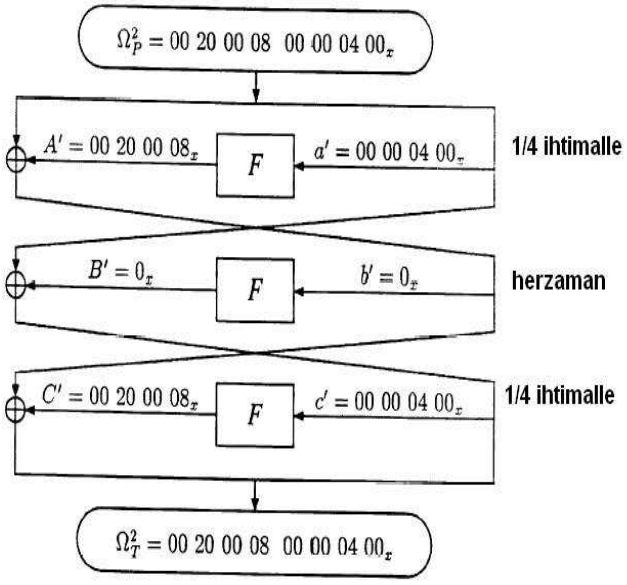
3-round (step) Karakteristik



(iterative) karakteristik tekrar edilebilen



3-round karakteristik



3-round karakteristik

8.4.2 Doğrusal Kriptanaliz

Doğrusal Kriptanaliz 1993 yılında DES sistemini kırmak için Matsui tarafından geliştirilmiş bir bilinen açık metin atak çeşididir. 2^{47} açık metin kullanılarak DES sistemi kırılmıştır. Doğrusal olmayan (nonlinear) fonksiyonlara doğrusal (linear) fonksiyonlarla yaklaşarak yapılmıştır. Bu yaklaşım olasılık üzerine dayalı olduğu için 0.5'ten ne kadar sapılırsa fonksiyonun yerine doğrusal fonksiyonlar kullanmak bu sapma miktarı kadar avantaj kazandırır.

$$P[i_1, i_2, ..., i_a] \oplus C[j_1, j_2, ..., j_b] = K[k_1, k_2, ..., k_c] \ ,$$

$i_1, i_2, ..., i_a, j_1, j_2, ..., j_a$ ve $k_1, k_2, ..., k_c$ belirli bit yerlerini göstermektedir, ve yukarıdaki denklemin tutma ihtimali $p \neq \frac{1}{2}$.

Yukarıdaki efektif doğrusal ifadeye ulaşıldıktan sonra anahtar bitleri aşağıdaki maksimum yakınlık metodu ile bulunur.

Algoritma

Adım 1 : Yukarıdaki denklemin sol tarafının 0'a eşit olduğu açık metinlerin sayısı T olsun.

Adım 2 : Eğer $T > N/2$ (Denenen açık metin sayısı = N),

$\longrightarrow K[k_1, k_2, ..., k_c] = 0$ (eğer $p > \frac{1}{2}$) veya 1 (eğer $p < \frac{1}{2}$)

$\longrightarrow K[k_1, k_2, ..., k_c] = 1$ (eğer $p > \frac{1}{2}$) veya 0 (eğer $p < \frac{1}{2}$)

Aşağıdaki tabloda N ve p cinsinden atağın başarı oranları verilmiştir.

N	$\frac{1}{4}\|p-\frac{1}{2}\|^{-2}$	$\frac{1}{2}\|p-\frac{1}{2}\|^{-2}$	$\|p-\frac{1}{2}\|^{-2}$	$2\|p-\frac{1}{2}\|^{-2}$
Başarı oranı	84.1%	92.1%	97.7%	99.8%

Doğrusal kriptanaliz aşağıda kısaca özetlenmiştir.

- Efektif doğrusal ifadenin bulunması,
- Başarı oranının N ve p cinsinden ifadesi,
- En iyi doğrusal ifadenin ve anahtar için en iyi tutma ihtimalinin hesaplanması.

Kaynaklar:

[1] J.Daeman,R. Govaerts and J. Vandewalle, *Weak Keys for IDEA*, Advances in Cryptology, Proc. EUROCRYPTO'93, LNCS 773, Springer-Verlag, pp. 224-231, 1994.

[2] A. Biryukov, *Introduction to cryptology and cryptanalysis* (Ders Notu), http://www.wisdom.weizmann.ac.il/ albi/cryptanalysis/lectures.htm

[3] X. Lai, J. L. Massey and S. Murphy, *Markov Cipher and Differential Cryptanalysis*, Advances in Cryptology, EUROCRYPTO'91, Lecture Notes in Computer Science, Springer Verlag, Berlin-Heidelberg, 547, pp. 17-38, 1991

[4] X. Lai, *On the design and security of block cipher*, ETH Series in Information Processing, V.1, Konstanz: Hartung-Gorre Verlag, 1992.

[5] *NESSIE Project: New European Schemes for Signatures, Integrity and Encryption at*, http://cryptonessie.org

[6] A. Menezes, P. van Oorschot, S. Vanstone, *Handbook of Applied Cryptography*, http://www.cacr.math.uwaterloo.ca/hac/

[7] L. R. Knudsen, *Ph.D. thesis: Block ciphers - Analysis, Design and Applications* , http://www.mat.dtu.dk/people/Lars.R.Knudsen/thesis.html

[8] *Crypto papers*, http://www.funet.fi/ bande/docs/crypt/

BÖLÜM 9

HASH FONKSİYONLARI

Hash fonksiyonları $h : \{1, 2, \ldots, 2^m\} \rightarrow \{1, 2, \ldots, 2^n\}$ ve asağıdaki özelliklere sahip olan fonksiyonlardır:

1. **sıkıştırma:** h fonksiyonu, uzunluğu sonlu ve değişken olabilen girdiyi alıp sabit bir uzunlukta çıktı vermelidir,
2. **kolay hesaplanabilirlik:** herhangi bir girdi için $h(x)$ değerini hesaplamak kolay olmalıdır.

Hash fonksiyonları anahtarsız hash fonksiyonları ve anahtarlı hash fonksi-yonları olmak üzere ikiye ayrılır:

1. **Anahtarsız hash fonksiyonları** $h : \{0, 1\}^* \rightarrow \{0, 1\}^n$
   - Blok şifreleme sistemleri tabanlı
   - Modüler aritmetik tabanlı
   - Customized (MD4,MD5,SHA-1,RIPE-MD,HAVAL)
2. **Anahtarlı hash fonksiyonları** $h_k : \{0, 1\}^* \rightarrow \{0, 1\}^n$
   - Blok şifreleme sistemleri tabanlı

- Anahtarsız hash fonksiyonları tabanlı
- Customized (MAA,MD5-MAC)
- Akan şifreler için üretilen

Customized hash fonksiyonları sadece hash için kullanılan anahtarlı veya anahtarsız olarak üretilen hash fonksiyonlarıdır. Ayrıca güvenilir-liği teorik olarak ispatlanan fakat pek pratik olmayan evrensel hash fonksiyonlarıda farklı bir grup olarak görülebilir.

Anahtarsız hash fonksiyonlarının üç temel özelliği aşağıda belirtilmiştir(h bir hash fonksiyonu, x ve $x^{'}$ girdileri, y ve $y^{'}$ çıktıları göstermektedir):

1. **preimage resistance:** $h(x) = y$ değeri bilindiğinde, x'i hesaplamak sonlu zamanda mümkün degil. y biliniyor, $h(x^{'}) = y$ olacak bir $x^{'}$ bulmak zor(hesaplamak sonlu zamanda mümkün degil).
2. **2nd-preimage resistance:** $h(x) = y$ biliniyor, $h(x^{'}) = y$ olacak farklı bir mesaj $x \neq x^{'}$ bulmak zor.
3. **collision resistance:** $h(x) = h(x^{'})$ olacak şekilde iki farklı mesaj x ve $x^{'}$ bulmak zor.

Örnek 1 *Mod-32 checksum (Mod 32 kontrol toplamları). Mesajın içerisindeki bütün 32-bit'lik parçaların toplamı alınarak kullanılan fonksiyon. Hesaplaması kolay, sıkıştırma var, fakat preimage resistant değil.*

Örnek 2 *$g(x) = x^2 \ mod \ n = pq$ p, q büyük asal sayılar (n'nin çarpanları bilinmiyorsa tek yönlü fonksiyondur.) Hesaplaması kolay, sıkıştırma yok, preimage resistant (çünkü preimage bulmak n'yi çarpanlarına ayırmaya denk), fakat 2nd preimage ve collision var $(x, -x)$.*

Örnek 3 *DES tabanlı tek yönlü fonksiyon. $f(x) = E_k(x) \oplus x$,sabit bir anahtar(k) için. E rasgele bir permütasyon olarak kabul edilirse f fonksi-yonu tek yönlü olur. y bilindiğinde $y = E_k(x) \oplus x$ olacak şekilde x ve k bulmak zor (E'nin rasgele olamasından dolayı), $E_k^{-1}(x \oplus y) = x$ bulmak zor, Dolayısıyla f tek yönlü bir fonksiyon. Fakat fonksiyon belli mesaj uzunlukları için çalışıyor.*

- collision resistant ise 2nd preimage resistantdır:

  Fonksiyonumuzun collision resistant olduğunu kabul edelim. 2nd preimage resistant degilse $\Rightarrow$ Sabit $x, h(x)$ için $h(x) = h(x')$ olan $x \neq x'$ bulabiliriz, fakat bu collision resistant olamadığını gösteririr, kabulümüzle çelişir.

