T.C.
SELÇUK ÜNİVERSİTESİ
FEN-EDEBİYAT FAKÜLTESİ
MATEMATİK BÖLÜMÜ

KRİPTOLOJİ

(Mezuniyet Çalışması)

Danışman
Yrd. Doç. Dr. Aynur KESKİN

Hazırlayan
Süleyman ÖĞREKÇİ
030320047

KONYA-2007

ÖNSÖZ

Selçuk Üniversitesi Fen-Edebiyat Fakültesi Matematik Bölümü Mezuniyet Çalışması olarak hazırlanan bu tezde genel kavramlarıyla "Kriptoloji" konusu incelenmiştir.

Kriptoloji alanı uygulamalı matematikte yoğun olarak çalışılan, günlük yaşantımızı yakından ilgilendiren bir alandır. Teknolojinin çok hızlı ilerlemesiyle, kişisel verilerimizin şifrelenerek istenmeyen kişiler tarafın elde edilmesini engelleme gereği kriptolojinin de çok hızlı gelişmesini sağlamıştır. Bu konu matematiğin hemen hemen her dalıyla bağlantılı olup matematiğin güzel bir uygulamasıdır. Bu çalışma hazırlanırken okuyucunun yeterli matematik kültürüne sahip olduğu, özellikle Sayılar Teorisi, Lineer Cebir ve Soyut Cebir konularına hakim olduğu varsayılmıştır.

Kriptoloji branşı çok geniş bir branştır. Bu çalışmada konu özet olarak verilmiş olup yeterli sayıda örnekle beslenmiştir. Kriptoloji, kriptografi ve kriptanaliz olmak üzere iki branşa ayılmaktadır. Kriptografi; şifre yazımı, Kriptanaliz ise kriptografiyi analiz edip şifreleri çözme işlemlerini inceler. Bu çalışmada Kriptografi detaylı olarak incelenmiş olup, Kriptanaliz konusu da kısaca özetlenmiştir. Kiptografik sistemler (kriptosistemler) de incelenirken blok şifreleme yöntemleri denilen sistemler incelenmiş olup, akan şifreler denilen diğer sistem incelenmemiştir. Bu çalışmada kriptografik sistemler anahtar yapılarına göre iki sınıfa ayrılarak incelenmiştir. Kriptosistemleri çeşitli özelliklerine göre değişik şekillerde sınıflandırmak mümkündür. Ayrıca kriptoloji branşına bağlı olmayan, diğer veri gizleme teknikleri olan Hash Fonksiyonları ve Stegonagrafi konusu da kısaca özetlenmiştir.

Bu çalışmamın hazırlanmasında yardımlarını esirgemeyen Sayın Hocam Yrd. Doç. Dr. Aynur KESKİN'e teşekkürlerimi sunarım.

Süleyman ÖĞREKÇİ

İÇİNDEKİLER

1. GİRİŞ

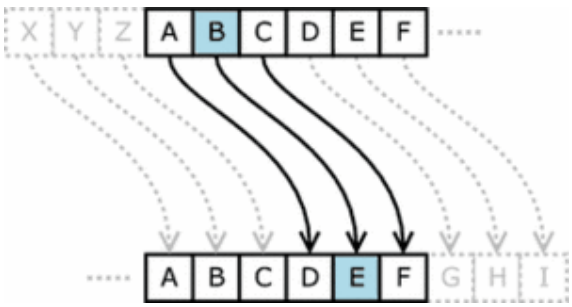
Kriptoloji, şifre bilimidir. Çeşitli iletilerin, yazıların belli bir sisteme göre şifrelenmesi, bu mesajların güvenlikli bir ortamda alıcıya iletilmesi ve iletilmiş mesajın deşifresiyle uğraşır.

Günümüz teknolojisinin baş döndürücü hızı göz önünde alındığında, teknolojinin gelişmesiyle ortaya çıkan güvenlik açığının da taşıdığı önem ortaya çıkmaktadır. Kriptoloji; askerî kurumlardan, kişiler arası veya özel devlet kurumları arasındaki iletişimlerden, sistemlerin oluşumunda ve işleyişindeki güvenlik boşluklarına kadar her türlü dalla alakalıdır. Kriptoloji bilimi kendi içerisinde iki farklı branşa ayrılır *Kriptografi*; şifreleri yazmak ve *Kriptanaliz*; şifreleri çözmek ya da analiz etmek ile uğraşır. Biz daha çok Kriptografi ile ilgileneceğiz ama kriptanalize de değineceğiz.

Modern Kriptoloji temelde üç görevi yerine getirmek için kullanılır. Verinin okunmasını ve değiştirilmesini engelleme ve verinin belirtilen kişi tarafından gönderildiğinin garanti altına alınması. Bir *hacker* yada kötü niyetli herhangi bir kişi internet gibi güvenli olmayan ortamlarda iki merkez arasındaki haberleşmeleri dinleyebilir ve bu haberleşmelerde gönderilen veriler üzerinde işlemler yapabilir. Bu işlemler genelde üç şekilde olabilir. Verinin gitmesini engelleme (intercept), veriyi sadece okuma (read), veriyi değiştirme (modify). Bu tip işlemlerin yapılmasını engellemek amacıyla şifreleme bir bilim olarak çalışmaktadır.

Kriptolojik yapılara basit birkaç örnek verecek olursak tarihten günümüze kullanılan bazı şifreleme teknikleri şunlardır:

Tarihin ilk kriptolojik fikirleri İngilizcede *transposition and substitution cipher* adını taşır, yani yer değiştirme ve harf değiştirme şifrelemesi. Bu yöntemlerden ilki bir yazıdaki harflerin yerlerini değiştirerek, ikincisi ise harfleri başka harflerle değiştirerek elde edilir. Bu şifrelemeyi kullanan belki de



en ünlü teknik Sezar Şifresi'dir: bu şifrede, her harf o harften birkaç sonraki harf kullanılarak yazılır. Örneğin, 3 harf atlamalı Sezar Şifresi'nde "deneme" yerine "ggrğpğ" yazılır. Öte yandan, Sezar Şifresi kırmak son görece kolay olmaktadır: bir filolog bir dilde en çok geçen harfleri bulabilir. O harfler ile mesajda en sık geçen harfleri karşılaştırarak hangi harfin hangi harf ile değiştirildiği bulunabilir. Bu adımların ardından, mesaj çözülmüş olur.

Sezar şifrelemesinin ardından en popüler şifreleme yöntemlerden biri Rotasyonel Şifreleme (*Rotor Machine*) olmuştur. Bu makinelerin en popüler örneği, Nazi Almanyası'nın İkinci Dünya Savaşı sırasında kullandığı *Enigma makinesi* adlı cihazdır. Rotasyonel şifre makinesinin en önemli özelliği, birkaç rotor'un bir araya getirilmesiyle birlikte şifrelemenin dinamik olarak değiştirilebilmesidir: örneğin, ilk harfler bir şifreleme çeşidi, ikinci harfler başka bir şifreleme çeşidi, üçüncü harfler ise başka bir şifreleme çeşidiyle şifrelenebilir. Bu tür şifrelemelerin kırılması için en popüler harfler yerine en popüler harf sekansları bilinmelidir; örneğin İngilizcede NG ve ST harflerine arka arkaya sık rastlanır.

Şimdi Biraz da kriptografik yapıların üzerinde duralım. Krpitografik yapılar nasıl çalışır, iş akış şeması nasıldır, çeşitler nelerdir gibi soruların üzerinde duralım.

Kriptografik yapılar, yani kriptosistemler; verilen bir verinin şifrelenmesi ve alıcı tarafından deşifre edilmesi için yapılan işlemler dizisidir.

Açık bir verini şifrelenmesi temel olarak: verinin, daha önceden belirlenmiş ve adına şifreleme algoritması denilen, bir işlem dizisinden geçirilmesiyle olur. Aşağıdaki şemada temel kriptografi şeması verilmiştir. Şemada; M: Açık metin (veri), E: Şifreleme için kullanılacak olan önceden belirlenmiş işlemler dizisi (şifreleme algoritması), C: M olarak verilen verinin, E şifreleme algoritmasından geçtikten sonraki şifrelenmiş hali, olarak kullanılmıştır.

Bir kriptografik yapının temel işleyiş mekanizması.

Örnek 1.1: üç harf atlamalı Sezar algoritmasında **S** harfinin şifrelenmesini sembolize edersek:

$$M = ''S'' \quad , \quad E = M \oplus_3 \quad , \quad C = ''\dot{U}''$$

Olarak ifade edebiliriz. Burada $\oplus_3$ işlemini, verilen harfin alfabeye göre üç harf ilerisindeki harfi döndüren işlem olarak tanımladık. ($M + 3(mod\ 29)$)

Kriptanaliz ataklarına karşı kriptografi zamanla gelişmekte, yeni yapılar geliştirilmektedir. Günümüzde kullanılan kriptosistemler temel olarak yukarıda bahsettiğimiz yapıda olmakla beraber biraz daha karmaşık yapılardadırlar. Günümüzde kriptografik yapıların çoğu ***blok şifreleme*** ve ***akan şifreleme*** denilen yapılardadır. Biz akan şifreler üzerinde durmayacağız. Detaylı olarak blok şifreleme yapılarını inceleyeceğiz. Şimdi bu yapı üzerinde biraz duralım.

1.1 Blok Şifreleme Yapıları:

Blok şifrelemenin en basit tanımı; verilen açık verinin tamamını şifreleme algoritmasından geçirerek şifrelemek yerine, veriyi parçalara ayırıp, her parçayı ayrı ayrı şifreleme algoritmasından geçirip her parçanın şifrelenmiş halini birleştirip tek bir şifrelenmiş çıktı elde etme yöntemi, olarak verilebilir. Bu yöntemde blok uzunluğu ne kadar büyük ise sistem o kadar güvenlidir.

Örnek 1.2: "**Senay'a kitabı sen ver**" cümlesini, blok uzunluğu 3 olacak şekilde bölüp herhangi bir şifreleme algoritmasıyla şifrelersek:

Açık Metin: sen-aya-kit-abı-sen-ver

Şifreli Metin: axk-bcg-xkt-ase-axk-hyt

Şeklinde sonuç alınıp şifreli çıktı: **axkbcgxktaseaxkhyt** olarak bulunur.

Örnek 1.3: "Matematik" sözcüğünü üç harf atlamalı Sezar kriptografisine göre şifreleyip sistemin işleyişini sembolize edelim:

Şifrelenecek veri Matematik sözcüğü ve Sezar kriptografisi verinin her elemanına, yani her harfine, ayrı ayrı işlem uyguladığı için veriyi harflere bölmemiz gerekiyor. O halde, bir blok bir harf olacak şekilde veriyi bloklara ayıralım:

$$M_1 = m,\ M_2 = a,\ M_3 = t,\ M_4 = e,\ \ldots, M_9 = i,\ M_{10} = k$$

Olarak veriyi tanımlayabiliriz. Şimdi şifreleme algoritmasını:

$$E = M_n \oplus_3$$

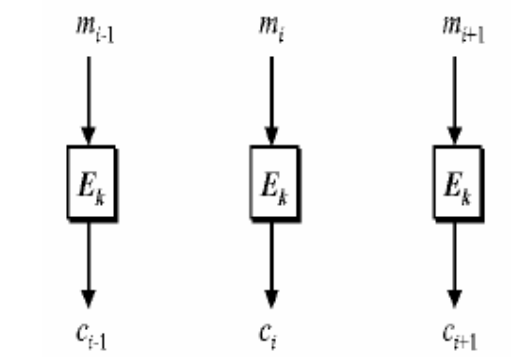
Olarak tanımlayalım, burada $\oplus_3$ işlemi: karakteri alfabeye göre 3 harf ilerisindeki harfe dönüştüren işlem olarak tanımlanmıştır.

Sonuç olarak işlemin bize döndüreceği şifrelenmiş çıktı ise: **Pdvhpdvln** olacaktır.

Yukarıdaki örneklere dikkat edilirse: "**Senay'a kitabı sen ver**" cümlesinde "**sen**" harf dizisi iki defa geçmektedir. Buna karşılık "**sen**" bloğunun şifrelenmiş çıktısı olan "**axk**" bloğu da çıktıda iki defa geçmektedir. Aynı şekilde diğer örnekte "**Matematik**" sözcüğünde "**mat**" dizisi iki defa geçmektedir, burada tekrarlayan blok olmamasına rağmen yine de çıktıda "**mat**" dizisinin çıktısı olan "**pdv**" çıktısının tekrarlandığını görmekteyiz. Bu problem, blok şifreleme yapılarının önemli bir güvenlik problemi olmuştur. Bu problemi çözmek için zaman içinde değişik blok şifreleme yapıları geliştirilmiştir. Bunlardan bazıları aşağıda verilmiştir.

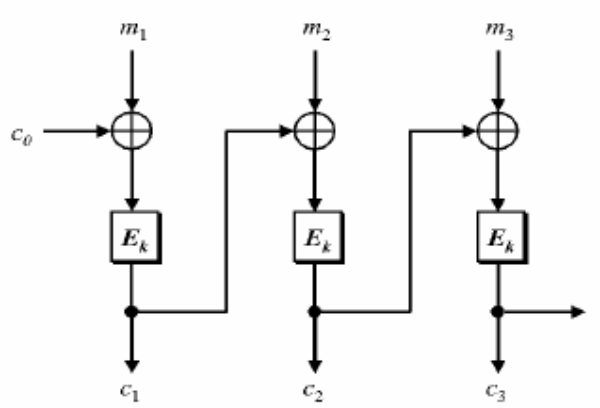
1.1.1 Elektronik Kod Modeli (Electronic Code Mode):

Bu model temel blok şifreleme modelidir. Yukarıda bahsettiğimiz örnekler bu yapıdadır. İşleyiş şemasını gösterirsek:



Elektronik Kod Modeli.
M_{i-1}, M_i, M_{i+1} *blokları,* E_k *algoritmayı,*
C_{i-1}, C_i, C_{i+1} *çıktıları sembolize etmektedir.*

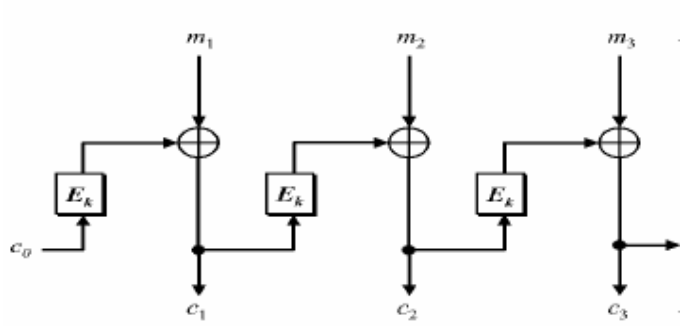
1.1.2 Kapalı Metin Zincirleme Modeli (Cipher Block Chaining Mode):



Kapalı zincir modeli iş akış şeması

Şemada da görüldüğü gibi bu modelde bir blok şifreleme algoritmasına tabi tutulmadan önce bir önceki bloğun şifrelenmiş çıktısı ile bir işleme sokuluyor. Bu işlem herhangi bir işlem olabilir (ekleme, toplama, çıkarma, …). Daha sonra algoritmaya sokuluyor. Böylece her blok şifrelenirken bir önceki bloktan etkileniyor. Sonuçta her bloğun şifrelenmesinde önceki blokların etkisi olduğu için veride tekrarlanan bloklar çıktıda tekrarlanmıyor.

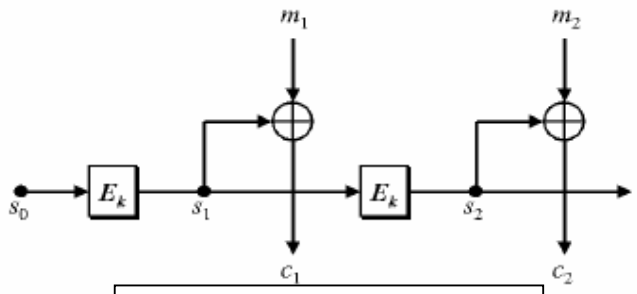
1.1.3 Çıktıyı Geri Besleme Modeli (Output Feedback Mode):



Çıktıyı geri besleme modeli şeması

Bu yöntemde de her blok şifrelendikten sonra bir sonraki veri bloğuyla işleme sokulup daha sonra şifrelenmektedir. Böylece tekrarlanan veri bloklarına karşın çıktıda tekrarlanan blokların önüne geçilmiş olur.

1.1.4 Girdiyi Geribesleme Modeli (Input Feedback Mode):



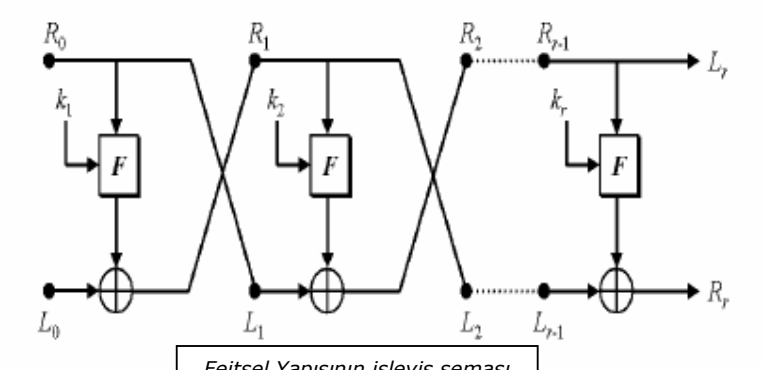
Girdiyi geri besleme modeli şeması

Bu yöntemde ilk veri bloğu, her bloğun şifrelenmesinden önce algoritmaya tabi tutulup şifrelenecek blokla işleme sokuluyor. Böylece veri ile çıktı arasındaki tekrar benzerlikleri engelleniyor.

1.1.5 Feıstel Yapıları:

Blok şifreleme yapısının özel bir hali olan feistel yapıları modern kriptografide çok büyük önem taşır. Günümüzde kullanılan DES gibi çok önemli kriptosistemler feitsel yapıdadır (DES sistemi üzerinde ileride ayrıntılı olarak durulacaktır).

Horst Feistel tarafından tasarlanan Fiestel yapısı şifrelenecek bloğun iki parçaya bölünmesi ve her aşamada sadece biri üzerinde işlem yapılması ve bu işlemin sonucunun da bir sonraki aşamada verinin ikinci yarısına etkimesi esasına dayanan sarmal bir yapıdır.



Feitsel Yapısının işleyiş şeması

Şemada da görüldüğü gibi veri bloklara ayrılmış ve her blok da R_n ve L_n gibi iki parçaya bölünmüştür. Burada F şifreleme algoritmasıdır. k_n'ler de F algoritmalarının birbirinden farklı olmasını sağlayan anahtarlardır. Her adımda anahtarlar değiştiği için her adımdaki algoritmalar da değişmektedir (anahtar yapıları hakkında ilerde detaylı olarak durulacaktır).

2. KRİPTOSİSTEMLER

Şifreleme sistemleri (kriptosistemler), açık anahtarlı (asimetrik) 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.

Simetrik sistemlerde ise tek bir anahtar hem şifreleme hem de deşifreleme amacıyla kullanılır. Güvenli bir şekilde iletişim kurmadan önce gönderici ile alıcının gizli anahtar olarak adlandırılan bir anahtar üzerinde uzlaşmaları gerekir. Simetrik sistemlerde temel problem, göndericinin ve alıcının üçüncü bir kişinin

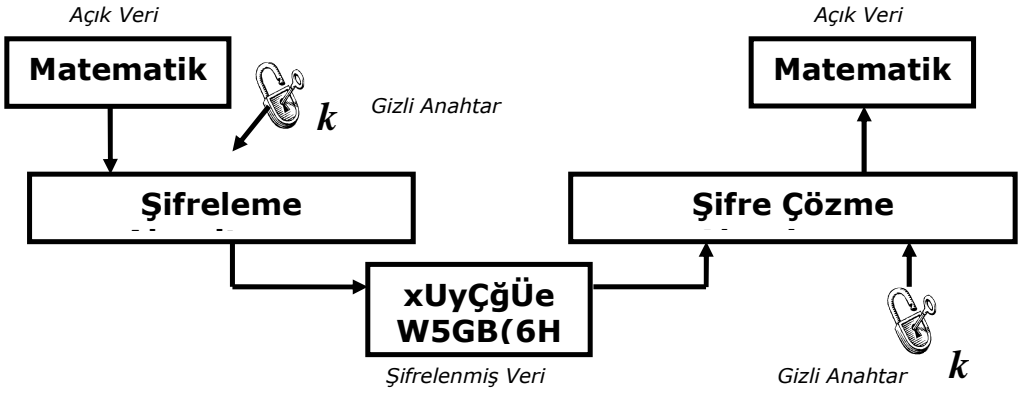
eline geçmesini engelleyerek ortak bir anahtar üzerinde anlaşmalarıdır. Ancak simetrik sistemlerin avantajı da, açık anahtarlı sistemlere göre daha hızlı olmalarıdır. Bir sistemin güvenliği anahtarda yatar. Şifre çözmeye yönelik ataklar anahtarı bulmaya yöneliktir.