- collision resistant ise preimage resistant olmak zorunda degildir:

  $g : (0,1)^* \rightarrow (0,1)^n$ collision resistant olsun, h fonksiyonunu aşağıdaki şekilde tanımlanırsa preimage resistant olmaz;

  $$h(x) = \begin{Bmatrix} 1\|x & if \ |x|=n \\ 0\|g(x) & if \ |x|\neq n \end{Bmatrix},$$

  $h : (0,1)^* \rightarrow (0,1)^{n+1}$ $n+1$ bit hash fonksiyon.

- preimage resistant ise 2nd preimage resistant olmak zorunda degildir:

  Örnek 2'de görülebilir.

Ekstra Şartlar:

1. **Non-correlation:** Girdi ve çıktı bitleri arasında korelasyon olmamalı, blok şifre sistemlerindeki gibi avalanche özelliği sağlanmalı(bütün girdi bitleri bütün çıktı bitlerini etkilemeli),

2. **Near-collision resistance:** $w(h(x) \oplus h(x^{'})$ küçük olacak farklı x ve $x^{'}$ çiftlerini bulmak zor olmalı (w:hamming agırlığı),

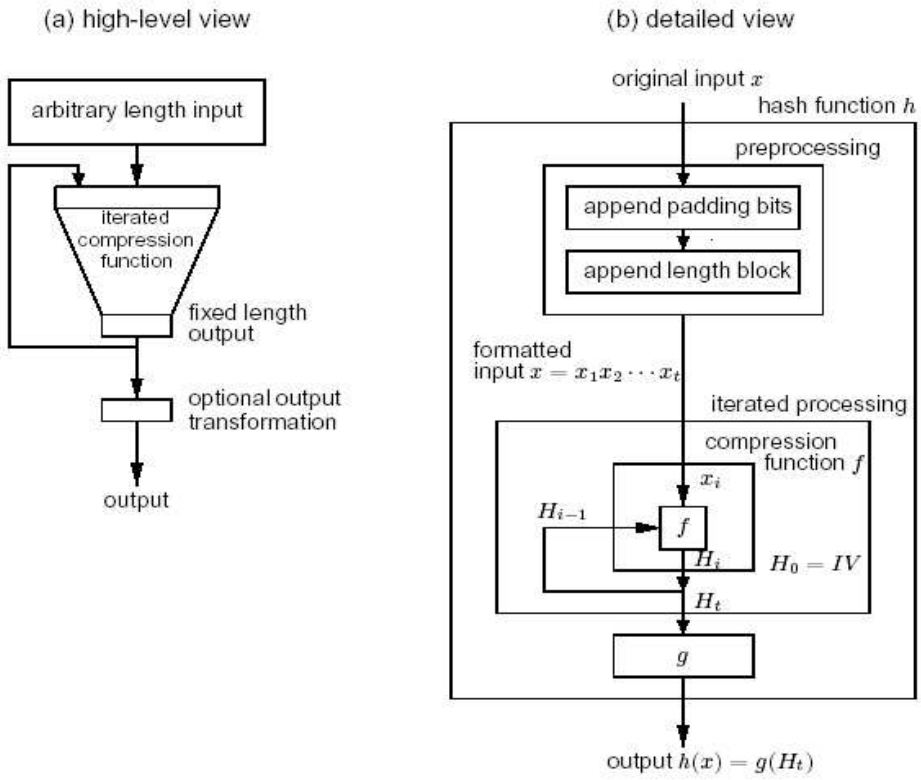
3. **Partial-preimage resistance(local one-wayness):** Girdi bitlerinin bir kısmını dahi bulmak zor olmalı, girdinin t uzunluğundaki kısmını bulmak için yaklaşık $2^t - 1$'lik hesaplama yapmak gerekmeli (girdinin belli bir kısmı bilinse dahi diger kısmını bulmak zor olmalı).

Anahtarsız hash fonksiyonlarının çogu girdi ve çıktı uzunluğu sabit olan bir f hash fonksiyonunun tekrarlı olarak uygulanmasıyla elde edilir. Bu fonksiyonlara **Iterative hash fonksiyonları**(h) adı verilir. Herhangi bir uzunluktaki x girdisi, sabit r-bit uzunluklara bölünür(x_i), x'in uzunluğunun r'nin katı olması için belli bir kurala bağlı olarak x'e padding (bit ekleme) yapılır. Girdi parçaları x_i'ler sırasıyla f'ye sokulur, f'nin çıktısı ve x_{i+1} tekrar f'nin girdisi olarak kullanılır ve son girdi bloğuna kadar bu işlem tekrarlanır. Bu durumda aşağıdaki işlemler yapılmış olur: $x = x_1 x_2 \dots x_t$,

$H_0 = IV$; $H_i = f(H_{i-1}, x_i)$, $1 \leq i \leq t$; $h(x) = g(H_t)$,

IV :başlangıç değeri

Iterative hash fonksiyonlarının genel ve detaylı yapıları aşağıdaki şekillerde verilmiştir:



NOT: f fonksiyonunun collision resistant olması h fonksiyonunun collision resistant olmasını garantiler.

Anahtarsız hash fonksiyonları:

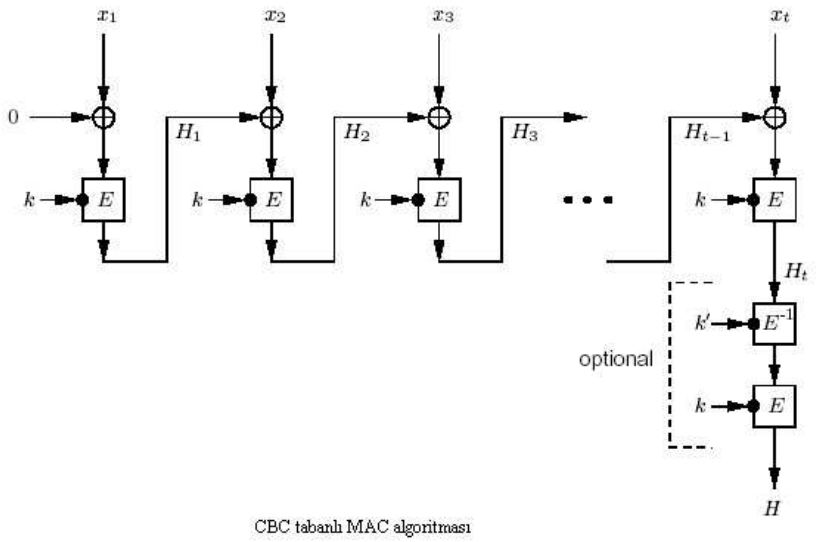
- **Blok şifre sistemleri tabanlı:** İterasyonda kullanılan f fonksiyonu herhangi bir blok şifre sistemi olarak seçilir. Kullanılan makinanın içinde bir blok şifreleme sistemi varsa hash fonksiyonu olarakta kullanılabilir.

- **Moduler Aritmetik tabanlı:** İterasyon fonksiyonu (f) mod M aritmetiğini baz alan bir fonksiyon olarak seçilir, çarpanlara ayırma ve discrete logaritim problemlerini temel alan sistemler seçilebilir.

- **Customized:** Özel olarak hash için tasarlanmış ve optimize hıza sahip olan fonksiyonlardır. Pratik olarak kullanılmaktadır, MD ailesi ve SHA örnek olarak verilebilir. Güvenilirlikleri hesaplama gücüne dayalı olarak ispatlanır, matemetiksel olarak güvenilir oldukları ispatlanmamıştır.

Anahtarlı hash fonksiyonları:

- **Blok şifreleme sistemleri tabanlı:** CBC tabanlı MAC'lar örnek olarak verilebilir.



CBC tabanlı MAC algoritması

- **Anahtarsız hash fonksiyonları tabanlı:** Gizli bir anahtarın anahtarsız hash

fonksiyonlarının girdisinin bir parçası olarak kullanılmasıyla üretilen hash fonksiyonlarıdır.

Aşağıdaki yöntemler örnek olarak verilebilir(k anahtar ve h bir anahtarsız hash fonksiyonu olmak üzere):

1. secret prefix metod: $M(x) = h(k||x)$,
2. secret suffix metod: $M(x) = h(x||k)$,
3. envelope metod with padding: $h_k(x) = h(k||p||x||k)$ p :padding $k||p$ bir blok uzunluğunda olacak şekilde padding yapılıyor,
4. hash tabanlı: $HMAC(x) = h(k||p_1||h(k||p_2||x))$, p_1, p_2 :padding, $k||p_1$ ve $h(k||p_2||x)$ birer tam blok uzunluğunda olacak şekilde padding yapılıyor.

- **Customized MAC'lar:** Sadece hash yapmak için tasarlanmış ve içerisinde gizli anahtar barındıran hash fonksiyonlarıdır. Örnek olarak MAA ve MD5-MAC verilebilir.

Hash fonksiyonları ile ilgili detaylı bilgiler aşağıdaki kaynaklarda bulunabilir:

- Handbook of Applied Cryptography,Chapter 9, by A. Menezes, P. van Oorschot, and S. Vanstone, CRC Press, 1996. http://www.cacr.math.uwaterloo.ca/hac
- Cryptographic Hash Functions: A Survey, by S. Bakhtiari, R. Safavi-Naini, J. Pieprzyk
- Hash functions based on block ciphers: a synthetic approach, by B. Preneel,R. Govaerts, and J. Vandewalle

BÖLÜM 10

TEST YÖNTEMLERİ

Giriş

Rassallığın tanımı kısaca tahmin edilemeyen, belirli bir kalıba sahip olmayan olarak verilebilir. Rassal olarak üretilen sayılar, şans oyunlarında, istatistiksel örneklemelerde ve simulasyon uygulamalarında sıkça kullanılır.

Rassallık kriptografide kullanılan en temel özelliklerden biridir. Atak yapan kişiye, bir kriptosistem çıktısının olabildiğince tahmin edilemez olması gerekir. Rassal sayılar birçok kriptografik uygulamanın temelini oluşturur. Oluşturulması en gerekli ve aynı zamanda en zor olan kısımdır. Neredeyse bütün kriptografik protokollerde gizli ve tahmin edilmesi zor değerlere ihtiyaç duyulur, örneğin asimetrik şifreleme yöntemlerinde (RSA, Diffie Hellman) anahtar oluşturulurken, rassal sayılar kullanılır. Anahtar gizliliği kriptosistemlerde çok önemli olduğu için, programlama dillerinde standart olarak kullanılan rassal sayı üreteçlerinin kriptografik amaçlar için kullanılması sakıncalıdır. Genelde, bu algoritmalar istatistiksel rassallık için tasarlanmıştır, kriptanalize karşı dayanıklı değillerdir). Temel olarak rassal sayılar iki farklı yöntemle oluşturulur:

Gerçek rassal sayı üreteçleri içinde rassal bir yapı bulunduran fiziksel sinyal kaynakları kullanarak dizi üretirler. Bu üreteçlerin en önemli avantajları:

- Dizinin bir kısmına sahipken, farklı bir kısmını elde etmenin mümkün olmaması;

- Üretilen diziler kendi içinde herhangi bir gizli bağıntının bulunmaması;
- Periyodik olmamalarıdır.

Bu avantajların yanı sıra, gerçek rassal sayı üreteçlerinin önemli dezavantajları da bulunur. Bu üreteçler çoğunlukla verimsizdir, uzun sayı dizileri elde etmenin maliyeti yüksektir. Deneyi tekrarlayıp bir sayı dizisini yeniden elde etmek mümkün değildir.

Sözde-rassal (pseudo-random) sayı üreteçleri matematiksel algoritmalar kullanarak diziler üretirler. Bu algoritmalar kendi içlerinde herhangi bir rassallık barındırmazlar, algoritmalar da genelde açıktır. Buradaki rassallık algoritmaların girdileri (seed) ile sağlanır, bu yüzden algoritmaların girdileri gizli tutulmalıdır ve kolay tahmin edilemez olmalıdır. Algoritma ve girdi bilinirse, dizinin tümü elde edilebilir. Bu üreteçler verimlidir ve uzun diziler üretmenin maliyet düşüktür. Kriptografik olarak kullanılabilecek sözde-rassal sayı üreteçleri ile üretilen bir dizinin bir kısmı biliniyorsa, bu dizinin diğer kısımları ile ilgili bir bilgi vermemelidir. Aynı üreteçle üretilen farklı diziler birbirleri ile ilişkileri olmamadılır (correlation). Dizilerin periyotları mümkün olduğunca uzun olmalıdır.

$0-1$ Dizileri

Kritografide kullanılan sayı dizileri 0 ve 1'lerden oluşur. Sayı üretecinin ürettiği her bitin 0 veya 1 olma ihtimali $\frac{1}{2}$ye eşit olmalıdır. Golumb, periyodik bir dizinin rassallığını test eden üç tane kural geliştirmiştir:

1. Dizi içerinse bulunan 0'ların ve 1'lerin sayısının farkı maksimum 1 olmalı.

2. Dizi içerisinde bulunan öbeklerin (kendisini tekrarlayan bitler) sayısı, n dizi uzunluğu olarak verildiğinde, $(n+1)/2$ olmalıdır. Bir bitten oluşan öbeklerin sayısı toplam öbek sayısının yarısı kadar olmalıdır.

3. Dizinin kendisi ile olan ilişkisi düşük olmalıdır.

Bir dizinin rassallığı test edilirken Golumb kuralları yeterli değildir. LFSR'lar ile üretilen sayı dizileri Golumb kurallarını sağlamalarına rağmen kriptografik olarak sayı üreteçleri olarak kullanılmazlar. LFSR ile üretilen sayı dizilerinin lineer karmaşıklıkları düşüktür.

Sözde-Rassal Sayı Üreteçlerinin Test Edilmesi

Sözde-rassal sayı üreteçlerini test etmek için istatistiksel testler kullanılır. Aşağıda istatistiksel hipotez testleri ile ilgili önbilgi bulunmaktadır.

İstatistiksel Testler

İstatistiksel çıkarım yapmak için istatistiksel hipotez testleri kullanılır. Bu testlerde bir hipotez (null hypothesis, H_0) öne sürülür, bu hipotezin tersi de alternatif hipotez, Ha olarak kabul edilir. İstatistiksel test sonucunda varılabilecek iki farklı temel karar vardır: - H_0' yu reddet. - H_0' yu reddetme. Birinci karar, H_0 aleyhine güçlü bir kanıt elde edildiğinde verilir. Bu güçlü kanıt bulunamadığında ise ikinci karar verilir.

Bütün istatistiksel testlerde kaçınılmaz hata yapma payı vardır. Test sonucunda iki farklı hata, birinci tip (alfa) ve ikinci tip (beta) yapılabilir. Birinci tip hata hipotezimiz doğruyken, kararımız H_0'yu reddet olduğunda gerçekleşir. İkinci tip hata ise hipotezimiz yanlışken, kararımız H_0'yu reddetme olduğunda gerçekleşir. Hipotez testinde birinci

tip hata yapma olasılığını sınırlamak gerekir. Test sonucunda birinci tip hata yapma olasılığımız, testimizin güvenilirlik seviyesini verir. Bu değer genel olarak 0.01-0.05 olarak seçilir. İstatistiksel bir testin gücü, ikinci tip hatayı yapmama olasılığına eşittir. Testin gücünü arttırmak için daha fazla örnekleme yapılır.

İstatistiksel bir test yapılacağında ilk olarak, H_0 ve H_a belirlenir. Daha sonra testin güvenilirlik seviyesine karar verilir. Bir örnekleme yapılır ve test istatistiği ve buna bağlı olarak p-değeri hesaplanır. P-değeri, birinci tip hata yapma olasılığını kontrol etmek yerine, H_0'ın doğru olduğu varsayımı ile test istatistiğinin gözlemleme değeri veya daha uç bir değer olması olasılığına karşılık gelir. Bu tanıma uygun olarak hesaplanan olasılık p-değerini verir. Eğer bu değer seçilen güvenilirlik değerinden küçükse H_0 hipotezi reddedilir.

İstatistiksel testlerde en çok kullanılan dağılımlar Normal ve Ki-kare dağılımlarıdır.

Normal Dağılım

Gauss Dağılımı adı ile de bilinen normal dağılım ilk kez De Moivre tarafından bulunmuştur. Genelde, hipotez testleri dağılımın normal olduğu varsayımına göre düzenlenir. Dağılımın ortalama ve standart sapma olmak üzere iki parametresi vardır. Çan eğrisi olarak da bilinir. Eğrinin tepe noktası ortalamasına denk gelir. Ayrıca bu dağılımda ortalama, medyan ve mod aynı değerdir. Ortalamaya göre simetrik bir grafiği vardır. Dağılımın standart sapması eğrinin genişliğini belirler.

Ortalaması sıfır ve standart sapması 1 olan normal dağılıma sahip bir değişkenin dağılımına standart normal dağılım denir. Standart normal dağılıma sahip değişkenler

Z ile gösterilir.

Ortalamadan iki yöne 1,2 ve 3 standart sapma kadar uzaklaşıldığında, toplam alanın sırasıyla %68.26 , %95.44 ve %99.74'ü kapsanır.