Şimdi bu iki yapı üzerinde detaylı olarak duracağız.

2.1 Gizli Anahtarlı Kriptografi:

Bu tür sistemlerde hem şifreleme hem de çözme işlemi için tek bir anahtar kullanılır. Aynı zamanda bu teknik; şifreleme işleminde kullanılan anahtarı, gönderen ve alıcıdan başka kimsenin bilmemesi gerektiği için, *gizli anahtar*, *anonim anahtar* ya da *özel anahtar* şifreleme olarak da adlandırılır. Ayrıca şifreleme ve çözme işlemlerinde tek bir anahtarla birbirinin simetriği olan algoritmalar kullanarak gerçekleştirildiğinden *simetrik şifreleme* teknikleri olarak da bilinirler. Bu şifreleme tekniğini kullanan algoritmalara örnek olarak **XOR Şifreleme**, **DES**, **3DES**, **IDEA**, **DESX**, **SKIPJACK**, **RC2**, **RC4**, **RC5** algoritmaları verilebilir. Bunların en meşhuru Veri Şifreleme algoritması (Data Encryption Standart) olarak da bilinen DES'tir. DES 64 bit blok boyutu olan bir blok şifrelemedir. 64 bitlik düzyazı bloklarını 56 bitlik anahtarlar kullanarak 64 bitlik şifreli yazı bloklarına çevirir. Simetrik şifreleme tekniğinde gizli anahtarın taraflar arasında iletilmesi problemi bulunmaktadır. Bu gizli anahtar taraflar arasında iletilirken istenmeyen kişiler tarafından ele geçirilebilir. Simetrik şifreler ise hız avantajına sahiptir; genellikle asimetrik şifrelerden 100, 1000 kat arası daha hızlı çalışırlar. Bu avantaj simetrik şifreleri yoğun ve hız gerektiren mesajlaşmalarda tek seçenek haline getirmiştir. Ancak başka bir yere güvenli bir mesaj göndermek istendiğinde, anahtarı karşı tarafa ulaştırmak gerekmektedir.

SİMETRİK ŞİFRELEME SİSTEMİ ŞEMASI



Şekilde de görüldüğü gibi simetrik şifreleme sisteminde veriyi şifrelemek için de şifreli metni deşifrelemek içinde aynı anahtar kullanılır.

Bir simetrik şifrenin kuvveti anahtar uzunluğu ile doğru orantılıdır. 40 bit genelde zayıf kabul edilirken 128 bit ve üzeri bir anahtar uzunluğu kuvvetli kabul edilir. Aşağıdaki tabloda Simetrik şifreleme sisteminde güvenlik ile anahtar uzunluğu arasındaki bağıntı net olarak vurgulanmıştır.

Anahtar uzunluğu	**Üretilebilen anahtar sayısı**		**Bir deşifreleme için gereken zaman**	
32 bit	2^{32}	$= 4.3 \times 10^{9}$	2^{31} µs	= 35.8 dakika
56 bit	2^{56}	$= 7.2 \times 10^{16}$	2^{55} µs	= 1142 yıl
128 bit	2^{128}	$= 3.4 \times 10^{38}$	2^{127} µs	$= 5.4 \times 10^{24}$ yıl
168 bit	2^{168}	$= 3.7 \times 10^{50}$	2^{167} µs	$= 5.9 \times 10^{36}$ yıl
26 karakter	26!	$= 4 \times 10^{26}$	2×10^{26} µs	$= 6.4 \times 10^{12}$ yıl

Şimdi simetrik şifrelemeye sistemlerine birkaç örnek verelim.

2.1.1 Sezar Şifreleme Sistemi:

Daha önce de incelediğimiz bu sistem basit bir simetrik şifreleme sistemidir. Bir örnek üzerinde görelim:

Örnek 2.1 :"Matematik" sözcüğünü üç harf atlamalı Sezar kriptografisine göre şifreleyip sistemin işleyişini sembolize etmiştik:

$$M_1 = m,\ M_2 = a,\ M_3 = t,\ M_4 = e,\ \ldots, M_9 = i,\ M_{10} = k$$

bloklar ve

$$E_k = M_n \oplus_k$$

Şifreleme algoritması olsun. Burada k :gizli anahtarımızdır. Bu sayı şifreleme işleminde harfin kaç harf ilerisindeki harfle yer değiştireceğini belirtmektedir. $\oplus_k$ işlemi de k sayısı kadar harf atlatarak algoritmayı uygulayan işlemdir.

$k = 3$ seçip şifrelersek alacağımız çıktı **Pdvhpdvln** olacaktır.

Şifreli mesajı ilettiğimiz bir diğer kişi gizli anahtarımızı kullanarak açık veriye ulaşırken kullanacağı algoritma şifreleme algoritmasının tersi olacaktır. Yani:

$$E_k = M_n \oplus_{-k}$$

Algoritmasını kullanarak "Matematik" sözcüğünü elde edecektir.

Örnekten de net olarak anlaşıldığı gibi simetrik şifreleme sistemiyle şifrelenmiş bir veriyi, gizli anahtarı bilen herkes deşifre edebilir. Bu da anahtarın çok güvenli taşınması problemini ortaya çıkarmıştır. Gizli anahtarın güvenliği sağlanabildiği sürece bu sistem güvenli sayılır.

2.1.2 Vıgenere Şifreleme Sistemi:

Gizli anahtarlı sistemlere verebileceğimiz bir başka örnek de Vigenere sistemidir. Metod uygulanırken n harften oluşan bir anahtar kelime seçilir, bu kelimenin harflerinin dizisi $l_1, l_2, \ldots, l_n$ ve sayısal karşılıkları da $k_1, k_2, \ldots, k_n$ olmak üzere, şifrelenecek metin n uzunluğunda olan bloklara ayrılır. Sayısal değerleri $m_1, m_2, .., m_n$ olan açık metnin harflerinin blokları şifrelenirken

$i = 1,2,...,n$ için

$c_i \equiv p_i + k_i(29),\ 0 \le c_i \le 28$

Bağıntısı kullanılarak şifre metninin harf bloklarının $c_1, c_2, ..., c_n$ sayısal değerleri bulunur. Bu sayısal değerler de harflere dönüştürülerek şifre metni elde edilir. Bu yöntem aynı zamanda blok şifreleme sistemlerine de bir örnektir.

Örnek 2.2: $L(x)$; x verisinin karakter uzunluğunu sayısal olarak döndüren fonksiyon, m_i; açık veri, c_i; şifreli metin olmak üzere, aşağıdaki tablodan faydalanarak, $k = kalem$ Anahtarını kullanarak, "**Anlaşma sağlandı**"

verisini $L(k)$ uzunluğunda bloklara ayırarak, $c_i \equiv m_i + k_i (26)$ algoritmasına göre şifreleyelim;

m	*c*	*m*	*c*	*m*	*c*	*m*	*c*	*m*	*c*	*m*	*c*
A	0	**F**	5	**J**	10	**O**	15	**Ş**	20	**Y**	**25**
B	1	**G**	6	**K**	11	**Ö**	16	**T**	21	**Z**	**26**
C	2	**H**	7	**L**	12	**P**	17	**U**	22		
D	3	**I**	8	**M**	13	**R**	18	**Ü**	23		
E	4	**İ**	9	**N**	14	**S**	19	**V**	24		

$L(k) = 5$ olup veriyi 5 harften oluşan bloklara ayırırsak:

ANLAŞ MASAĞ LANDI

Elde edilir. Tablodan harflere karşılık gelen m_i ler bulunursa:

$$m_1 = 0,\ m_2 = 14,\ m_3 = 12,\ \ldots,\ m_{14} = 3,\ m_{15} = 8$$

Olarak elde edilir.

Algoritmaya göre her m_i değerini k anahtarının harflerinin tablodaki karşılıkları olan k_i ile toplayım 26 ya göre modülünü alarak şifrelenmiş metin olan c_i değerine ulaşılır.

$$k_1 = 11,\quad k_2 = 0,\ k_3 = 12,\ k_4 = 4,\ k_5 = 13$$

Olup c_i ler:

$$c_1 = 11,\ c_2 = 14,\ c_3 = 24,\ \ldots,\quad c_{14} = 7,\ c_{15} = 0$$

Elde edilir. Bu değerlerin harf karşılığını yazarsak şifreli metnimiz:

KNVEH VAAEÜ ÜAAIZ

olarak elde edilir.

2.1.3 Hill Şifreleme Sistemi:

Gizli anahtarlı sistemlere verebileceğimiz bir diğer örnek de Hill yöntemidir. Hill şifreleme yöntemi Lester Hill tarafından bulunmuş ve 1929 yılında yayınlanmıştır. Bu sitem de veriyi bloklara ayırarak şifreleme yapmaktadır. Bu sistemde gizli anahtar $n \times n$ bir matristir. Bu matrisi seçerken

dikkat edilmesi gereken nokta, matrisin $mod(26)$ (Türkçe için 29 alınır) da tersi alınabilir bir matris olmasıdır. Anahtar matris seçildikten sonra şifrelenmek istenen açık veri n karakter uzunlukta olan bloklara ayrılır. Daha sonra şifreli metnin her bloğu bir vektör olarak ifade edilip anahtar matris mod(26) tabanında çarpılarak şifreli hali elde edilir. Daha sonra elde edilen şifre blokları birleştirilerek şifreli çıktı elde edilmiş olur. **A** anahtar matrisi, **P** matrisi açık verinin bir bloğu ve **C** de şifreli blok olmak üzere

$$A=\begin{bmatrix} k_{1,1} & k_{1,2} & \cdots & k_{1,n} \\ k_{2,1} & k_{2,2} & \cdots & k_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ k_{n,1} & k_{n,2} & \cdots & k_{n,n} \end{bmatrix},\quad P=\begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix} \text{ ve } C=\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} \text{ için,}$$

$$C=\begin{bmatrix} k_{1,1} & k_{1,2} & \cdots & k_{1,n} \\ k_{2,1} & k_{2,2} & \cdots & k_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ k_{n,1} & k_{n,2} & \cdots & k_{n,n} \end{bmatrix}.\begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix} mod(26)$$

olur, yani;

$$\begin{aligned} c_1 &= (k_{1,1}.p_1 + k_{1,2}.p_2 + \cdots + k_{1,n}.p_n)\,mod(26) \\ c_2 &= (k_{2,1}.p_1 + k_{2,2}.p_2 + \cdots + k_{2,n}.p_n)\,mod(26) \\ &\vdots \\ c_n &= (k_{n,1}.p_n + k_{n,2}.p_2 + \cdots + k_{n,n}.p_n)\,mod(26) \end{aligned}$$

olarak bulunur. **A** tersi alınabilen bir matris olduğundan;

$$C = A.P \Leftrightarrow A^{-1}.C = A^{-1}.A.P \Leftrightarrow A^{-1}.C = I.P \Leftrightarrow A^{-1}.C = P$$

bağıntısı ile deşifre algoritması elde edilir.

Örnek 2.3: Aşağıda verilen **A** matrisini ve tabloyu kullanarak "**paymoremoney**" sözcüğünü Hill sistemiyle şifreleyelim.

A	B	C	D	E	F	G	H	İ	J	K
0	1	2	3	4	5	6	7	8	9	10
L	**M**	**N**	**O**	**P**	**R**	**Q**	**S**	**T**	**U**	**V**
11	12	13	14	15	16	17	18	19	20	21
W	**X**	**Y**	**Z**							
22	23	24	25							

ve

$$A = \begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix}$$

için, **paymoremoney** kelimesini $\boldsymbol{n = 3}$ uzunlukta bloklara ayırırsak;

Pay-mor-emo-ney şekline ayırdıktan sonra ilk blok için şifreleme işlemini uygulayalım. Tabloya göre;

$$P = \begin{bmatrix} 15 \\ 0 \\ 24 \end{bmatrix}$$

olarak bulunur. O halde;

$$C = A.P = \begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix}.\begin{bmatrix} 15 \\ 0 \\ 24 \end{bmatrix} mod(26) = \begin{bmatrix} 375 \\ 819 \\ 486 \end{bmatrix} mod(26) = \begin{bmatrix} 11 \\ 13 \\ 18 \end{bmatrix} = LNS$$

olup $\boldsymbol{C = LNS}$ olarak bulunur. Diğer bloklar da aynı yolla bulunabilir.

Şimdi de şifrelediğimiz ilk bloğu deşifre edelim:

$$A^{-1} = \begin{bmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{bmatrix}$$

dir. Gerçekten de

$$\begin{bmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{bmatrix}.\begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix} mod(26) = \begin{bmatrix} 443 & 442 & 442 \\ 858 & 495 & 780 \\ 494 & 52 & 365 \end{bmatrix} mod(26) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

olur.

$$P = (A^{-1}.C) mod(26) = \begin{bmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{bmatrix} \begin{bmatrix} 11 \\ 13 \\ 18 \end{bmatrix} mod(26) = \begin{bmatrix} 15 \\ 0 \\ 24 \end{bmatrix} = PAY$$

olarak bulunur.

2.1.4 Playfair Şifreleme Sistemi:

Gizli anahtarlı sistemlere verilebilecek bir örnek de *Playfair şifreleme sistemi* dir. Lord Peter Wimsey ve Lord Montoya tarafından yazılmış olan bu yöntem de bloklara ayırarak şifreleme mantığıyla işlemektedir. Bu yöntemde gizli anahtar olarak bir 5×5 boyutunda bir matris seçilir. Bu matrisin elemanları alfabenin harfleri olmalıdır. İngiliz alfabesinde 26 harf olduğu için bir harf bu matriste yer almaz. O harf de **J** harfidir. Yani bu şifreleme sisteminde J harfi şifrelenmez. Verideki j harfleri şifreli çıktıda aynen yazılır. Alfabenin harflerinin rasgele yerleştirildiği bir matris anahtar olarak seçildikten sonra açık veri iki karakter uzunluğunda bloklara ayrılır. Daha sonra her blok sırayla şifrelenip sonuçta birleştirilerek şifreli metin elde edilir. Şifreleme işlemi şu şekilde olur;

- Eğer bloktaki iki harf da aynı ise matristen o harf bulunur ve bulunduğu kolonda kendisinden sonraki gelen iki harf şifrelenmiş metin olarak alınır.
- Eğer bloktaki harfler matriste aynı sütunda/satırda ise harflerin bulunduğu sütunda/satırda kendilerinden sonra gelen harfler şifreli metin olarak alınır.
- Eğer bloktaki harfler matriste aynı satır veya aynı sütunda değilse ilk harfin bulunduğu satır ile ikinci harfin bulunduğu sütunun kesişiminde bulunan harf ve ilk harfin bulunduğu sütun ve ikinci harfin satırın kesişimdeki harf şifreli metin olarak alınır.

Örnek 2.4:

$$A = \begin{bmatrix} M & O & N & A & R \\ C & H & Y & B & D \\ E & F & G & I/J & K \\ L & P & Q & S & T \\ U & V & W & X & Z \end{bmatrix}$$

anahtar matrisi ile **SAYI** sözcüğünü playfair algoritması ile şifreleyelim.

İlk önce sözcüğü **SA-YI** olarak bloklara ayıralım. Ve ilk blok olan **SA** bloğunu şifreleyelim. **S** ve **A** harfleri aynı sütunda bulunmaktadırlar. 2. kurala göre harflerin bulunduğu sütunda kendilerinden sonra gelen harfleri almalıyız. Yani **S** den sonra gelen harf **X** , ve **A** dan sonra gelen harf **B** olduğu içib ilk bloğun şifrelenmiş hali **XB** olarak bulunur. Şimdide **YI** bloğunu şifreleyelim. **Y** ve **I** harfleri farklı sütun ve satırda bulunmaktadırlar. 3. kurala göre **Y** nin satırı ile **I** nın sütunun kesişiminde bulunan **B, ve I** nın satırı ile **Y** nin sütununun kesişimindeki harf olan **G** harfi alınır. Yani ikinci bloğun şifrelenmiş hali **BG** dir. Sonuç olarak **SAYI** sözcüğünün şifrelenmiş hali **XBBG** olarak bulunur.

2.1.5 DES Kriptosistem:

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.

Veri Şifreleme Standardı (Data Encryption Standard), Ulusal Standartlar ve Teknoloji Enstitüsü (*National Institute of Standards and Technology - NIST*) tarafından yayınlanan ABD Hükümeti Federal Bilgi İşleme Standartları (*US Government Federal Information Processing Standards - FIPS*) 46-3 içinde anlatılmaktadır. DES geliştirildiği 1970lerden beri tamamen analiz edilmiştir ve yeni önemli bir kusuru bulunamamıştır.

İlk tasarladığında donanım uygulamalarında kullanılması amaçlanmıştır. 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. 1998

yapımı 200.000 Amerikan Doları değerinde bir makine 6 günde olası bütün anahtarları deneyebilmektedir; daha fazla parayla daha hızlı sonuçlar elde edilebilmektedir. Bu basit DES'i birçok amaç için güvensiz kılmaktadır ve NIST, US hükümet sistemlerinde basit DES'in kullanım iznini kaldırmıştır. DES'in diğer bir zayıflığı da yavaş olmasıdır.

Şimdi biraz da DES algoritmasının çalışma sistemini inceleyelim.

DES algoritması Feistel yapısındadır. DES'i 16 döngüden oluşan bir döngüye benzetebiliriz. DES algoritmasının detaylarına geçmeden önce bit, XOR, permütasyon işlemi gibi kavramları açıklayalım:

Bilgisayar sisteminde en küçük veri depolama birimine 1 bit denir. b harfi ile gösterilir. Bir veri kaydettiğimz zaman bilgisayarda bu verinin her karakteri bir bit içine kaydedilir. Kısacası bilgisayarın hafızasını çok büyük bir depo gibi düşünelim. Bu depoyu ilk önce eş odalara, her odayı da eş bölmelere, her bölmeyi de eş kutucuklara bölersek, en küçük birim olan kutucuklardan her biri 1 bit olur. Bilgisayarlar ikilik tabanda çalıştığı için hafızasına kaydettiği her şey ikilik tabandadır. Yani her bit bir karakterlik değer aldığına göre, bir bitin içerdiği değer ya 0, ya da 1 değeridir. Örneğin bilgisayarın hafızasına bir cümle kaydettiğimiz zaman, bilgisayar ilkönce cümlenin her harfinin ikilik sistemdeki karşılığını bulur, mesela cümlenin ilk harfi A olsun. Bu harfin ikilik sistemdeki karşılığı 00000 olduğu için bu harfi hafızaya kaydetmek için 5 bit kullanılır ve her bir bit 0 değeri içerir. Ayrıca bilgisayar terimi olarak 8 bitlik veri depolama birimine Byte (B) denir. 1024 Byte lik birime Kilobyte (KB), 1024 KiloByte lık birime Megabyte (MB), 1024 Megabyte lik birime Gigabyte (GB), 1024 Gigabyte lik birime de Terrbyte (TB) denir. İkilik sistemde çalışıldığı için birimler de 2'nin katları ile değişmektedir.

XOR işlemi de iki veriyi ikilik tabanda toplamak demektir. Mesela iki kelimeyi XOR işlemine tabi tutmak demek; her kelimenin harflerinin ikilik değerini bulup her kelimenin ikilik değerine ulaşmak, daha sonrada bu ikilik değerleri toplayıp yeni bir ikilik sayıyı elde etmemiz, daha sonrada bu ikilik sayıyı tekrar harflere dönüştürmek demektir. XOR işlemi $\oplus$ işareti ile gösterilir. Harflerin ikilik değerleri aşağıdaki tabloda verilmiştir.

HARF	*DEĞERİ*	*HARF*	*DEĞERİ*	*HARF*	*DEĞERİ*	*HARF*	*DEĞERİ*	*HARF*	*DEĞERİ*
A	00000	F	00110	J	01100	Ö	10010	U	11000
B	00001	G	00111	K	01101	P	10011	Ü	110014
C	00010	Ğ	01000	L	01110	R	10100	V	11010
Ç	00011	H	01001	M	01111	S	10101	Y	11011
D	00100	I	01010	N	10000	Ş	10110	Z	11100
E	00101	İ	01011	O	10001	T	10111		

Şimdi de permütasyon işlemine değinelim. Bir veriyi permütasyon işleminden geçirmek demek, verinin tüm bitlerinin, verilen permütasyona göre yerinin değiştirmek demektir. Örneğin **ALİ** verisinin bitsel karşılığı **00000111001011** dir. Bu veriyi **7 5 1 12 15 8 9 10 2 4 11 14 3 6** permütasyonuna sokmak demek ; 1. bit yerine 7. biti, 2. bit yerine 5. biti, … , 15. bit yerine de 6. biti getirerek bir nevi bir karıştırma işlemi yapmak demektir.