Ki-Kare Dağılımı

Standart normal bir dağılımdan seçilen bir birimin x değerinin karesi bir ki-kare değeri olur. Bu şekilde tek bir birimden elde edilen ki-karelerin dağılımı bir serbestlik derecelidir. Standart normal bir dağılımdan seçilen n değerin karelerinin toplamı n serbestlik dereceli bir ki-kare dağılımı olur. Dağılımın şekli serbestlik derecesine göre değişir ve asimetriktir. Sürekli bir dağılıma sahiptir.

NIST Test Paketi

Sayı üreteçlerinin rassallığını ölçmek için bir istatistiksel test yeterli değildir. Bu konuda birçok test paketi üretilmiştir (FIBS 140 - Queensland University, DieHard - Florida State University, NIST). NIST paketinden seçilen bazı testlerin açıklaması aşağıda verilmiştir. NIST paketindeki testler ile ilgili daha ayrıntılı bilgi için : http://csrc.nist.gov/rng/

Frekans Testi

Verilen bir dizide bulunan 0 ve 1'lerin oranını kontrol eder. Testin herhangi bir parametresi yoktur. Testte kullanılan referans dağılım yarım normal dağılımdır. Testin sonunda elde edilen p-değeri çok küçük çıkması, dizideki 1'lerin yada 0'ların sayısının beklenenden fazla olduğunu gösterir. Testin geçerli olabilmesi için dizi uzunluğunun enaz 100 olması gerekir.

Blok Frekans Testi

Verilen bir dizide bulunan 0 ve 1'lerin oranını M bitlik bloklar içinde kontrol eder. Testin tek parametresi blok uzunluğudur (M). Blok uzunluğu 1 olarak alındığında blok frekans testi, frekans testine dönüşür. Herbir bloktaki 1'lerin beklenen oranı $M/2$'dir. Testte kullanılan referans dağılım ki-kare dağılımdır. Testin sonunda elde edilen p-değeri çok küçük çıkması, dizideki bloklarda 1'lerin ve 0'ların oranının $\frac{1}{1}$'den fazlasıyla saptığını gösterir. Testin geçerli olabilmesi için blok uzunluğunun en az 20, dizi uzunluğunun da en az 100 olması gerekir.

Öbek Testi

Dizide bulunan öbeklerin (birbirlerini tekrarlayan bitlerin) sayısını kontrol eder. Test, frekans testinden geçmiş dizilere uygulanır. Dizideki değişimler ne çok hızlı (örn. 01010101), ne de çok yavaş (örn. 00001111) olmalıdır. Testte kullanılan referans dağılım ki-kare dağılımdır. Testin geçerli olabilmesi için dizi uzunluğunun en az 100 olması gerekir.

Bloktaki En Uzun Birler Testi

Test, M-bitlik bloklarda bulunan en uzun birler grubu üzerinde odaklaşır. Testin tek parametresi blok uzunluğudur (M). Dizi M-bitlik n tane bloğa bölünür ve her blok içerisindeki en uzun birler öbeğinin uzunluğuna bakılır. Bu değerlerin frekansları beklenen değerlerle kıyaslanır ve ciddi bir sapma olup olmadığı kontrol edilir. Testte kullanılan referans dağılım ki-kare dağılımdır. Dizi uzunluğuna göre blok uzunluğu ve blok sayısına karar verilir.

Matris Rank Testi

Test içerisinde dizi $M * M$-bitlik matrislere bölünür ve oluşturulan herbir matrisin rankı hesaplanır. Diziden oluşturulan matrislerin ranklarının frekansları hesaplanır, beklenen frekansla kıyaslanır ve ciddi bir sapma olup olmadığı kontrol edilir. Testte kullanılan referans dağılım ki-kare dağılımdır. Dizi uzunluğuna göre matrisin boyutlarına karar verilir. Önerilen $M = 32$ değeri için dizi uzunluğu en az 38,912 bit olmalıdır.

Evrensel Test

Verilen dizinin yeterince sıkıştırılıp sıkıştırılamayacağını kontrol eder. Dizinin fazlasıyla sıkıştırılması, dizinin rassallıktan uzak olduğunu gösterir. Testte, dizi L bitlik bloklara ayrılır. Bu blokların bir kısmı testin başlangıç kısmında uygulanır. Testte kullanılan referans dağılım yarım normal dağılımdır. L-bitlik kalıpların birbirlerini ne kadar sıklıkla tekrar ettiği hesaplanır ve bu değerler beklenen değerler ile karşılaştırılır. Blok uzunluğu 6 seçildiğinde, dizi uzunluğu en az 387,840 olmalıdır.

Lineer Karmaşıklık Testi

Test dizinin rassallık için yeterince karmaşık olup olmadığını kontrol eder. Diziler LFSR çıktıları olarak kabul edilir ve diziyi oluşturabilecek en küçük LFSR'ın boyu küçükse, dizinin rassal olmak için yeterince karmaşık olmadığına karar verilir. Testte dizi M bitlik bloklara ayrılır ve bloktaki bitlerin lineer karmaşıklıkları Berlekamp-Massey algoritması kullanılarak hesaplanır. Hesaplanan lineer karmaşıklıkların beklenen dağılıma uygun olup olmadıklarına bakılır. Testte kullanılan referans dağılım ki-kare dağılımıdır. Testin geçerli olabilmesi için dizinin boyu en az 1,000,000; blok uzunluğu da 500 ve 5000 arasında olmalıdır.

Entropi Testi

Test, m bitlik kesişen blokların frekansları üzerinde odaklaşır ve bu frekansları m ve $(m+1)$-bitlik bloklar için beklenen değerler ile karşılaştırır. Testte kullanılan referans dağılım ki-kare dağılımıdır. Diziden kesişen n tane m-bitlik blok üretilir. Bu blokların frekansları ve entropisi hesaplanır. Aynı işlemler blok uzunluğu $m+14$ için tekrarlanır. m ve $m+1$ bit için hesaplanan değerlerin farkına bağlı test istatistiği hesaplanır. Bu farkın düşük olması rassallıktan uzaklığı gösterir.

Değerlendirme Stratejileri

Verilen rassal sayı üreteci kullanılarak uzunluğu n olan m adet sayı dizisi oluşturulur. Bu m dizi verilen testlere girdi olarak kullanılır. Testlerin sonucunda $m*$(test sayısı) kadar p-değeri hesaplanır. Bu p-değerlerinin analizi sonucunda üretecin sağladığı rassallık hakkında karara varılır. Burada 3 farklı karar verilebilir; (1) Rassallıktan sapma belirlenmedi, (2) Açıkca rassallıktan sapma belirlendi, (3) Belirli bir sonuca varılmadı. Bu kararlar verilirken öncelikle testin güvenilirlik seviyesinden düşük kaç tane p değeri bulunduğu ölçülür, bunun beklenen değeri $m*$(test sayısı)$*$(güvenilirlik seviyesi)'dir. Buna ek olarak bulunan üretecin kabul edilmesi için p-değerlerinin dağılımının da tek düze (uniform) olması beklenir.

BÖLÜM 11

KRİPTOGRAFİK PROTOKOLLER

KRİPTOGRAFİK PROTOKOL NEDİR?

Protokol, basitçe iki veya daha fazla kişi arasındaki önceden belirlenmiş belli bir amaca yönelik haberleşme metodu olarak tanımlanabilir. Kriptografik protokol ise protokol olarak kriptografik bir algoritma içeren bir protokol kastedilir. Ancak genelde amaç temel gizliliğin ötesindedir. Haberleşen taraflar düşman ya da dost olabilirler.

KRİPTOGRAFİK PROTOKOLLERİN ÖZELLİKLERİ

Kriptografik algoritmalarda olduğu gibi bir protokolün de güvensiz olduğunu ispatlamak güvenliliği ispatlamaya göre çok daha kolaydır. Bir protokolü incelerken yine tıpkı algoritmalardaki gibi protokolü nasıl bir cihazda hayata geçireceğimizden çok temel çalışma prensipleri ile ilgilenilir.

HAKEM (arbitrator) ve DÜZENLEYİCİ (adjudicator)

Kimi zaman protokollerde haberleşmenin düzgün bir şekilde işleyebilmesi için hakem olarak adlandırdığımız güvenilir 3. kişilere ihtiyaç duyulur. Hakemlerin en önemli özelliği tüm protokolün onların gözetiminde yürütülüyor olmasıdır. Hakemi çıkardığınız zaman protokol işlemez. Düzenleyici ise hakemin aksine sadece bir anlaşmazlık durumunda başvurulan güvenilir 3. kişilere denir.

DÜZENLEYİCİLİ (adjudicated) PROTOKOLLER

Bu protokoller temelde taraflarin dürüstlüğüne dayanır. Bir anlaşmazlık ya da birisinin hile yapması durumunda düzenleyici kişi bunu farkeder. Denilebilir ki iyi bir düzenleyicili protokolde düzenleyici aynı zamanda hile yapan tarafın kim olduğunu da farkeder. Dolayısı ili bu kişinin varlığı, hile önünde engel teşkil eder.

KENDİ İŞLEYİŞİNİ ZORLAYAN (Self-Enforcing) PROTOKOLLER

Protokol herhangi bir hileye yer bırakmaz. Protokolün sorunsuz işlemesi otomatik olarak herhangi bir hile yapılmadığı anlamına gelir. Hile halinde protokol durur. Devam edilemez bir hal alır. Ve bunu herhangi bir hakem ya da düzenleyiçiye ihtiyaç olmadan sağlayacak şekilde tasarlanmıştır. Bir protokol için her zaman istenen bir özelliktir. Ancak her durumda kendi işleyişini zorlayan bir protokol bulmak mümkün olamayabilir.

AKTİF ve PASİF SALDIRILAR (attacks)

Bir protokol işlerken her zaman saldırı olması olasıdır. Pasif saldırıca kötü amaçlı kişi sadece arada gelip giden trafiği dinleyerek protokol tasarlanırken kendisinin ulaşması umulmayan bir bilgiye erişmeye çalışır. Aktif saldırıda ise kötü niyetli kişi sadece dinlemekle kalmaz. Mesajları ya da kayıtları okumanın ötesinde kesebilir, bozabilir ya değiştirebilir. Aktif saldırgan tamamen dışardan birisi olmak zorunda değildir. Sistemdeki başka legal bir kullanıcı ve hatta sistem yöneticisi olabilir. Saldırganın protokolü uygulayan taraflardan birisi olması durumunda ise daha çok hile ve saldırıyı uygulayan kişi

içinse hilekar tabirleri kullanılır. Saldırganlarda olduğu gibi hilekarları da pasif hilekar ve aktif hilekar olarak ikiye ayırmak mümkündür.

TİPİK SİMETRİK ALGORİTMA HABERLEŞME PROTOKOLÜ

Simetrik kriptografi kullanılarak yapılan tipik bir haberleşmede A ve B kişileri bir kriptosistem ve bir anahtar üzerinde anlaşırlar ve A kişisi bu anahtarı kullanarak şifrelediği mesajını B kişisine gönderir. B kişisi de yine aynı anahtarı kullanarak mesajı deşifre eder ve haberleşme tamamlanır.

Bu sistemin önemli dezavantajları vardır. Öncelikle A ve B kişiler anahtar değiştirmek için biraraya gelmelidirler (anahtar değişimi için başka bir algoritma kullanılmadığını varsayarsak). Araya giren kötü niyetli Z kişisi mesajlara ulaştığı taktirde mesajın diğer tarafa gitmesini engelleyebilir ve hatta anahtarı bilmesi durumunda bu mesajı kendisininkilerle değiştirerek iki haberleşme tarafını da farkında olmaksızın bambaşka mesajlar gönderebilir.

TİPİK AÇIK ANAHTAR KRİPTOSİSTEMİ İLE HABERLEŞME

A ve B kişileri bir açık anahtar kriptosistemi üzerinde anlaşırlar. Simetrik anahtarlı sistemde olduğu gibi burdada kriptosistem üzerinde anlaşma gizli yapılmak zorunda değildir. B kişisi A kişisine kendi açık anahtarını gönderir. A kişisi mesajını B'nin açık anahtarı ile şifreler ve gönderir. B kişisi ise kendi gizli anahtarı ile mesajı çözer ve mesaja ulaşır.

Açık anahtarlı tipik haberleşme sistemlerinin en önemli dezavantajı ortalama olarak simetrik bir algoritmadan 1000 kat yavaş olmalarıdır. Uzun mesajları bu yolla göndermek

pratik değildir.

TİPİK KARMA (hybrid) KRİPTOSİSTEMLER

B, A'ya açık anahtarını gönderirir. A bir oturumda kullanılmak üzere rastgele bir oturum anahtarı üretir ve B'nin açık anahtarı ile şifreleyerek B'ye gönderir. B kişisi ise gizli anahtarı aracılığıyla edindiği oturum anahtarını açar ve daha sonra bu anahtarı belirledikleri simetrik kriptosistemde kullanarak haberleşmelerini sağlarlar.

Tahmin edileceği üzere bu sistemde taraflar, simetrik kriptosistem anahtarı belirlemek için biraraya gelmek zorunda kalmamaktadırlar.

HASH FONKSİYONLARI

Hash fonksiyonlarının kriptografide birazdan bir miktar inceleyeceğimiz üzere çok geniş bir kullanım alanı vardır. Hash fonksiyonları, öncelikle tek-yönlü (one-way) fonksiyonlardır. Yani bir verinin fonksiyon altında görüntüsünü hesaplamak kolaydır ama görüntüden fonksiyonun tersi aracılığıyla ana veriyi elde etmek hesaplama gücü anlamında zordur. Hash fonksiyonları ile elde ettiğimiz değer, yani hash değeri, fonksiyon girdisinin değişken boyuta sahip olmasına karşın sabit boyuta sahiptir ve genelde hash değeri, girdiye göre çok daha ufak boyuttadır. Kullanılan Hash fonksiyonlarından birbirine çok benzer girdi değerlerini için dahi çok farklı çıktılar üretmesi beklenir. Ve yine önemli bir özellik olarak hash fonksiyonlarından çakışmasız (collision-free) olmaları umulur. Yani hash fonksiyonumuz altında aynı hash değerini veren iki girdinin bulunması hesaplama gücü göze alındığında çok zor olmalıdır.

Yukarda bahsedilen özellikler ışığında hash değerleri, verilerin parmak izi olarak düşünülebilir. Dolayısıyla birisinde olan bir dökümanın sizde de olduğunu dökümanı o kişiye göndermeden ispat etmek isterseniz hash değerini ona söyleme yolunu kullanabilirsiniz. Nitekim dokümanın elinizde gerçekten olmaması durumunda o dökümana ait hash değerini üretmeniz çok zordur. En çok kullanılan hash fonksiyonları arasında SHA ve MD5 algoritmaları sayılabilir.

GÜNLÜK HAYATTA İMZA

Günlük hayatta sıkça kullandığımız imzalarda aradığımız bizim için çok önemli özellikleri gözden geçirelim;

1) İmza otentiktir(authentic), imzalayanın kimliğini gösterir
2) İmza çoğaltılamaz.
3) İmza tekrar kullanılamaz.
4) İmza değiştirilemez. Üzerinde oynama yapılamaz.
5) İmza atan kişi attıktan sonra imzasını inkar edemez.

AÇIK ANAHTAR KRİPTOSİSTEMLERLE DİJİTAL İMZA

Tipik bir açık anahtar kriptosisteminde A kişisi kendi gizli anahtarı ile imzalamak istediği dökümanı şifreler. B kişisi ise A'nın açık anahtarı ile dökümanı açar. Eğer açamazsa imza geçerli değildir, imza birisi tarafından ya da doğal etkenlerle bozulmuştur.

Bu basit imza protokolüne kötü taraftan bakacak olursak imzalı dökümanı alan kişi dijital ortamda bu imzalı dökümanı dilediği kadar çoğaltabilir. İmzalı döküman bir anlaşma

metni ise bu elbette ciddi bir sorun teşkil etmeyebilir ancak dökümanın para kaşılığı olan bir çek olduğunu kabul edersek durum farklı olabilir. B kişisinin farklı zamanlarda farklı çeklermiş gibi aynı imzalı çeki bozdurmasını istemeyiz. Bu gibi tekrar kullanımları önlemek için dijital dökümanlar çoğu zaman hangi tarihe ait olduklarını gösteren bir zamanpulu (timestamp) ile beraber kullanılır. Mesaj algoritma ile imzalanıp kapatılmadan önce içine tarih bilgileri de eklenir.

AÇIK ANAHTAR KRİPTOSİSTEM VE HASH FONKSİYONUYLA İMZA

A kişisi bu defa dökümanın kendisini (zamanpulu ile veya zamanpulsuz) imzalamak yerine dökümanın bir hash fonksiyonu sonucu üretilmiş hash değerini imzalar. Ve A kişisi imzalamak üzere kullandığı dökümanla beraber imzalanmış hash değerini B'ye gönderir. Alıcı olan B ise imzayı A'nın açık anahtarı ile açtıktan sonra dökümanın hashini kendisi hesaplar ve bu değerin gönderilen hash değeri ile uyup uymadığına bakar. Eğer uyuyorsa imza geçerlidir. Aksi halde geçersizdir.

Bu protokol dökümanın kendisi yerine sadece hash değerini imzaladığımız için dökümanın boyutuna bağlı olarak bir öncekine göre inanılmaz ölçüde hızlı olabilir. İki farklı dökümanın hash değerlerinin örneğin 160-bit'lik bir çıktı üreten bir hash fonksiyonunda $1/2^{160}$ olduğu düşünülürse hash değerini imzalamakla dökümanın kendisini imzalamanın rahatlıkla eşleştirilebileceği ortaya çıkar. Ancak hash fonksiyonunun temel özellikleri sağlaması gerektiği unutulmamalıdır. Şayet tek-yönlü olmayan bir hash fonksiyonu kullanılsaydı, bir döküman imzalandığı zaman onunla aynı hash değerini veren tüm dökümanlar da beraberinde aynı kişi tarafından imzalanmış olacaktı.