Daha önce de bahsettiğimiz gibi DES algoritması her veriyi bloklara ayırarak şifreleme yapar. Algoritma veriyi 64 bitlik parçalara ayırır. Algoritma her bloğu şifrelerken 16 döngü uygular. Döngülerde anahtarlar, permütasyonlar, XOR işlemleri ve bir f fonksiyonu kullanılır. DES algoritmasını incelerken, aynı anda çalışan üç ana parçaya ayıracağız. Bunlar; veriyi şifreleyen temel algoritma, algoritmanın kullandığı gizli anahtara bağlı olarak 16 döngünün her biri için ayrı anahtarlar üreten anahtar üretim algoritması ve ana algoritmada her döngüde kullanılan f fonksiyonunu üreten algoritmadır.

a) Temel algoritma:

Aşağıdaki şemadan da anlaşılacağı gibi algoritma ilk olarak veriyi ***başlangıç permütasyonu*** *(initial permutation)* denilen bir permütasyondan geçirmektedir. Daha sonra veriyi 32 şer bit uzunluğunda iki parçaya ayırır. Bu parçalar L_i ve R_i olarak adlandırılır. Tüm işlemler R_i bloğu üzerinde yapılır. Bu 16 döngünün sonunda elde edilen veri tekrar bir permütasyondan geçirilir. Bu permütasyon algoritmanın başında kullandığımız başlangıç permütasyonunun tersidir. Bu permütasyondan sonra verimiz şifrelenmiş olur. Bu işlem tüm bloklara

aynı şekilde uygulanarak tüm veri şifrelenir. Şimdi algoritmada uygulanan 16 döngülük bölümü biraz daha detaylı inceleyelim.

Başlangıç permütasyonunda çıkan veri 32 bit uzunluğunda iki parçaya ayrılır. Birinin adı L_0, diğerininki de R_0 olur. Birinci döngü burada sona erer. Bahsettiğimiz gibi tüm işlemler R_i bloğu üzerinde yapılacaktır. Birinci döngü işlerken anahtar üretim algoritmasında K_1 anahtarı üretilir ve ikinci döngü için $R_1 = L_0 \oplus f(R_0, K_1)$ hesaplanır. İkinci döngüde $L_1 = R_0$ olur. Üçüncü döngüde de benzer şekilde $L_2 = R_1$ ve $R_2 = L_1 \oplus f(R_1, K_2)$ olarak hesaplanır. 16 döngünün her adımında $L_{i+1} = R_i$ ve $R_{i+1} = L_i \oplus f(R_i, K_{i+1})$ işlemleri tekrarlanır. 16. döngünün sonunda elde edilen 32 şer bit uzunluğundaki $L_{16} = R_{15}$ ve $R_{16} = L_{15} \oplus f(R_{15}, K_{16})$ değerleri birleştirilerek tekrar 64 bit uzunluk elde edilir ve bu veri de başlangıç permütasyonundan geçirilerek blok veri şifrelenmiş olur.

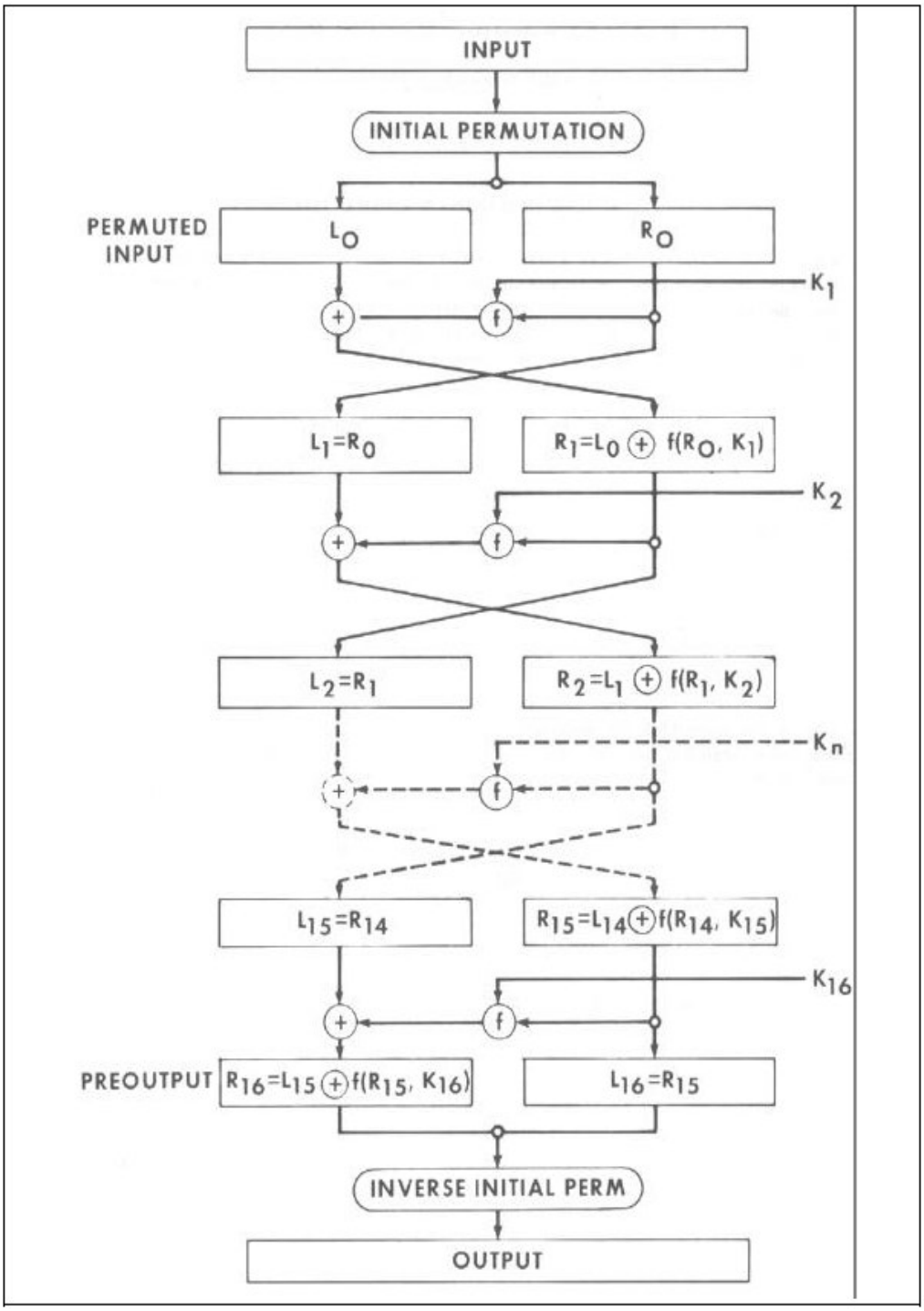
Algoritmada kullanılan başlangıç permütasyonu ve başlangıç permütasyonunun tersi aşağıda verilmiştir.

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

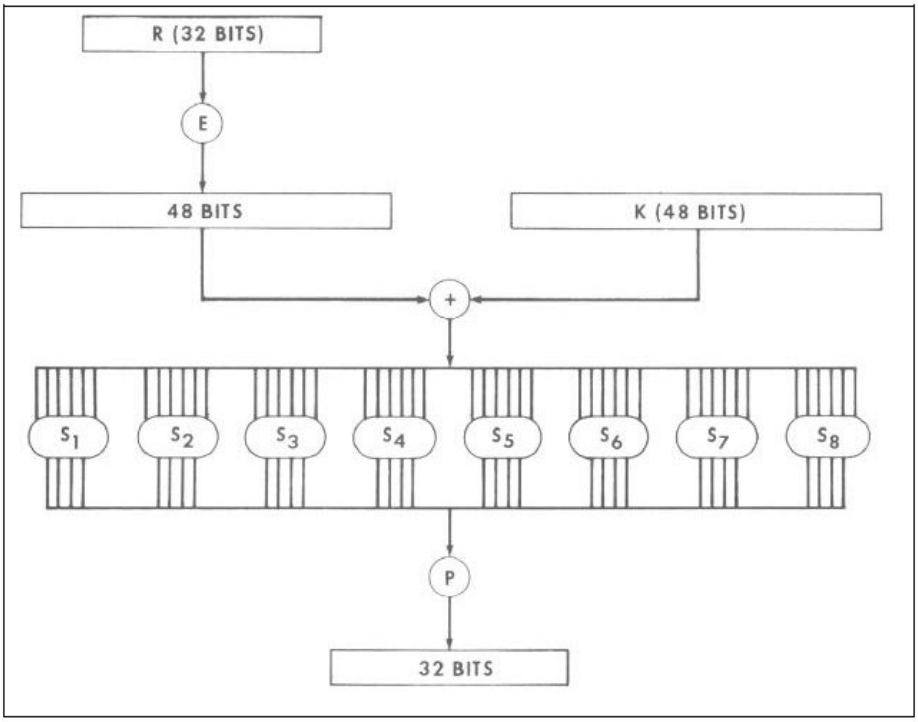
Başlangıç permütasyonu

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

Başlangıç permütasyonunun tersi


INPUT
INITIAL PERMUTATION
PERMUTED INPUT
L_0
R_0
K_1
+
f
$L_1=R_0$
$R_1=L_0 \oplus f(R_0, K_1)$
K_2
+
f
$L_2=R_1$
$R_2=L_1 \oplus f(R_1, K_2)$
K_n
+
f
$L_{15}=R_{14}$
$R_{15}=L_{14} \oplus f(R_{14}, K_{15})$
K_{16}
+
f
PREOUTPUT
$R_{16}=L_{15} \oplus f(R_{15}, K_{16})$
$L_{16}=R_{15}$
INVERSE INITIAL PERM
OUTPUT

b) f fonksiyonu algoritması:



f fonksiyonu algoritması

Şimdi de f fonksiyonunun yapısını inceleyelim. Daha önce de bahsettiğimiz gibi algoritmada sürekli 32 bitlik veri parçalarının birisiyle işlem yapılıyordu. $f(R,K)$ Fonksiyonu anlaşılacağı üzere 32 bit uzunluğundaki R veri parçasını ve K anahtarını kullanarak değer döndürmektedir. Fonksiyon ilk önce 32 bit uzunluğunda olan R fonksiyonunu ***E Permütasyonu*** denilen özel bir permütasyondan geçirerek 48 bit uzunluğunda bir veriye dönüştürür. Bu permütasyon aşağıda verilmiştir.

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

E permütasyonu

Daha sonra 48 bit uzunluğundaki bu veri yine 48 bit uzunluğunda olan ve anahtar üretim algoritmasında üretilen ***K*** anahtarı ile **XOR** işlemine tabi tutulur. Bu işlemden sonra elde edilen 48 bit uzunluğundaki veri 6 şar bit uzunluğunda 8 bloğa ayrılır ve her blok **S-Kutusu** denilen kutulara gönderilip işleme tabi tutulur. Bu işlem sonucunda, 6 şar bitlik veri blokları 4 er bloğa indirgenmiş olarak Kutudan çıkarlar. S-kutuları aşağıda verilmiştir.

S1

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

S1

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

S3

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

S_4

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

S_5

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

S_6

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

S_7

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

S_8

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

S-Kutularının çalışma mantığı şu şekildedir;

S-Kutusuna gelen veri 6 bit uzunluğundadır. Bu verinin ilk ve son biti birleştirilip ikilik tabanda bir sayı elde edilir. Bu sayı onluk tabana çevrilir. Bu sayı S-Kutusunda kullanılacak satırı belirler. Sonra verinin ilk ve son biti dışında kalan sayıları ikilik tabanda bir sayı olarak alınır ve onluk tabandaki karşılığı hesaplanır. Bu değer de S-Kutusunda kullanılacak sütünu belirler. Son olarak da elde ettiğimiz satır ve sütunun kesişimindeki sayı S-Kutusundan bulunur ve ikilik

tabandaki karşılığı elde edilir. Bu şekilde 6 şar bitlik tüm parçalar S-Kutusundan geçirilir. Daha net anlaşılması için bir örnek ile açıklayalım:

Örnek 2.5: 101110 verisini S_1 kutusuna tabi tutalım:

Verinin ilk ve son bitini ikilik tabanda bir sayı kabul edip onluk sistemdeki karşılığını hesaplayalım:

$$(10)_2 = 0.2^0 + 1.2^1 = 2$$

Şimdi de diğer bitleri bir sayı kabul edip onluk tabandaki karşılığını hesaplayalım:

$$(0111)_2 = 1.2^0 + 1.2^1 + 1.2^2 + 0.2^3 = 7$$

S_1 Kutusunda 2. satır ve 7. sütunun kesişimindeki sayı 11 dir. Bu sayının ikilik tabanda karşılığını hesaplarsak:

$$11 = (1011)_2$$

Sonuç olarak 6 bit uzunluğundaki 101110 verisini S_1 kutusuna tabi tutarak 4 bit uzunlukta 1011 çıktısını aldık.

Son olarak S-Kutularının çıktılarının birleştirilmesiyle 32 bitlik bir veri elde edilir. Bu elde edilen veri **P Permütasyonu** denilen bir permütasyondan daha geçerek elde $f(R,K)$ değeri elde edilmiş olur. Bu permütasyon aşağıda verilmiştir.

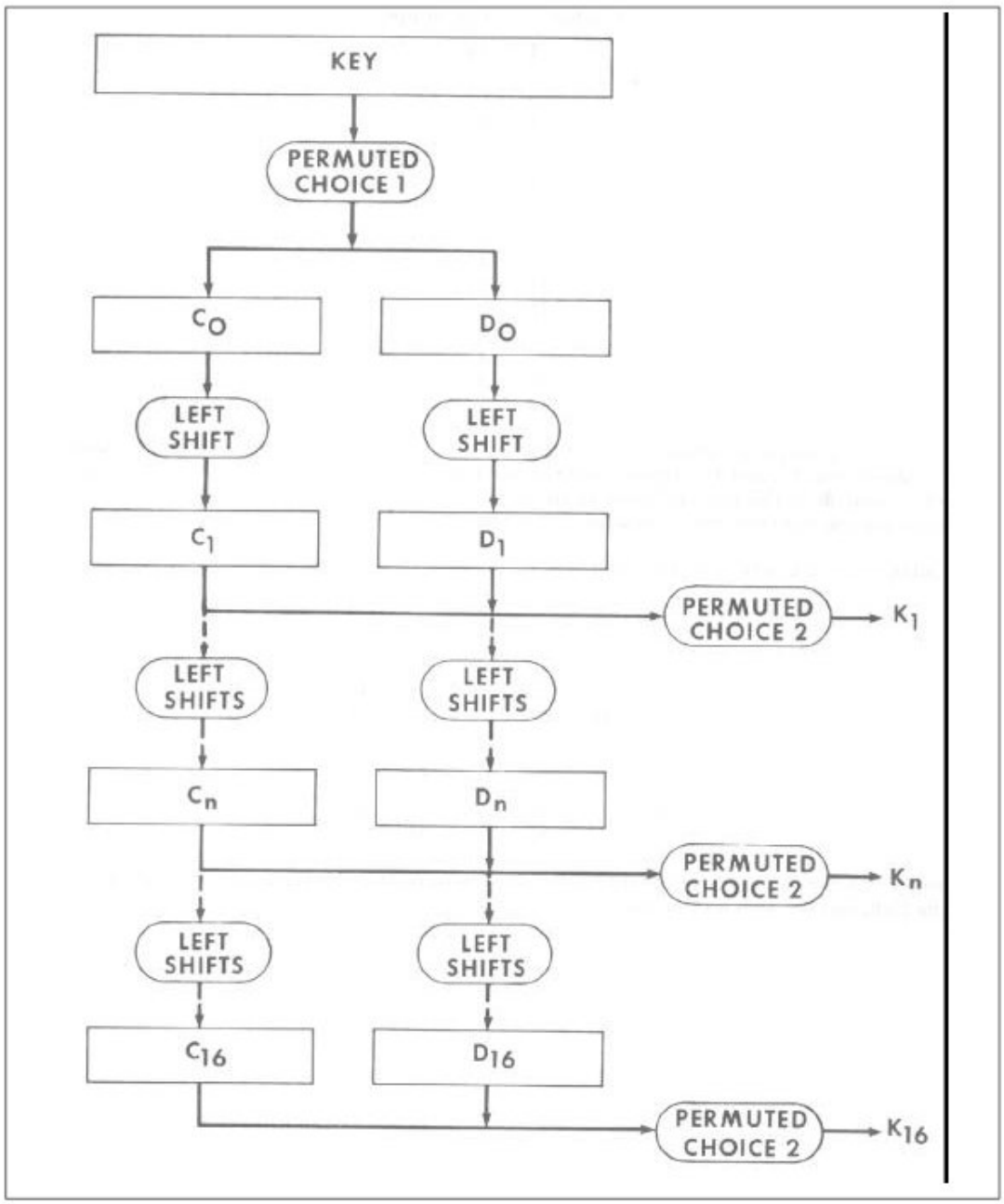
P

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

P Permütasyonu

Bildiğimiz gibi bu veri L parçası ile XOR işlemine tabi tutularak algoritma devam edecektir.

c) Anahtar üretim algoritması:



Anahtar üretim algoritması DES algoritmasının kullandığı gizli anahtar üzerinde işlemler yaparak anahtar üretir. İlk önce 64 bit uzunluğundaki anahtarı **Permuted choice 1 (PC1)** denilen özel bir permütasyondan geçirerek 56 bit uzunlukta olan bir veriye dönüştürür. Bu permütasyon 64 bitlik veriyi 8 er bitlik

bloklara ayırıp her bloğun 8. bitini atarak 56 bitlik veriye ulaşır. Bu **PC1** permütasyon aşağıda verilmiştir.

***PC*-1**

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

Daha sonra elde edilen bu 56 bit uzunluğundaki veri 28 er bitlik iki bloğa ayrılır. Bu blokların her biri **döndürme** denilen bir işleme tabi tutulur. Bu işlemde bloğun her biti bir ya da iki bit sola kayar. Kalan bitler de sona eklenir. Yani bu işlem düzgün kurallı bir permütasyon işlemidir diyebiliriz. Bitlerin kayma miktarları her döngüde farklılık göstermektedir. Döngü sırasına göre kayma miktarları aşağıdaki tabloda gösterilmiştir.

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

Döndürme işleminden sonra veri **Permuted choice 2 (PC2)** denilen bir permutasyona daha girer. Bu permutasyon sonucunda 56 bit uzunluğundaki veri 48 bite iner. Bu permütasyon aşağıda verilmiştir.

PC-2

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

Sonuç olarak 48 bit uzunluğundaki veri anahtar olarak ana algoritmadaki döngüde kullanılır. Hatırlarsak anahtarlar $f(R,K)$ fonksiyonunda kullanılıyordu. Anahtarın 48 bit uzunlukta olacağı anahtar üretim algoritması konusunda da verilmişti.

DES algoritmasıyla şifrelemeyi i,inceledikten sonra deşifrelemeye de değinelim. Deşifreleme için kullanacağımız algoritma, şifreleme yaparken kullandığımız üç temel algoritmadan birincisi olan, 16 döngüden oluşan temel algoritmadır. Fakat deşifreleme yapılırken anahtar sırası ters olur. Yani K_1 yerine K_{16} anahtarı kullanılır. Ve algoritma sonunda şifreli veri deşifre edilmiş olur.

2.1.6 AES Kriptosistem:

AES (Rijndael) algoritması şu ana kadar bilinen algoritmalar içerisinde en güçlülerinden biridir. Bu gücü nerden aldığını incelemekte fayda vardır. DES (Data Encryption Standard) 1970'li yıllarda IBM ve NSA ile birlikte öne sürüldüğünde 1990'lı yıllara kadar kendini başarı ile savundu. Ancak onun en büyük zaafı 56 bit anahtara sahip olması ayrıca anahtarın bir şekilde daha büyük uzunlukta kullanılabilecek bir yönteminin olmamasıydı. Daha sonraları paralel işlemcili bilgisayarların kullanılması ve geniş anahtar arama saldırısı ile kısa bir zaman içinde kırılması mümkün hale gelmişti. Ayrıca güçlü kriptanaliz saldırılarına karşı da yetersiz kalmaya başlamıştı. DES algoritması Feistel mimarisine sahipti. Yani veri bloğu iki parçaya bölünerek şifreleme işlemi yapılıyordu. Heys, 1994 yılındaki çalışmasında, bir SPN algoritması kullanarak

diferansiyel ve lineer kriptanalize karşı güçlü DES algoritmasına eşit güçte bir algoritmayı 8 döngüde sağladı. Bu algoritma 64 bit blok uzunluğunda, 64 bit anahtar ve 8 bit girişli - 8 bit çıkışlı random S kutuları kullanmaktaydı. Bunun yanında S kutularının güçlü hale getirilmesi için bazı çalışmalar da gerçekleştirildi. Random S kutuları, S kutularının sırasını değiştirme, anahtar bağımlı S kutularının sırasını değiştirme, anahtar bağımlı S kutularının transformasyonu ve matematiksel bağlamda saldırılara karşı güçlü S kutuları tasarlanmaya çalışıldı. Nyberg S kutularının tasarlanmasında alan terslerinin (field inverse) kullanılmasını önerdi. Bu çalışmaların yardımıyla da tasarlanan AES algoritmasında feistel mimarisinden yer değiştirme-permütasyon mimarisine geçilmiş oldu.

AES algoritması matematiksel bir altyapı üzerine kurulmuştur. Özellikle algoritmada kullanılan S-Kutusunu incelerken ihtiyacımız olacak olan matematiksel altyapıdan biraz bahsedelim.

1. Matematiksel altyapı:

8 bitten oluşan bit Byte 16lık tabanda yazılabildiği gibi bir polinom olarak da ifade edilebilir**.** *Bir byte ı 16 lık tabanda yazarken 10 yerine A, 11 yerine B, 12 yerine C, 13 yerine D, 14 yerine E ve 15 yerine F harfleri kullanılır***.** İki byte ı toplamak çarpmak veya bir byte nin tersini almayı polinomlarca ifade edeceğiz.

Bir $b = (b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0)_2$ baytının polinom olarak gösterimi:

$$p(b) = b_7 x^7 + b_6 x^6 + b_5 x^5 + b_4 x^4 + b_3 x^3 + b_2 x^2 + b_1 x + b_0$$

olup byte'ın değeri 10 luk tabandaki karşılığıdır.

Örnek 2.6: $(53)_{16}$ byte nının değeri :

$(53)_{16} = 3.16^0 + 5.16^1 = 83$ tür.

İkilik sistemdeki karşılığı da:

$83 = (01010111)$ dir.

o halde polinom olarak gösterimi:

$$p_a(x) = x^6 + x^4 + x^2 + x + 1$$

dir.

Örnek 2.7: $8A$ byte nı polinom olarak ifade edersek:

$$8A = 10.16^0 + 8.16^1 = 138 = (10001010)_2 = x^8 + x^3 + x$$

Olarak elde edilir.

Şimdi de byte lar üzerinde bazı işlemlerin nasıl yapıldığını inceleyelim.

a) Toplama İşlemi:

Byte leri toplamak demek, byte lara karşılık gelen polinomları **mod2** de toplamak demektir. Bu işlem byte ları XOR lamaya denktir.

Örnek 2.8: $a = (10011010)$ ve $b = (10101011)$ byte larını toplayalım:

$$p(a) = x^7 + x^4 + x^3 + x \text{ ve } p(b) = x^7 + x^5 + x^3 + x + 1 \text{ dir.}$$

$$\begin{aligned} p(a) + p(b) &= (x^7 + x^4 + x^3 + x + x^7 + x^5 + x^3 + x + 1) mod\ 2 \\ &= (2.x^7 + x^5 + x^4 + 2.x^3 + 2.x + 1) mod\ 2 \\ &= x^5 + x^4 + 1 \\ &= (00110001) \end{aligned}$$

olarak elde edilir. Dikkat edilirse:

$$a \oplus b = (10011010) \oplus (10101011) = (00110001)$$

olduğu görülür.

b) Çarpma işlemi:

i. İki byte ı çarpma:

İki byte'ı çarparken, ilk önce byte'ler polinom olarak ifade edilir ve polinomlar $GF(2^8)$ (Galois alanı/cismi) alanında çarpılır. Yani çarpım ikilik tabanda uygulanır, dolayısıyla katsayısı 2 olan terimler sıfırlanır. Byte ların polinom gösterimlerini hatırlarsak, bir bytın polinom ifadesinde en fazla 7. dereceden terim bulunması gerekir. Fakat burada iki byte ın polinom ifadelerini çarptığımızda 8 ve daha yüksek dereceden terimler de oluşabilir. Bu problemi çözmek için iki byte ın çarpımını $mod\ m(x)$ de yapılır. Buradaki $m(x)$ mod 2 de çarpanlarına ayrılamayan bir polinomdur. Yani $m(x)$ polinomu katsayıları 0 ve 1

olan polinomların çarpımı şeklinde yazılmayan bir polinomdur. İki byte ı $mod\, m(x)$ de çarpmak demek byte ların çarpımından elde edilen polinomun $m(x)$ e bölümünden kalanı elde etmek demektir. $m(x)$ çarpanlarına ayrılamadığı için çarpımdan elde edilen polinomun 8 ve daha yüksek dereceli terimleri yok edilmiş olur. AES algoritmasında kullanılan $m(x)$ polinomu:

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

dir. Burada $m(x)$ polinomu mod2 de çarpanlarına ayrılamayan başka herhangi bir polinom olarak da seçilebilirdi. $mod\, m(x)$ de işlem ayaprken kısa yol olarak $x^8 + x^4 + x^3 + x + 1$ görülen yere $\mathbf{0}$ yazılır, ya da $\mathbf{x^8}$ görülen yere $x^4 + x^3 + x + 1$ yazılır. Bir örnek üzerinde iki byte nın çarpının gösterelim:

Örnek 2.9:

$$a = (01010101) \Rightarrow p_a(x) = x^6 + x^4 + x^2 + 1$$

$$b = (10000011) \Rightarrow p_b(x) = x^7 + x + 1$$

byte larını çarparsak:

$$(a).(b) = p_a(x).p_b(x) mod\, m(x)$$

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

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

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

$$= x^5 x^8 + x^3 x^8 + x x^8 + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$$

$$= x^5(x^4 + x^3 + x + 1) + x^3(x^4 + x^3 + x + 1) +$$
$$x(x^4 + x^3 + x + 1) + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$$

$$= x^9 + x^8 + x^6 + x^5 + x^7 + x^6 + x^4 + x^3 + x^5 + x^4 + x^2$$
$$+ x + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$$

$$= x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + 1$$

$$= x(x^4 + x^3 + x + 1) + (x^4 + x^3 + x + 1) + x^7$$
$$+ x^6 + x^5 + x^4 + 1$$

$$= x^5 + x^4 + x^2 + x + x^4 + x^3 + x + 1 + x^7 + x^6 + x^5 + x^4 + 1$$

$$= x^7 + x^6 + x^4 + x^3 + x^2$$

$= (\ 11011100\ )$

olarak bulunur.

ii. Bir byte ın çarpmaya göre tersi:

Bir $a = (\ a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0\ )$ byte ının çarpmaya göre tersi:

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

denklemini sağlayan $p_b(x)$ polinomuna karşılık gelen byte dır. Yani ;

$$p_a^{-1}(x) = p_b(x) \bmod m(x) \text{ dir.}$$

iii. Dört byte uzunluğundaki vektörlerin çarpımı:

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

$$p_{\vec{a}}(x) = a_3 x^3 + a_2 x^2 + a_1 x + a_0$$

Burada a_3, a_2, a_1, a_0 ın byte oldukları unutulmamlıdır.

$\vec{a} = (\ a_3, a_2, a_1, a_0\ )$ ve $\vec{b} = (\ b_3, b_2, b_1, b_0\ )$ vektörleri için;

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

olarak tanımlanır. Diğer bir ifade ile;

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

olarak da tanımlanabilir.

iv. Dört byte uzunluğundaki vektörün çarpmaya göre tersi:

$\vec{a} = (\ a_3, a_2, a_1, a_0\ )$ vektörünün çarpmaya göre tersi:

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

bağıntısını sağlayan $\vec{b}$ vektörüdür. Yani;

$$p_{\vec{a}}^{-1}(x) = p_{\vec{b}}(x) \bmod M(x) \text{ dir.}$$

2) Algoritma:

AES (Rijndael) algoritması veriyi bloklara ayırarak şifreleme yapan bir algoritmadır. AES algoritması veriyi 128 bit uzunluğunda bloklara ayırarak şifreleme işlemine tabi tutar. Algoritma ilk tasarlandığında 128, 192 veya 256 bit uzunluğunda bloklarla çalışabilecek şekilde tasarlandı fakat NIST 128 bit uzunlukta bloklara ayrılmasına karar verdi. Şifreleme işlemi DES sisteminde olduğu gibi tekrar eden döngüler içerisinde olur. Ve yine DES deki gibi her döngü için özel anahtarlar üretilir. AES algoritmasında anahtar uzunluğu 128, 192 veya 256 bit olabilir. Döngü sayısı sabit değildir, anahtar uzunluğuna göre değişmektedir. Aşağıdaki tabloda anahtar uzunluğuna göre döngü sayıları belirtilmiştir.

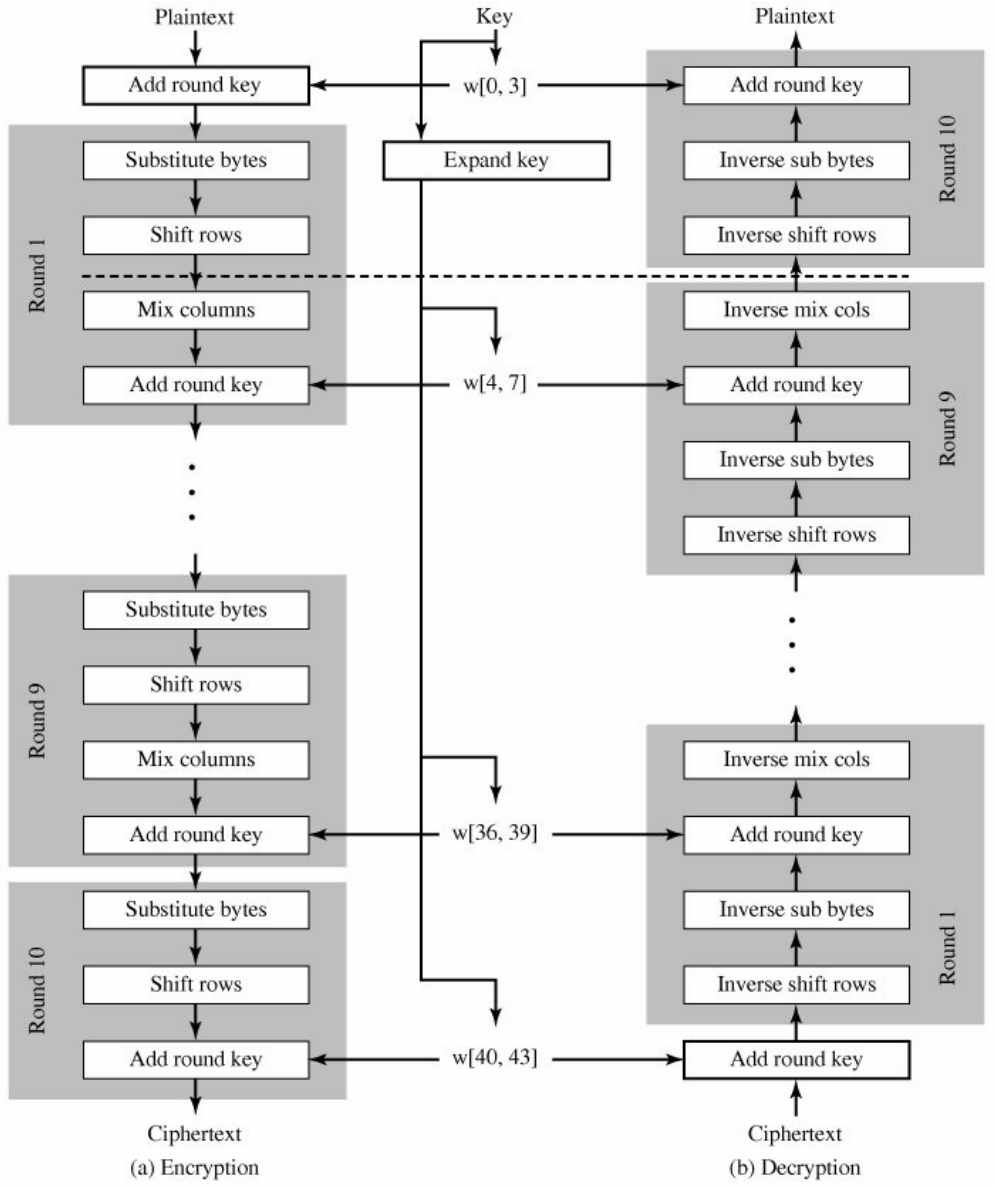
Anahtar uzunluğu	**Blok uzunluğu**	**Döngü sayısı**
128 bit (16 Byte)	128 bit (16 Byte)	10
192 bit (24 Byte)	128 bit (16 Byte)	12
256 bit (32 Byte)	128 bit (16 Byte)	14

Alogritmada son döngü dışındaki tüm döngülerde aynı işlemler uygulanır. Daha önce de değindiğimiz gibi 8 bit, 1 Byte değerine eşitti. AES algoritmasını incelerken Byte birimini kullanacağız.

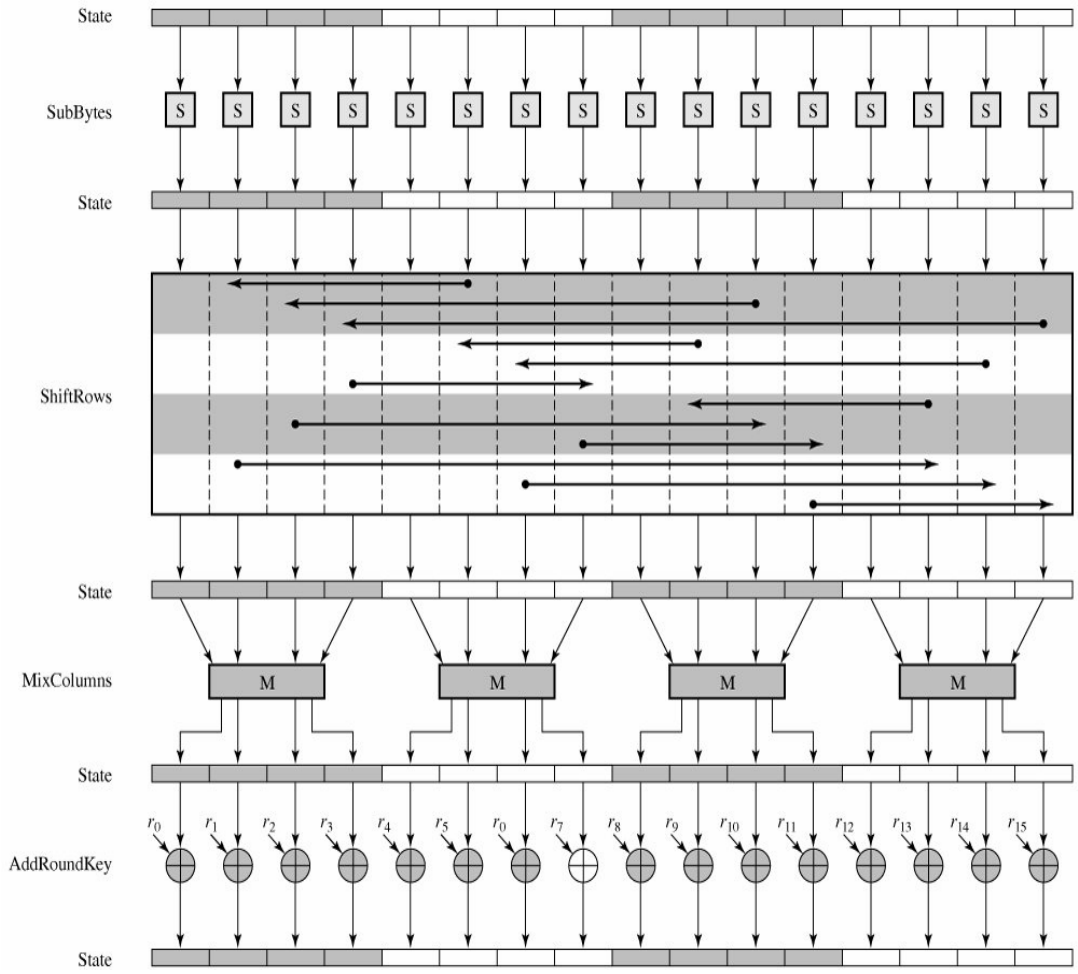
Algoritma ilk önce veri bloğunu 4×4 kare matris formuna çevirir. Bu matris şeklindeki bloğa ***state*** denir. Algoritma her döngüde aynı işlemleri uygular. Bu işlemler*:* ***Substitue bytes*** *(byte lerin yer değiştirmesi)*, ***Shift ro*****ws** *(satırların ötelenmesi)*, ***Mix columns*** *(sütunların karıştırılması)* ve ***Add round key*** *(anahtar üretim algoritmasından gelen anahtarla XOR lanması)* işlemleridir.

Aşağıdaki şekilde de görüldüğü gibi veri matrisi döngülere sokulmadan önce *Add round key* işleminden geçer, daha sonra döngüler başlar ve son döngü hariç diğer tüm döngülerde yukarıda bahsettiğimiz işlemler sırasıyla uygulanır. Son döngüde algoritmanın *tersi alınabilirlik* özelliğinin korunması amacıyla *mix columns* işlemi uygulanmaz. Şekilden de anlaşılabileceği gibi AES sistemi Feitsel

yapıda değildir. Anahtarların kullanıldığı tek işlem add round key işlemidir. S-kutularının kullanıldığı işlem ise substitue bytes işlemidir.

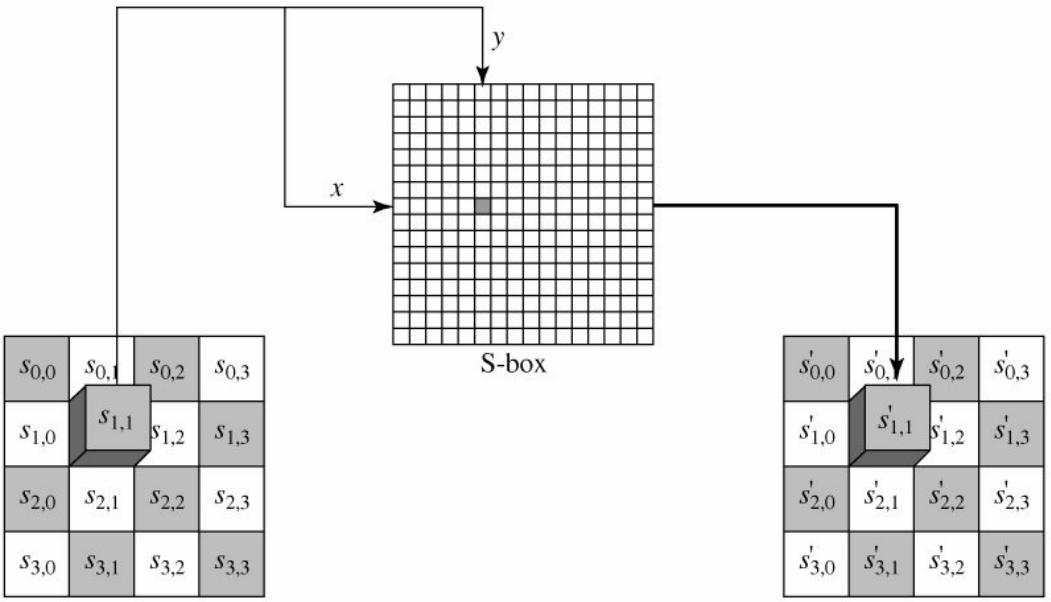


Aşağıdaki şekilde de tek bir döngü içinde gerçekleşen işlemler daha detaylı verilmiştir. Şimdi döngüler içinde uygulanan işlemleri daha detaylı inceleyelim.



a) Substitue Bytes (Byte ların yer değiştirmesi):

Aşağıdaki şekilde de gösterildiği gibi veri matrisinin her elemanı ayrı ayrı S-kutusundan geçerek karşılığı bulunur ve elde edilen değerlerden yeni matris elde edilir.



(a) Substitute byte transformation

S-kutusu kullanımı şöyledir: S-kutusundan geçirilecek olan elemanın satır numarası ve sütün numarası tespit edilip S-ktusundan o satır ve sütunun kesişimindeki veri seçilir. Bu veri yeni matriste aynı satır ve sütun numarasıyla yerini alır. Aşağıda S-kutusu verilmiştir.

(a) S-box

		y															
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
x	**0**	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Burada kullanılan S-kutularının çok şık bir matematik altyapısı vardır. Şimdi bu kutuların matematiksel altyapısına değinelim.

S-Kutusu:

S-Kutusu algoritmanın lineer olmayan, yani algoritmayı güçlü kılan en önemli kısmıdır. Görüldüğü gibi S-Kutusunda her karede bir byte vardır. *Tablodaki her karede bulunan byte; kesişiminde bulunduğu satır ve sütundaki karakterlein oluşturduğu byte ın $GF(2^8)$ de çarpmaya göre tersidir.* Yani her byte ın ifade ettiği polinomun $mod\, m(x) = x^8 + x^4 + x^3 + x + 1$ e göre tersi tabloda verilmiştir. Bu işlemi daha önce incelemiştik. Hatırlarsak: Her byte ı ikilik sistemde (bit birimi cinsinden) ifade edebiliyorduk. Kutudaki her byte 8 bitlik yer kapladığına göre her byte ; $(b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0)$ şeklinde bitlerden oluşmaktadır.