Bu metodla A kişisi bir dökümanı açığa çıkarmadan dökümana sahip olduğunu ispatlaya-

bilir. Örneğin 'copyright' hakkına sahip olmak isteyeceğimiz bir dökümanı dökümanın kendisini açığa çıkamadan imzalayabiliriz.

DİJİTAL İMZALARDA İNKAR EDEMEMEZLİK (nonrepudiation)

Yukardaki algoritmanın eksik bir yanı, belgeyi imzalayan kişiye her zaman hile olanağı vermesidir. Şöyle ki, eğer dökümanda bir zamanpulu mevcut değilse A kişisi dökümanı imzaladığını inkar etmek istediği bir durumda her zaman için gizli anahtarını herhangi bir dijital ortamda bırakarak daha sonra da gizli anahtarının çalındığını ve dökümanın kendisi değil de imzasını çalan kişi tarafından imzalandığını öne sürebilir. Bu gibi durumlarda çoğu zaman 3. güvenilir kişi olan hakemlerin kontrolünde olan bir protokole ihtiyaç duyulabilir.

İNKAR EDİLEMEYEN DİJİTAL İMZALAR

Protokolümüz bu defa bir parça daha karışık ve de 3. güvenilir T kişisine dayanıyor. Şöyle ki; A kişisi mesajı imzalar ve mesaja kendisini tanımlayan bir başlık ekler. Elde ettiği bu yeni birleştirilmiş mesajı tekrar imzalar ve T kişisine gönderir. 3. kişide herkesin açık ve gizli anahtarı bulunmaktadır dolayısıyla dıştaki mesajı A kişisinin açık anahtarı ile açarak A kişisinin kendisini tanımladığı eklentiye erişir. T daha sonra mesajı gerçekten a kişisinin gönderdiğini B'ye onaylamak için mesajın ne zamana ait olduğu bilgisini de ekleyerek imzalar ve hem A'ya, hem de B'ye gönderir. B kişisi T'nin imzasını açar ve A'nın kimlik bilgilerine ulaşarak imzayı onaylar. İmzanın A'ya da gönderiliyor olmasının sebebi bir anlamda A'ya "Senin adına şöyle bir mesaj gönderildi, şayet bunu gerçekten

gönderen sen değil de (örneğin anahtarını çalan) bir başkası ise acele konuş, daha sonra bunu inkar edemezsin" denmesidir.

Bahsedilebilecek bir nokta da şudur ki bu protokolde zaman bilgileri açısından T'ye güvenilmektedir. Dolayısı ile T ile bir şekilde anlaşan birisi dökümanının tarihi hakkında hileye girişebilir. Protokolde B'nin A'da mesajı aldıktan sonra gerçekten ona dair olup olmadığını onaylaması için T'ye sorması gibi değişiklikler yapılması suretiyle alternatifleri de türetilebilir.

ŞİFRELEME ve DİJİTAL İMZA

A ve B kişilerinin her ikisinin de açık ve gizli anahtarlarının bulunduğu bir ortamda bir mesajı hem şifrelemek ve hem de imzalamak istediğimiz bir durumda A kişisi, mesajı, basitçe söylemek gerekirse önce kendi gizli anahtarı ile imzalar ve daha sonra da imzalanmış mesajı B'nin açık anahtarı ile şifreleyerek B'ye gönderir. B de mesajı kendi kapalı anahtarı ile deşifre eder ve daha sonra da A'nın açık anahtarı ile imzaıyı onaylar. Şayet mesajı sorunsuz bir şekilde aldığını A'ya iletmek isterse A'nın yaptığı işin tersini yaparak mesajı tekrar A'ya gönderebilir. Açıkça ifade etmek gerekirse, B kişisi mesajı kendi gizli anahtarı ile mesajı imzalayıp daha sonra da A'nın açık anahtarı ile imzayı şifreleyerek mesajı A'ya gönderebilir. A'nınsa geri ters operasyonlarla gönderdiği orjinal mesajı elde etmesi durumunda işlemlerin başarıyla gerçekleştiği kabul edilir.

Bu operasyonlarda mesajın şifrelenmeden önce imzalanması doğal gözükmektedir. Tersi bir sıra uygulanması durumunda tıpkı kağıt yerine zarfı imzalayan birisinin, bir başkasına zarfın içinden kağıdı çıkararak başka bir kağıt koyup, koyduğu kağıdı imzalanmış göstermesi gibi bir imkan verdiği açıktır.

Burada dikkat edilmesi gereken bir nokta da şudur ki bu protokol de bir takım ataklara maruz kalabilir. Örneğin başkalarından gelen rasgele mesajların imzalanmaması gerekir. Aynı şekilde başkalarından gelen rasgele mesajların deşifre edilip sonucun başkalarına verilmesi de direkt olarak saldırıya meydan verebilmektedir. Ancak genel olarak şifreleme ve imzalama için farklı algoritmalar kullanıldığı taktirde ya da aynı algoritma kullanılsa bile bu iki operasyon için farklı anahtarlar kullanıldığı zaman bu gibi basit bir saldıyıa meydan verilmediği söylenebilir. Benzer şekilde zamanpulu eklentisi de mesajları farklılaştırdığı için hash fonksiyonlarını burada kullanmak saldırılara karşı iyi bir çözüm olabilmektedir. Protokol, değişik amaçlar ve ortamlar doğrultusunda şekillendirilebilir.

ANAHTAR DEĞİŞİMİ (Key Exchange)

Kriptografide en temel uygulamalardan birisi anahtar değişimidir. Standart şifreli haberleşme için, karşılıklı anahtar değişimi başarı ile gerçekleşmişse, çoğu zaman herhangi bir güvenilir 128-bit (yada daha geniş) blok şifre kullanılarak iletişim sağlanabilir. Ancak değinildiği gibi öncelikli olarak anahtar değişiminin güvenilir bir şekilde gerçekleşmesi gereklidir. Sadece belli bir oturum (yada haberleşme seansı) için kullanılan anahtara *oturum anahtarı* denir.

Anahtar değişimi için simetrik algoritmalar kullanılırsa, güvenilir üçüncü kişinin de yardımıyla anahtar değişimini gerçekleştirmek mümkündür. Ancak burada, ilk aşamada güvenilir 3. bir kişiye ihtiyaç bırakmayan, açık anahtar metoduyla gerçekleştirilen ve çok daha işlevsel olan bir yöntem ele alınacaktır.

AÇIK ANAHTAR METODUYLA ANAHTAR DEĞİŞİMİ

Bu uygulamada A kişisi önce B'nin açık anahtarını temin eder, rastgele oluşturduğu oturum anahtarını B'nin açık anahtarı ile şifreleyerek B'ye gönderir. B kişisi ise kendi gizli anahtarı ile A'nın mesajını açar ve haberleşmelerini gönderilen oturum anahtarını kullanarak, simetrik bir algoritma aracılığı ile yürütürler. Burada açık ve kapalı anahtarlar bir açık anahtar algoritmasına, oturum anahtarı ise bir simetrik şifreleme algoritmasına (DES, SAFER, Vigenere vb) aittir.

Burada pasif bir saldırgan için, yani herhangi bir müdahalede bulunmayan kötü niyetli kişi E için kullanılan açık anahtar algoritmasını kırmaya çalışmaktan başka bir yol görünmemektedir. Ancak aktif saldırgan için aynısı söz konusu değildir. Aktif saldırgan, eğer iki kişi arasındaki mesajlara erişme ve hatta değiştirme imkanına sahipse, B kişisi A'ya açık anahtarını gönderirken araya girip ona kendi açık anahtarını gönderebilir. A kişisi de oturum anahtarını, B'nin anahtarı sandığı bu anahtarla şifreleyerek geri gönderdiğinde ise, kötü niyetli kişi zaten kendi açık anahtarı ile şifrelenmiş bu mesajı rahatlıkla açıp okuyabilir. Dahası E, mesajı B'nin gerçek açık anahtarı ile tekrar şifreleyerek B'ye göndererek onların hiçbirşey olmamışcasına anahtarı kendisinde olan bir simetrik algoritmayla haberleşmelerini sağlayabilir. Aktif saldırgan, hissedilir bir yavaşlamaya sebep olmadığı sürece bu şekilde A ve B kişilerinin haberi bile olmadan onların şifreleyerek gönderdikleri tüm mesajları okuyabilir. Ortadaki 3. kişi tarafından gerçekleştirilen bu saldırıya "ortadaki-adam saldırısı" (man-in-the-middle attack) adı verilir.

Burada bir nokta ise protokolün adım adım işlemesi yerine A kişisi tek bir gönderimle, B'nin açık anahtarı ile şifrelediği oturum anahtarını, o oturum anahtarı ile şifrelediği mesajla birlikte B'ye gönderebilir. B de önce kendi gizli anahtarı ile oturum anahtarını elde eder, sonra da bu anahtarı kullanarak simetrik algoritma ile şifrelenmiş mesajı açarak orjinal mesaja ulaşır.