$a \to a^{-1} = b = (b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0)$ ve S-Kutusundan elde edilen çıktı $y = (y_7 y_6 y_5 y_4 y_3 y_2 y_1 y_0)$ ise, kısa yoldan;

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

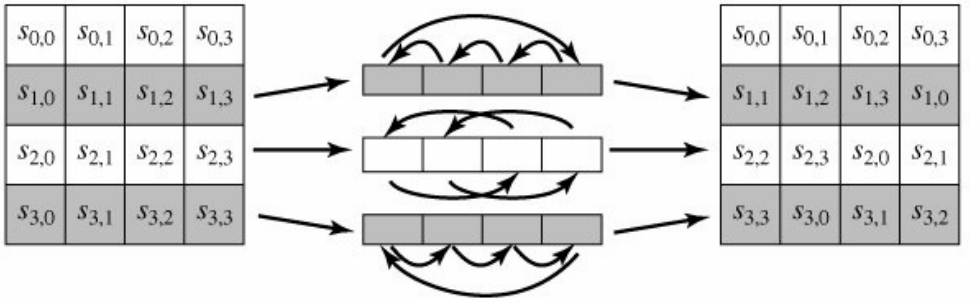
olarak formülize edilir. Ya da diğer bir ifade şekli ile:

$$y_i = b_i \oplus b_{(i+4) mod\, 8} \oplus b_{(i+5) mod\, 8} \oplus b_{(i+6) mod\, 8} \oplus b_{(i+7) mod\, 8} \oplus c_i$$

olarak verilebilir. Burada $\boldsymbol{c_i}$; özel olarak tanımlanmış $c = 63 = (01100011)$ byte nın i nci biti dir.

b) Shift Rows (satırların ötelenmesi) :

Adından da anlaşılacağı gibi bu aşama, veri matrisinin satırlarının ötelendiği basit bir permütasyon aşamasıdır. Aşağıdaki şekilde verildiği gibi bu aşamada 2. ve 4. satırlar birer eleman ötelenmiştir.



(a) Shift row transformation

c) Mix columns (sütunların karıştırılması) :

Bu aşamada veri matrisinin her sütunu bir vektör olarak alınır ve özel olarak tanımlanmış bir 4 byte lık $\boldsymbol{c}$ vektörüyle $GF(2^8)$ de çarpılır. Hatırlarsak bu işlem vektörlerin polinom olarak ifadelerinin $mod\, M(x) = x^4 + 1$ de çarpılması olarak tanımlanıyordu. Buradaki c vektörü: $c(x) = 3x^3 + x^2 + x + 2 \Rightarrow c = (03,01,01,02)$ olarak tanımlanmıştır. O halde mix columns aşamasını,

$$\begin{bmatrix} b_{00} & b_{01} & b_{02} & b_{03} \\ b_{10} & b_{11} & b_{12} & b_{13} \\ b_{20} & b_{21} & b_{22} & b_{23} \\ b_{30} & b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix}$$

olarak ya da, $\bullet$ işlemi $GF(2^8)$ de çarpma işlemi olmak üzere,

$$b_{0,j} = (2 \bullet a_{0,j}) \oplus (3 \bullet a_{1,j}) \oplus a_{2,j} \oplus a_{3,j}$$
$$b_{1,j} = a_{0,j} \oplus (2 \bullet a_{1,j}) \oplus (3 \bullet a_{2,j}) \oplus a_{3,j}$$
$$b_{2,j} = a_{0,j} \oplus a_{1,j} \oplus (2 \bullet a_{2,j}) \oplus (3 \bullet a_{3,j}$$

olarak verilebilir.

d) Add round key (döngü anahtarıyla işlem) :

Bu aşamada döngü için üretilen, matris formundaki, anahtar ile matris formundaki veri bloğu XOR işlemine tabi tutulur. Yani;

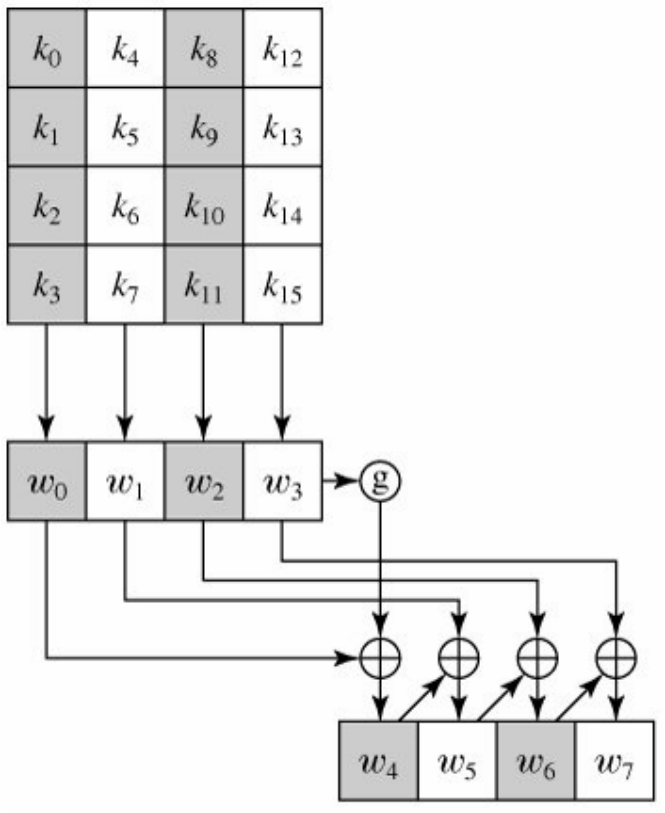
$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \oplus \begin{bmatrix} k_{00} & k_{01} & k_{02} & k_{03} \\ k_{10} & k_{11} & k_{12} & k_{13} \\ k_{20} & k_{21} & k_{22} &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

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

işlemi uygulanır.

e) Anahtar üretim algoritması:

AES te gizli anahtar bir matris formundadır. Anahtar algoritmasında 4 byte bir kelime olarak alınır. Şekilde de görüldüğü gibi anahtar matrisin her sütunu bir kelime olarak alınır ve sonuçta 4 kelimelik (16 byte) bir veri elde edilir. Şekilde görüldüğü gibi bu 4 kelimelik veri, özel olarak tanımlanmış bir g fonksiyonu ve XOR işlemi kullanılarak yeni bir 4 kelimelik veri elde edilir. Bu veri o döngünün anahtarı olur.



Her döngüde bu şekilde 4 kelimelik anahtarlar elde edilir ve bu anahtarlar algoritmadaki add round key aşamasında kullanılır. Toplamda 44 kelimelik anahtar kullanılmış olur (4 kelime gizli anahtar ve 10 döngüde üretilen toplam 40 kelime).

Burada kullanılan özel g fonksiyonu aşağıda verilmiştir.

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

Bu fonksiyonda kullanılan bileşenler:

subbyte : substitue bytes aşamasındaki işlemler.

Rotbyte : verilen kelimeyi ters çevirir (örnek:

$Rotbyte((\ a,b,c,d\ )) = (\ d,c,b,a\ ))$

$RoundConstant[j]$:

$$RJ[1], RC[j] = x.RC[j-1] \Rightarrow RoundConstant[j] = \begin{bmatrix} RC[j] \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

olarak tanımlanır.

Sonuç olarak anahtar algoritması:

$N_k / i \in Z$ ise;

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

$N_k / i \notin Z$ ise ;

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

olarak sembolize edilebilir.

2.2 Açık Anahtarlı Kriptografi:

Şimdiye kadara gizli anahtarlı (simetrik) şifreleme teknikleri üzerinde durduk. Simetrik yapıda DES, AES, Playfair gibi bazı algoritmalar üzerinde durduk. Hatırlayacak olursak simetrik şifrelemede açık veriyi şifrelemek için bir şifreleme algoritması ve birde gizli anahtar kullanıyorduk. Çoğu sistemin algoritması gizli değildi. Yani veriyi şifrelerken kullanılan algoritma kamuya açılmıştır ve herkes tarafından bilinmektedir. Şifreli verinin herkes tarafından deşifre edilmesini önleyen öğe gizli anahtardır. Gizli anahtar algoritmada bir parametre ya da değişken gibi kullanılıyordu. Yani algoritma gizli anahtara göre çalışıyordu. Bu nedenle algoritma herkes tarafından bilinmesine rağmen gizli anahtar bilinmediği için herkes deşifreleme yapamıyordu. Simetrik şifreleme sistemlerini incelerken değinmiştik. Bu sistemlerin güvenliği anahtarda yatıyordu. Bunun için çok güvenli bir anahtar dağıtım sistemine sahip olması gerekiyordu. Yani ne kadar güçlü bir algoritma ile şifrelersek şifreleyelim anahtarı bilen bir kişi şifreli metni çok kolay bir şekilde deşifre edebilir. Mesela AES algoritması bilinen en sağlam simetrik şifreleme sistemidir. Bir veriyi AES le şifreledikten sonra şifreyi arkadaşımıza göndersek, arkadaşımız bu şifreyi açık metne dönüştürüp

okumak için bizim şifreleme yaparken kullandığımız gizli anahtara ihtiyaç duyacağı için bizim güvenli bir şekilde, kimsenin eline geçmeden, arkadaşımıza anahtarımızı iletmemiz gerekecektir. Anahtarı arkadaşımıza ilettikten sonra arkadaşımız çok rahat bir şekilde deşifreleme yapıp mesajımızı okuyacaktır. Şayet anahtar iletilirken istenmeyen başka bir kişinin eline geçerse o kişi de mesajı arkadaşımız gibi kolay bir şekilde deşifre edebilecektir. İşte bu noktada, anahtar iletimindeki güvenlik sorunlarını ortadan kaldırmak için *açık anahtarlı kriptografi* geliştirilmiştir. Açık anahtarlı kriptografinin gelişmi ile artık anahtarın başkası tarafından bilinmesi kaygısı ortadan kalkmıştır. Çünkü anahtarın başkası tarafından bilinmesinde bir sakınca yoktur.

Açık anahtarlı kriptografi, gerçek anlamda daha önceki gelişmelerden radikal bir kopuştur. Açık anahtarlı kriptografik sistemlerin en önemli noktaları matematiksel işlevler üzerine temellenmiş olmalarıdır. Aslında açık anahtarlı kriptografi için matematiğin çözüm getiremediği bir takım durumları (örneğin çok büyük bir sayının iki asal çarpanının bulunmasının matematikte herhangi bir doğrudan çözümü olmaması gibi) kullanarak güvenlik sağlar, bu yüzden de incelenmesi ayrı keyif ve heyecan verir.

Daha da önemlisi, açık anahtarlı kriptografi, tek anahtar kullanan simetrik geleneksel şifreleme algoritmalarının tersine, iki ayrı anahtarın asimetrik kullanımını öngörür. Birazdan göreceğimiz gibi, anahtar dağıtımı ve kimlik denetimi gibi gizlilik ve güven gerektiren durumlarda, iki anahtar kullanımı etkili sonuçlar ortaya koymuştur.

Az önce bahsettiğimiz gibi açık anahtarlı kriptografi de iki adet anahtar kullanılır. Birisi, algoritmaya adını veren, herkes tarafından bilinmesinde bir sakınca olmayan, hatta bazı durumlarda herkes tarafından bilinmesi gereken ***açık anahtar (public key),*** diğeri de sadece mesaj gönderilen kişinin bilmesi gereken, deşifreleme işleminde kullanılacak olan gizli anahtar. Açık anahtarla gizli anahtar birbirinden tamamen bağımsız değildir. İkisi arasında matematiksel bir bağ olması gerekir. Açık anahtarlı kriptografinin çalışma sistemini şu örnek üzerinde açıklayalım:

Örneğin **A**, **B** ve **C** kişileri aralarında gizli mesajlar alıp vermek istiyorlar. Bu mesajları açık anahtarlı kriptografi ile şifrelemeye karar veriyorlar. Bunun için

mesajları alacak olan her kişi ilk önce iki adet anahtar oluşturuyor. Bunlardan birisi K anahtarıdır. Bu anahtar, sisteme adını veren açık anahtardır. Başkaları tarafından bilinmesinde bir sakınca yoktur. Bu anahtarla sadece şifreleme yapılır. Diğeri de deşifreleme işleminde kullanılacak olan K' anahtarıdır. Bu anahtar gizli anahtar olarak isimlendirilir. Bu anahtarla şifreleme değil deşifreleme yapılır. Bu iki anahtar arasında matematiksel bir bağ vardır. **B** kişisi K ve K' anahtarlarını oluştursun. **C** kişisi de L ve L' anahtarlarını oluştursun. **B** kişisi, **A** kişisine K açık anahtarını, **C** kişisi de L açık anahtarını gönderir. **A** kişisi **B** kişisine mesaj göndermek istediği zaman mesajını K anahtarıyla şifreleyip **B** ye gönderir. **B** kişisi de K' anahtarıyla mesajı deşifre edip okuyabilir. **A** kişisi **C** kişsine mesaj göndereceği zaman da mesajı L ile şifreler. **C** kişisi mesajı L' ile deşifre edip okuyabilir.

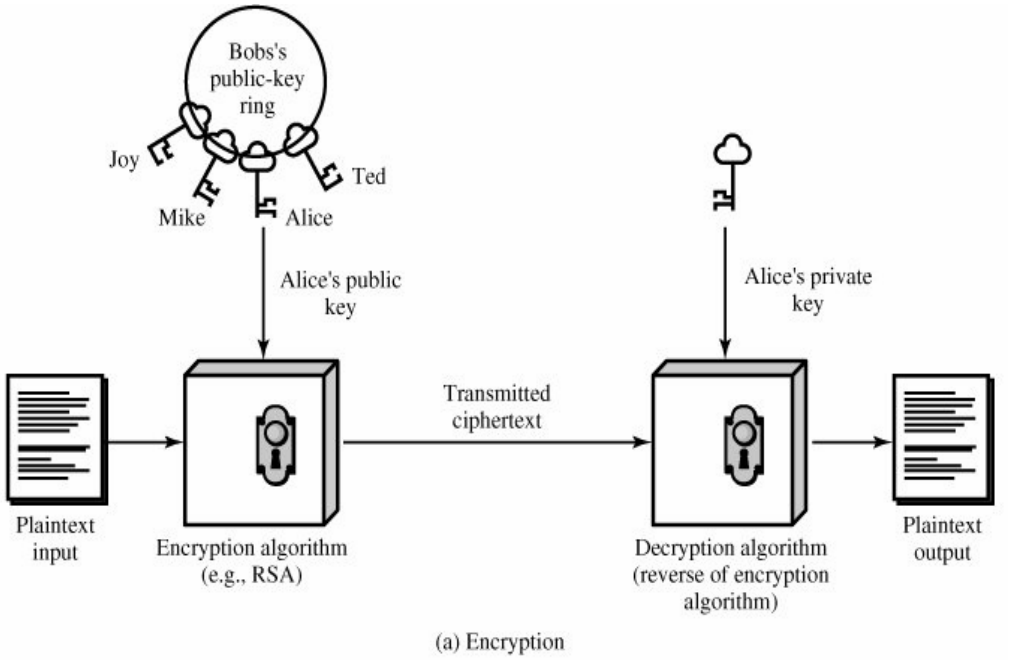
Başka basit bir örnekle açıklayacak olursak;

İki kişi arasında bir veri alışverişi yapıldığını varsayalım (bu iki kişi hemen bütün şifreleme kitaplarında Alice ve Bob diye geçer. Nedendir bilinmez?) Alice, Bob'a şifreli bir mesaj göndermek istemektedir. Bob kendi bilgisayarı üzerinde bir adet public key (şifreleme için) ve bir adet de private key (şifre çözmek için) oluşturur. Ve Bob oluşturmuş olduğu public key i Alice gönderir. Yolda public key in başkaları tarafından görülmesinde herhangi bir sakınca yoktur.(secret key yada simetrik algoritmalardan farkı buradadır.) Çünkü public key sadece şifreleme yapar. Alice Bob'a ait public key (açık anahtar) ile mesajını şifreler ve şifreli bir şekilde Bob'a yollar. Bob'un elinde de kendisine ait olan public key e karşılık gelen private key vardır (ve bu private key sadece oluşturulmuş olan public key ile şifrelenmiş mesajları çözer ve bu sebebten ötürü hep gizli kalmalıdır.) Bob şifrelenmiş mesajı private key (gizli anahtar) i ile çözerek okur. Yolda metin şifrelenmiş olarak gittiğinden ve private key (özel anahtar) her zaman Bob un elinde gizli bir şekilde tutulduğundan mesajın güvenliği sağlanmıştır.

Görüldüğü gibi açık anahtarlı şifreleme sistemlerinde anahtarın iletilmesi sorunu yoktur. Açık anahtar herkes tarafından bilinse de bir sorun çıkmaz çünkü açık anahtar deşifreleme yapamaz. Sadece şifreleme yapar. Açık anahtarla gizli anahtarın arasında bir matematiksel bağ olduğundan bahsetmiştik. Bu

matematiksel bağ kolay kolay çözülemeyen bir denklem üzerine kurulmalıdır. Çünkü bu sistemde açık anahtar herkes tarafından bilinir. Eğer matematiksel ilişki basit bir denklem üzerine kurulu olsaydı bu denklem çözülmek suretiyle, açık anahtarı kullanarak gizli anahtar elde edilebilinirdi. Bunun için bu matematiksel bağ genelde matematikte doğrudan çözümü olmayan problemler üzerine kurulmuştur.

Aşağıdaki şekilde açık anahtarlı kriptografinin çalışma sistemi net olarak gösterilmiştir.



Açık anahtarlı kriptografik sistemlerin en önemli karakteristik özelliklerini şu şekilde sayabiliriz:

- Sadece kriptografik algoritma ve deşifreleme anahtarı verilmişken, bir takım hesaplamalar yolu ile şifreleme anahtarını bulmak mümkün değildir.
- Her iki benzer anahtar da şifreleme ve deşifreleme için kullanılabilir. Bununla beraber, bir anahtar şifreleme için kullanılmışsa, deşifreleme için diğer anahtar kullanılmalıdır.

Açık anahtarlı kriptografi mesajların şifrelenerek gönderilmesi yoluyla güvenlik sağladığı gibi kimlik denetimi yoluyla da bir güvenlik uygulaması olarak kullanılabilir. Yani bu sistemi iki şekilde kullanılır. Bunlardan birisi yukarıda da bahsettiğimiz mesaj şifreleme ve deşifreleme algoritması olarak kullanılmasıdır. Yukarıda bunu örnek üzerinde açıklamıştık. Bir kez daha açıklayacak olursak bu işlem kabaca aşağıdaki adımlarla yürür.

- Her ağdaki her son sistem, kullanıcı ya da benzeri, mesaj alındığında şifreleme ve de şifreleme için kullanacak olduğu anahtar parçalarını yaratır.
- Her sistem, şifreleme anahtarını herkesçe erişilebilecek bir dosya ya da yazmaç içerisine kaydederek ya da duyurarak herkesçe erişilebilecek şekilde paylaşır. Bu anahtarın, genel olan kısmıdır (public key diye geçer). Özel anahtar saklı tutulur.
- Eğer, herhangi bir **A**, herhangi bir **B**'ye, **B**'nin bu mesajı kendisinden başka kimsenin görüntüleyemediğine emin olabileceği bir mesaj yollamak isterse, mesajı **B**'nin genel anahtarını kullanarak şifreler.
- **B**, mesajı aldığında, bu mesajı kendi özel anahtarını kullanarak de şifre eder. Diğer hiçbir alıcı (diyelim ki bu ağlardan herhangi birindeki bir sniffer) mesajı deşifreleyemez, çünkü mesajı de-şife edecek olan özel anahtarı sadece **B** bilir.