ARAKİLİT PROTOKOLÜ (Interlock Protocol)

Yukarıda değinilen ortadaki-adam saldırısı, A ve B kişilerinin gerçekten birbirleri ile konuştuklarını onaylama şansları yokken kullanılabilir. Arakilit protokolünde A ve B kişileri birbirlerine karşılıklı olarak açık anahtarlarını gönderirler. Burada aktif saldırgan hala araya girip tarafların açık anahtarlarını kendi belirlediği bir açık anahtar ile değiştirebilir) Ancak sonrasında protokol farklı işler. A kişisi oturum anahtarını (veya herhangi bir mesajı da olabilir) yine B'nin açık anahtarı ile şifreler ancak bu defa şifrelenmiş mesajın kendisi yerine *yarısını* B'ye gönderir. Aynı şekilde B kişisi de kendi şifrelediği mesajın ilk yarısını A'ya gönderir. Ve daha sonra, yine önce A kişisi ve sonra da B kişisi olmak üzere her iki taraf da şifrelediği mesajın kalan yarısını birbirine gönderirler ve iki yarımı birleştirerek kendi gizli anahtarlarıyla mesajın hepsini deşifrelerler.

Burada mesajın ikinci yarısı olmadan ilk yarısı kullanışsızdır. Taraflar da ortadaki adam gibi mesajın ikinci yarısı gelmeden şifreli mesajı deşifre etme şansına sahip değildirler. Burada ortadaki-adamı safdışı bırakan etken nedir? Aktif saldırganın daha önce açık anahtarları kendisininki ile değiştirerek daha sonra şifreli metinlere nasıl eriştiğini hatırlayınız.

Ancak burada açık anahtar ile giden mesaj, mesajın tamamı değil. Dolayısıyla saldırgan, mesajı kendi gizli anahtarı ile açıp B kişisinin açık anahtarı ile tekrar şifreleme şansına sahip değil. Çünkü şifreli mesaj *yarım* olduğu için mesajı herhangi bir şekilde açma/ deşifre etme şansına sahip değil. Peki bu özelliği sağlayan yarım mesaj nedir? Mesajı iki 'yarım'a bölmek için nasıl bir mekanizma kullanılabilir?

Burada kullandiğımız mesajı iki parçaya bölüp gönderme metodundaki parçalardan ilki şifreli mesajın bir hash fonksiyonu ile elde edilmiş bir hash değeri olurken ikincisi ise şifreli mesajın kendisi olabilir örneğin. Ya da gerçekten bir 64-bit blok şifre algoritması ile şifrelenmiş mesajın 32'şer bit uzunluğunda iki yarısı olabilir. Her iki durumda da ikinci mesaj gelmeden deşifreleme yapılamayacaktır. Ve yine her iki durumda da aktif saldırganın 2. şifreli mesaj yarısını da bilmeden mesajı değiştirmesine olanak yoktur. İkinci şifreli mesaj, taraflardan birisi tarafından gönderildiğinde ise çoktan mesajın onay için kullanılacak olan ilk yarısı diğer tarafın eline ulaşmıştır. Dolayısıyla ortadaki adam için saldırı vakti çoktan geçmiştir. Nitekim protokolün işleyişi bunu gerektirmektedir. Arakilit protokolü Rivest ve Shamir tarafından geliştirilmiştir.

DİJİTAL İMZALARLA ANAHTAR DEĞİŞİMİ

Şu ana kadar mesajımızın kendisini ya da mesajımızı iletmede kullanacağımız simetrik algoritmaya ait anahtarımızı göndermekte kullandığımız metodlarda açık anahtarımızı gönderirken aktif bir saldırganın her zaman için gerçek dünya uygulamaları ile son derece örtüşen bir ihtimal olan araya girip tarafların açık anahtarlarını değiştirmek suretiyle

çeşitli saldırılarda bulunabileceğini kabul ettik. Bu sebeple açık anahtarları dağıtmanın nasıl daha güvenli bir şekilde olabileceği sorusuna cevap olarak 3. güvenilir T kişisinin imzasını da protokole dahil edelim. Şöyle ki; basit bir ifade ile, A ve B kişileri anahtarlarını direkt olarak birbirlerine göndermeleri yerine, 3. güvenilir T kişisi onlar için anahtar belirleyip bu anahtarı imzalayabilir. Ve bu şekilde A ve kişileri sadece T kişisinden imzalı anahtarlar kullanmak üzere karşılıklı anlaşmışlardır. Ve T kişisinin güvenli olduğunu, yani onun anahtar veri tabanına kimse tarafından erişilemediğini kabul edersek (ki bu belli ve tek bir anahtar dağıtıcısının güvenliği sağlamak ayrı ayrı pek çok kullanıcının güvenliğini sağlamaktan her zaman için daha kolaydır) aktif saldırgan araya girip açık anahtarları kendi anahtarı ile değiştiremez.

Bu arada her zaman için protokol detayları üzerinde düzenlemeler yaparak protokolde faydalı değişiklikler yapabilmemizin olası olduğunu unutmayalım. Örneğin yukarda açık ve gizli anahtarlarımızı ilk aşamada bizim için T kişisinin değil de kendimizin ürettiğini ve T kişisinin sadece bizim ürettiğimiz anahtarların imzası için protokolde yer aldığını düşünürsek, anahtarı göndereceğimiz B kişisi anahtarımızda her zaman T'nin imzasını arayacağı için ve aktif saldırgan da dijital imza metodlarının güvenilirliği ölçüsünde T kişisinin imzasını değiştiremeyeği için açık anahtar gönderimlerimizi çok daha güvenilir bir şekilde yürütebiliriz. Üstelik bu durumda aktif saldırgan T'nin dijital anahtarını ele geçirse bile bu anahtarı A ve B taraflarının mesajlarını okumakta kullanamaz, sadece yeni açık anahtarlar imzalayabilir. Çünkü T'de sadece bizim açık anahtarlarımızı imzalamada kullandığı bir dijital imza anahtarı bulunmaktadır.

OTENTİKASYON(kimlik doğrulama)

Bir bilgisayara kendimizi tanıtmak için kullandığımız yegane metod parola kullanmaktır. Ancak ilk kez Needham ve Guy ikilisinin ortaya koyduğu şekliyle otentikasyon yapan bilgisayarda parolaların kendilerinin bulunmasına gerek yoktur. Dolayısıyla sistem yöneticisinin (ya da sizin parolalarınıza direkt erişme şansına sahip birisinin) parolanızı farklı amaçlar için kullanmasını (örneğin aynı parolayı kullandığınız başka bir sisteme sizin adınıza erişmesini) önemli ölçüde zorlaştırmaya yol açan gerçek şudur ki, bilgisayar için gerekli olan içinde tüm kullanıcı parolalarının bulunması değil, sadece yanlış girilen parolaları diğerlerinden ayırt edebilmesidir. Bu da hash fonksiyonları sayesinde gerçekleşmektedir, şöyle ki; kullanıcı otentikasyon yapılacak bilgisayara parolasını girer, bilgisayar bu parolanın hash değerini hesaplar ve kendi veri tabanındaki aynı kullanıcı için tutulan hash değeri ile uyup uymadığını kontrol eder.

Burada otentikasyon yapılan bilgisayara erişimi olan bir saldırgan hala şunu yapabilir; parola olarak kullanılabilecek çok miktarda sözcük üreterek bunların hash değerlerini hesaplar ve veri tabanındakilerle karşılaştırır. Bu metodla yeterli miktarda süreye sahip olduğu taktirde tüm olası kelimeleri deneyerek parolalara ulaşabilir. Bu operasyonu pratik olarak zorlaştırmak için *tuzlama* (salting) işlemi uygulanır. Tuzlama, parolalara hash fonksiyonundan geçirmeden önce rastgele metin eklemektir. Örneğin UNIX sistemlerde 12-bit tuzlama uygulanmaktadır. Tuzlama metoduyla otentikasyon işlemini önemli oranda yavaşlatmaksızın bir sözlük saldırısını çok ciddi biçimde zorlaştırmak mümkündür. Ancak her zaman için kolay tahmin edilebilir parolalar böyle bir saldırıda önce kırılanlar

olacaktır ve tuzlama onları daha güçlü hale getirmemektedir.

AÇIK ANAHTAR KRİPTOGRAFİ İLE OTENTİKASYON

Otentikasyonun çok uzaktaki bir makina tarafından yapıldığı bir ortam düşünelim. Örneğin bizden birkaç ülke uzaklıkta olan bir internet sitesine kendimizi tanıtmak için kullandığımız parolamızın açık bir şekilde onca yolu gitmesi ne kadar sağlıklı olabilir? Bunun için yukardaki basit otentikaston protokolümüzü açık anahtar sistemli otentikaston kullanacak nasıl değiştiririz?