Yukarıdaki adımlar gerçekleştiğinde **B**, sadece kendisinin okuduğundan ve başka herhangi bir kimsenin görüntüleyemediğinden emin olduğu bir mesaj alır. Fakat bunun kimden geldiğinden emin olamaz. Gizlilik sağlanmış olur. Yukarıdaki son iki adımın şu şekilde gerçekleşmiş olduğu bir senaryoya bakalım bir de:

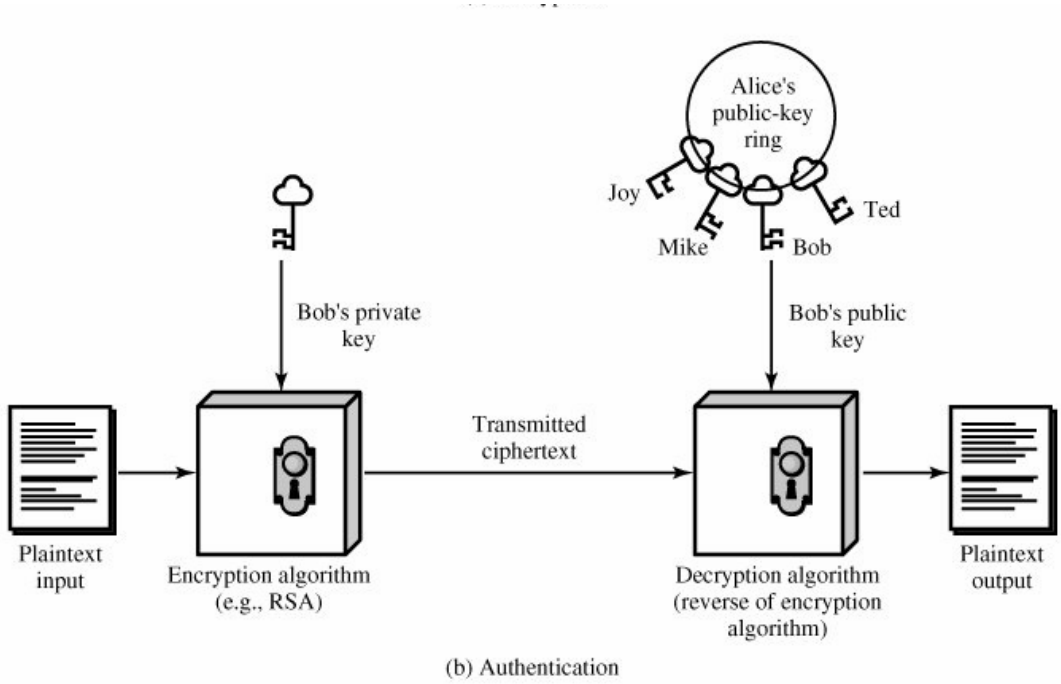
- Eğer, herhangi bir **A**, herhangi bir **B**'ye, **B**'nin **A**'dan geldiğine emin olarak okuyabileceği bir mesaj yollamak isterse, mesajı kendisinin gizli anahtarını kullanarak şifreler.
- **B**, mesajı aldığında, bu mesajı A'nın genel anahtarı ile de-şifreler. Diğer üçüncü parti alıcıların her biri de bunu

yapabilir, çünkü **A**'nın genel anahtarı herkesçe bilinmektedir. Bu durumda **B**, bu mesajın **A**'nın ta kendisinden geldiğinden ve kendisine ulaşana kadar yolda herhangi bir yerinin değiştirilmediğinden emin olur. Çünkü **A**'nın genel anahtarı ile deşifrelediği mesajın sadece **A**'nın bilebileceği özel anahtar ile şifrelenmiş olabileceğini bilir.

Bu senaryo ile de gizlilik yerine kimlik denetimi sağlanmış olur. Hem gizliliğin hem de kimlik denetiminin sağlanabileceği bir senaryo da şu şekilde olabilir bu durumda:

- Eğer, herhangi bir **A**, herhangi bir **B**'ye, **B**'nin **A**'dan geldiğine ve yolda kendisinden başka kimsenin içeriğini görüntüleyemediğine emin olarak okuyabileceği bir mesaj yollamak isterse, mesajı kendisinin gizli anahtarını kullanarak şifreler, daha sonra ortaya çıkan mesajı da **B**'nin genel anahtarını kullanarak şifreler.

Bu sayede de hem gizlilik hem de iki taraflı kimlik denetimi sağlanmış olur. Aşağıdaki şekilde de kimlik denetiminin çalışma sistemi gösterilmiştir.



(b) Authentication

Açık anahtarlı kriptografinin, pratik olarak uyarlanışı gösterilmeden, tekniğin mimarisi geliştirildi ve doğru kabul edilerek yayınlandı. Kimse açık anahtarlı kriptografinin pratiğini görmeden, sistemin teorisi kabul gördü.

Açık anahtarlı kriptografinin mucitleri olan Diffie ve Hellman, herhangi bir açık anahtarlı algoritmanın varlığını göstermeksizin bu bahsettiğimiz sistemi "varsaymışlardır". Bununla beraber, ilerde yazılması muhtemel algoritmaların yerine getirmeleri gereken durumları şöyle sıralamaktan da geri kalmamışlardır:

- Bir **B** için, anahtar parçalarını (genel anahtar ve özel anahtar) yaratmak, hesapsal olarak kolay olmalıdır.
- Gönderenin, mesajı göndereceği kişinin genel anahtarını bildiği ve şifrelenecek olan mesajı bildiği durumda, uygun şifreli metni yaratmak hesapsal olarak kolay olmalıdır.
- Alıcı, **B**`nin, özel anahtarını kullanarak, şifrelenmiş mesajı orijinal haline getirmesi hesapsal olarak kolay olmalıdır.
- Herhangi bir üçüncü kişi için, genel anahtarı bilerek, özel anahtarı bulması hesapsal olarak imkânsız olmalıdır.
- Herhangi bir saldırgan için, genel anahtarı, şifreli metini bilerek orijinal mesajı elde etmesi hesapsal olarak imkânsız olmalıdır.

Bunlar, sağlanması gerçekten zor olan gerekliliklerdir. Bu yüzden, açık anahtarlı kriptografi fikrinin ileri sürüldüğünden bu yana geçen yıllar süresince sadece bir kaç algoritma geniş bir kitle tarafından kabul edilmiştir.

Tabi belirtilmesi gereken bir nokta da şudur, "kolay" dan kasıt, fonksiyonun girdi uzunluğuna bağlı olarak polinomsal bir zaman süresi içerisinde çözülebilir olmasıdır. Şöyle ki, eğer girdi uzunluğu n bit kadarsa, fonksiyonun hesaplanması için gereken süre, a bir sabit sayı iken, n^a gibi bir fonksiyonla orantılı olmalıdır. "İmkânsız" ise, oldukça bulanık bir durumu ifade etmek için kullanılır. Bir problemin çözümünün matematiksel olarak olanaksız olduğundan, giriş büyüklüğüne bağlı olarak çözüm için harcanan çabanın, polinomsal zamandan daha hızlı arttığı durumda bahsedebiliriz. Örneğin, girdi n bit ile

gösterilirken, fonksiyonun çözülme zamanı 2^n gibi bir fonksiyona bağlı olarak artıyorsa, bu fonksiyonun çözümünün imkânsız olduğunu düşünebiliriz.

Şimdi, açık anahtarlı kriptosistemlere örnek olarak verilebilecek bazı önemli algoritmaları inceleyeceğiz.

2.2.1 RSA Kriptosistem:

Diffie ve Hellman'ın 1976 yılında yayınladıkları meydan okumaya (açık anahtarlı sistemlerin taşıması gerektiği özellikler yayınlanmıştı. Daha önce bahsetmiştik bu özelliklerden.) cevap fazla gecikmeden, 1977 yılında MIT'deki Ron Rivest, Adi Shamir ve Len Adleman'dan geldi ve 1978 yılında yayınlanan ünlü makaleleri "***Dijital imza elde etme metodu ve açık anahtarlı kriptosistemler***" (*A Method for Obtaining Digidal Signatures and Public-Key Cryptosystems, Feb. 1978*) ile Diffie ve Hellman'ın bahsettikleri gereklilikleri yerine getiren bir algoritma olan RSA duyuruldu. Bu sistem adını, tasarlayıcıların soyadlarının baş harflerinden almaktadır.

1. Algoritma:

Bir açık anahtarlı kriptografik yapı olan RSA kriptosistem, göndericinin bir metotla ve herkesçe bilinen açık bir anahtarla mesajlarını şifrelediği bir şifre sistemi olarak tanımlanır. Daha önceki gizli anahtarlı (simetrik) 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.

Şimdi biraz da RSA algoritmasının detaylarını inceleyelim. RSA' da düz metin, bloklar içinde şifrelenir, her blok bir n sayısından daha az bir ikili değere sahiptir. Bloğun büyüklüğü $log_2(n)$ değerine eşit ya da daha az olmalıdır; pratikte blok büyüklüğü 2^k bittir, bu durumda n için sağlanması gereken durum da $2^k < n \leq 2^{k+1}$ eşitsizliğidir.

RSA algoritmasını formülize edersek, C: şifreli metin, M: düz metin, (d,n) sayı çifti gizli anahtar, (e,n) sayı çifti de açık anahtar olmak üzere;

$$C \equiv M^e (mod\, n)$$

Ve

$$M \equiv C^d (mod\, n)$$

dir.

Şimdi anahtar algoritmasını inceleyelim;

a) Anahtar Algoritması:

RSA algoritmasında gizli ve açık anahtar aşağıdaki adımlar takip edilerek üretilir;

- İki tane farklı, rasgele p ve q asal sayıları seçilir.
- $n = pq$ ve $\phi = (p-1)(q-1)$ sayıları hesaplanır.
- $1 < e < \phi$ ve $gcd(e,\phi) = 1$ şartını sağlayan herhangi bir $\boldsymbol{e}$ sayısı seçilir. (gcd=e.b.o.b)
- Öklid algoritması kullanılarak $1 < d < \phi$ ve $ed \equiv 1 (mod\, \phi)$ koşulunu sağlayan d sayısı hesaplanır.
- Bu adımlar sonucu (n,e) çifti açık anahtar, (n,d) çifti de gizli anahtar olarak ele alınır.

$\boldsymbol{e}$ şifreleme üssü, d deşifreleme üssü ve n de modül olarak da adlandırılır. Şimdi bir örnek üzerinde RSA' nın çalışmasını daha net açıklamaya çalışalım;

b) Şifreleme:

B şahsı, **A** ya bir $\boldsymbol{M}$ mesajını şifreleyip göndermek istiyorsa aşağıdaki adımları sırasıyla uygular;

- Öncelikle **A** nın açık anahtarı olan (n,e) çiftini alır.
- M mesajını $[0, n-1]$ aralığında yazar.
- Sonra $C \equiv M^e (mod\, n)$ formülü ile metni şifreler.
- Son olarak da $\boldsymbol{C}$ şifreli metnini, **A** ya gönderir.

c) Desşifreleme:

A şahsı da bu şifreyi çözmek için aşağıdaki adımları uygular;

- A şahsına gönderdiği (n,e) açık anahtarın eşi olan (n,d) gizli anahtarını alır,
- Sonra da $M \equiv C^d (mod\, n)$ formülü ile şifreli metni açık metne dönüştürür.

2. Matematiksel Altyapı:

Deşifre sisteminin matematiksel ispatına değinelim;

$ed \equiv 1 (mod\, \phi)$ olduğu için $ed = 1 + k\phi$ olacak şekilde bir $\boldsymbol{k}$ tamsayısı vardır. Eğer $gcd(M, p) = 1$ ise Fermat teoreminden dolayı;

$$M^{p-1} \equiv 1 (mod\ p)$$

dir. Bu denkliğin her iki tarafının $k(q-1)$'inci kuvvetini alırsak;

$$M^{k(q-1)(p-1)} \equiv 1 (mod\ p)$$

elde edilir. Her iki tarafı $\boldsymbol{M}$ ile çarparsak;

$$M^{1+k(q-1)(p-1)} \equiv M (mod\ p)$$

sonucuna ulaşırız.

Diğer taraftan $gcd(M, p) = p$ olursa yukarıdaki denklik yine geçerli olur. Çünkü farz edelim ki bir k tamsayısı için $M = kp$ olsun;

$$M^{p-1} = (kp)^{p-1} = k^{p-1} p^{p-1} \equiv p (mod\ p)$$

olur. Bu denkliğin her iki tarafının da $k(q-1)$'inci kuvvetini alırsak;

$$M^{k(p-1)(q-1)} = k^{k(p-1)(q-1)} \equiv p (mod\ p)$$

olur. Ve her iki tarafı $\boldsymbol{M}$ ile çarptığımızda;

$$M^{1+k(q-1)(p-1)} \equiv Mp = kp^2 \equiv kp = M (mod\ p)$$

olup, iki durumda da

$$M^{ed} \equiv M (mod\ p)$$

olduğu görülür. Aynı şekilde;

$$M^{ed} \equiv M (mod\ q)$$

olduğu da gösterilir. Sonuçta p ve q farklı asallar olduğundan dolayı;

$$M^{ed} \equiv M (mod\ n)$$

olup, böylelikle;

$$C^d = M^{ed} \equiv M (mod\, n)$$

olur.

Şimdi bir sayısal örnek üzerinde RSA algoritmasını görelim;

Örnek 2.10:

A şahsı **B** ye bir mesaj gönderecektir. Aşağıdaki adımları uygulayarak RSA algoritması uygulanır.

1. Anahtar Oluşturma:

A şahsı $p = 2357$ ve $q = 2551$ asal sayılarını seçsin.

$$n = pq = 2357.2551 = 6012707$$

Ve

$$\phi = (p-1)(q-1) = 2356.2550 = 6007800$$

değerlerini hesaplasın. A şahsı, bir $e = 3674911$ sayısı seçer. Bu sayı $gcd(\phi = 6007800, e = 3674911) = 1$ ve $1 < e = 3674911 < \phi = 6007800$ şartlarını sağlıyor. Daha sonra öklid algoritmasını kullanarak,

$$ed \equiv 1 (mod\, \phi)$$

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

denklemini çözüp $d = 422191$ değerini elde eder. Sonuç olarak A nın açık anahtarı $(n,e) = (6012707, 3674911)$ ve gizli anahtarı da $(n,d) = (6012707, 422191)$ olarak elde edilir.

2. Şifreleme:

B şahsı, $M = 5234673$ metnini şifrelemek için, A nın açık anahtarı olan $(n,e) = (6012707, 3674911)$ yi alır ve aşağıdaki gibi $\boldsymbol{C}$ yi hesaplar.

$$C = M^e (mod\, n) = 5234673^{3674911} (mod\, 6012707) = 3650502$$

Ve bu şifreli metni A ya gönderir.

3. Deşifreleme:

A şahsı bu şifreli metni çözmek için gizli anahtarı olan $(n,d) = (6012707,422191)$ yı alır ve aşağıdaki şekilde şifreyi çözer;

$$M = C^d (mod\ n) = 3650502^{422191} (mod\ 6012707) = 5234673$$

olarak elde edilir ve deşifre işlemi sona erer.

RSA algoritmasının dijital imzalar için de kullanıldığını söylemiştik. Şimdi RSA dijital imza sistemini inceleyelim.

2.2.2 RSA İmza Sistemi:

RSA algoritması dijital imzalar için de kullanılabilir. A şahsının açık anahtarı (n,e) ve gizli anahtarı da (n,d) olsun. Öncelikle M mesajının imzalanabilmesi için $\{0,1,2,...,n-1\}$ olması istenir. Daha sonra hesaplamalar yapılır.

1. İmzalama:

A şahsı, B ye mesajını göndermek isterse metne kendi gizli anahtarını uygular, yani;

$$\rho \equiv M^d (mod\ n)$$

Daha sonra (ρ, M) imzalı mesajı B ye gönderir.

2. İmza Doğrulama:

B, A dan aldığı (ρ, M) imzalı mesajı doğrulamak için

$$M \equiv \rho^e (mod\ n)$$

değerini hesaplar. Çıkan sonuç M ise imza doğrulanmış olur.

RSA imza şemasını bir örnek üzerinde görelim:

Örnek 2.11:

Anahtar oluşturma:

A kişisi $p = 7927, q = 6997$ asal sayılarını seçsin. Daha sonra $n = pq = 7927.6997 = 55465219$ ve $\phi = (p-1)(q-1) = 55450296$ değerlerini hesaplasın. $e = 5$ seçsin. $gcd(e = 5, \phi = 55450296) = 1$ ve $1 < e = 5 < \phi = 55450296$ şartları sağlanıyor. Daha sonra $ed = 5d \equiv 1 (mod\ \phi = 55450296)$ denklemini sağlayan $d = 44360237$ sayısı bulunur. Böylece açık anahtar $(n, e) = (55465219, 5)$ ve gizli anahtar da $(n, d) = (55465219, 44360237)$ olur.

1. İmzalama:

$M = 31229978$ metnini imzalamak için A şahsı aşağıdakini hesaplar:

$$\rho = M^d (mod\ n) = 31229978^{44360237} (mod\ 55465219) = 30729435$$

Ve elde ettiğim $(\rho, M) = (30729435, 31229978)$ imzalı mesajını B şahsına gönderir.

2. İmza doğrulama:

A dan gelen $(\rho, M) = (30729435, 31229978)$ imzalı mesajı onaylamak için B şahsı aşağıdaki işlemi yapar:

$$M = \rho^e (mod\ n) = 30729435^5 (mod\ 55465219) = 31229978$$

değerini elde eder. $\boldsymbol{M}$ değerini elde ettiği için imza doğrulanmış olur.

Açık anahtarlı bir algoritmalarda açık anahtar herkes tarafından bilinen bir anahtardı. Burada gizli kalması gereken gizli anahtardır. Açık anahtar ile gizli anahtar arasında matematiksel bağın çok kuvvetli olması gerekir. RSA algoritmasında açık anahtar ile gizli anahtar arasındaki matematiksel bağ, çok büyük bir tamsayının asal çarpanlarına ayrılması problemidir. Yani $\boldsymbol{p}$ ve $\boldsymbol{q}$ asallarının bulunması problemidir. Matematikte bunun kolay bir yolu yoktur. Verdiğimiz örneklerde küçük sayılar aldık fakat günümüz sistemlerinde kullanılan sayılar çok büyüktür ve asal çarpanlarına ayrılması çok zordur. Açık anahtarlı

şifrelemenin bir diğer güvenliği de burada yatar; şifrenin çözülmesi için gerekli olan ihtimal sayısı çok fazladır. Kullanılan asal sayılar çok büyük olduğu için deneme yanılma yöntemi saldırılarında saldırganın denemesi gereken sayı çok fazladır. Asimetrik bir algoritmanın çözülmesi için gerekli olan, p ve q sayılarının bulunmasıdır. p ve q sayıları bilinen (ve herkese açık olan) açık anahtar içerisindeki n sayısından bulunabilir. Ama bunun bulunması teorik olarak mümkün kabul edilir. Az önce bahsettiğimiz gibi örneklerimizde kolay hesaplayabilelim diye çok küçük asal sayılarla çalıştık, ama günümüzde neredeyse standart hale gelen 128 bit şifrelemede 45 hanelik asal sayılar kullanılır ki asal çarpanlarına ayrılması gereken n sayısı, iki tane 45 hanelik asalın çarpımından oluşmaktadır.

128 bit bir şifrenin çözülmesi için ihtimal sayısı 2^{128} dir. Ve bu kadar ihtimal normal bir bilgisayar ile ortalama 2 trilyon yılda çözülür. (Saniyede 1 trilyon işlem yapabilen 1 milyon dolarlık süper bilgisayarlarla bu süre 1190 yıl dır). Japonya'da 64 bit bir şifre yaklaşık 100.000 (yüz bin) bilgisayarın paralel bağlanarak ortak çalışması sonucunda yaklaşık **3,5** yılda çözülmüştür. Günümüzde internet üzerinde çok önemli bilgilerimiz vardır. Elektronik alışveriş sitelerinden alışveriş yaparken kullandığımız kredi kartımızın numarası ve şifresi gibi. Alışveriş sitesine hiç çekinmeden yazdığımız kart numaramız ve şifremiz sitede bulunan **SSL** adı verilen bir protokolden geçerek bankaya gönderilir ve banka tarafından onaylanması istenir. Bu SSL protokolünün görevi mesajları şifrelemektir. Ve bu protokoller RSA algoritması ile şifreleme yapar. Bugün internet üzerinde **SSL** ile kurulu sayfalarda 128 bit şifreleme kullanılır. (sizin kredi kartı bilgilerinizin korunması için) ve her zaman şifrelemede verinin geçerlilik süresi çözülme süresinden az ise güvendesiniz demektir. Yani SSL kullanılan bir sayfada şifrelenerek yollanan kredi kartı bilgileriniz 1190 yıldan fazla süre yürürlükte kalmayacağı için **1190** yıl sonra çözülmüş olmasının bir mahzuru yoktur. Özel şifreleme programları ile **4096** bit şifreleme yapılabilir bu şekilde gerçekleştirilmiş bir şifrenin çözülme ihtimali 2^{4096} , yani yaklaşık olarak 10^{500} . Bu sayının büyüklüğünü anlamak için şöyle bir örnek verilebilir. Dünyada bilinen atom sayısı 10^{70} dir.

Bahsettiğimiz gibi günümüzde RSA algoritması güvenli sayılmaktadır. Çünkü temellendiği matematik probleminin çözümü bilinmemektedir. Bu yüzden şifreyi kırmanın tek yolu deneme yanılma yöntemidir. Çok büyük sayılarla çalışılınca bu da pratik olarak imkânsız olmaktadır. Fakat çok büyük asal sayıların bulunması da çok zor bir iştir. Ve bundan dolayı RSA algoritması ticari bir boyut kazanmıştır. Çok büyük asal sayılar bulunup sayılmaktadır. Fakat şu da bir gerçek ki günümüz matematik dünyasında, asal sayılar, üzerinde yoğun olarak çalışılan bir konudur. Asal sayılar hakkında keşfedilecek olan her şey RSA sisteminin güvenliğini tehdit edecektir. Eğer günün birinde asal sayılar tamamen analiz edilirse, tamsayılar arasındaki dağılımlar keşfedilirse yukarıda bahsettiğimiz olasılıklar ve yıllar hükümsüz kalacaktır. Çünkü dediğimiz gibi RSA algoritmasının güvenliği asal sayıların bulunamamasıyla sağlanıyor.

2.2.3 Knapsack Kriptosistem:

Knapsack açık anahtarlı şifreleme sistemi de diğer açık anahtarlı sistemler gibi matematiksel bir problemin çözülme zorluğuna dayanarak güçlenir. Knapsack sistemi alt küme toplam problemini temel alır. Buradaki temel düşünce, çözümü kolay olan bir alt küme toplam problemini alıp, bunu çözümü çok daha zor olan bir alt küme toplam problemine dönüştürmektir. Burada ilk alınan orijinal knapsack kümesi gizli anahtar, daha sonra bunun zorlaştırılmasıyla elde edilen dönüştürülmüş kanpsack kümesi de gizli anahtar olacaktır. Merkle-Hellman ve Chor-Rivest knapsack kriptosistem olmak üzere yaygın olan iki knapsack sistemi vardır. Biz burada Merkle-Hellman sistemi üzerinde duracağız. Yalnız ilk önce yapının matematiksel altyapısı üzerinde duracağız. Şimdi süperartan dizi ve alt küme toplam problemindan bahsedelim.

1. Matematiksel Altyapı:

Süperartan dizi:

Tanım 2.1: $B=(b_0,b_1,...,b_n)$ dizisinda her sayı kendisinden önce gelen sayıların toplamından büyükse, yani

$$b_i > \sum_{j=1}^{i-1} b_j, 2 < i < n$$

şartını sağlıyorsa bu diziye *süperartan dizi* denir.

Alt küme toplam problemi ve çözüm algoritması:

Tanım 2.2: $B = (b_0, b_1, ..., b_n)$ gibi süperartan bir dizi ve bir s sayısı verildiğinde, $x_i \in \{0,1\}$ olan ve $\sum_{i=1}^{n} x_i = s$ şartını sağlayan $(x_1, x_2, ..., x_n)$ dizisini bulma problemine *alt küme toplam problemi* ve ya *knapsack problemi* denir.

Bu gibi bir problemi çözmek için aşağıdaki adımlar uygulanır:

1. $i \leftarrow n$ olarak atanır. Bunun anlamı i ye n in değerini atamaktır. $i = n$ şeklinde düşünülünebilir ama anlam olarak eşittir diyemeyiz.
2. $i \geq 1$ ise aşağıdaki adımlar uygulanır:
3. Eğer $s \geq b_i$ ise $x_i = 1$ yazılır. Aksi takdirde $x_i = 0$ yazılır.
4. $s \leftarrow s - b_i$ ve $i \leftarrow i - 1$ alınarak 2. adımdan başlayarak tekrar uygulanır. Bu tekrar $i = 1$ oluncaya kadar devam eder.
5. Bulunan x_i lerden $(x_1, x_2, ..., x_n)$ dizisi oluşturulur. Ve bu dizi istenilen şartı sağlar.

Merkle-Hellman Knapsack algoritması:

Merkle-Hellman sistemi en yaygın kullanılan kapsack kriptosistemidir. Şimdi bu yapının üzerinde duralım.

a) Anahtar Üretim:

Bu kriptosistemde bir kişi kendi açık anahtarını ve buna bağlı olarak gizli anahtarını aşağıdaki adımları uygulayarak oluşturur.

1. Bir n sayısı alır.

2. Bir $(b_1,b_2,...,b_n)$ süper artan dizisi ve $M > b_1 + b_2 + ... + b_n$ şartını sağlayan bir M *mod sayısı seçer*.
3. $1 < W < M - 1$ ve $gcd(W,M) = 1$ şartlarını sağlayan herhangi bir W sayısı seçer.
4. $\{1,2,...,n\}$ sayılarıyla ifade edilen bir π permütasyonu seçer.
5. $i = 1,2,...,n$ için $a_i = Wb_{\pi(i)} (mod\ M)$ değerleri hesaplanır.
6. Açık anahtar $(a_1,a_2,...,a_n)$, gizli anahtar da $(\pi, M, W, (b_1,b_2,...,b_n))$ olur.

b) Şifreleme:

B şahsı **A** şahsı için mesajını şifreliyorsa aşağıdaki işlemleri uygular:

1. A nın açık anahtarı olan $(a_1,a_2,...,a_n)$ yı alır.
2. m mesajını n uzunluğunda ikilik dizi olarak: $m = m_1 m_2 ... m_n$ olarak ifade eder.
3. Daha sonra $c = m_1 a_1 + m_2 a_2 + ... + m_n a_n$ olarak hesaplar ve A ya gönderir.

Deşifreleme:

A şahsı mesajı deşifrelemek için de aşağıdaki işlemleri uygular:

1. Öncelikle $d = W^{-1} c (mod\ M)$ değerini hesaplar.
2. Süperartan alt küme toplam problemini çözerek $d = r_1 b_1 + r_2 b_2 + ... + r_n b_n$ ve $r \in \{0,1\}$ şartlarını sağlayan r_i değerlerini hesaplar.
3. Mesaj bitleri: $m_i = r_{\pi(i)}$, $i = 1,2,...,n$ olarak elde edilir.

Örnek 2.12: $n = 6$ olsun. A şahsı $(12,17,33,74,157,316)$ süperartan dizisini ve $M = 737 > 12 + 14 + 33 + 74 + 157 + 316 = 609$ modülünü seçsin. Daha sonra $gcd(W = 635, M = 609) = 1$ koşuluna uyan bir $W = 635$ sayısını seçsin. Son olarak da $1,2,3,4,5,6$ sayılarından oluşan $\pi = \begin{pmatrix} 123456 \\ 361254 \end{pmatrix}$

permütasyonunu seçsin. Bu permütasyona göre; $\pi(1)=3, \pi(2)=6, \pi(3)=1, \pi(4)=2, \pi(5)=5, \quad \pi(6)=4$ olmaktadır. A şahsı açık anahtarını $a_i = Wb_{\pi(i)}(mod\ M)$ eşitliğini kullanarak aşağıdaki şekilde oluşturur.

$$a_1 = Wb_{\pi(1)}(mod\ 635) = Wb_3(mod\ 635) = 635.33(mod\ 635) \equiv 319$$
$$a_2 = Wb_{\pi(2)}(mod\ 635) = Wb_6(mod\ 635) = 635.316(mod\ 635) \equiv 196$$
$$a_3 = Wb_{\pi(3)}(mod\ 635) = Wb_1(mod\ 635) = 635.12(mod\ 635) \equiv 250$$
$$a_4 = Wb_{\pi(4)}(mod\ 635) = Wb_4(mod\ 635) = 635.17(mod\ 635) \equiv 477$$
$$a_5 = Wb_{\pi(5)}(mod\ 635) = Wb_5\ mod\ 635) = 635.157(mod\ 635) \equiv 200$$
$$a_6 = Wb_{\pi(6)}(mod\ 635) = Wb_4(mod\ 635) = 635.74(mod\ 635) \equiv 559$$

Sonuç olarak A nın açık anahtarı $(319,196,250,477,200,559)$ knapsack dizisidir. Gizli anahtarı da $(\pi, M, W(12,17,33,74,157,316))$ dır. B kişisi A ya $m = 101101$ mesajını göndermek istiyorsa mesajı şifrelemek için aşağıdaki işlemi uygular:

$$c = 1.319 + 0.196 + 1.250 + 1.477 + 0.200 + 1.599 = 1605$$

elde eder ve bunu A ya gönderir. A şahsı da deşifreleme için aşağıdaki işlemleri uygular. Öncelikle

$$d = W^{-1}c(mod\ M) = 635^{-1}(mod\ 737) = 513(mod\ 737) \equiv 136$$

değerini hesaplar. Daha sonra

$$136 = 12.r_1 + 17.r_2 + 33.r_3 + 74.r_4 + 157.r_5 + 316.r_6$$

süperartan altküme toplam problemini çözer ve

$$(r_1, r_2, r_3, r_4, r_5, r_6) = (111100)$$

olarak bulur. Son olarak π permütasyonu uygulanarak $m_1 = r_3 = 1$, $m_2 = r_6 = 0$, $m_3 = r_1 = 1$, $m_4 = r_2 = 1$, $m_5 = r_5 = 0$, $m_6 = r_4 = 1$ olarak bulunur ve $m = 101101$ olarak elde edilir.

2.2.4 El-Gamal Kriptosistem:

El-Gamal açık anahtarlı sistem matematiksel temel olarak ayrık logaritma problemini almıştır. Bu sistemin anahtar transferi modeli *Diffie-Hellman Anahtar*

Anlaşması yöntemiyle aynıdır. Bu algoritma güvenliğini ayrık logaritma ve Diffie-Hellman problemlerinin doğrudan çözümlerinin olmaması ile sağlamaktadır. Ayrık logaritma kavramına değinelim.

1. Matematiksel Altyapı:

Tanım 2.3: G**;** n**.** Mertebeden sonlu bir devirli grup, α; G'nin bir üreteci ve $\beta \in G$ olsun. $0 < x < n-1$ için $x = log_{\alpha} \beta$ değerine β*'nın* α *tabanına göre ayrık logaritması* denir ve $\beta = \alpha^x$ dir. α ve β verildiğinde x'i bulma problemine de *ayrık logaritma problemi* denir.

Şimdi algoritmayı inceleyelim.

2. Algoritma:

a) Anahtar oluşturma:

Sistemdeki her kişi açık anahatrını ve gizli anahtarını oluşturmak için aşağıdaki adımları uygular:

1. Çok büyük bir p asalı ve $mod\ p$ ye göre tamsayıların oluşturduğu Z_p çarpımsal grubunun bir üreteci olması koşuluyla bir α seçilir.
2. $1 < a < p-2$ olacak şekilde bir a tamsayısı seçilir ve $a^{\alpha}\ mod\ p$ değeri hesaplanır.
3. Açık anahtar (p, α^a, α), gizli anahtar da a olur.

b) Şifreleme:

B şahsı A için m mesajını şifrelemek isterse aşağıdaki adımları uygular:

1. A nın açık anahtarı olan (p, α^a, α) yı alır.
2. m mesajını $\{0,1,...,p-1\}$ aralığında m tamsayısı olarak ifade eder.
3. $0 \le k \le p-2$ şartına uygun rastgele bir k tamsayısı seçer.

4. $\gamma = \alpha^k \ mod \ p$ ve $\delta = m.(\alpha^a)^k \ mod \ p$ değerlerini hesaplar.
5. Son olarak $c = (\gamma, \delta)$ şifreli metnini A ya gönderir.

c) Deşifreleme:

A şahsı aldığı şifreli mesajı deşifrelemek için aşağıdaki işlemleri uygular:

1. a gizli anahtarını kullanarak $\gamma^{-a} \ mod \ p$ değerini hesaplar. $(\gamma^{-a} = \alpha^{-ak} \ mod \ p)$
2. $\gamma^{-a}.\delta \ mod \ p$ değerini hesaplayarak m metnine ulaşır: $\gamma^{-a}.\delta \ mod \ p \equiv \alpha^{-ak}.m\alpha^{-ak} \ mod \ p \equiv m \ mod \ p$

Şimdi bir örnek üzerinde bu algoritmayı inceleyelim.

Örnek 2.13: A şahsı bir $p = 2357$ asal sayısını ve Z_{2357} grubunun bir üreteci olan $\alpha = 2$ tamsayısını seçer. Ve gizli anahtar olarak $a = 1751$ seçer. Şimdi de

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

değerini hesaplar. Böylece açık anahtar $(p =, \alpha^a = 1185, \alpha = 2)$ olarak belirlenmiş olur. B şahsı $m = 2035$ metnini şifrelemek için rastgele bir $k = 1820$ tamsayısı seçer ve

$$\gamma = 2^{1520} \ mod \ 2357 \equiv 1430$$
$$\delta = 2035.1185^{1520} \ mod \ 2357 \equiv 697$$

değerlerini hesaplar. Ve son olarak $c = (\gamma = 1430, \delta = 697)$ şifreli mesajını A şahsına gönderir. Bu şifreli mesajı alan A şahsı da deşifre işlemini şu şekilde yapacaktır:

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

değeri hesaplanır ve

$$m = 872.697 \ mod \ 2357 \equiv 2035$$

şeklinde m metnine ulaşır.

El-Gamal kriptosistem şifreleme-deşifreleme işlemleri dışında imzalama-doğrulama işlemleri içinde kullanılır. Yani bir mesajı şifreleyerek güvenliğini

sağlamak yerine mesajın istenilen kişiden geldiğini garanti altına almak için imzalama amacıyla ve bu imzanın doğrulanması amacıyla da kullanılır. RSA algoritması da bu amaçla kullanılabiliyordu. İmza algoritması olarak RSA kriptosistem incelenmişti.

2.2.5 Diffie-Hellman Anahtar Anlaşması:

Gizli anahtarlı kriptosistemlerdeki anahtar dağıtım problemine karşı geliştirilen ilk pratik çözüm *Diffie-Hellman anahtar değişimi (Diffie-Hellman key agreement)* algoritmasıdır. Üs olarak anahtar değiştirme olarak da bilinen bu sistem, daha önce hiç haberleşmemiş iki tarafın açık kanal üzerinden mesajlarını birbirine göndererek ortak bir anahtar yaratma temeline dayanır.

p, öyle büyük bir asal sayı olsun ki; Z_p de ayrık logaritma probleminin çözümü olmasın. Bir de Z_p de basit kök olan g sayısı olsun. Bu p ve g tamsayıları herkes tarafından bilinirken A ve B şahıslar aşağıdaki adımları uygulayarak ortak bir anahtar oluşturabilirler:

1. A şahsı $0 < a < p-2$ şartına uyan rastgele bir a sayısı seçer. $c = g^a$ değerini hesaplar ve B şahsına gönderir.
2. B şahsı $0 < b < p-2$ şartına uyan rastgele bir b sayısı seçer. $d = g^b$ değerini hesaplar ve A şahsına gönderir.
3. A şahsı aşağıdaki şekilde bir anahtar oluşturur:
   $k = d^a = (g^b)^a = g^{ba}$
4. B şahsı da aşağıdaki gibi anahtar oluşturur:
   $k = c^b = (g^a)^b = g^{ab}$

Yukarıda görüldüğü gibi A ve B şahısları ortak bir k anahtarı üzerinde anlaşmış olurlar.

3. KRİPTANALİZ

Kriptanaliz, kriptolojinin, şifreleme algoritmalarını analiz eden ve şifreleri anahtara sahip olmadan kırmanın yollarını arayan, dolayısıyla algoritmaların güvenliğini test eden, istenilirse kötü amaçlı olarak da kullanılan, dalıdır. Bu kesimde ilk olarak açık anahtarlı kriptosistemlerin analizine değineceğiz.

3.1 Açık Anahtarlı Sistemlerin Analizi:

Hatırladığımız gibi bu sistemlerin analizi tamamen matematiksel bir analizdir. Örneğin RSA algoritmasında temel problem büyük bir tamsayının asal çarpanlarının doğrudan bir formülle bulunamayışıydı. Benzer şekilde El-Gamal kriptosistemi ayrık logaritma problemine dayanıyordu. Şimdi bu problemlerin çözüm yollarına değinelim.

3.1.1 Asal Çarpanlara Ayırma Problemi:

RSA ve Rabin algoritmalarının da arasında bulunduğu birçok simetrik sistem gücünü bu problemden alır. Bu kesim de bu problemi kısacan tanıtıp çözümü için geliştirilen bazı algoritmaları vereceğiz.

Tanım 3.1: Bir n pozitif tamsayısı verildiğinde, $i = 1,2,...,k$ için, p_i ler farklı asallar ve $e_i \geq 1$ olmak üzere, $n = p_1^{e_1} p_2^{e_2} ... p_k^{e_k}$ şeklinde asal sayıların kuvvetlerinin çarpımı şeklinde yazımını bulma problemine, asal çarpanlara ayırma problemi *(integer factorization problem) denir.*

Bir tamsayının asal çarpanlar şeklinde yazılması için, sayının bileşik sayı olması gerekir. Aksi takdirde kendisi de bir asal sayı olduğu için başka asal sayıların çarpımı şeklinde yazılamaz. Sayının asal olup olmadığını anlamak için çeşitli asallık testleri bilinmektedir. Konumuz gereği bileşik sayılarla çalışacağımız için bu asallık testlerini kullanmayacağız ve testler hakkında bilgi verilmeyecektir.

Şimdi asal çarpanlara ayırma problemini çözmek için geliştirilmiş bazı algoritmaları inceleyelim.

1. Pollard'ın RHO çarpanlara ayırma algoritması:

Pollard'ın RHO algoritması bileşik bir tamsayının küçük asal çarpanlarını bulmak için geliştirilmiştir. Büyük asalların çarpımından oluşan bileşik tamsayıları asal çarpanlarına ayırmak için bu algoritma kullanışlı değildir.

S: n elemanlı (sonlu) bir küme ve $f : S \to S$ bir fonksiyon olsun. x_0, S'in herhangi bir elemanı olsun. $x_{i+1} = f(x_i)$ ile verilen $(x_0, x_1, ...)$ dizisini düşünelim. S kümesi sonlu olduğu için verilen seri devirli olmak zorundadır. Yani dizinin belirli bir teriminden sonra, dizinin ilk terimi, sonra ikinci terimi ve bu şekilde belirli bir terimden sonra dizi ilk terimden başlayarak devretmeye başlayacaktır. Örneğin 5. terimden sonra başa dönen bir devirli dizi $(x_0, x_1, ..., x_5, x_6 = x_0, x_7 = x_8, ...)$ seklinde terimlerden oluşur. Bu diziyi incelemek istersek, tüm terimleri değil de sadece bir devirdeki terimleri incelememiz bize kolaylık sağlayacaktır. Fakat devreden kısmı tespit etmemiz gerekir. Bunun için de $i \neq j$ olmak üzere $x_i = x_j$ şartını sağlayan terimler tespit edilir. Bu işi için kullanılan en önemli algoritma Floyd'un devir bulma algoritması *(Floyd's cycle-finding angorithm)* dır. Bu algoritmada ilk olarak (x_1, x_2) çifti alınır. Daha sonra (x_{i-1}, x_{2i-2}) çiftlerinden faydalınalarak (x_i, x_{2i}) çiftleri hesaplanır. $x_m = x_{2m}$ şartını sağlayan bir çift bulununcaya kadar hesaplama devam eder. Ve böylece istenilen özelliği saplayan iki terim elde edilir. Eğer dizinin uzunluğu λ ve devirin uzunluğu μ ise, $\lambda < m \geq \lambda + \mu$ şartıyla, $x_m = x_{2m}$ şartını sağlayan ilk m sayısı $m = \mu[1 + (\lambda / \mu)]$ formülü ile hesaplanabilir.

Şimdi p; n bileşik tamsayısının bir asal böleni (çarpanı) olsun. Pollard'ın RHO algoritması p'yi bulmak için, $x_0 = 2$ ve $x_{i+1} = f(x_i) = x_i^2 + 1 (mod\ p), i \geq 1$ ile verilen $(x_0, x_1, x_2, ...)$ dizisindeki, $i \neq j$ olmak üzere $x_i = x_j$ şartını sağlayan (tekrar eden) terimleri bulmaya çalışır. Floyd'un devir bulma algoritması uygulanarak $x_m = x_{2m} (mod\ p)$ şartını sağlayan x_n ve x_{2m} terimleri hesaplanır. Eğer x_n ve x_{2m} sayıları $1 < gcd(x_m - x_{2m}, n) < n$

şartını sağlarsa $p = gcd(x_m - x_{2m}, n)$ sayısı n sayısının bir asal çarpanıdır. ($gcd(x_m - x_{2m}, n) = n$ olması ihtimali ihmal edilebilir.)

Bir n bileşik sayısının bir asal çarpanını bulmak için pollard'ın RHO algoritması aşağıdaki gibi kısaca uygulanabilir:

1. $a \leftarrow 2, b \leftarrow 2$ olarak alınır.
2. $i = 1,2,...$ için aşağıdaki adımlar uygulanır:
3. $a \leftarrow a^2 + 1 (mod\ n)$ ve $b \leftarrow b^2 + 1 (mod\ n)$ değerleri hesaplanır.
4. $d = gcd(a - b, n)$ değeri hesaplanır.
5. $1 < d < n$ ise d bir asal çarpandır. Aksi takdirde asal çarpan değildir, 3. adıma dönülerek iterasyona devam edilir.

Örnek 3.1: $n = 455459$ sayısı için Pollard'ın RHO algoritması uygulanmıştır. Alınan çıktılar aşağıdaki tabloda verilmiştir.

a	b	d
5	26	1
26	2871	1
677	189675	1
2871	15560	1
44380	416250	1
17968	43670	1
121634	164403	1
155260	274944	1
44567	68343	743

Bu yüzden $n = 455459$ sayısının asal çarpanları 743 ve $455459 / 743 = 613$ olarak bulunur.

Bu algoritmanın hatalı sonuç vermediği durumlarda $f(x_i) = x_i^2 + c (mod\ p), i \geq 1$ şeklinde c 1 den farklı bir tamsayı seçilerek tekrar denenir.

2. Pollard'ın p-1 çarpanlara ayırma algoritması:

Tanım 3.2: B ve n birer pozitif tamsayı olsun. Eğer n tamsayısının tüm asal çarpanları B tamsayısından küçükse, *n tamsayısı B sınırına göre düzgündür (smooth with respect to a bound B)* denir, *B-düzgün* şeklinde gösterilir.