Açık anahtar kullanan otentikaston metodunda Biz parola yerine kendi gizli anahtarımızı bilmekteyiz. Makinada ise bizim açık anahtarımız bulunur. Otentikasyon şöyle işler; makina bize rastgele bir metin gönderir. Biz de gizli anahtarımızla bu metni şifreler ve makinaya geri göndeririz. Makina, bizim açık anahtarımız ile gönderdiğimiz mesajı açtığında kendi gönderdiği rastgele metni bulursa bize sistem erişimi verir. Bu metodun kullanıldığı ortamda, ne otentikasyon makinasının (gizli anahtarımız sadece bizde bulunuyor, makina veritabanında mevcut değil), ne de iletişim yolunun (açık anahtar kriptosisteminin temel özelliğinden dolayı) güvenli olması gerekmemektedir.

SIR PARÇALAMA (secret splitting)

Bir bilgiyi iki kişi arasında paylaştırmak istediğimizi düşünelim. Öyle ki; her ikisi de biraraya gelmeden bilgiye ulaşamasınlar. Paylaştırılacak olan M mesaji için aynı bit uzunluğunda R gibi bir rastgele metin oluşturduğumuzu düşünürsek $M \oplus R = S$ şeklinde

bir xorlama operasyonu ile S metini elde ettiğimizi düşüenlim. Bu durumda bir kişiye R'ı ve de diğer kişiye de S'i verdiğimizi düşünürsek her ikisi de biraraya gelmeden M mesajını elde edemezler. Biraraya geldiklerinde ise $R \oplus S = M$ ile mesajı geri dönüştürebilirler.

Bu basit metodu ikiden fazla kişide uygulamak içinse bir sırrı 4 kişi arasında paylaştırmak istediğimizi düşünelim. Bu defa sır metni ile yine aynı uzunlukta R,S ve T rastgele metinlerine ihtiyacımız olacak. $M \oplus R \oplus S \oplus T = U$ şeklinde bir operasyonla, R,S,T ve U metinlerini farklı 4 kişiye dağıtarak ancak biraraya geldiklerinde tüm sahip oldukları 4 metni xorlayarak M'i elde etmelerine izin verebiliriz. Dahası bu 4 kişiden birisini safdışı bırakmak istediğimizde onun metinini diğerlerine dağıtmamız yeterli olacaktır. Bu sayede o olmadan da bu xor operasyonunu yaparak ana metne ulaşabilirler.

SAKLI İLETİŞİM YOLU (subliminal channel)

Birisiyle haberleşmek istediğiniz, ancak bunun 3. bir başka kişi aracılığıyla gerçekleşmesi dışında mümkün olmadığını bir senaryo düşünelim. Ve bu 3. kişi de mesajın şifreli olmasına izin vermiyor, kendisi mesajınızı iletmeyi ancak o içeriğini okuyabildiği zaman kabul ediyor. İşte böylesi bir durumda bir saklı iletişim yolu metodu kullanımı çözüm olabilir. Yani ilettiğimiz mesajda gizli bir algoritmayla saklanmış, görünenin ardındaki başka bir mesajla hem diğer tarafa mesajımızı ulaştırıp hem de 3. aracı kişinin yalnızca görünürdeki mesajı gönderdiğimizi sanmasına yol açarak gerçekte ne gönderdiğimiz hakkında hiç bir fikri olmamasını sağlayabiliriz.

Çok basit bir saklı iletişim yolu metodu olarak, kurduğumuz cümlelerde tek sayıda kelime olması durumunda 1-bitini, çift sayıda kelime içeren cümleler ile de 0-bitini gönderdiğimizi düşünebiliriz. Saklı iletişim yoluyla, tıpkı steganografide olduğu gibi herhangi bir anahtar bulunmamakta ve mesajın gizliliği tamamen algoritmanın gizliliğine dayanmaktadır. Saklı iletişim yollarına çok çeşitli uygulama alanları bulmak olasıdır. Örneğin dijital imzalama sırasında saklı iletişim yolu metodları ile mesaji bir taraftan imzalarken diğer taraftan imzanın içine "ben bu mesajı baskı altında imzalıyorum" gibi bir ifade saklamak mümkündür.

BİT VADETME (bit commitment)

Şöyle bir senaryo düşünelim; Bir yatırımcıya hisse senedi önereceğiz. Yükselecek olan hisseleri önerebilirsek, karşılığında ondan bir iş alacağız. Ancak önerdiğimiz hisse senetlerini hemen bilmesini istemiyoruz, çünkü o zaman bize işi vermeden o senetlere kendisi para yatırıp bizi yüzüstü bırakabilir. Ancak diğer taraftan, yatırımcı da bizim samimiyetimize inanmak istiyor. Eğer hisse senedi sonuçları belli olduktan sonra ona açıklarsak, sonucu baştan tahmin ettiğimize dair onu ikna etme şansımız kalmaz. Özetle durum şu ki; A kişisi, B kişisine, belli bir tarihte belli bir bilgiyi bildiğini aradan bir süre geçtikten sonra ona ispatlamak istiyor, ve kendisi izin verene kadar bilginin ne olduğuna dair B'ye ipucu vermek istemiyor.

B kişisi rastgele bir R metni üretir ve bunu A'ya gönderir. A'nın belli bir grup bit içerisinden seçimini gösteren bite 'b' dersek A kişisi, simetrik bir algoritma kullanarak

kendi belirlediği bir anahtar ile (R,b) ikisisini şifreler ve geri B'ye gönderir. Ve seçtiği biti açığa çıkarmak istediği gün geldiğinde ise şifrelemede kullandığı anahtarı B'ye vererek onun paketi açmasını sağlar. B kişisi ise kendi ilk başta gönderdiği rastgele R metninin pakette seçim biti ile beraber bulunup bulunmadığını kontrol eder. Eğer protokolde böyle bir rastgele R metni olmasaydı, sözkonusu olan yalnızca bir bit olabileceği için A kişisi son aşamada gönderdiği paketi rahatlıkla istediği bite (veya hatta bit grubuna) deşifre edebilen bir anahtar gönderebilirdi. Ancak paketin içinde fazladan bir rastgele metnin farklı anahtarlı deşifre etmeler için bozulacak olması A kişisini böylesi bir hileden alıkoyar.

SIFIR-BİLGİ İSPAT METODU (zero-knowledge proof)

Sıfır bilgi ispat, kısaca belli bir bilgiye sahip olduğumuzu, o bilginin ne olduğunu açığa çıkarmadan ispatlamamızı sağlayan metodtur. Şöyle ki, ispatlayacağımız problemi onunla eşyapıda (isomorphic) başka bir probleme dönüştürerek problemin kendisi yerine yeni problemi ispatlıyoruz. Ancak dönüştürdüğümüz problemi, öyle bir şekilde seçiyoruz ki bu yeni problemin çözümü orjinal problemin çözümü hakkında bilgi vermiyor. Burada orjinal problem hakkında bilgi vermeyecek sözü ile ifade edilen şu ki; hesapsal olarak karşı tarafın yeni çözümü orjinal problemin çözümüne dönüştürmesi, en az orjinal problemin kendisini çözmesi kadar zor olmalıdır. Ancak diğer taraftan, bizim aslında çözümüne sahip olmadığımız bir probleme eşyapıdaki bir problem için çözüm sahibiymişiz gibi davranmamız olasılığına karşı diğer taraf her seferinde bize şu soruyu soruyor. "Ya bu eşyapıdaki problemin çözümünü bana ver, ya da onun asıl probleme eş yapıda olduğunu göster". Ve eğer iddiamızda haklı isek bu iki isteği de gerçekleştirebiliriz. Ancak bu iki önermeden birini

gerçekten sağlayamıyo olmamız, yani hile yapmamız durumuna yer bırakmamak içinse bu işlemler n kez (karşı taraf ikna oluncaya kadar) tekrarlanır. Öyle ki, gerçekten orjinal problemin cevabını bilmediğimiz bir durumda her defasında sorulan soruya doğru cevabı verme olasılığımız $1/2$ dir. n kez tekrarlanma durumunda ise bu şekilde hile ihtimalimiz $1/n$ e düşer ki yeterli tekrardan sonra ihmal edilebilecek bir olasılıktır. Sıfır-bilgi ispat'ın dayandığı nokta ise bu olasılığın düşüklüğüdür. Dolayısıyla sıfır-bilgi ispatta yeterli miktarda tekrar için mutlak manada bir ispattan çok karşı tarafın iknası söz konusudur.

Sıfır-bilgi ispat metodları ne yazık ki her matematiksel probleme aynı verimde uygulanamamaktadır. Kimi zaman ispat için çözümün bir kısmını açığa çıkarmak zorunda olduğumuz problemler sözkonusudur. Bu yüzden sıfır bilgi ispat metodları, karşı tarafın sorusuna ne kadar sıklıkta doğru cevap verebileceğimiz ölçüsüne göre kusursuz, istatistiksel, hesapsal ya da kullanışsız gibi kategorilere ayrılmıştır.