B bir düzgünlük sınırı olsun. Q da, B den küçük olan tüm asal sayıların ve asalların kuvvetleri olan sayıların en küçük ortak katı olsun. n de B den büyük bir tamsayı olsun. Eğer $q^t \le n$ ise $t.\ln q \le \ln n$ olup $t \le \left(\frac{\ln n}{\ln q}\right)$ olur. Sonuç olarak $Q = \prod_{q \le B} q^{(\ln n / \ln q)}$ olur.

Eğer bir p sayısı bir n sayısının asal bir çarpanı ve $p-1$ de $B-düzgün$ ise $p-1 \mid Q$ dir ve $gcd(a, p) = 1$ şartını sağlayan bir a tamsayısı için Fermat Teoreminden dolayı $a^Q \equiv 1 (mod\ p)$ dir. Sonuç olarak $d = gcd(a^q - 1, n)$ dir. Bir asal çarpan budur.

Pollard'ın p-1 çarpanlara ayırma algoritması yukarıdaki mantıkla çalışır. Algoritmayı madde madde özetleyecek olursak, aşağıdaki adımlardan oluştuğu görülür:

1. Asal olmayan bir B sayısı verildiğinde, il olarak bir B düzgünlük sınırı seçilir.
2. $2 \le a \le n-1$ olacak şekilde herhangi bir a sayısı seçilir ve $d = gcd(a,n)$ hesaplanır. Eğer $d \ge 2$ ise d bir asal çarpan olarak bulunur. Aksi takdirde işlemlere aşağıdaki gibi devam edilir.
3. $\forall q \le B$ asalı için, $t = \left(\frac{\ln n}{\ln q}\right)$ ve $a \leftarrow a^{q^t} \bmod n$ değerleri hesaplanır.
4. $d = gcd(a-1, n)$ değeri hesaplanır. $d \ne 1$ ve $d \ne n$ ise d bir asal çarpandır. Aksi halde değildir.

Örnek 3.2: $n = 19048567$ sayınsın asal çarpanlarını bulalım. Yukarıdaki algoritmayı adım adım uygulayalım. İlk olarak düzgünlük sınırı olarak $B = 19$ seçelim. $gcd(3,19048567) = 1$ olduğu için $a = 3$ seçelim. Aşağıdaki tabloda tüm $q \le B = 19$ asal sayılarına karşılık gelen $t = \left(\frac{\ln n}{\ln q} \right)$ sayıları (yaklaşık olarak), ve $a \leftarrow a^{q^t} \bmod n$ değerleri verilmiştir.

q	t	a
2	24	2293244
3	15	13555889
5	10	16937223
7	8	15214586
11	6	9685355
13	6	13271154
17	5	11406961
19	5	554506

$gcd(55403 - 1,19048567) = 5281$ değeri hesaplanır. Bir asal çarpan 5281 dir, diğeri de $19048567 / 5281 = 3607$ dir. 55403 dışındaki a değerleri için $d = gcd(a - 1,n) = 1$ oluyor. Onun için 55406 yı seçtik. Bunda da $d = gcd(a - 1,n) = 1$ olsaydı B sınırını arttırıp tekrar deneyecektik.

Asal çarpanlarına ayırma probleminde kullanılan algoritmalardan bazılarını burada inceledik. İncelediklerimizden başka; *quadratik elek algoritması (quadratik sieve algorithm), sayısal elek algoritması (number sieve algorithm), deneyerek bölme algoritması (trial divison)* gibi bazı algoritmalar da kullanılmaktadır.

3.1.2 Ayrık Logaritma Problemi:

Bu bölümde El-Gamal kriptosistem ve Diffie_hellman Anahtar Anlaşması sistemleri gibi bazı sitemlerin temelini oluşturan ayrık logaritma problemi ve problemin çözüm yolları üzerinde durulacaktır.

Tanım 3.3: G**;** n**.** Mertebeden sonlu bir devirli grup, α; G'nin bir üreteci ve $\beta \in G$ olsun. $0 < x < n-1$ için $x = log_{\alpha} \beta$ değerine β*'nın* α *tabanına göre ayrık logaritması* denir ve $\beta = \alpha^x$ dir. α ve β verildiğinde x'i bulma problemine de *ayrık logaritma problemi* denir.

Şimdi bu problemin çözümü için geliştirilmiş olan bir algoritmayı verelim:

1. Baby-step Giant-step algoritması:

Baby-step Giant-step algoritmasını özetleyecek olursak, n. Mertebeden bir G devirli grubu, bu grubun bir α üreteci ve bir $\beta \in G$ elemanı verildiğinde $x = log_{\alpha} \beta$ değerini hesaplamak için aşağıdaki adımlar uygulanır:

1. $m \leftarrow \sqrt{n}$ (yaklaşık olarak tamsayı değeri) atanır.
2. $0 < j < m$ olmak üzere $(j, \alpha^j (mod\ p))$ çiftleri hesaplanır ve tablo olarak yazılır. (Tablo yazılırken α^j satırına göre artan sırada yazılması kolaylık sağlayacaktır.)
3. $\alpha^{-m} (mod\ p)$ değeri hesaplanır ve $0 < i < m-1$ için $\gamma = \beta\alpha^{mi} (mod\ p)$ değerleri hesplanarak tablo halinde yazılır.
4. Yazdığımız iki tablo karşılaştırılarak $\gamma = \alpha^j$ şartını sağlayan i ve j değerleri kullanılarak $x = im + j$ değeri bulunur.

Örnek 3.3: $p = 113$ olsun. $n = 112$ olur. Z^*_{113} ün bir üreteci olan $\alpha = 3$ seçelim. $\beta = 57$ seçelim ve $x = log_3 57$ değerini hesaplayalım:

1. $m \leftarrow \sqrt{112} \approx 11$ olarak atayalım.
2. $0 < j < 11$ için aşağıdaki tabloyu elde ederiz:

j	0	1	8	2	5	9	3	7	6	10	4
$3^j (mod\ 112)$	1	3	7	9	17	21	27	40	51	63	81

3. $\alpha^{-m} = 3^{-11} = 38^{11} (mod\ 113)$ değeri hesaplanır.

4. $i = 1,2,...$ için $\gamma = \beta\alpha^{-mi} = 57.58^{i} (mod\ 113)$ değerleri hesaplanarak aşağıdaki tablo oluşturulur:

i	0	1	2	3	4	5	6	7	8	9
$\gamma = 57.58^{i} (mod\ 113)$	57	29	100	37	112	55	26	39	2	3

5. $i = 9$ için $\gamma = \beta\alpha^{-9m} = 3 = \alpha^{1}$ olduğundan $x = im + j = 9.11 + 1 = 100$ olarak elde edilir. Yani $log_{3} 57 = 100$ olarak bulunur.

3.2 Gizli Anahtarlı Sistemlerin Analizi:

Bu kesimde gizli anahtarlı (asimetrik) sistemlerin analizine değinilecektir. Akan şifrelerden bahsetmediğimiz için bu kesimde blok şifreleme sistemlerinin analizine değinilecektir. Fazla detaya inmeden kısaca genel kavramlar verilecektir.

Kerkchoff Prensibi ne göre: *kriptanalizci şifreleme algoritmasının detaylarına ulaşma gücüne sahiptir ve sistemde sadece anahtar gizlidir.* Bu prensibe göre tasarlanmış ve günümüzde çok etkili olan birçok algoritma vardır. İkinci dünya savaşında İngiliz ve Polonyalı matematikçiler Alman enigma makinelerinin analizini yaparak algoritmayı kırmayı başarmışlardır. Benzer şekilde farklı bir algoritma olan RC4 algoritması da kırılmıştır. Bir kriptanalizcinin amacı şifreli metni herhangi bir algoritma kullanarak açık metne çevirmektir. Kriptanaliz bu işi, genellikle, gizli anahtarın tamamını veya birkısmını elde ederek yapar. Günümüzde kullanılan bazı kriptanaliz senaryoları aşağıdaki gibi sıralanabilir:

1. Sadece Şifreli Metin Atağı (Ciphertext Only): En etkili kriptanaliz atağıdır. Sadece haberleşme yeteri kadar dinlenip yeterli miktarda şifreli metin elde edilerek uygulanır.

2. Bilinen Açık Metin Atağı (Known Plaintext): Dinlemeler sonucu bir miktar şifreli-açık metin çifti elde edilince uygulanabilir. Bilinenlere dayanılarak bazı şifreli metinlerin açık halinin tahmini esasına dayanır.

3. Seçilmiş Açık Metin Atağı (Choosen Plaintext): Analizcinin seçtiği açık metinleri şifreleyebildiği varsayılan bir atak senaryosudur. Analizci şifreleme algoritmasının güvenli olarak yerleştirildiği mekanizmayı elde edebilir. Analizci haberleşmede aktif rol oynar.

3.2.1 Blok şifrelerin analizi:

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

1. Diferansiyel Kriptanaliz:

Diferansiyel Kriptanaliz methodu DES, GDES, Lucifer, FEAL, PES, IDEA, LOKI'89, REDOC ve Khafre dâhil olmak üzere birçok 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. Diferansiyel Kriptanaliz, kriptosistemlerin yeniden gözden geçirilmesine, tekrar dizayn 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 yapar. Bu farklar olası anahtarları ihtimal atamak ve ihtimali en yüksek anahtarları belirlemek için kullanılır. Aynı farka sahip olan birçok açık metin ikilisini ve karşı gelen kapalı metin ikililerini kullanır.

Şimdi 2n bit blok uzunluğundaki blok şifreleme sistemlerini analizinin özetini vereceğiz. İlk önce eşit uzunluktaki iki bit dizinin X ve $\dot{X}$ arasındaki **farkı** (difference) tanımlayalım.

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

Burada $\otimes$; bit dizi grupları üzerinde, döngü fonksiyonu üzerinde anahtar ile metin girdisinin birleştirilmesini sağlayan bir grup operasyondur ve $\dot{X}^{-1}$ de $\otimes$ işlemine 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:

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

Feistel yapısındaki blok şifre sistemlerinin birçoğu 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 metnimiz $P = C_0$ ve C_i de i döngü şifrelemesinden oluşan kapalı metnimiz olsun. α_i, ΔC_i'nin beklenen değeri ve α_0 seçilen $\Delta P = \Delta C_0$ olmak üzere bir **r-döngü karakteristiği** $(\alpha_0, \alpha_1, ..., \alpha_r)$ dir. Burada ΔP açık metin farkı ve ΔC_i de i döngüden sonraki kapalı metin farkıdır. Bir karakteristiğin olasılığı verilen $i-1$ döngü şifrelemesinden oluşan $\Delta C_{i-1} = \alpha_{i-1}$ farkına göre, i döngü şifrelemesinden sonra elde edilen $\Delta C_i = \alpha_i$ farkının elde edilmesi koşullu olasılığıdır. Rasgele, hep aynı şekilde seçilmiş döngü anahtarları K_i ler için bir karakteristiğin olasılığı:

$$Pr(\Delta C_{i-1} = \alpha_{i-1}, \Delta C_i = \alpha_i, ..., \Delta C_1 = \alpha_1 \mid \Delta C_0 = \alpha_0)$$

Koşullu olasılığıdır. Bu olasılığı hesaplamak çok zor olabilir. Bununla beraber bazı blok şifre sistemleri için bu olasılık her bir döngünün olasılıkları kullanılarak hesaplanabilir (Markov şifre sistemleri).

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.

4. STEGONAGROFİ

Steganografi, eski Yunancada "gizlenmiş yazı" anlamına gelir ve bilgiyi gizleme (şifreleme değil) bilimine verilen addır. Steganografi'nin şifrelemeye göre en büyük avantajı bilgiyi gören bir kimsenin gördüğü şeyin içinde önemli bir bilgi olduğunu fark edemiyor olmasıdır, böylece içinde bir bilgi aramaz (oysaki bir şifreli mesaj, çözmesi zor olsa bile, gizemi dolayısıyla ilgi çeker).

Tarihte steganografi, hem şifreleme öncesi dönemde hem de sonrasında (ilgi çekmeme avantajından dolayı) kullanılmıştır. Birkaç örnek verecek olursak:

- Eski Yunanistan'da, insanlar mesajları tahtaya yazıp üzerini mumla kaplarlardı. Böylece cisim kullanılmamış bir tablete benzerdi öte yandan mumun eritilmesiyle birlikte içindeki gizli mesaj okunabilirdi.
- Herodotos'un bir hikâyesine göre Pers saldırısının öncesinde saçları kazınan bir kölenin kafasına yazılan uyarı mesajı, saçlarının uzaması sayesinde saklanmıştır. Bu sayede, mesaj dikkat çekmeden gerekli yere ulaşabilmiş, ulaştığında da kölenin saçları tekrar kesilerek uyarı okunabilmiştir.
- İkinci Dünya Savaşı sırasında, New York'taki bir Japon ajanı (Velvalee Dickinson) oyuncak bebek pazarlamacısı kılığı altında saklanmaktaydı. Bu ajan, Amerikan ordusunun hareketlerini bebek siparişi içeren mektuplar içine saklayarak Güney Amerika'daki adreslere gönderiyordu.

- Özellikle 1960'larda mor ötesi boya ile yazı yazabilen sprey ve kalemler moda idi. Bu kalemlerin yazdığı yazılar, sadece bir mor ötesi ışıkla görülebiliyordu.

Steganografiye basit bir örnek de verebiliriz: aşağıdaki yazı, normal bir şekilde okunduğunda normal bir yazı gibi gözükürken satır atlanarak okunduğunda daha değişik bir mesaj içerdiği görülmektedir:

Benim için futbolda önemli olan centilmenlik
ve dostluktur. Hedefim illa ki kazanmak
falan değildir. Ben sadece kendi reklamını düşünen
kişiliğe sahip olsam başka olurdu. Ben net
birisiyim arkadaş. Takımım kazanırsa mal-
zemecisine kadar mutlu oluruz. Ben de sporcu
varlığımı geliştiririm. Hakemlere baskı uygulamak
sportmenliğe yakışmaz. Fair-play için mücadele
gerekirse onu da yaparım. Medyayı da bağ-
rıma basmışım, spor uğruna gülmüşüm ve ağ-
lamışım, kafam rahat!

Steganografi teknikleri elektronik ortamda da sıklıkla kullanılmaktadır. Bilgisayar dünyasında en sık kullanılan steganografi tekniği resim, ses, veya video dosyaları içine veri gizlemektir. Resim, ses ve video gibi verilerde dosya boyutları çok büyüktür. Öte yandan, seste ve görüntüdeki küçük bozuklukları insan beyni fark edemediği için, kasıtlı olarak periyodik bozukluklar şeklinde dosyanın içine başka bir dosya saklanabilir.

Örneğin Resimlerde, 24-bit'lik bir kanallama kullanılır. Bu kanallar kırmızı, yeşil ve mavi'dir ve her bir kanal 8 bit'lik bir değere sahip olabilir. Öte yandan, bir insan her renkteki 8 bit'in son iki (hatta üç) bitindeki değişiklikleri göremeyecektir, zira bu değişiklik 3 (veya son üç bit için 7) ton değişikliğe eşdeğerdir (oysaki toplamda 255 ton var). Bu durumda, son üç bit asıl rengin detayları yerine başka bir bilgi saklamak için kullanılabilir: hesaplayacak olursak görüyoruz ki günümüzde

cep telefonlarının çektiği (dolayısıyla sıkça paylaşılan) 1600 * 1200 çözünürlükteki bir resmin içine bile 2 Megabayta kadar veri saklanabilir!

5. HASH FONKSİYONLARI

Tek yönlü fonksiyonlar (*one way functions*) olarak bilinen bu fonksiyonlar şifreleme algoritmaları değildirler. Çünkü bu fonksiyonlar yapıları gereği veriyi sıkıştırarak sabit uzunlukta çıktı verirler ve çıktının geri dönüşü yoktur. Yani bu fonksiyonlarla şifrelenen çıktılar deşifre edilemezler. Bu fonksiyonların çıktılarına hash denir.

Hash fonksiyonları şifreleme için kullanılmazlar. Genelde kimlik doğrulama için kullanırlar. Elektronik ortamda, özellikle internet ortamında çok sık kullanılırlar. Örneğin neredeyse tüm e-mail servislerinde bu yöntemle kimlik doğrulanır. Kimlik doğrulama işlemi şu şekilde gerçekleşir. Sisteme üye olunurken kaydettirdiğimiz şifremiz, sistem veritabanına açık metin olarak kaydedilmez. Veritabanına istenmeyen bir kişinin girme ihtimali düşünülerek bu bilgiler herhangi bir hash fonksiyonundan geçirilip elde dilen hash değerleri kaydedilir. Daha sonra bis sisteme (örneğin mail kutumuza) gireceğimizde, şifre sorusuna cevap verdiğimizde sistem bizim verdiğimizin hashını hesaplar ve veritabanına önceden kaydettirdiğimiz hash ile aynı olup olmadığını kontrol eder. Başka şekilde kontrol edemez. Çünkü veritabanındaki hashımızı eski haline getirmek mümkün değildir. Bu şekilde sistem güvenliği sağlanmış olur. Veritabanına herhangi bir üçüncü kişi girse bile elde ettiği değerleri deşifre edemeyeceği için güvende oluruz.

En iyi bilinen hash fonksiyonları MD2, MD4, MD5, SHA-1 dir. Günümüzde en yaygın kullanılanı MD5 tir.

Önceleri hash fonksiyonları anahtarsız olurken günümüzde gizli anahtarlı hash fonksiyonları da üretilmiştir. Bir hash fonksiyonunun sağlaması gereken özellikleri şu şekilde sıralayabilirz:

1. Girdi 'M' herhangi uzunluktaki bir veri bloğu olabilmelidir.
2. Sabit uzunlukta bir çıktı verir (*compression*)
3. Verilen herhangi bir M için, hash değeri olan $h = H(M)$'nin hesaplanması çok kolay olmalıdır.

Bunun yanında aşağıdaki özellikler de aranmaktadır.

4. Verilen herhangi bir hash çıktısı, h, için *H(x)=h* olan *x*'i bulmak kolay hesaplanabilir olmamalıdır. (*pre-image resistance*)
5. Verilen herhangi bir veri bloğu, *x*, için *x*'ten farklı *H(x)=H(y)* özelliğini sağlayan *y* kolaylıkla bulunamamalıdır. (*2nd pre-image resistance*)
6. *H(x)=H(y)* özelliğini gösteren herhangi *(x,y)*, veri ikilisi kolaylıkla bulunamamalıdır. (*collision resistance*)

1., 2. ve 3. özellikler genel olarak bir hash fonksiyonunu tanımlamaktadır. 4 ve 5 numarali özellik hash fonksiyonlarını tek yönlü yapmaktadır. Ayrıca 3, 4 ve 5. özellikler *checksum*, *fingerprint, message digest* gibi isimler için yetmektedir. Fakat birçok uygulama için tek-yönlülük yeterli değildir. Onun için 6. özellik sayesinde çakışma-direnci(bağışıklığı) (*collision-resistance*) özelliği eklenmiş olur. Bunun yanında bu özelliklere alternatif isimler de verilebiliyor. Bazi kaynaklarda Bazi kaynaklarda 5. özellik için zayıf çakışma direnci (*weak collision resistance*), 6. özellik için güçlü çakışma direnci (*strong collision resistance*) denmekedir.

KAYNAKLAR

[1] W. STALLINGS, *Cryptography and Network Security Principles and Practices, Fourth Edition, Prentice Hall Publication, 2005.*

[2] A. MENEZES, P. VAN OORSCHOT, ve S. VANSTONE, *Handbook of Applied Cryptography, CRC Press, 1996.*

[3] FIPS PUB 46-3 *(Federal Informatıon Processing Standards Publicatıon)*, *DATA ENCRYPTION STANDARD (DES)* , *1999.*

[4] FIPS PUB 197 *(Federal Informatıon Processing Standards Publicatıon)*, *Announcing the ADVANCED ENCRYPTION STANDARD (AES)* , *2001.*

[5] A. MURAT EREN, *Açık Anahtarlı Kriptografi, Penguence Dergisi Sayı-2, 2005.*

[6] *Kriptolojiye Giriş Ders Notları***,** *Uygulamalı Matematik Enstitüsü, Kriptografi Bölümü, ODTÜ, TÜRKİYE, 2004.*

[7] *http://tr.wikipedia.org/wiki/Şifreleme, Vikipedi, Özgür Ansiklopedi.*

[8] *http://www.mutasyon.net/makaleler.asp?id=30, Mutasyon.net-Teknoloji.*

[9] *http://www.uekae.tubitak.gov.tr/ , UEKAE Web Sayfası- TUBİTAK.*

[10] *http://kriptoloji.tr.cx/ , KRİPTOLOJİ.*

[11] *http://www.makale.gen.tr/Category/27.html , Makale.gen.tr-Kriptoloji.*

[12] *http://www.kriptoloji.net/ , Kriptoloji.net (İlk Türk Kriptoloji Sitesi).*

[13] *http://en.wikipedia.org/wiki/Cryptography , Wikipedia, The Free Encyclopedia.*